INFORMATION SYSTEMS DEVELOPMENT

Advances in Theory, Practice, and Education

- ❐ ISD Methodologies
- ❐ Method Engineering
- ❐ Web Systems Engineering
- ❐ Information Analysis and Data Mining
- ❐ Costing Methods
- ❐ Security Issues

Edited by
Olegas Vasilecas, Albertas Caplinskas,
Wita Wojtkowski, W. Gregory Wojtkowski,
Jože Zupančič, and Stanisław Wrycza

Information Systems Development ISD'2004

Proceedings of the Thirteenth International Conference on Information Systems Development

Advances in Theory, Practice and Education

Vilnius, Lithuania, September 9–11, 2004

Edited by

Olegas Vasilecas
Vilnius Gediminas Technical University
Lithuania

Albertas Caplinskas
Institute of Mathematics and Informatics
Lithuania

Wita Wojtkowski
Boise State University
USA

W. Gregory Wojtkowski
Boise State University
USA

Joze Zupancic
University of Maribor
Slovenia

and

Stanislaw Wrycza
University of Gdansk
Poland

Springer

Information Systems Development

Advances in Theory, Practice, and Education

Information Systems Development

Advances in Theory, Practice, and Education

Edited by

Olegas Vasilecas
Vilnius Gedinimas Technical University
Vilnius, Lithuania

Albertas Caplinskas
Institute of Mathematics and Informatics
Vilnius, Lithuania

Wita Wojtkowski and W. Gregory Wojtkowski
Boise State University
Boise, Idaho, USA

Jože Zupančič
University of Maribor
Kranj, Slovenia

Stanisław Wrycza
University of Gdansk
Gdansk, Poland

Springer

Olegas Vasilecas
Vilnius Gedinimas Technical University
Sauletekio 11
LT-10223 Vilnius
Lithuania
olegas@fm.vtu.lt

Albertas Caplinskas
Institute of Mathematics and Informatics
Akademijos 4
LT-08663 Vilnius
Lithuania
alcapl@ktl.mii.lt

Wita Wojtkowski
1910 University Drive
Boise State University
Boise, Idaho 83725
USA
wwojtkow@boisestate.edu

W. Gregory Wojtkowski
1910 University Drive
Boise State University
Boise, Idaho 83725
USA
gwojtkow@boisestate.edu

Jože Zupančič
Systems Development Laboratory
Faculty of Organizational Sciences
University of Maribor
4000 Kranj
Slovenia
Joze.Zupancic@POV.Uni-Mb.si

Stanisław Wrycza
Department of Information Systems
University of Gdansk
81-824 Sopot
Armii Krajowej 119/121
Poland
swrycza@univ.gda.pl

Proceedings of 13th International Conference on Information Systems Development—Advances in Theory, Practice, and Education (ISD 2004), held in Vilnius, Lithuania, September 9–11, 2004

Library of Congress Control Number: 2005925766

ISBN-10: 0-387-25026-3
ISBN-13: 978-0387-25026-7

Printed in the United States of America (BS/DH)

9 8 7 6 5 4 3 2 1

springeronline.com

PREFACE

This volume contains papers presented during 13th International Conference on Information Systems Development – Advances in Theory, Practice and Education (ISD'2004), held in Vilnius, Lithuania, September 9–11, 2004. The intended audience for this book comprises researchers and practitioners interested in current trends in the Information Systems Development (ISD) field. Papers cover a wide range of topics: ISD methodologies, method engineering, business and IS modelling, web systems engineering, database related issues, information analysis and data mining, quality assessment, costing methods, security issues, impact of organizational environment, and motivation and job satisfaction among IS developers. The selection of papers was carried out by the International Program Committee. All papers were reviewed in advance by three reviewers and evaluated according to their relevance, originality and presentation quality. Papers were evaluated only on their own merits, independent of other submissions. Out of 117 submissions Program Committee selected 75 research papers to be presented at the Conference. 39 best papers and 5 papers presented by invited speakers are published in this volume.

The 13th International Conference on Information Systems Development continues the tradition started with the first Polish-Scandinavian Seminar on Current Trends in Information Systems Development Methodologies, held in Gdansk, Poland in 1988. Through the years this seminar has evolved into one of most prestigious conferences in the field. ISD Conference provides an international forum for the exchange of ideas between the research community and practitioners and offers a venue where ISD related educational issues are discussed.

ISD progresses rapidly, continually creating new challenges for the professionals involved. New concepts and approaches emerge in research as well as in practice. Emerging ideas need to be disseminated in order to stimulate the exploration of new solutions. Advanced ISD curricula and new teaching and learning approaches need to be developed. That is why ISD'2004 Conference theme was *Advances in Theory, Practice, and Education*. In addition to regular presentations and invited talks, the Conference provided two discussion panels: *Issues of Business Rules Approach in IS Development* and *Issues in Information Systems Education: Capitalizing on Recent Advances in Learning Theory.*

Many people contributed to the organization of the ISD'2004. We would like to express our thanks to all authors, members of the Program Committee, and external reviewers for their efforts. We also wish to acknowledge the support of the members of the Organizing Committee, the administration of Vilnius Gediminas Technical University, Institute of Mathematics and Informatics (Vilnius, Lithuania), Lithuanian Science and Studies State Foundation, Lithuanian Ministry of Education and Science, and all other institutions that actively supported the Conference.

Olegas Vasilecas
Albertas Caplinskas
Wita Wojtkowski
W. Gregory Wojtkowski
Joze Zupancic
Stanislaw Wrycza

ACKNOWLEDGEMENTS

The following acknowledges the organizations, tradenames, trademarks and products referenced throughout this book:

Tradename	***Organisation***
BEA®	BEA Systems, Inc.
Borland®	Borland International, Inc.
Casewise®	Casewise Ltd.
Cisco Systems®	Cisco Systems, Inc.
DCMI®	Dublin Core Metadata Initiative
DMTF	Distributed Management Task Force, Inc.
EDS®	Electronic Data Systems
EMC®	EMC Corporation
HP®	Hewlett-Packard Company
IBM®	IBM Corporation
IDC®	International Date Corporation
Information Builders	Information Builders, Inc.
ISACA®	Information Systems Audit and Control Association
ISACF®	Information Systems Audit and Control Fundation
ISO®	International Organization for Standardization
META®group	META group, Inc.
Microsoft®	Microsoft Corporation
Nokia®	Nokia Corporation
OCLC®	OCLC Online Computer Library Center, Inc.
OMG®	Object Management Group, Inc.
Oracle®	Oracle Corporation
PeopleSoft®	PeopleSoft, Inc.
PMI®	Project Management Institute, Inc.
RightNow®	RightNow Technologies, Inc.

Salesnet®	Salesnet, Inc.
SAP®	SAP AG.
SAS®	SAS Institute, Inc.
SCITOR®	Scitor Corporation
SGI™	Silicon Graphics, Inc.
Siebel®	Siebel Systems, Inc.
SPSS®	SPSS, Inc.
STG®	Structured Technology Group, Inc.
Sun®, Sun Microsystems®	Sun Microsystems, Inc.
VERITAS®	VERITAS Software Corporation
W3C®	World Wide Web Consortium (MIT, INRIA, Keio)

Abreviation	***Organisation***
BMBF	Bundesministerium für Bildung und Forschung
BRG	Business Rules Group
CICYT	Comisión Interministerial de Ciencia y Tecnología
DARPA	Department of Advanced Research Projects Agency
CEN	European Committee for Standardization
FIPA	The Foundation for Intelligent Physical Agents
GGF	The Global Grid Forum
ICAR-CNR	L'Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche
ISOC	The Internet Society
MIT	Massachusetts Institute of Technology
NASA	National Aeronautics and Space Administration
NCSA	National Center for Supercomputing Applications (University of Illinois)
NISO	National Information Standards Organization
NLB	Nova Ljubljanjska banka d.d.
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Office of Government Commerce
Salesforce.com	Salesforce.com Foundation
SIGWEB	ACM Special Interest Group on Hypertext, Hypermedia and the Web (former SIGLINK)
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business'

Trademark, Product Name, Project Name	***Organisation***
ABR	IBM
ADO	Microsoft
ADONIS®	BOC Information Systems GmbH

Agent Factory	ICAR-CNR, Agentcities.NET
Ant	The Apache Software Foundation
Apache™ HTTP Server	The Apache Software Foundation
APM	The EXTERNAL Consortium
APM BoK	Association for Project Management
Argo/UML	TIGRIS Open Source Comunity
Aris®	IDS-Scheer AG
ASP®	Microsoft
Atlas database™	ATLAS Database Group
Axiom-SYS™	STG
Balanced Scorecard®	Balanced Scorecard Collaborative
BML™	IBM
BPEL4WS	BEA, IBM, Microsoft, SAP, Siebel
BPML™	The Business Process Management Initiative
BP Win™	Unitek Information Technologies
BRBeans	IBM
BROCOM	Institut fur Wirtschaftsinformatik (University of Bern)
Casewise Corporate Modeler™	Casewise
Chimaera	Knowledge Systems Laboratory (Stanford University)
CIMOSA™	AMICE Consortium, CIMOSA association
Clementine®	SPSS, Inc.
Cloudscape®	Cloudscape, Inc.
COBIT	ISACA, ISACF, IT Governance Institute
COSA® Workflow	TRANSFLOW AG
CVS	JBoss, Inc.
CWM™	OMG
C++™	American Telephone and Telegraph, Inc.
DAML	DARPA, Joint US/EU ad hoc Agent Markup Language Committee
DB2 OLAP Server®	IBM
Dewey Decimal Classification®	OCLC
DOORS®	Telelogic AB
DSDM®	Dynamic Systems Development Method Ltd.
DTF	W3C
Dublin Core℠	OCLC
ebXML	United Nations, OASIS
Eclipse™	Eclipse Foundation
Enterprise JavaBeans®	Sun Microsystems
Excel®	Microsoft
EXTERNAL	The EXTERNAL Consortium
FLINTS™	West Midlands Police
FrameSolutions™	Computas AS

GATE™	NLP group (University of Sheffield)
GERAM™	IFIP/IFAC Task Force on Architectures for Enterprise Integration
GME	Institute for Software Integrated Systems
GRADE™	Grade Development Group, INFOLOGISTIK GmbH
GRAI-GIM	University of Bordeaux
HTML	W3C
HTTP™	W3C
IDC®	International Data Group, Inc.
IDEF	IDEF Information
IEM	Fraunhofer Institute for Production Systems and Design Technology
IMPRESS™	NLP group (University of Sheffield)
Intelligent Miner™	IBM
ISAC™	Indicus Software Pvt. Ltd.
ISDM™	Insource IT Consultancy Ltd.
ITIL®	The Office of Government Commerce
Java®	Sun Microsystems
JBoss®	JBoss, Inc.
JDBC®	Sun Microsystems
JMining	Vilnius Gediminas Technical University
JMS	Sun Microsystems
JSP™	Sun Microsystems
J2EE®	Sun Microsystems
KACTUS	KACTUS consortium
KIF	InterLingua Working Group (DARPA Knowledge Sharing Initiative)
LCSH	Library of Congress
LegoDB™	Lucent Bell Labs, Indian Institute of Science
Linux™	Linus Torvalds
LOCARD™	Advantage Systems Solutions Ltd.
Lotus Notes®	IBM, Lotus Development Corporation
MDA®	OMG
MediNET®	MediNet Systems, Inc.
METIS®	Computas AS
Microsoft Access®	Microsoft
Microsoft Internet Explorer®	Microsoft
Microsoft.NET™	Microsoft
Microsoft Solutions Framework®	Microsoft
Microsoft.NET	Microsoft
MineSet™	SGI
MOF™	OMG
MS Project®	Microsoft
MSQM	Microsoft

MS Word®	Microsoft
MySQL®	MySQL AB
NCSA™	University of Illinois Board of Trustees
Netscape Navigator®	Netscape Communications Corp.
NetSuite™	NetSuite Inc.
OCL	OMG
ODBC®	Microsoft
OMT++	Nokia
OntoEdit™	Ontoprise GmbH
On-To-Knowledge	On-To-Knowledge Consortium
Ontolingua	Knowledge Systems Laboratory (Stanford University)
Ontosaurus	Information Sciences Institute (University of Southern California)
OPEN	Open Consortium
Opera®	Opera Software AS
Oracle CASE*Method™	Oracle Corporation
Oracle Designer™	Oracle Corporation
OS/390™	IBM
PERA	Purdue Laboratory for Applied Industrial Control (Purdue University)
PMBOK®	PMI
Power Point®	Microsoft
Prometheus	Computer Science and IT School (RMIT University)
PROSIT™	WM Data
Protégé	Stanford Medical Informatics
ProVision®	Proforma Corporation
QStudio®	QA Systems BV.
Rational Rose®	IBM, Rational Software Corporation
RequisitePro®	IBM, Rational Software Corporation
Reuters-21578	Carnegie Group Inc., Reuters Ltd.
RFC	Network Working Group
RUP®	IBM, Rational Software Corporation
SADT®	Softec, Inc.
SAS IDP	SAS
SCITOR® Process	Scitor Corporation
SDLC™	IBM
SILVERRUN®-BM	CSA Research Pte Ltd.
SIMULA®	The Norwegian Computing Center
SimVision	The EXTERNAL Consortium
SOAP	W3C
SPSS for Windows®	SPSS, Inc.
SSADM®	OGC
SWEBOK	IEEE Computer Society,

	Association of Computing Machinery
Telelogic DOORS®	Teleologic AB
TGN	The J. Paul Getty Trust
TOVE	Enterprise Integration Laboratory (University of Toronto)
UDDI	OASIS
UEML	IFAC TC5.3 WG UEML
UML™	OMG
UNIX®	The Open Group
URL	The Internet Society
Visio®	Microsoft
WAP®	Wireless Application Protocol Forum Ltd.
WebML	Stefano Ceri, Piero Fraternali, Aldo Bongio
WebOnto™	The Open University
Weka™	The University of Welkato
Workflow BPR™	IBM, HOLOSOFX
WORKWARE	The EXTERNAL Consortium
WSCI	W3C
WSDL	W3C
XCHEAPS	The EXTERNAL Consortium
XMI®	OMG
XML®	Massachusetts Institute of Technology, World Wide Web Consortium
XSL™	World Wide Web Consortium
YSM™	Yourdon, Inc.

Invention	*Developer*
ANOVA	Ronald Fisher
AOM	Joseph W. Yoder, Fedenco Balaguer, Ralph Johnson
Cassiopeia	Alexis Drogoul, Jean-Daniel Zucker
Coordination Contract	Luís Filipe A. Andrade, José Luiz L. Fiadeiro
CoPaV^2	M. Hacid, C. Decleir, J. Kouloumdjian
CPI	D. Lee, W. W. Chu
EAR	Peter P-S Chen
EORM	Michael Lang
ETHICS	Enid Mumford
FOOPS	Joseph A. Goguen, David A. Wolfram
FTP	MIT
Gaia	Michael Wooldridge, Nicholas R. Jennings, David Kinny
HTML	Tim Berners-Lee
IE	Clive Finkelstein, James Martin
IPSD	Willibrordus M. P. van der Aalst, Kees M. van Hee
IRC	Jarkko Oikarinen

ISAC	M. Lundeberg
JCM	J. R. Hackman, G. R. Oldham
JUnit	Erich Gamma, Kent Beck
JSD	Michael A. Jackson
Language Action Perspective (Communicative Action Perspective)	F. Flores, J. J. Ludlow, T. Winograd
MASE	Mark F. Wood, Scott A. DeLoach
MAVIS	ChingMiin Duh, LiarnRurng Wen, John Sillince, Masoud Saeedi
MERISE	Hubert Tardieu
Multiview	David Avison, Trevor Wood-Harper
NIAM	G. M. Nijssen
OMT	James R. Rumbaugh, Michael R. Blaha, William Premerlani, Frederick Eddy, William Lorensen
OntoWeber	J. Yuhui, S. Decker, G. Wiederhold
OOHDM	Daniel Schwabe, Gustavo Rossi
QUICKethics	Enid Mumford
RMM	T. Isakowitz, E. A. Stohr, P. Balasubramanian
RPC	A. D. Birrell, B. J. Nelson
Rule Object	Ali Arsanjani
SSM	Peter Checkland
TCP/IP	Bob Kahn, Vinton Cerf
TOOR	Francisco A. C. Pinheiro, Joseph A. Goguen
Tropos	Jaelson Castro, Manuel Kolp, John Mylopoulos
VHDM	H. Lee, J. Kim, Y. G. Kim, S. H. Cho
Workflow Evolution	F. Casati, S. Ceri, B. Pernici, G. Pozzi
WSDM	O. M. F. De Troyer, C. J. Leune

ABBREVIATIONS

The following abbreviations are used throughout this book:

Abbreviation	*Full text*
ABC	Activity-Based Costing
ABM	Activity-Based Management
ABR	Accessible Business Rules
ADO	ActiveX Data Objects
AF	Agent Factory
AI	Artificial Intelligence
AIFS	Australian Integrated Forecast System
AMICE	European CIM Architecture
ANOVA	ANalysis Of VAriance
AO	Agent-Oriented
AOM	Adaptive Object Model

APL	Agent Programming Language
APM	Action Port Modeling
APM BoK	Body of knowledge of the Association for Project Management
ASP	1. Active Server Pages 2. Application Service Providing
ATM	Asynchronous Transfer Mode
BDI	Belief-Desire-Intention
BML	Bean Markup Language
BPEL4WS	Business Process Execution Language for Web Services
BPI	Business Process Improvement
BPML	Business Process Modeling Language
BPN	Backpropagation Neural Network
BPO	Business Process Outsourcing
BPR	Business Process Reengineering
BPWin	Business Processes for Windows
BRBeans	Business Rule Beans
BROCOM	Business Rule-Oriented Conceptual Modeling
BRP	Business Rules Propagation
BSC	Balanced Scorecard
B2B	Business-to-Business
CASE	Computer-Aided Software Engineering
CB	Codebook
CIM	1. Computer Integrated Manufacturing 2. Computational Independent Model
CIMOSA	CIM Open System Architecture
CIO	Chief Information Officers
CM	Conceptual Modeling
CMS	Content Management System
COBIT	Control Objectives for Information and related Technology
COTS	Commercial Off the Shelf Software
CPI	Constraints-Preserving Inlining
CRM	Customer Relationship Management
CRO	Chief Resource Officer
CRUD	Create, Read, Update, Delete
CSCW	Computer-Supported Cooperative Work
CSF	Critical Success Factors
CSV	Comma-Separated Value
CVS	Concurrent Versions System
CWM	Common Warehouse Metamodel
DAML	DARPA Agent Markup Language
DAML-S	DAML Services
DBMS	Data Base Management System

DC	Dublin Core
DCSV	Dublin Core Structured Value
DIM	Design Independent Model
DM	Data mining
DoI	Diffusion of Inovations
DSDM	Dynamic Systems Development Method
DSS	Decision Support System
DTD	Document Type Definition
DTF	Date Time Format
ebXML	Electronic Business XML
EAR	Entity-Attribute-Relationship
ECA	Event-Condition-Action
ECG	ElectroCardioGram
EEG	ElectroEncephaloGram
EER	Extended Entity–Relationship
EGS	Electronic Grocery Store
EM	Enterprise Modeling
EORM	Enhanced Object-Relationship Model
EPC	Engineering, Procurement & Construction
ER	Entity-relationship (model)
ERP	Enterprise Resource Planning
ETHICS	Effective Technical and Human Implementation of Computer-based Systems
EUIS	End-User Information System
EUC	End-User Computing
EXTERNAL	EXTended Enterprise Resources, Network Architectures and Learning
FD	Facet Data
FLINTS	Force Linked Intelligence System
FOOPS	Functional Object Oriented Programming System
FTP	File Transfer Protocol
GATE	General Architecture for Text Engineering
GERAM	Generalised Enterprise Reference Architecture and Methodology
GIS	Geographical Information Systems
GME	The Generic Modeling Environment
GRAI-GIM	GRAI Integrated Methodology
GRAI	Graphes et Résultats et Activités Interreliés
HASSIP	Harmonic Analysis and Statistics for Signal and Image Processing
HCP	Healthcare provider
HLA	High Level Architecture
HPRN	High Plains Research Network
HR	Human Resource
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
HW	Hardware
IC	Information Centre
ICT	Information and Communications Technology
IDEF	Integrated DEFinition Language
IDP	Information Delivery Portal
IE	Information Engineering
IEM	Integrated Enterprise Modelling
IMPRESS	IMPRession Evidence & Serial-crime profiling System
Info Mgt	Information Management
IPSD	Interactive, Process-Oriented system development
IRC	Internet Relay Chat
IRM	Information Resource Management
ISAC	Information Systems Work and Analysis of Changes
IS/ICT	Information Systems and Information and Communications Technology
ISAD	Information Systems Analysis and Design
ISD	Information Systems Development
ISDM	1.Information Systems Development Methodology 2. Insource Software Development Method
IT	Information Technology
JCM	Job Characteristics Model
JDBC	Java Data Base Connectivity
JIT	Just In Time
JMS	Java Message Queue Service
JSD	Jackson Systems Development
JSP	Java Server Pages
J2EE	Java 2 Platform, Enterprise Edition
KDD	Knowledge Discovery in Databases
KIF	Knowledge Interchange Format
LCSH	Library of Congress Subject Headings
LDAP	Light-weight Directory Access Protocol
LOC	Lines of Code
LOC/PM	Lines of Code per Person-Month
LPR	Longitudinal Process Research
LVO	Learning Virtual Organization
MAIS	Multi-Agent Information System
MARC	Machine-Readable Cataloging
MASE	Multiagent Systems Engineering
MAVIS	Multi-Agent Virtual Seminar System
MBRM	Manchester Business Rules Management
MDA	Model Driven Architecture
MDL	Minimum Description Length
MDU	Multimedia Description Unit

ME	Method Engineering
MEG	MyoElectroGraph
MERISE	Methode d'Etude et de Realisation Informatique pour les Systemes d'Enteprise
MGS	Mars Global Surveyor
MISQ	Management Information Systems Quarterly
MO	Modus Operandi
MOF	The Meta-Object Facility
MPEG	Moving Picture Experts Group
MSF	Microsoft Solutions Framework
MSG	Meteosat Second Generation
MSMQ	Microsoft Message Queuing
NDA	National Digital Archive (Hungary)
NIAM	Natural Language Information Analysis Method
NWP	Numerical Weather Predictions
OAG	Office of the Auditor General
OAI	Open Archives Initiative
OB	Organisational Behaviour
OCL	Object Constraint Language
ODBC	Open Data Base Connectivity
ODC	Orthogonal Defect Classification
OPEN	Object-Oriented Process, Environment and Notation
OLAP	Online Analytical Processing
OMT	Object Modeling Technique
OO	Object-Oriented
OOA&D	Object-Oriented Analysis and Design
OOHDM	Object-Oriented Hypermedia Design Model
OOISD	Object-Oriented Information System Development
OPEN	Object-Oriented Process, Environment and Notation
OTK	On-To-Knowledge
OSAD	Object Systems Analysis and Design
P2P	Peer-to-peer
PC	Personal Computer
PDA	Personal Digital Assistant
PDOM	Persistent Document Object Model
PERA	Purdue Enterprise-Reference Architecture
PIM	Platform Independent Model
PLA	Piece-Linear Formalism
PLS	Partial Least Square
PM	Process Management
PMBOK	Project Management Body of Knowledge
PO2	Professional Officer Level 2
PROSIT	PROcesS oriented IT audit support
PSM	Platform-Specific Model
QA	Quality Assesment

QUICKethics	Quality Information from Considered Knowledge ETHICS
RAD	1. Role Activity Diagram 2. Rapid Application Development
RAM	Random Access Memory
RE	Requirements Engineering
REAL	Resources, Events, Agents and Locations
RFC	1. Request For Comments 2. Regional Forecasting Centre
RMM	Relationship Management Methodology
R.O.	Regional Office
RPC	Remote Procedure Call
RUP	Rational Unified Process
R&D	Research and Development
SADT	Structured Analysis and Design Technique
SD	System Development
SDLC	Synchronous Data Link Communication
SI	Statuary Instrument
SIV	Standard Systems in Enterprises ("Standardsystem i Verksamheter" in Swedish)
SLA	Service Level Agreement
SME	1. Situational Method Engineering 2. Small and Medium size Enterprises
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SPOC	Senior Professional Officer Grade C
SPSPR	Strategy, business Processes, Service, information Processes and information Recourses
SPSS	Statistical Package for the Social Sciences
SQL	Structured Query Language
SSADM	Structured Systems Analysis and Design Methodology
SSM	Soft System Methodology
SW	Software
SWEBOK	Software Engineering Body of Knowledge
SWOT	Strengths, Weaknesses, Opportunities and Threats
S^3	Strategy-Service-Support
TAM	Technology Acceptance Model
TbKM	Task-based Knowledge Management
TCO	Total Cost of Ownership
TCP/IP	Transmission Control Protocol / Internet Protocol
TGN	Thesaurus of Geographic Names
TIB	Time-Interval-Based model
TLOCS	Tanker Loading Operative Control System

TOOR	Traceability of Object-Oriented Requirements
TQM	Total Quality Management
TSV	Tab Separated Value
UDA	User-Developed Application
UDDI	Universal Description Discovery and Integration
UDT	User-Defined Type
UEC	User-End Computing
UEML	Unified Enterprise Modeling Language
UI	User Interface
UML	Unified Modeling Language
UML/UP	Unified Modeling Language/Unified Process
UoD	Universe of Discourse
UP	Unified Process
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
VaR	Value at Risk
VHDM	View-Based Hypermedia Design Methodology
VO	Virtual Organization
VQ	Vector Quantization
WADM	Workflow Application Development Method
WAP	Wireless Application Protocol
WebML	Web Modeling Language
WfMS	Workflow Management System
WIS	Web-based Information System
WM	Workflow Modeling
WSCI	Web Service Choreography Interface
WSDL	Web Services Description Language
WSDM	Web Site Design Method
WWW	World Wide Web
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XP	eXtreme Programming
XQL	Extensible Query Language
XSL	Extensible Stylesheet Language
YSM	Yourdon Structured Method

Organisation

The Thirteenth International Conference on Information Systems Development (ISD'2004) held on September 9–11, 2004 in Vilnius, Lithuania, was organised by Vilnius Gediminas Technical University and Institute of Mathematics and Informatics.

Conference Chair

Olegas Vasilecas	Vilnius Gediminas Technical University	Lithuania

Local Organising Committee members

Algirdas Ciucelis	Vilnius Gediminas Technical University	Lithuania
Olegas Vasilecas	Vilnius Gediminas Technical University	Lithuania
Albertas Caplinskas	Institute of Mathematic and Informatics	Lithuania
Sergejus Sosunovas	Vilnius Gediminas Technical University	Lithuania
Saulius Maskeliunas	Institute of Mathematic and Informatics	Lithuania
Dale Dzemydiene	Law University of Lithuania, Institute of Mathematics and Informatics	Lithuania
Audrone Lupeikiene	Institute of Mathematic and Informatics	Lithuania
Diana Bugaite	Vilnius Gediminas Technical University	Lithuania
Arunas Ribikauskas	Vilnius Gediminas Technical University	Lithuania

Program Committee

Gary Allen	University of Huddersfield	United Kingdom
David Avison	ESSEC Business School	France
Janis Barzdins	University of Latvia	Latvia
Juris Borzovs	Information Technology Institute	Latvia
Frada Burstein	Monash University	Australia
Rimantas Butleris	Kaunas University of Technology	Lithuania
Sharma Chakravarthy	The University of Texas at Arlington	USA
Heitor Augustus Xavier Costa	Universidade Federal de Lavras	Brazil
Darren Dalcher	Middlesex University	United Kingdom
Vitalijus Denisovas	Klaipeda University	Lithuania

Klaus R. Dittrich	Institut für Informatik – Universität Zürich	Switzerland
Julie Fisher	Monash University	Australia
Chris Freyberg	Massey University	New Zealand
John Gammack	Griffith University	Australia
Janis Grundspenkis	Riga Technical University	Latvia
Fitzgerald Guy	Brunel University	United Kingdom
Remigijus Gustas	Karlstad University	Sweden
Janis Eiduks	Riga Technical University	Latvia
Hele-Mai Haav	Institute of Cybernetics at Tallinn Technical University	Estonia
G. Harindranath	University of London	United Kingdom
Igor Hawryszkiewycz	Sydney University of Technology	Australia
Alfred Helmerich	Research Institute for Applied Technology	Germany
Juhani Iivari	University of Oulu	Finland
Mirjana Ivanovic	University of Novi Sad	Serbia and Montenegro
Marius Janson	University of Missouri – St. Louis	USA
Leonid Kalinichenko	Institute for Problems of Informatics of the Russian Academy of Science	Russia
Roland Kaschek	Massey University	New Zealand
Marite Kirikova	Riga Technical University	Latvia
Gabor Knapp	Budapest University of Technology and Education	Hungary
John Krogstie	Norwegian University of Science and Technology	Norway
Marian Kuras	Cracow Academy of Economics	Poland
Rein Kuusik	Tallinn Technical University	Estonia
Sergei Kuznetsov	Institute for System Programming of Russian Academy of Science	Russia
Robert Leskovar	University of Maribor	Slovenia
Henry Linger	Monash University	Australia
Björn Lundell	University of Skoevde	Sweden
Audrone Lupeikiene	Institute of Mathematic and Informatics	Lithuania
Kalle Lyytinen	Case Western Reserve University	USA
Leszek A. Maciaszek	Macquarie University	Australia
Yannis Manolopoulos	Aristotle University	Greece
Sal March	Vanderbilt University	USA
Majed Al-Mashari	King Saud University	Saudi Arabia
Heinrich C. Mayr	University of Klagenfurt	Austria
Elisabeth Métais	CNAM University	France
Mike Metcalfe	University of South Australia	Australia
Giorgio De Michelis	University of Milano	Italia
Daniel Moody	Czech Technical University	Czech Republic
Robert Moreton	University of Wolverhampton	United Kingdom
Pavol Navrat	Slovak University of Technology	Slovakia

Lina Nemuraite	Kaunas University of Technology	Lithuania
Anders G. Nilsson	Karlstad University	Sweden
Jorgen F. Nilsson	Technical University of Denmark	Denmark
Jacob Norberg	Copenhagen Business School	Denmark
Jari Palomaki	Massey University	New Zealand
Malgorzata Pankowska	The Karol Adamiecki University of Economics In Katowice	Poland
George A. Papadopoulus	University of Cyprus	Cyprus
Olga L. Perevozchikova	Glushkov Institite of Cybernetics	Ukraine
Alain Pirotte	University of Louvain	Belgium
Jaroslav Pokorny	Charles University in Prague	Czech Republic
Boris Rachev	Technical University of Varna	Bulgaria
Vaclav Repa	Prague University of Economics	Czech Republic
J. W. Schmidt	Technische Universität Hamburg-Harburg	Germany
David Schwartz	Bar-Ilan University	Israel
Timothy K. Shih	Tamkang University	Taiwan
Klaas Sikkel	Universiteit Twente	Netherlands
Guttorm Sindre	Norwegian University of Science and Technology	Norway
Ignas Skucas	VytautusMagnus University	Lithuania
Arne Soelvberg	Norwegian University of Science and Technology	Norway
Carsten Sorensen	London School of Economics	United Kingdom
Larry Stapleton	Waterford Institute of Technology	Republic of Ireland
Eberhard Stickel	Bonn University of Applied Sciences	Germany
Uldis Sukovskis	Riga Technical University	Latvia
Janis Tenteris	Riga Technical University	Latvia
Bernhard Thalheim	Kiel University	Germany
Jacek Unold	Wroclaw University Of Economics	Poland
Douglas Vogel	City University of Hong Kong	Hong Kong
Jiri Vorisek	Prague University of Economics	Czech Republic
Gottfried Vossen	University of Münster	Germany
Gert-Jan de Vreede	University of Nebraska at Omaha	USA
Benkt Wangler	University of Skoevde	Sweden
Roel Wieringa	University of Twente	Netherlands
Carson C. Woo	University of British Columbia	Canada

Additional Reviewers

Manu Aery	The University of Texas at Arlington	USA
Akshay Arora	The University of Texas at Arlington	USA
Per Backlund	University of Skoevde	Sweden
Christos Berberidis	Aristotle University	Greece
Dhawal Bhatia	The University of Texas at Arlington	USA

Antanas Cenys	Semiconductor Physics Institute	Lithuania
Pascal van Eck	University of Twente	Netherlands
Ajay Eppili	The University of Texas at Arlington	USA
Rik Eshuis	University of Twente	Netherlands
Matt Germonprez	Case Western Reserve University	USA
Jaap Gordijn	Vrije University	Netherlands
Sari Hakkarainen	Norwegian University of Science and Technology	Norway
Algimantas Juozapavicius	Vilnius University	Lithuania
Yun Lin	Norwegian University of Science and Technology	Norway
Antanas Mitasiunas	Vilnius University	Lithuania
Didier Nakache	CNAM University	France
Alexandros Nanopoulos	Aristotle University	Greece
Vamshi Pajjuri	The University of Texas at Arlington	USA
Rimantas Plestys	Kaunas University of Technology	Lithuania
A. Raman	The University of Texas at Arlington	USA
Adam Rechbinder	University of Skoevde	Sweden
Sunit Shreshta	The University of Texas at Arlington	USA
Michael Skusa	Technische Universität Hamburg-Harburg	Germany
Darijus Strasunskas	Norwegian University of Science and Technology	Norway
David Tilson	Case Western Reserve University	USA
Anders Toms	University of Skoevde	Sweden
Grigoris Tsoumakas	Aristotle University	Greece
Tarmo Veskioja	Tallinn Technical University	Estonia

Co-Editors

Olegas Vasilecas	Vilnius Gediminas Technical University	Lithuania
Albertas Caplinskas	Institute of Mathematics and Informatics	Lithuania
Wita Wojtkowski	Boise State University	USA
W. Gregory Wojtkowski	Boise State University	USA
Joze Zupancic	University of Maribor	Slovenia
Stanislaw Wrycza	University of Gdansk	Poland

CONTENTS

SYSTEM CO-DEVELOPMENT THROUGH REQUIREMENTS SPECIFICATION

Pericles Loucopoulos[*]

1. INTRODUCTION

The relatively recent emphasis on process-centred approaches has highlighted the need for appropriate mechanisms for the elicitation, representation and validation of requirements that focus on the *co-development* activity whose aim is to ensure alignment between business processes and support technical systems.

A key challenge in the development of systems is the engagement of domain experts in their articulation, agreement, and validation of requirements. This challenge is particularly pronounced at the so-called *early requirements* phase when multiple stakeholders from different divisions and often different organisations need to reach agreement about the intended systems. Decisions taken at this stage have a profound effect on the technical and economic feasibility of the project. It is no longer appropriate for information systems professionals to focus only on functional and non-functional aspects of the intended system and somehow assume that organisational context and needs are outside their remit.

Traditional requirements engineering (RE) techniques that have emerged over the years in the field of information systems engineering are also being applied to business process modelling. In both cases modelling is advocated as a way of facilitating analyst-client communication. In both cases the focus of the modelling paradigm is on requirements for support systems rather than on co-development issues. In neither case does a modelling paradigm is specifically deployed to facilitate communication between stakeholders and since early requirements elicitation is founded on the need for stakeholders to co-operatively reach agreement on the interactive and inter-dependent nature of all requirements in their totality, neither class fully tackles the problems of early requirements.

This paper examines contemporary approaches to early requirements modelling, discusses the main issues that should be addressed by modelling and proposes an approach that arguably provides a solid and usable framework for addressing the key problem of

[*] Department of Computation, University of Manchester Institute of Science and Technology, P.O. Box 88, Manchester, M60 1QD, U.K., pl@co.umist.ac.uk, http://www.co.umist.ac.uk/~pl.

early requirements namely, engaging stakeholders in defining, exploring and validating early requirements.

Section 2 introduces the topic of early requirements and discusses the needs for methods and tools for this critical development activity. Section 3 introduces a framework, known as the S^3 framework, that addresses early requirements from three viewpoints. Section 4, describes a large industrial-strength application that made use of the S^3 approach, and discusses and the lesson learned from it.

2. EARLY REQUIREMENTS

There is a high degree of consensus amongst information systems researchers and practitioners that the development of systems is not solely a technical activity but rather organisational factors very often have a profound effect on both the delivered system and the design process. This is particularly acute in today's turbulent business environment where powerful forces such as deregulation, globalisation, mergers, advances in information and telecommunications technologies, and increasing education of people provide opportunities for organising work in ways that have never before been possible.[1]

Traditional requirements engineering (RE) techniques that have emerged over the years in the field of information systems engineering are also being applied to business process modelling. Some pay attention to procedural matters[2–5] whilst others adopt an intentional viewpoint.[6–12] The emphasis of the former is on support systems whereas that of the latter is on organisational issues.

In both cases modelling is advocated as a way of facilitating analyst-client communication. In both cases the focus of the modelling paradigm is on requirements for support systems rather than on co-development issues. In neither case does a modelling paradigm is specifically deployed to facilitate communication between stakeholders and since early requirements elicitation is founded on the need for stakeholders to co-operatively reach agreement on the interactive and inter-dependent nature of all requirements,[13] in their totality, neither class fully tackles the problems of early requirements.

What distinguishes *early* requirements from *late* (or support system) requirements, is the degree of involvement of client stakeholders. Early requirements are almost exclusively driven by client stakeholders' communication. Issues of early requirements include: (a) the customer profiles of a business process, (b) the likely demand for product or service made by each type of customer, (c) the level of desirable service that the business process should strive to achieve, (d) the resources that are required in order to achieve these levels of service and (e) the trade-off between levels of service and requisite resource between all client stakeholders of a business process. Only when these issues have been resolved can one then begin to develop specifications of requirements for support systems. The analyst will need to know how the support system interacts with other systems, what kind of levels of service it must achieve and so on before engaging into further analysis on functional and non-functional properties of the intended system.

These issues of early requirements are specifically tackled by the *strategy-service-support* (referred to as S^3) modelling approach.[14] It is based on the premise that informal (c.f.,[15, 16] semi-formal (c.f.[12, 17] or formal approaches (c.f.[18, 19] to business process modelling do not fully address the issues relating to early requirements. The S^3 modelling approach advocates a process cycle of *hypothesis formulation*, *testing*, and *re-formulation* until stakeholders have enough confidence about the efficiency of the proposed design. Essentially, one is developing theories, externalised as conceptual models, about the Universe of Discourse and tests these theories for their validity. In terms of the 6 classes of requirements elicitation techniques identified in[20] (i.e. traditional, group, prototyping, model-driven, cognitive and contextual), the contribution of S^3 is in the *model-driven* and *group* classes. In S^3 models are used to establish the structural aspects of a business process. These models are subsequently subjected to scenario generation in consensus-building stakeholder workshops.

3. THE S^3 APPROACH TO SYSTEM CO-DEVELOPMENT

In terms of methodology the S^3 approach supports a reasoning cycle of *hypothesis formulation*, *testing*, and *re-formulation*. Within this reasoning cycle, S^3 deals with strategic, service and support issues.

a. **S**trategic issues are organisational and stakeholder goals for improving the organisation's performance.
b. **S**ervice issues are the levels of improvement considered by the organisation.
c. **S**upport issues are the resources policies and actions required to reach the desired service levels.

The approach is summarised in terms of its *philosophy*, *models* delivered and *processes* adopted, in Figure 1.

The S^3 approach is motivated by four 'principles: (a) Systems thinking considers independent components that form a unified whole[21, 22]; for example the business process

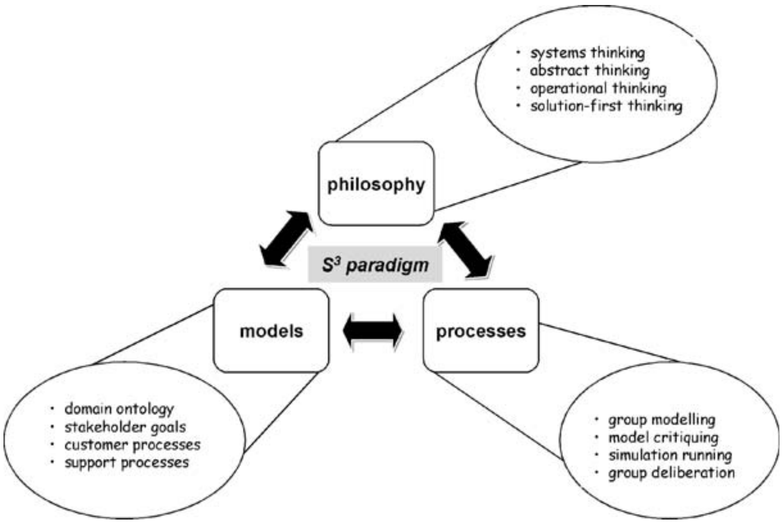


Figure 1. The S^3 methodology.

`spectator services` involves `arrival handling`, `security checking`, `crowd management`, `services` etc and the behaviour of each component affects all others, potentially in a profound manner. (b) Abstract thinking implies that one moves away from the physical manifestation of processes.[23] (c) Operational thinking considers the dynamics of a business process and in particular its behaviour over time.[24, 25] (d) Solution-first thinking implies that in attempting to identify requirements one generates a provisional design whose purpose is to highlight potential functionality.[26]

On the basis of these four fundamental principles, three types of model may be developed: (a) domain ontologies[27] that scope the problem space and define the key business objects that participate in a business process; (b) stakeholder goals[28] that determine the high-level intentions of each class of stakeholder; (c) processes[29] that fall into the two sub-categories of customer-oriented processes and support-oriented processes where the former provides the basis for defining the level of service and the latter the manner in which this will be met.

In terms of processes involved there are essentially two activities: (a) *model building and critiquing*[30, 31] and (b) *simulation and group deliberation*.[32, 33] Models are mainly built by analysts with input from domain experts but are critiqued and revised by stakeholders. Analysts also facilitate simulation sessions where model parameters are instantiated by stakeholders. Consensus building stakeholder workshops develop scenarios that facilitate deliberation of alternative future realizations.

4. AN EXAMPLE OF USING THE S^3 APPROACH

4.1. The Problem Domain

All the issues with early requirements, outlined in Section 2, were present in a project regarding the design of systems supporting venue operations for the Athens 2004 Olympic Games. The action research on which this paper is based was carried out at the Organising Committee for the Athens 2004 Olympic Games (ATHOC). The project underwent 3 phases during a period of 2 years and involved 175 person months of effort. Initially, elicitation of early requirements was attempted using informal stakeholder facilitation. This was followed by the use of a traditional business process modelling approach which in turn gave way to the S^3 approach that eventually became the main vehicle for eliciting early requirements for 21 applications. In preparing towards the staging of the Games, ATHOC planed, coordinated and designed for systems most of which were delivered by external contractors. The problem domain is one of *process integration* the characteristics of which may be summarised as follows:

- There are many different *co-operating agents* (systems, ATHOC functional areas, sub-contractors, personnel and procedures). For example, the distribution of results involves the co-ordination of systems concerned with timing, information structuring, information communication, reprographics, and physical distribution.
- Different stakeholders have *distinct goals* and expectations of systems. For example, transportation is solely concerned with safe and timely arrival and departure of spectators whereas catering is concerned with meeting demand during the operation of a venue.

- Although different stakeholders have their own distinct concerns, from a total venue perspective all these need to be considered *holistically*. For example, the demand on catering is influenced by the behaviour of spectators in terms of their arrival and departure, their attendance of sports events etc.

The effect of these characteristics is the transformation that every Games Organising Committee needs to undertake. An Organising Committee is established a few years prior to the staging of the Games. Functional areas such as transportation, security, logistics, marketing etc. are established approximately 5–6 years prior to the staging of the Games, in order to organise and develop the necessary human, information and physical resources for the Games. Whilst at the outset the structure of an Organising Committee is hierarchical and *function-oriented*, this needs to gradually get transformed, as the Games approach, to a venue-based *process orientation* in order to shift emphasis away from internal organisational efficiency towards *venue operation* efficiency. The term 'venue operations' concerns the systems, actors and procedures that will be implemented in a venue (be it a competition venue or a support venue) within spatial and temporal constraints, for different types of service e.g. security control, crowd management, catering etc. The design of systems supporting venue operations needs to address both their functional requirements (the resources and procedures for their management) and their non-functional requirements (the quality of service provision).

A central issue is the interaction and coordination of domain experts from 27 different functional areas in order to design systems that interact properly and are fully co-ordinated. For example, transportation needs to be co-ordinated with crowd queuing control which in turn need to co-ordinate with security etc. Typically, the stakeholders involved are experts from the Organising Committee, technical suppliers and subcontractors.

The complexity of venue operations increases as the variability of demands increases according to different 'customer' demands. For example, coordinating the processes concerned with spectator services involves many stakeholders from the 27 functional areas of ATHOC and for large venues there are in excess of 100 factors that influence the behaviour of the system.

Co-development of business processes and support systems for this problem led to a number of complex decision making activities during early requirements. The following are few examples of frequently faced, typical questions:

- What are the appropriate types and level of resources that will be needed by each functional area for each venue?
- What are the interrelationships and interdependencies of the various functional areas in providing the required services to the various customer groups?
- What is the trade-off between the desired service level and the resource allocation / requirements for each activity involved in a process?
- What is the optimal operational smoothing / resource levelling for the overall process pertaining to a given customer group?

Ultimately, ATHOC had to specify the physical infrastructure (e.g. building works, public transportation etc), support systems (e.g. security systems, reporting systems, catering systems, ATMs etc) and procedures (e.g. protocols for dissemination of results, crowd control etc) in such a way so as to satisfy both the *individual* requirements of each functional area and the *systemic* requirements that arise from the process-oriented view of venue operations. It was therefore, profoundly important to reach agreement between stakeholders from all 27 functional areas on the way that their *interdependent requirements* were to be dealt in the most transparent, quantifiable and effective way possible.

4.2. Model Building and Critiquing

In conceptual modelling two viewpoints are accommodated: (a) stakeholder goals that determine the high-level intentions of each class of stakeholder and (b) business processes that determine the service that should be provided and the support mechanisms for providing the service.

4.2.1. Stakeholders' Goals

Goal modelling is about describing the causal structure of a system (be it a business system, or a software system, etc.), in terms of the *goals-means* relations from the "intentional" objectives that control and govern the system functions to the actual "physical" processes and activities available for achieving these objectives. Goal modelling aims at providing the means for describing the *purpose* of the system under consideration, why it came into being.

In eliciting the goals for the venue operations system, the aim was to understand what determined the successful operation of the system. This involved helping the various stakeholders externalise the (sometimes implicit) goals that they had, capturing these goals, and synthesising that knowledge with information from other sources, such as existing documentation, abstract descriptions of various systems and procedures, and so forth. Stakeholders' goals were thus an initial, high-level expression of system requirements viewed from the perspective of ATHOC, i.e. the *service provider* (as opposed to that of the *user*).

With respect to goal categorisation, we found that it was often relatively straightforward to capture goals about the functions that the system should provide (i.e. the functional requirements), while in most cases it was difficult to accurately define goals regarding the quality of venue operations. In both cases, the multitude of stakeholders (i.e. functional areas) involved in the requirements specification often resulted to competing, and sometimes clearly conflicting, goals about the system. Furthermore, it was especially difficult for stakeholders to express their goals in specific (i.e. measurable) terms. Indeed, while each functional area found it relatively easy to identify distinct functional/quality aspects of the system, it was much more difficult to quantify each of these aspects. This difficulty unfailingly complicates subsequent stages of system design because it has a decisive influence on the type and amount of resources required, and by extension on the final cost of the system.

To deal with the complex situation of goal elicitation we progressed in a stepwise, cyclical manner, starting from high-level, sometimes fuzzy, goals. We then elaborated on these goals with the help of the functional areas affected by them. By modelling the processes that the venue operations system comprises, and by testing different scenarios on how quality goals can be implemented in each of these processes, we could identify different ways of refining goals into specific quality requirements on the basis of which the system could be developed.

An example of a high-level goal that all functional areas invariably expressed was related to the quality of service provision to various customer groups. Irrespective of customer group and service type, the goal was usually expressed in this form: 'Minimise the time that a customer has to wait in order to get serviced'. This translates into goals such as 'minimise the time that a spectator has to wait in order to go through security checking' or 'minimise the time that a staff member has to wait in order to check in upon arrival to the venue'.

This type of goal does not translate very well into operational terms because it does not specify a concrete target for the waiting time. To complicate matters further, there is not a single acceptable waiting time as that depends on the service type and the customer group for which it is intended. What is acceptable for spectators or staff, for instance, may not be acceptable for members of the Olympic family or for athletes. In other words, what is the *level of service* that each functional area is aiming to offer to the customers it is going to service? Should we be happy with 30 seconds waiting time or with 15 minutes? In some cases even that answer was not ready, so it had to be negotiated.

A different type of high-level goal was expressed with respect to the overall presence of spectators in a venue. Given that a venue may hold more than one event (e.g. competition session) during a day, at any time there may be spectators arriving at the venue area for one of the upcoming sessions, spectators leaving the venue from one of the past sessions, and spectators participating in a current session. The total number of spectators present has to be somehow controlled for practical reasons such as the availability of resources (e.g. space), but also due to safety concerns. This translates into the goal 'manage the total presence of spectators in the venue area'. Again this is an abstract goal that needs to be made more specific; to refine it, the stakeholders examined the factors influencing the presence of spectators in the venue and their distribution in the various areas of which it consists. These factors include the competition schedule at each venue, the transportation capabilities to/from the venue, the availability of open spaces and/or service areas within the venue, and so forth. Addressing issues such as those concerning these two high-level goals was the first step towards visualising an operational system.

4.2.2. Business Processes

There was a wide range of process-related problems to be studied while addressing the issue of venue operations. At one end of the spectrum, there were problems with 'local' impact, i.e. affecting a single customer group, a small area of the venue, and a small part of venue resources (workforce, machinery, consumables). At the other end of the spectrum, there was the problem of the 'behaviour' of an entire venue as a complex, interconnected system. This corresponds to process models focusing on the dynamic profiling of all venue components, over an extended time frame (e.g. an entire day of the Games), possibly with respect to the needs of more than one customer group. A distinguishing feature of this type of situation is the large number of different service types that the model must represent, since the behaviour of the venue operations system is affected by each of these service sub-components. As a result, the degree of complexity in the resulting process model rises dramatically.

To demonstrate the use of business process modelling, consider the application of spectators' processes.

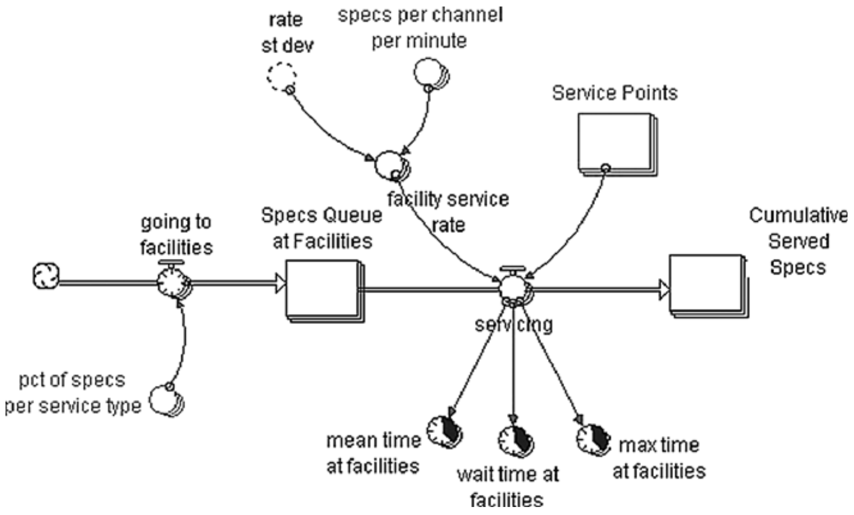


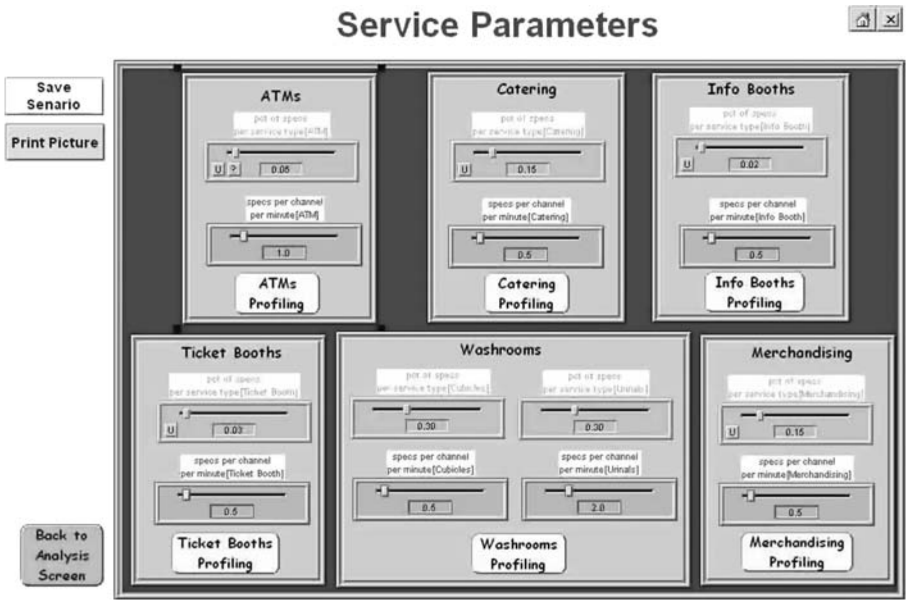
Figure 2. Model fragment regarding venue service facilities.

The behaviour of specific components of the system is examined by model components like the one presented in Figure 2.

The behaviour of the services component is determined by two issues: the *demand* for each type of service and the *supply* offered by the service provider. The demand is determined in part through the `pct of specs per service type` variable, which expresses the number of customers expected at each type of service facility per unit of time as a percentage of total spectator presence. Total spectator presence depends on overall spectators' behaviour in the venue area, which interacts with this model fragment through a number of feedback loops (not shown here due to the complexity of the complete model).

The supply is determined by two parameters: the number of `Service Points` available (e.g. 10 stands selling food), and the `specs per channel per minute` service rate (e.g. two spectators serviced per service point per minute). According to this representation, spectators arrive at the service facility (`going to facilities`), queue there for a while if no service point is available (`Specs Queue at Facilities`), and eventually get serviced (`servicing`).

Using this model fragment we can elaborate on the way that stakeholder goals were refined through the use of process modelling. We previously mentioned the high-level goal 'Minimise the time that a customer has to wait in order to get serviced'. The realisation of this goal for a given type of service facility, and for a given demand, depends on the availability of supply for that facility. Supply is composed of two independent factors, the number of service points and the service rate. Therefore, the initial goal was decomposed into two complementary (i.e. non-competing) goals: 'Maximise the number of service points' and 'maximise the service rate'. These goals are more accurate than the initial one, however they need to be analysed further in order to become quantifiable.



ATMs
pct of specs per service type[ATM] 0.05
specs per channel per minute[ATM] 1.0

Catering
pct of specs per service type[Catering] 0.15
specs per channel per minute[Catering] 0.5

Info Booths
pct of specs per service type[Info Booth] 0.02
specs per channel per minute[Info Booth] 0.5

Ticket Booths
pct of specs per service type[Ticket Booth] 0.03
specs per channel per minute[Ticket Booth] 0.5

Washrooms
pct of specs per service type[Cubicles] 0.30
pct of specs per service type[Urinals] 0.30
specs per channel per minute[Cubicles] 0.5
specs per channel per minute[Urinals] 2.0

Merchandising
pct of specs per service type[Merchandising] 0.15
specs per channel per minute[Merchandising] 0.5

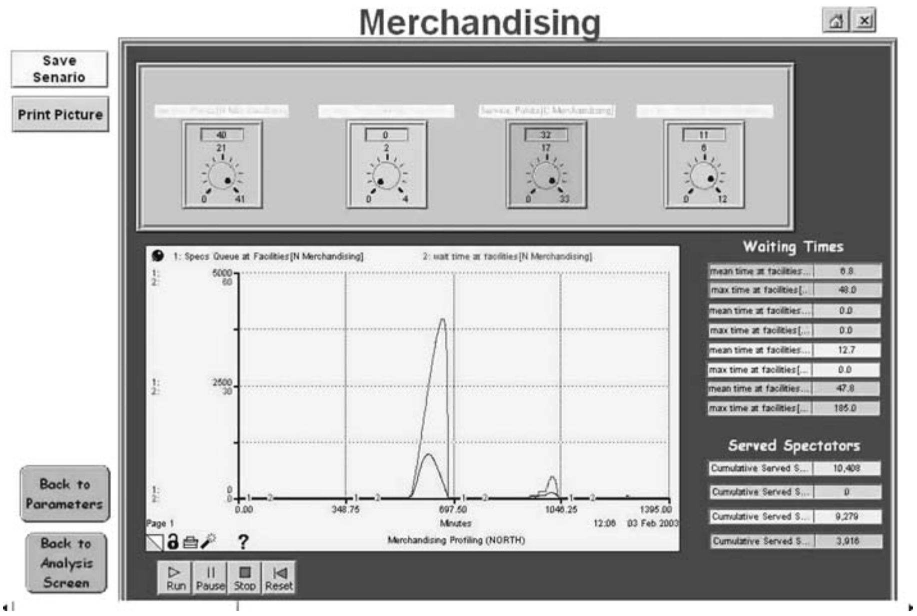
Figure 3. Stakeholder-defined parameters for service.

4.2.3. Scenarios Building

The generation of different scenarios concerning each problem studied, and the simulation of these scenarios with the help of the process models developed, is an essential part of requirements definition. In our experience, scenarios are an indispensable tool for truly understanding the implications of stakeholders in their deliberation of requirements. In our application, as the models were being developed and the stakeholders were becoming more aware of the different factors influencing each problem, the range of possible values for each of these factors became more evident, thus creating the initial ideas for different scenarios.

For example, in the components of the system model that deals with services (ATMs, merchandising, catering, etc), fragment of which is shown in Figure 2, we find a plethora of *stakeholder defined assumptions* regarding demand and supply for each service facility. One set of parameters is to do with the expected percentage of spectators demanding the service and the expected productivity per unit of service. An example of the system developed to support scenarios is shown in Figure 3.

For merchandising, for example, stakeholders have defined that `15% of the spectators` will seek this service and that on average it takes `2 minutes` to service each spectator. These values, just like all other stakeholder-defined values are obviously assumptions on their part. But given that there is some past experience from previous Olympics, stakeholders can begin their scenarios from a reasonably defendable position.


Merchandising
Save Senario
Print Picture
Back to Parameters
Back to Analysis Screen
Service Points[C Merchandising]
1: Specs Queue at Facilities[N Merchandising]
2: wait time at facilities[N Merchandising]
Page 1
Minutes
12:06 03 Feb 2003
Merchandising Profiling (NORTH)
Run
Pause
Stop
Reset

Waiting Times

mean time at facilities...	6.8
max time at facilities[...	48.0
mean time at facilities...	0.0
max time at facilities[...	0.0
mean time at facilities...	12.7
max time at facilities[...	0.0
mean time at facilities...	47.8
max time at facilities[...	185.0

Served Spectators

Cumulative Served S...	10,408
Cumulative Served S...	0
Cumulative Served S...	9,279
Cumulative Served S...	3,916

Figure 4. Simulation results for the 'Merchandising' service.

An additional set of parameters that influence the behaviour of the system focus on the resources that support the service provision e.g. "how many merchandising outlets?", or "how many ATMs?" etc. An example of such parameters for merchandising in the `north area` is shown in Figure 4. This also shows the simulation that results from the instantiation of parameters for the demand and supply set by stakeholders.

A key point about the simulation process was the realisation by stakeholders that their specific requirements and their specific decisions influenced those of others. Merchandising for example, does not exist in isolation. It is influenced by other components and in turn it influences others. Other relevant factors, such as spectators' arrival and departure patterns, were taken into account. The stakeholders involved in scenario generation investigated the range of probable values for each of these parameters, as well as some 'extreme' values that were less probable but worth investigating nonetheless. Each scenario was characterised by the values of *all* independent variables; the number of possible scenarios thus depended on the number of their feasible combinations.

The models were subjected to testing through simulation sessions, in workshops involving stakeholders in groups ranging from 5 to as many as 40. In all workshops the models were presented to project stakeholders together with the corresponding scenarios and simulated runs. These features enabled stakeholders to reach a consensus about the underlying processes and the implications that each choice would have on overall system behaviour. The first type of result, i.e. results concerning specific components of the system, helped to answer operational questions concerning the rational allocation of resources and the resulting service provision capabilities of the system. The second type of result proved useful for understanding the overall behaviour of a venue, thus answering higher-level, management questions concerning customer presence and distribution, arrival and departure patterns etc.

5. CONCLUSIONS

Requirements engineering is considered by many as the most critical of all development activities for socio-technical systems. The sensitive area of early requirements is only recently beginning to be addressed in a methodological sense. Considerable effort is required to bridge the semantic islands that are often formed between different communities of client stakeholders, designers, regulators, etc. Indeed the entire system development process seems to be disadvantaged by lack of techniques to assist with effective communication.[13, 34] An in-depth study on industrial practice[35] provides evidence that communication is crucial to the entire design process.

In early requirements, when there is a great deal of vagueness and uncertainty about system goals that are often set against a background of social, organizational and political turbulence, the need for a systematic and systemic way of dealing with all co-development aspects seems to be of paramount importance.

The work presented in this paper is an attempt to highlight this need from experiences from a substantial project that involved many stakeholders. These experiences confirm that informal and textual descriptions need to give way to conceptual modelling languages with clear semantics and intuitive syntax so that an application can be defined at an appropriate level of abstraction. This would greatly enhance visualisation of processes that will in turn contribute to a more informed discussion and agreement between stakeholders.

Whilst qualitative-based conceptual modelling approaches seem to be an improvement on purely linguistic-based approaches, they fail to bridge the communication gap between client stakeholders and analysts. The issue of analyst-client relationship has been highlighted by many authors.[36, 37] This type of modelling paradigm that has evolved from work on Databases, Software Engineering or Object-oriented Design, with its analyst orientation, does little to enhance communication.

Empirical evidence from the Athens 2004 Olympics Games project on early requirements concurs with the premise on which the S^3 approach is based namely, that qualitative models need to be enhanced with quantitative capabilities. These capabilities provide opportunities for the generation and evaluation of alternative scenarios with respect to stakeholder choices on their requirements. In the ATHOC application, the S^3 approach proved to encourage group brainstorming through which participants could focus on alternative solutions and to envision potential behaviour of the system prior to its implementation. This way of working supports the way experts work on ill-structured problem settings such as planning and design.[26]

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to his many colleagues, analysts and stakeholders, at the Athens 2004 Olympic Games Organising Committee. In particular to Nikos

Prekas for managing the team of analysts and liaising with stakeholders in the most effective manner and to Dimitris Beis and Gregory Vgontzas for having the insight to adopt new innovative approaches towards the design of venue operations. The author would also like to acknowledge the contribution of Kostas Zografos of the Athens University of Economics and Business for the work on the design framework.

REFERENCES

1. T. W. Malone, R. Laubacher, and M. S. S. Morton, *Inventing the Organizations of the 21st Century* (MIT Press, Cambridge, Massachusetts, 2003).
2. A.-W. Scheer and A. Hars, Extending data modelling to cover the whole enterprise, *Communications of the ACM* **35**(9), 166–175 (1992).
3. A. W. Scheer, *ARIS – Business Process Frameworks* (Springer, 1999).
4. M. Lundeberg, "The ISAC approach to specification of information systems and its application to the organisation of an IFIP working conference," in: *Information Systems Design Methodologies: A Comparative Review*, edited by T. W. Olle, H. G. Sol, and A. A. Verrijn-Stuart (IEEE Computer Society Press, North-Holland, 1982), pp. 273–234.
5. IDEF0, "Integration Definition for Function Modeling (IDEF0)," Computer Systems Laboratory, National Institute of Standards and Technology FIPS Pub 183 (December 21, 1993).
6. E. S. K. Yu and J. Mylopoulos, Using goals, rules and methods to support reasoning in business process reengineering, *Intelligent Systems in Accounting, Finance and Management* **5**(2) (1996).
7. A. I. Anton, W. M. McCracken, and C. Potts, Goal Decomposition and Scenario Analysis in Business Process Reengineering, in: *Proceedings of 6th International Conference on Advanced Information Systems Engineering (CAiSE'94)*, edited by G. Wijers, S. Brinkkemper, and T. Wasserman (Utrecht, The Netherlands, 1994), 94–104.
8. A. van Lamsweerde, Goal-Oriented Requirements Engineering: A Guided Tour, in: *Proceedings of 5th IEEE International Symposium on Requirements Engineering, RE'01* (Toronto, 2001), pp. 249–263.
9. C. Rolland, C. Souveyet, and C. Ben Achour, Guiding goal modeling using scenarios, *IEEE Trnansactions on Software Engineering* **24**(12), 1055–1071 (1998).
10. C. Rolland, G. Grosz, and K. Regis, Experience with Goal-Scenario Coupling in Requirements Engineering, in: *Proceedings of IEEE International Symposium on Requirements Engineering* (Limerick, Ireland, 1999), pp. 74–83.
11. P. Loucopoulos and E. Kavakli, Enterprise modelling and the teleological approach to requirements engineering, *International Journal of Intelligent and Cooperative Information Systems* **4**(1), 45–79 (1995).
12. V. Kavakli and P. Loucopoulos, Goal-driven business process analysis – application in electricity deregulation, *Information Systems* **24**(3), 187–207 (1999).
13. J. Coughlan and R. D. Macredie, effective communication in requirements elicitation: a comparison of methodologies, *Requirements Engineering* **7**(2), 47–60 (2002).
14. P. Loucopoulos, The S^3 (Strategy-Service-Support) Framework for Business Process Modelling, in: *Proceedings of Workshop on Requirements Engineering for Business Process Support (REBPS'03)*, edited by J. Eder, R. Mittermeir, and B. Pernici (Klagenfurt/Velden, Austria, 2003), pp. 378–382.
15. R. D. Galliers, Towards a flexible information architecture: integrating business strategies, information systems strategies and business process redesign, *Journal of Information Systems* **3**(3), 199–213 (1993).
16. F. Leymann and W. Altenhuber, Managing business processes as an information resource, *IBM Systems Journal* **33**(2), 326–348 (1994).
17. M. Ould, *Business Processes: Modelling and Analysis for Re-engineering and Improvement* (Chichester, John Wiley & Sons, 1995).
18. A. Fuxman, L. Liu, M. Pistore, M. Roveri, and J. Mylopoulos, Specifying and Analysing Early Requirements: Some Experimental Results, in: *Proceedings of 11th IEEE International Conference on Requirements Engineering* (Monterey Bay, California, USA, 2003), pp. 105–116.
19. A. Fuxman, J. Mylopoulos, M. Pistore, and P. Traverso, Model Checking Early Requirements Specifications in Tropos, in: *Proceedings of 5th IEEE International Symposium on Requirements Engineering* (Toronto, Canada, 2001), pp. 174–181.

20. B. Nuseibeh and S. Easterbrook, Requirements Engineering: A Roadmap, in: *Proceedings of 22nd International Conference on on Software Engineering* (Limerick, Ireland, 2000), pp. 35–46.
21. P. B. Checkland, *Soft Systems Methodology: A 30-year Retrospective*, New edition (Chichester, Wiley, 1999).
22. B. M. Richmond, Systems Thinking: Critical Thinking Skills for the 1990s and beyond, *System Dynamics Review* **9**(2), 113–133 (1993).
23. G. Walsham, Virtual organization: an alternative view, *Information Society* **10**(4), 289–292 (1994).
24. J. D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World* (Irwin/McGraw-Hill, Boston, 2000).
25. J. W. Forrester, *Principles of Systems* (Waltham, MA, Pegasus Communications Inc., 1999).
26. J. M. Carroll, Scenarios and Design Cognition, in: *Proceedings of IEEE Joint International Conference on Requirements Engineering (RE'02)*, edited by E. Dubois and K. Pohl (Essen, Germany, 2002), pp. 3–5.
27. B. Aune, *Knowledge of the External World* (Routledge, London, New York, 1991).
28. E. Kavakli and P. Loucopoulos, "Goal Modelling in Requirements Engineering: Analysis and Critique of Current Methods," in: *Information Modeling Methods and Methodologies*, edited by J. Krogstie, T. Halpin, and K. Siau (IDEA Group Inc., 2004).
29. P. Loucopoulos and N. Prekas, A Framework for Requirements Engineering Using System Dynamics, in: *Proceedings of 21st International Conference of the System Dynamics Society*, New edition (York City, 2003).
30. D. F. Andersen, G. P. Richardson, and J. A. M. Vennix, Group model building: adding more science to the craft, *System Dynamics Review* **13**(2), 187–201 (1997).
31. J. A. M. Vennix, Group model-building: tackling messy problems, *System Dynamics Review* **15**(4), 379–401 (1999).
32. A. Fowler, Simulations's Evolving Role in Management, in: *Proceedings of 1996 International System Dynamics Conference*, edited by G. P. Richardson and J. D. Sterman (Cambridge, Massachusetts, 1996), pp. 162–165.
33. E. F. Wolstenholme, Decision Analysis Using Dynamic Simulation, in: *Proceedings of System Dynamics '95*, edited by T. Shimada and K. Saeed (Tokyo, 1995), pp. 937–945.
34. C. Urquhart, Analysts and clients in organisational contexts: a converstational perspective, *Strategic Information Systems* **10**(3), 243–262 (2001).
35. B. Curtis, H. Krasner, and N. Iscoe, A field study of the software design process for large systems, *CACM* **31**(11), 1268–1287 (1988).
36. S. Kennedy, Why users hate your attitude, *Informatics*, pp. 29–32 (February 1994).
37. B. Bashein and M. I. Markus, A credibility equation for IT specialists, *Sloan Management Review* **38**(4), 35–44 (1997).

INFORMATION SYSTEMS AS A DESIGN SCIENCE

Some concerns

Juhani Iivari*

1. INTRODUCTION

There are increasing concerns about the relevance and identity of Information Systems (IS) as a scientific discipline (Benbasat and Zmud, 1999; 2003; Galliers, 2004; Hirschheim and Klein, 2003; Markus, 1999; Weber, 2003; Whinston and Geng, 2004). Benbasat and Zmud (2003, p. 191) suggest that the IT artefact and related immediate nomological net should form the core of the discipline:

> "our focus should be on how *to best design IT artifacts and IS systems* to increase their compatibility, usefulness, and ease of use or on how to best manage and support IT or IT-enabled business initiatives". [*italics* added by JI]

To this end, the IS discipline should produce knowledge to support IS development (Iivari, 2003).† But how can we provide this knowledge? The North American IS research community seems to believe that the best way to support practice is to focus on descriptive-explanatory theories, hoping that they will lead to practical implications relevant to practitioners.‡ To increase the direct utilization of research results in practice, Benbasat and Zmud (1999) recommend that we should

> "develop cumulative, theory-based, context-rich bodies of research to be able to make prescriptions and be proactive" (p. 14).

* Department of Information Processing Science, University of Oulu, P.O. Box 3000, 90014 Oulun yliopisto, Finland, juhani.iivari@oulu.fi.

† I use the term "development" in a broader sense than "design", to cover all the "life-cycle" activities of requirements construction, design, technical implementation and organizational implementation of an IS artefact and its re-development (evolution). Note also that I interpret development as being capable of taking place in-house or being based on application packages.

‡ An alternative explanation is that the major objective of the IS community was to achieve legitimacy in terms of the criteria applied in organization/management studies, without much concern for practical relevance.

Even though desirable in principle, it seems that this theory-driven research has seriously failed to produce results which are of real interest in practice. A pilot analysis of the practical recommendations made in articles in MISQ between 1996 and 2000 showed that these were weak (Iivari et al., 2004). As a consequence, it is no wonder that practitioners are not interested in these major IS journals (Kock et al., 2002).

Iivari (2003) proposes three reasons for the IS discipline's perceived lack of practical relevance: overemphasis on the "theory-with-practical-implications" research strategy (Benbsat and Zmud, 1999, 2003), underemphasis on the nature of IS as an applied engineering-like discipline, as a science of the artificial or as a design science (Simon, 1969; March and Smith, 1995; Hevner et al., 2004), and underemphasis on constructive research (Iivari, 1991; Nunamaker et al., 1991, Burstein and Gregor, 1999). Interestingly, similar concerns have also been expressed in business economics (Kaasanen et al., 1991) and management research (van Aken, 2004).

This paper continues some of the argumentation in Iivari (2003). It is structured in terms of four concerns:

1. What are the core artefacts built and studied in IS when viewed as a discipline of computing (Section 2)?
2. Is the concept of "IT artefact" too design-oriented (Section 3)?
3. Are systems development methods valid methods of constructive research (Section 4)?
4. How can one theorize about "IT artefacts" (Section 5)?

2. WHAT ARE THE CORE ARTEFACTS BUILT AND STUDIED IN INFORMATION SYSTEMS?

It is currently quite popular in the IS literature to talk about IT artefacts rather than about information systems. Dahlbom (1996) suggested the name "Informatics" instead of "Information Systems", interpreting the latter as covering only a certain era of computer use (or type of application).* He argued that our focus should be on Information Technology (IT) rather than information systems, because the latter do not easily cover, personal computing, communication, electronic publishing, air traffic control or intelligent houses, for instance. More specifically, he claims that we should conceive of our discipline in terms of "using information technology" instead of "developing information systems" (p. 34). Contrary to Dahlbom (1996), I see development as primary if we emphasize IS as an applied discipline. The work of most practitioners, including our students, is in practice not only to study the use of information technology, but to develop new IT artefacts. We as researchers and the practitioners on the field attempt to understand the use of IT artefacts in order to be able to develop "better" ones. In fact, the difference in emphasis between the view of Dahlbom (1996) and my own is not so great, as he also points out that we are interested in "the use of technology because we are interested in changing that use" mainly by designing/developing the technology concerned (p. 42).

* The name "Informatics" has been widely adopted in Scandinavian universities instead of "Information Systems".

Although Dahlbom (1996) suggested that our focus should be on Information Technology (IT) artefacts rather than information systems, it was Orlikowski and Iacono (2001) who popularized the phrase "IT artefact" within the IS research community. They define IT artefacts as "those bundles of material and cultural properties packaged in some socially recognizable form such as hardware and/or software" (p. 121), and based on the 188 articles published in *Information Systems Research* in the decade beginning in 1990 and ending in 1999, they distinguish 13 views of IT artefacts. Most of these conceptualizations treat IT artefacts as black boxes without looking inside them. They simply focus on the computational capabilities of IT artefacts (the computational view of technology), their intended uses (the tool view of technology), technology as a variable (the proxy view of technology), how technologies come into being or how technologies come to be used (the ensemble view of technology).

If we conceive of Information Systems as a design science that also builds IT artefacts, a natural question is what sort artefacts we build, especially if we wish to distinguish Information Systems from Computer Science and Software Engineering. IT artefacts obviously include various computer hardware and software products. Gorgone et al. (2002) identify seven areas of knowledge in information technology: computer architectures, algorithms and data structures, programming languages, operating systems, telecommunications, databases and artificial intelligence. If one looks more at IT applications, one can identify embedded computer systems and non-embedded applications. Embedded systems may be totally invisible to human beings, such as the computer controlling fuel injection in a car.

One way to conceptualize existing applications is to distinguish different roles or functions that IT applications may serve (Table 1). The first four functions of Table 1 are close to "technology as a labour substitution tool", "technology as a productivity tool", "technology as a social relations tool" and "technology as an information processing tool" in the view of Orlikowksi and Iacono (2001). The roles "to automate" and "to informate" come from Zuboff (1988). Computer games illustrate the capability of IT applications to entertain. Finally, IT applications may also attempt to arouse artistic experience, and one can easily imagine a new sort of art that is essentially built on the interactive character of computer technology.

A single application may include several functions. For example, a word processor is primarily a tool intended to augment text production. At the same time it is a medium that

Table 1. Functions of IT applications

Role/function	Metaphors	Examples
To automate	Processor	Many embedded systems Many transaction processing systems
To augment	Tool (proper)	Many personal productivity systems
To mediate	Medium	E-mail Digital media
To informate	Information source	Information systems proper
To entertain	Game	Computer games
To artisticize	Piece of art	Computer art

may make use of the specific nature of computer technology as a medium (e.g. links) and it automates some aspects of text production (e.g. spelling). Zuboff (1988) claimed that to automate also allows one to informate. This can obviously be extended to cover other uses of IT applications, so that computer games, for example, could at least in principle collect information about the users' actions and reactions during playing. This can be used to develop the game further. A second point is that many IT applications with the primary role of automating, augmenting, entertaining or possibly artisticizing may include an information system that supports the use of the primary functionality. E-mail, for example, with the original function of communicating messages, includes mailboxes that allow one to build a directory to informate about previous communications. Thus it can be developed into a fairly sophisticated information system about one's electronically mediated social network and one's electronic communication within that network.

The major point in Table 1, however, is that the core purpose of an information system is to informate. In my vocabulary, information systems form a subcategory of IT artefacts. I interpret an information system as being a system whose purpose "is to supply its groups of users (...) with information about a set of topics to support their activities." (Gustafsson et al., 1982). The definition implies that an information system is specific to the organizational (or inter-organizational) context in which it is implemented. In the terms of Walls et al. (1992), March and Smith (1995) and Lee (1999), an information system is an instantiation of more general information technology. As a consequence, no prefabricated commercial software product is an information system as such. An information system cannot be bought, only software and hardware (and possibly data) to be used in its implementation can be bought.

Analysing the above definition, one can identify three levels of abstraction in an information system (Iivari, 1983; Iivari and Koskela, 1987), which closely correspond to the three contexts identified by Lyytinen (1987):

organizational level (organizational context):*	Users and their activities
conceptual/infological level (language context):	Information about a set of topics
datalogical /technical level (technology context):	Technology

Alter (2003) proposes that work systems, and especially IT-reliant work systems, should be taken as "units of analysis" in our field instead of IT artefacts. I agree with him that we often attempt when developing information systems to enhance work systems. Work systems represent users and their activities at the organizational level in the above framework, but to define the core and identity of our discipline in terms of IT-reliant work systems would be an attempt by IS experts to monopolize the development of work systems. I see the enhancement of work systems more as an interdisciplinary effort in which experts from different fields are required (expertise in organizations, expertise in the application domain, and expertise in IT). IS experts represent only IT-related expertise in this enhancement effort.

* It is not necessary to confine organizations here to "formal organizations" such as companies, for organizations may include various inter-organizational arrangements and informal organizations such as families, for example.

Weber (2003) proposes that information systems rather than IT artefacts should form the core of the IS discipline, and more specifically he puts forward the idea of information systems as representations that enable 'faithful' tracking of other systems. An information system as a state tracking system refers to the conceptual/infological level above, where the information system is assumed to include faithful information about the state of things (topics) in another system. One should note, however, that I do not confine information systems to "after-the-fact" tracking systems, but also allow them to include information about the future (e.g. forecasting).

Agreeing with Weber (2003), I am ready to suggest that information systems should form the core of the IS discipline rather than IT artefacts. In fact, I interpret the somewhat convoluted definitions of IT artefacts provided by Orlikowski and Iacono (2001)* and Benbasat and Zmud (2003)† as attempts to limit the focus to IT artefacts that are close to information systems. I have a number of reasons for this suggestion. First, this is consistent with the name of our discipline. Second, as pointed out above, the concept of information systems allows us to incorporate the ideas of Alter (2003) and Weber (2003) and more or less those of Orlikowski and Iacono (2001) and Benbasat and Zmud (2003) as well. Third, the concept of information system as defined above implies that information is a significant part of the system. Thus this fundamental concept (Alter, 2003) is not just an empty word in the name of a discipline as it often is in the case of IT.‡ Finally, I agree with Alter (2003) that even though the core of the discipline may be defined narrowly, the borders may be broad.

3. IS THE CONCEPT OF "IT ARTEFACT" TOO DESIGN-ORIENTED?

Simon (1969/1996) makes a distinction between artificial, or man-made things and natural things. He associates artefacts with design, in that they are designed (synthesized) by human beings, even though not necessarily with full forethought. Since theories can also be considered "artefacts" in a sense, I exclude them from the extension of the concept of artefact by presuming that artefacts do not have any truth or truth-like value as theories do (Niiniluoto, 1999)§ but are more or less useful as means of achieving certain ends.

I feel that the dichotomy between designed artefacts and natural objects is too simple. Many "artefacts" are only partly the work of a designer. Figure 1 aims at describing this continuum of "artefacts", starting from completely designable artefacts such as mathemat-

* "Bundles of material and cultural properties packaged in some socially recognizable form such as hardware and/or software."

† "The application of IT to enable or support some task(s) embedded within a structure(s) that itself is embedded within a context(s)." (p. 186).

‡ Space does not allow me to discuss here the concept of information and its relationship to knowledge. Note, however, that the often adopted view of "data" as pure raw inputs (facts) and "information" as processed, cooked output is totally untenable.

§ Even though Sutton and Staw (1995) point out that there is more consensus about what theory is *not* than about what theory is, I dare to characterize my interpretation of "theory" as a systematic explanation of certain phenomena that allows existing empirical findings about the phenomena to be explained and/or hypotheses to be generated which can at least in principle be subjected to empirical testing. By a truth value I mean that theories more or less approximate to the truth (Niiniluoto, 1999).

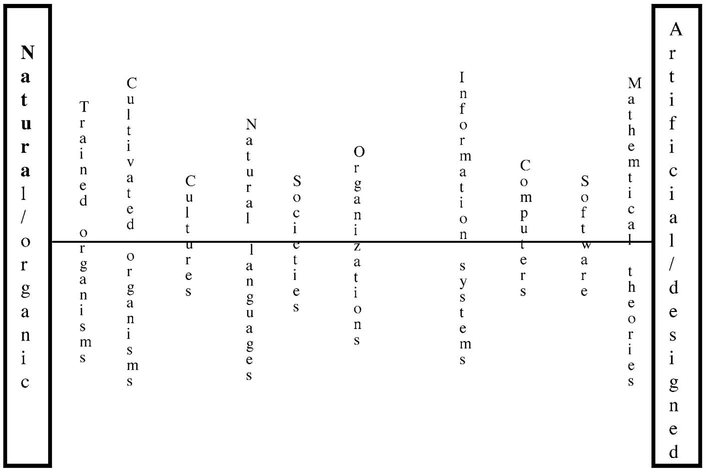


Figure 1. Natural-artificial as a continuum.

ical theories and ending with "natural" objects in our environment which are nevertheless partly man-made because of factors such as cultivation, breeding, genetic engineering and training.

The position of the phenomena on the continuum of Figure 1 is only an example. Software is interpreted as a system of algorithms close to mathematical structures. Computers have their physical implementations, which introduce a natural element into them. In addition to software and computers, information systems comprise an information base, which is only partially designable and additionally makes information systems organic/emergent. On the opposite side, trained organisms (such as human beings and some animals) are considered less "designed" than cultivated ones, because the influence of training is merely ontogenetic while the influence of cultivation and breeding is phylogenic. One should also note that there may be internal variation within each phenomenon, so that some societies may be more designed than others, for example. Similarly, organizations may also differ in the degree to which they are designed.

The purpose of the above exercise is to illustrate that the dichotomy between artificial and natural is a simplification. It is obvious that the term "artefact" emphasizes the artificial, designed end of the continuum, but unfortunately, I am unable to find a better term than "artefact". "Technology", when interpreted as "a design for instrumental action that reduces the uncertainty in the cause-effect relationship involved in achieving a desired outcome" (Rogers, 1995, p. 12), might be an alternative term, but it may have too technical a connotation. Järvinen (2002) prefers to speak about "innovations" rather than "artefacts", but the concept of "innovation" may lose the connotation of artificiality in contrast to theories.

At the same time, Figure 1 suggests that information systems differ in their degree of artificiality from other IT artefacts such as software and computers. Many IT artefacts are only partly the work of a designer. They may exhibit emergent features as an outcome of numerous local actions (e.g. use, interpretation, negotiation and redesign), but these emergent features cannot be anticipated by reference to any *a priori* design. At a more theoretical level, the literature on the social construction of technology (Bijker et al., 1989; Bijker and Law, 1992; Orlikowski and Gash, 1994) discusses this emergent aspect of many artefacts. The provocative article of Truex et al. (1999) suggests that emergent organizations need continuous redevelopment of their systems, but in spite of the title of their paper "Growing systems in emergent organizations", the authors fail to recognize emergent information systems which grow without any continuous redevelopment. More recently, Markus et al. (2002) have analysed the provision of IT support for emergent knowledge processes (EKPs), which they define as organizational activity patterns characterized by (1) an emergent process of deliberations with no best structure or sequence, (2) an actor set that is unpredictable in terms of job roles or prior knowledge, and (3) knowledge requirements for general and specific distributed expertise. Unfortunately, they are not very explicit in formulating the characteristics of EKP support systems (see Section 6). On the other hand, Knowledge Management Support Systems provide good examples of emergent information systems in which the support provided by the information system is much more dependent on the growth of the system than on its design. Answer Garden (Ackerman, 1998) provides a good example of this kind of growing information system. If a user does not find an answer to his/her question or is not satisfied with the answer, he/she may trigger the system to route to an appropriate human expert. The expert answers the user directly, but may also insert the answer with related diagnostic questions into the database. In that way Answer Garden provides a mechanism for growing a body of information over time.

4. ARE SYSTEMS DEVELOPMENT METHODS VALID METHODS OF CONSTRUCTIVE RESEARCH?

Benbasat and Zmud (1999) note that "Academic work could impact practice through the development of tools, techniques, and practices", noting that such research contributions are infrequently observed [in major journals such as MISQ and ISR] (p. 9). The development of such *"meta-artefacts"* to support the development of IS artefacts is a complementary approach to the "theory-with-practical-implications" type of research. Following Walls et al. (1992), meta-artefacts can be divided into *meta-artefacts for the IS product* and *meta-artefacts for the ISD (information systems development) process*. The former comprise technical implementation resources such as application domain-specific software components, application frameworks, application packages, ERP systems, development environments, IS generators, or their prototypes, which can be used in the technical implementation of an IS artefact,* and also more abstract models and principles such as various architectural models, analysis and design patterns, and application-dependent design principles for use in the design and implementation of the IS product.† The latter correspond

* Of course, meta-artefacts for the IS product also include programming languages, DBMSs, UIMSs, etc. I assume, however, that they belong more to the province of Computer Science and Software Engineering.

† Programming patterns, normal forms in relational databases, principles of modularization and information hiding may be more familiar examples, but again I interpret them as belonging more to Computer Science and Software Engineering.

to the "design process" in the information system design theory of Walls et al. (1992) and comprise systems development approaches, methods, techniques and tools, for example.

These "meta-artefacts" are often relatively complex, and their development involves constructive research (Iivari, 1991) for building the artefacts (March and Smith, 1995; Hevner et al., 2004). It is clear that constructive research is badly neglected in the IS field. Well-known classifications of IS research methods such as those of Benbasat (1985), Jenkins (1985) and Galliers and Land (1987), for example, do not recognize anything resembling constructive research methods. It also seems that the building of artefacts is also poorly understood. Hevner et al. (2004), for example, propose a number of guidelines for design science research and methods for design evaluation, but these are poor in addressing building activity.

Many authors associate constructive research with action research (e.g. Kaasanen et al., 1991; Burnstein and Gregor, 1999; Järvinen, 2002) or clinical research based on multiple case studies (van Aken, 2004). Authors such as Kaasanen (2001) and van Aken (2004) do not pay any special attention to the construction of complex artefacts. The implicit focus of van Aken (2004), for example, lies in improvement problems rather than construction problems. Action and clinical research, of course, provide an opportunity to evaluate the resulting artefacts and even to build them in close contact with the real context in which they are to be used. Most constructive research in Computer Science, Software Engineering and Information Systems has taken place outside this real context of use, however. Furthermore, action research and the typical engineering type of research are quite different philosophically.

Nunamaker et al. (1990) propose systems development as a specific research method to be used for constructing the meta-artefacts discussed above. If systems development methods are really to be applicable, this should put an end to the regression of meta-levels between artefacts. Systems development methods as meta-artefacts for the ISD process could be employed for developing other meta-artefacts. The question is, however, whether they allow sufficient room for creativity and serendipity, which are essential in research.

5. HOW CAN ONE THEORIZE ABOUT IT ARTEFACTS?

Simon (1969/1996) assumes artefacts to be interfaces between an "inner" and an "outer" environment, both being subject to natural laws. This idea would provide a promising starting point for a theory of artefacts. Weber (1987) also remarks that

> "it would be a strange quirk of nature if human assemblies of natural objects did not manifest order in the same way that natural assemblies of natural objects manifest order" (p. 13),

and then proceeds to outline a theory of discrete artefacts. Referring to Brooks (1987), one can claim, however, that software systems contain arbitrary complexity, because they are designed at will by different people. It is only constraints on current technology, other resources, and above of all imagination that limit software products. Thus the big question mark for me is whether we are able to combine theories of artefacts with theories of their inner, and more especially of their outer, environments.

March and Smith (1995) discuss research as a design and natural science, suggesting four research activities: build, evaluate, theorize and justify.* *Building* means the construction of an artefact for a specific task, demonstrating that such an artefact can be constructed. *Evaluation* is the process of determining whether the artefact represents any progress relative to the performance of existing artefacts. *Theorizing* explains why and how the effects occurred, i.e. why and how the artefact works, and *justification* performs empirical and/or theoretical research to test the theories proposed. According to March and Smith, theorizing can be *ex post* activity: given that the performance of an artefact has been evaluated, it is important to theorize on why and how it worked or did not work in its environment. Building and evaluation represent design science activities, and theorizing and justification natural science activities.

Walls et al. (1992) propose the interesting idea of an "IS design theory". This would be divided into two parts: the design product and the design process. They suggest that an IS design theory for a product should consist of meta-requirements (the class of goals to which the theory applies), meta-design (the class of artefacts hypothesized to meet the meta-requirements)†, kernel theories (theories from the natural and social sciences governing design), and testable design product hypotheses (used to test whether the meta-design satisfies the meta-requirements). An IS design theory for a process would comprise a design method (a description of the procedures for artefact construction)‡, kernel theories and testable design process hypotheses (used to verify whether the design method results in an artefact which is consistent with the meta-design).

It is difficult for Walls et al. (1992) to find any good examples of such IS meta-artefacts with well-defined kernel theories, however.§ Markus et al. (2002) adopt a simplified idea of design theory to introduce a design theory for systems that support emergent knowledge processes (EKPs). They propose three requirements for IT support of EKPs: 1) systems cannot target specific user roles, depend on training, or assume motivation to use the tool, 2) systems must accommodate complex, distributed, evolving knowledge bases, and 3) systems must support an unstructurable, dynamically changing process of deliberations and tradeoffs. Markus et al. (2002) do not clearly distinguish design theory for an IS design product from design theory for the IS design process, but suggest six principles for EKP support and the design process: 1) design for customer engagement by seeking out naïve users, 2) design for knowledge translation through radical iteration with functional prototypes, 3) design for offline action, 4) integrate expert knowledge with local knowledge sharing, 5) design for implicit guidance through a dialectical development process, and 6) componentize everything, including the knowledge base. They do not propose any testable design product and design process hypotheses.¶ Markus et al. (2002) also seem to play

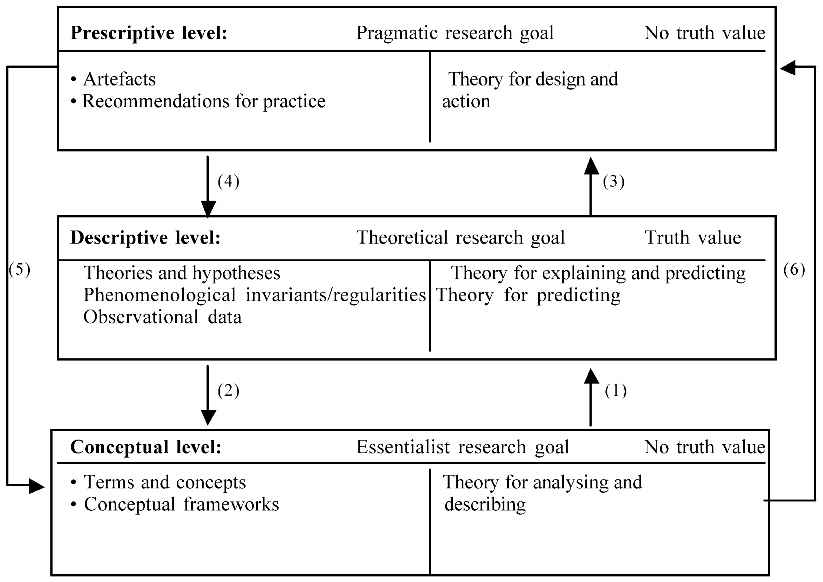
* Natural science in March and Smith (1995) covers traditional research in the physical, biological, social and behavioural domains (p. 253).

† Meta-designs correspond to meta-artefacts for the IS product in Section 5.

‡ Design methods correspond to meta-artefacts for the ISD process in Section 5.

§ They suggest that the relational database theory is the best-developed design theory in the IS discipline.

¶ Markus et al. (2002) suggest that their design principles can be restated as a set of hypotheses to be tested empirically in other situations where the same theoretical conditions hold. One should not confuse these with testable design product and design process hypotheses.



(1) Conceptual model of the research territory and its terminology
(2) Conceptual analysis of theories
(3) Theories, regularities and observations as foundations for artefacts and recommendations
(4) Empirical research investigating artefacts
(5) Conceptual analysis of artefacts
(6) Conceptual frameworks as methods (artefacts)

Figure 2. A hierarchy of conceptual, descriptive and prescriptive levels of research.

down the significance of kernel theories, claiming that these may be academic theories or practitioner theories-in-use. They suggest a kernel theory of EKPs with three characteristics: 1) it is almost impossible to predict in advance who will participate in the process and what tools they will use, 2) knowledge is distributed and includes both general expertise and local context knowledge, and 3) the process is emergent.

I would consider the existence of a kernel theory to be a key component of a design theory. To allow any practitioner theory-in-use to serve as a kernel theory implies that a design theory is not necessarily based on any scientifically validated knowledge. Furthermore, a “design theory” as introduced in Walls et al. (1992) does not necessarily include any explanation of why the “meta-design” satisfies the meta-requirements and the “design method” results in an artefact that is consistent with the meta-design (cf. theorizing and justification in March and Smith (1995)). It may therefore be misleading to regard “design theories” as a category of theories. Taking a cynical viewpoint, there is a danger that the idea of a “design theory” will be (mis)used just to make our field sound more scientific without any serious attempt to strengthen the scientific foundation of the meta-artefacts proposed.

Gregor (2002) proposes a hierarchy of five theories: 1) a theory for analysing and describing, 2) a theory for understanding, 3) a theory for predicting, 4) a theory for explaining and predicting, and 5) a theory for design and action. Theories for analysing and describing name and classify units of analysis (individuals, groups, situations, or events) based on commonalities found in the specific dimensions and characteristics of the units. Theories for predicting aim at foreseeing outcomes from a set of predictors, without necessarily understanding the causal connection. A theory for explaining and predicting represents the traditional view of theory, in that it implies both prediction and understanding of the underlying causes. A theory for design and action says "how to do things", and finally, theories for understanding refer to quite general conjectures or theories such as structuration theory and actor-network theory that can be used as "sensitizing devices". Because of their generality theories for understanding are not falsifiable.

Although it is again arguable whether all these should be called "theories", they nicely illustrate different types of knowledge involved in our discipline. Apart from the theory of understanding, they can easily be mapped on the three-layer model of the IS discipline proposed in Iivari (1983), a model influenced by Lehtovuori (1973) and more indirectly by Chmielewicz (1970).* The hierarchy of descriptive research is based on Törnebohm (1975). Concepts and conceptual frameworks at the conceptual level aim at identifying essences in the research territory and their relationships. The descriptive level aims at describing, understanding and explaining how things are, whereas the prescriptive level is interested in how to achieve the specified ends in an effective manner. Note, however, that artefacts and recommendations as such do not have a truth or truth-like value, but only statements about their efficiency and effectiveness do.

6. FINAL COMMENTS

This paper has argued that in order to increase the practical relevance of the IS discipline we should emphasize more the nature of Information Systems as a design science that develops various "meta-artefacts" to support the construction of information systems and other IT artefacts. The paper argues that, if we regard Information Systems as a design science, we should consider what sort of IT artefacts we are focusing on, and suggests that information systems form core objects of our research. If we accept that Information Systems is a design science, then the question of how to construct meta-artefacts becomes essential. Is it enough that the meta-artefacts are just invented, or should their development be made transparent (just as the development of novel theories is). Finally, the paper discussed how to theorize about IT artefacts. Even though the term "design theory" may be misleading, the idea that a design theory should be based on a kernel theory makes it possible to link the IT artefacts to theories. Unfortunately, it seems that, as the IS community, we lack good examples of design theories. In order the promote IS as a design science and constructive research into the building of meta-artefacts, we should take stock of good examples of design theories, reconstruct how the meta-artefacts were developed (invented and elaborated) and assess the role of kernel theories in their development. This could provide examples of how to do high-quality constructive research in practise.

* Lehtovuori (1973) and Chmielewicz (1970) also have a fourth, normative level. I have excluded this from consideration because it is still a controversial question whether one can make "ought-to" conclusions based on "what is".

REFERENCES

Ackerman, M., 1998, Augmenting organizational memory: A field study on Answer Garden, *ACM Transactions on Information Systems* **16**(3):203–224.

Alter, S., 2003, Sidestepping the IT artifact, scrapping the IS silo, and laying claim to 'systems in organizations', *Communications of the AIS* **12**:494–526.

Benbasat, I., 1985, An analysis of research methodologies, in: *The Information Systems Research Challenge*, F. W. McFarlan, ed., Harvard Business School Press, Boston, pp. 47–85.

Benbasat, I., and Zmud, R. W., 1999, Empirical research in information systems: The practice of relevance, *MIS Quarterly* **23**(1):3–16.

Benbasat, I., and Zmud, R. W., 2003, The identity crisis within the discipline: Defining and communicating the discipline's core properties, *MIS Quarterly* **27**(2):183–194.

Bijker, W. E., Hughes, T. P., and Pinch, T. J. (eds.), *The Social Construction of Technological Systems, New Directions in the Sociology and History of Technology*, The MIT Press, Cambridge, MA, 1989.

Bijker, W. E., and Law, J. (eds.), *Shaping Technology/Building Society*, The MIT Press, Cambridge, MA, 1992.

Brooks, F. P. Jr., 1987, No silver bullet: Essence and accidents of software engineering, *Computer* **20**(4):10–19.

Burstein, F., and Gregor, S., 1999, The systems development or engineering approach to research in information systems: An action research perspective, in: *Proceedings of the 10th Australasian Conference on Information Systems*, B. Hope and P. Yoong, eds., Victoria University of Wellington, New Zealand, pp. 122–134.

Chmielewicz, K., 1970, Forschungskonzeptionen der Wirtschaftswissenschaft, Stuttgart.

Dahlbom, B., 1996, The new informatics, *Scandinavian Journal of Information Systems* **8**(2):29–48.

Galliers, R. D., and Land, F. F., 1987, Choosing appropriate information systems research methodologies, *Communications of the ACM* **30**(11):900–902.

Galliers, R. D., 2004, Change as crisis or growth? Toward a trans-disciplinary view of information systems as a field of study: A response to Benbasat and Zmud's call for returning to the IT artifact, *Journal of the AIS* **4**(6):337–351.

Gregor, S., 2002, A theory of theories in information systems, in: *Information Systems Foudations: Building the Theoertical Base*, S. Gregor and D. Hart, eds., Australian National University, Canberra, pp. 1–20.

Gorgone, J. T., Davis, G. B., Valacich, J. S., Topi, H., Feinstein, D. L., and Longenecker, H. E., Jr., 2002, *IS 2002, Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems*, Association for Computing Machinery (ACM), Association For Information Systems (AIS), Association for Information Technology Professionals; (http://www.acm.org/education/curricula.html#IS2002).

Gustafsson, M. R., Karlsson, T., and Bubenko, J. Jr., 1982, A declarative approach to conceptual information modelling, in: *Information Systems Design Methodologies: A Comparative Review*, T. W. Olle, H. G. Sol, and A. A. Verrijn-Stuart, eds., North-Holland, Amsterdam, pp. 93–142.

Hevner, A. R., March, S. T., Park, J., and Ram, S., 2004, Design science in information systems research, *MIS Quarterly* **28**(1):75–105.

Iivari, J., 1983, *Contributions to the theoretical foundations of systemeering research and the PIOCO model*, Acta Universitatis Ouluensis, Ser. A 150, Oulu.

Iivari, J., 1991, A paradigmatic analysis of contemporary schools of IS development, *European Journal of Information Systems* **1**(4):249–272.

Iivari, J., 2003, The IS core – VII: Towards information systems as a science of meta-artifacts, *Communications of the AIS* **12**:568–581.

Iivari, J., Hirschheim, R., and Klein, H., 2004, Towards a distinctive body of knowledge for information systems experts: Coding ISD process knowledge in two journals, *Information Systems Journal* **14**(4):313–342.

Iivari, J., and Koskela, E., 1987, The PIOCO model for is design, *MIS Quarterly* **11**(3):401–419.

Jenkins, A. M., 1985, Research methodologies and MIS research, in: *Research Methods in Information Systems*, E. Mumford, R. Hirschheim, G. Fitzgerald, and A. T. Wood-Harper, eds., North-Holland, Amsterdam, pp. 103–117.

Järvinen, P., 2001, *On Research Methods*, Opinpajan kirja, Tampere, Finland.

Kaasanen, E., Lukka K., and Siitonen, A., 1991, Konstruktiivinen tutkimusote liiketaloustieteissä, *Liiketaloudellinen Aikakausikirja* **3**:301–327.

Kock, N., Gray, P., Hoving, R., Klein, H., Myers, M., and Rockart, J., 2002, IS research relevance revisited: Subtle accomplishment, unfulfilled promise, or serial hypocrisy?, *Communications of the AIS* **8**:330–346.

Lee, A. S., 1999, Researching MIS, in: *Rethinking Management Information Systems*, W. L. Currie and B. Galliers, eds., Oxford University Press, pp. 7–27.

Lehtovuori, J., 1973, *Liiketaloustieteen metodologista taustaa*, Turun kauppakorkeakoulun julkaisuja, AI-6.

Lyytinen, K., 1987, A taxonomic perspective of information systems development: theoretical constructs and recommendations, in: *Critical Issues in Information Systems Research*, R. Boland and R. Hirschheim, eds., John Wiley & Sons, Chichester, pp. 3–41.

March, S. T., and Smith, G. F., 1995, Design and natural science research on information technology, *Decision Support Systems* **15**:251–266.

Markus, M. L., 1999, Thinking the unthinkable: What happens if the IS field as we know it goes away?, in: *Rethinking Management Information Systems*, W. L. Currie and B. Galliers, eds., Oxford University Press, Oxford, pp. 175–203.

Markus, M. L., Majchrzak, A., and Gasser, L., 2002, A design theory for systems that support emergent knowledge processes, *MIS Quarterly* **26**(3):179–212.

Niiniluoto, I., 1999, *Critical Scientific Realism*, Oxford University Press, Oxford.

Nunamaker, J. F., Chen, M., and Purdin, T. D. M., 1990, System development in information systems research, *Journal of Management Information Systems* **7**(3):99–106.

Orlikowski, W. J., and Gash, D. C., 1994, Technological frames: Making sense of information technology in organizations, *ACM Transactions on Information Systems* **12**(2):174–207.

Orlikowski, W. J., and Iacono, C. S., 2001, Research commentary: Desperately seeking the "IT" in IT research – A call theorizing the IT artifact, *Information Systems Research* **12**(2):121–134.

Rogers, E. M., 1995, *Diffusion of Innovations*, fourth edition, The Free Press, New York.

Simon, H., 1969/1981/1996, *The Sciences of Artificial*, MIT Press, Cambridge, MA.

Sutton, R. I., and Staw, B. M., 1995, What theory is not, *Administrative Science Quarterly* **40**:371–384.

Törnebohm, H., 1975, Tieteellisestä tutkimusprosessista, in: *Yhteiskuntatieteiden abstrakti metodologia*, R. Tuomela, ed., Gaudeamus Ab, Hämeenlinna, Finland.

Truex, D. P., Baskerville, R., and Klein, H., 1999, Growing systems in emergent organizations, *Communications of the ACM* **42**(8):117–123.

van Aken, J. E., 2004, Management research based on the paradigm of design sciences: The quest for field-tested and grounded technological rules, *Journal of Management Studies* **41**(2):219–246.

Walls, J., Widmeyer, G. R., and El Sawy, O. A., 1992, Building an information system design theory for vigilant EIS, *Information Systems Research* **3**(1):36–59.

Weber, R., 1987, Toward a theory of artifacts: A paradigmatic base for information systems research, *Journal of Information Systems* **1**:3–19.

Weber, R., 2003, Still desperately seeking for the IT artifact (editor's comments), *MIS Quarterly* **27**(2):iii–xi.

Whinston, A., and Geng, X., 2004, Opertionalizing the essential role of the information technology artifact in information systems research: Gray area, pittfalls, and the importance of strategic ambiguity, *MIS Quartelly* **8**(2):149–159.

Zuboff, S., 1988, *In the Age of the Smart Machine, The Future of Work and Power*, Heineman, Oxford.

INFORMATION SYSTEMS DEVELOPMENT (ISD): PAST, PRESENT, FUTURE TRENDS

Anders G. Nilsson*

1. INFORMATION SYSTEMS DEVELOPMENT AS AN ACADEMIC FIELD

By information systems development we mean analysis, design and implementation of useful IT systems to support some kind of business in organisations (cf. Avison and Fitzgerald, 2003; Nilsson and Pettersson, 2001). By IT systems we mean the use of hardware and software solutions to improve the business activities within and between organisations. The IT systems can be of a various character – for example we can create information systems in organisations by using bespoke (tailor-made) software, application packages or component-based solutions. We are here focusing on computer-based systems for developing and changing the situation in concrete business cases.

Research on information systems development (ISD) has its roots back in the mid 1960ies. Scandinavian researchers have had a great influence on the evolution of information systems development as an academic field (see Iivari and Lyytinen, 1998; Davis, 2003). Personally, I had the privilege of being a member of the Scandinavian school and tradition of information systems development (Langefors, 1973; 1995). My main experiences are based from working with the ISAC approach for requirements specifications (Lundeberg et al., 1981), the SIV method for purchasing standard application packages (Nilsson, 1990), the Business Modelling framework for studying method combinations (Nilsson et al., 1999) and the ISD perspectives on multimedia development (Burnett et al., 2003). After practising in the ISD area for more than 30 years, as both a researcher/teacher (for the academia) and an advisor/counsellor (to the industry), I feel a great need to make some reflections on my findings.

This paper will focus on important trends in the area of Information Systems Development (ISD) from three time perspectives: the past (yesterday), the present (today) and the future (tomorrow) situation. The method for comparison is driven from ten important and relevant dimensions: philosophical approach, modelling area, view of model aspects, sys-

* Karlstad University, Information Systems, SE-65188 Karlstad, Sweden, Anders.Nilsson@kau.se.

tems scope, methodology package, actor orientation, IT perspective, subject matter focus, development strategy and range of work. My review is of course strongly biased of how I perceive the trends in our ISD field. The research work is a continuation of two earlier studies on evolution of ISD methodologies (Nilsson, 1995) and change work with ISD in organisations (Nilsson, 2003). The paper will follow the ten dimensions for comparison of trends in the ISD field. For each dimension the three time perspectives are presented with a focused area for the past, present and future trends respectively.

2. PHILOSOPHICAL APPROACH

The philosophical standpoint for the ISD field has changed over time from a pure systems approach to include relevant knowledge from network theories and recently from multi-perspective approaches.

2.1. Systems Approach – Past Trend

The systems approach has historically had a great impact on construction of methodologies for ISD. Characteristic for such a view is that ISD is accomplished in a systematic manner trying to do different tasks in a natural and logical order. The ISD work is therefore partitioned in a set of manageable phases and steps for building and acquiring new IT-systems. Each phase requires different types of specialist or expert knowledge. The systems approach has been regarded as a core theory for the ISD field.

2.2. Network Approach – Present Trend

The network approach is a school of thought which is developed by industrial marketing within the business administration area. Different actors or interest groups establish strong networks within or between companies. This view has shown a great relevance for extending the present methodologies for ISD work. As an example we can mention the construction of ISD approaches for taking care of issues regarding inter-organisational systems and outsourcing of IT-systems. The network approach can in a wider sense be seen as a special case of the systems approach.

2.3. Multi-Perspective Approach – Future Trend

A rather new view which has evolved more recently within the ISD field is labelled the multi-perspective approach. It could be a major future trend for our area! ISD work is placed in a larger context as part of organisational and cultural changes in companies. The methodologies for ISD will open up for looking at IT systems and the connected business processes from different angels or perspectives. For example human-centred and market-driven factors will heavily influence the development work. A multi-perspective view is a generic approach where systems and network thinking should be essential perspectives for analysis and design of new IT-systems in organisations.

3. MODELLING AREA

The practice of ISD has from the beginning been oriented towards modelling of some concrete subject of interest. The modelling area has been broader and broader over the years. Methodologies for ISD have originated in an information modelling paradigm which have evolved in a business modelling framework and now will be facing the challenges of modelling virtual markets.

3.1. Information Modelling – Past Trend

The justification for starting up the new field of ISD in the mid 1960ies was to propose the work with information analysis before "jumping down" to program construction and database design. Information modelling was invented as an efficient tool for analysing information flows in organisations which after that would be realised by so called computer-based systems. Information modelling is still an important kernel or corner-stone in the today's theories of ISD work.

3.2. Business Modelling – Present Trend

The present focus area in ISD research has been shifted to what is called business modelling or enterprise modelling. By business modelling we mean the use of models to understand and change business operations together with IT-systems in organisations. The area of interest is more on studying the business demands and commercial effects of IT-systems rather than analysing the information flows per se. Information support has been a more integrated part of business operations and, in many cases, a vital part of the business mission itself.

3.3. Virtual Modelling – Future Trend

In the future ISD will be oriented to model how companies will operate in the oncoming virtual markets. The modelling area of interest will become how different kinds of inter-organisational IT-systems can support electronic commerce applications and web-based business solutions. The future trend is towards more service-based operations in companies which mean that virtual modelling will be a necessary extension to the earlier tools of information and business modelling. There is a challenge to integrate work with ISD and service development in business settings of tomorrow.

4. VIEW OF MODEL ASPECTS

A closely related dimension to the "modelling area" is the issue regarding "view of model aspects". Methodologies for ISD are emphasising some kind of model aspect when describing business operations and their supporting IT-systems. This ends up in a useful document labelled requirements specification. In the ISD area the evolution of trends for model aspects has gone through a resource view to a more comprehensive process view and in the future to stress strategically a more intention view.

4.1. Resource View – Past Trend

Historically IT-systems have been regarded as a resource to make the business operations more efficient in a company or organisation. We have in the ISD area developed techniques for e.g. Conceptual Modelling (CM) and Information Resource Management (IRM). From a resource viewpoint we can model concrete aspects of data, concepts, components and objects together with their relationships. In other words a resource view illuminates the users' constructs and professional language. Resource-driven methodologies consider data or information as a valuable resource or asset for a company.

4.2. Process View – Present Trend

In the today's situation we see IT-systems in a broader sense as a part or way for making a purposeful process orientation of a company. We have in the ISD area developed techniques for e.g. Business Process Reengineering (BPR) and Process Management (PM). From a process viewpoint we can model concrete aspects of activities, events, rules, tasks and functions together with their logical order or sequence. In other words a process view illuminates coordination of organised activities in companies and the triggers for initiating different work tasks in business. Process-driven methodologies consider the business and information flows as a valuable platform for finding good structures in a firm.

4.3. Intention View – Future Trend

In the future we have to even more stress the strategic importance of guiding the IT-systems in the right direction; this will be done through carefully specifying the intentions behind the development work. We have to use techniques for e.g. SWOT analysis and Balanced Scorecard (BSC). We are in the ISD area also developing techniques for e.g. situation graphs and means/ends diagrams. From an intention viewpoint we can model concrete aspects of goals, visions, critical success factors (CSF), problems, strengths, measures, actors and force fields. In other words an intention view has the special role to "make life" to and change the prerequisites for the business operations. Intention-driven methodologies consider the business mission and the purpose of IT-systems as a valuable driving force for performing ISD in a right manner.

5. SYSTEMS SCOPE

The systems scope for studying IT-systems during ISD has evolved over the years from data systems to enterprise systems and in the future to what would be called global systems. The focus area for IT-systems has gradually widened up starting with systems for delimited business activities, moving to company wide solutions and then shifting to globally oriented measures.

5.1. Data Systems – Past Trend

The development work is focused on designing a set of well-specified data systems each supporting a certain type of business activity. These systems are interacting in a business context and the systems interfaces are therefore important to outline and specify.

The data systems could be implemented by tailor-made solutions or application packages. A productivity or cost-reducing motive lies behind the development work.

5.2. Enterprise Systems – Present Trend

Enterprise systems or ERP-systems are a recent trend since mid 1990ies with the aim of offering companies mega-based systems for their business operations. By enterprise systems we refer to large application packages that fully cover the provision of information required in a company. An important criterion is that the included parts or data systems are closely integrated with each other through a central database. A possible strategy for a company is to acquire best-of-breed solutions through selecting the most excellent parts from different enterprise systems on the market. An integration or coupling motive lies behind the development work with enterprise systems.

5.3. Global Systems – Future Trend

The challenge for the future is to design IT solutions for so called global systems. The overall purpose is to link or connect different companies' IT-systems to each other according to the value chain in a proper and effective manner. The global systems comprise solutions for the company and their suppliers, customers and business partners. Also a company group could have global systems for connecting IT-systems from their including business units. A transformation or enhanced value-creating motive lies behind the development work.

6. METHODOLOGY PACKAGE

Methodologies for ISD are launched as more or less packaged products. The first approaches were designed as formalised methodologies. Today we face a situation with a more flexible or softer use of methodologies. In the future more of a hands-on-practice will govern the use of methodologies for ISD.

6.1. Formalised Methodologies – Past Trend

The use of formalised methodologies builds on an engineering paradigm for ISD. The methodologies are precisely defined in coherent work steps and well-formulated description techniques for documentation. The development work is formalised in a planned or predictive manner. A requirements specification should reflect the user needs in a complete and consistent way. ISD is performed with a harmony perspective where actors are regarded to have common goals for the development work.

6.2. Soft Methodologies – Present Trend

The use of soft methodologies builds on a human-centred paradigm for ISD. Today we propose a more flexible application of methodologies taking into account the specific knowledge and prerequisites of the participating actors. The soft methodologies consist of both generic parts and situation specific constituents. These methodologies are extendable in character with a basic approach as a starting point and the possibility to use a set of advanced versions for special cases within ISD.

6.3. Hands-On Methodologies – Future Trend

The use of so called hands-on-methodologies builds on a rather pragmatic oriented paradigm for ISD. The leading idea behind this future trend is to use a combination of existing methodologies available on the market instead of innovating totally new ones. In this scenario it is useful to apply tool boxes of methodologies where you select suitable combinations adapted to specific development situations. These combinations can be worked out in different ways for example using the concepts of methodology chains (through the development cycle) or methodology alliances (across the same development phase).

7. ACTOR ORIENTATION

There are usually a lot of actors performing on the "stage" during the ISD process. The ISD trends have gone through a strong user orientation phase into recent ideas about customer orientation as a base for a more comprehensive view of stakeholder orientation.

7.1. User Orientation – Past Trend

From the ISD area we have for a long time learnt the lesson to proceed from user needs, requirements and terms during the development work. The simple argument is that there are the real users who in their daily work should live with the new IT-systems. They have the best knowledge of the business activities for creating efficient systems solutions. The principle of user orientation goes back to professor Börje Langefors' infological approach to the ISD field. The theory of infology states the significance of designing and operating IT-systems from a user point of view in order to achieve desired results in organisations.

7.2. Customer Orientation – Present Trend

From services marketing within the business administration area we have learnt the lesson to proceed from customer needs and demands during the development work. The motive behind this is that the customer of a specific service has the excellent knowledge of the business setting and is regarded as the "king" of the market arena. A customer orientation has also been influential for the ISD area during recent years. The customers behind an IT-system are identified and regarded as the main users. The ISD work delivers services for new IT-solutions with the aim to meet or satisfy the customers' expectations.

7.3. Stakeholder Orientation – Future Trend

Development work in organisations can be seen as a social field of forces between different interest groups or stakeholders. There usually exist communication gaps or misunderstandings when people from various interest groups try to deal with ISD issues. Therefore it is important to find constructive ways to bridge the communication gaps between key actors during the development process. One way to achieve this is to highlight and illuminate the needs and demands from each stakeholder's point of view. A future trend with a stakeholder orientation is a more general principle for ISD and in this sense includes the former ways of user and customer orientation for development work.

8. IT PERSPECTIVE

The perspective of the role of IT-systems is changing over the time. The original perspective has been to regard IT as a support for the business activities. The business world of today is looking at IT as a potential enabler for new market opportunities. In the future the role of IT will be more offensive and integrated with the business mission of a company or organisation.

8.1. IT as a Support – Past Trend

The focus of the development efforts in ISD has traditionally been on designing IT-solutions in order to support the various business operations in companies or firms. The IT-systems are regarded as resources in development work. Starting with the needs of the users, a business operation specification is made which provides both content-based and structural requirements on the IT-systems. IT is not an end in itself but rather a efficient tool to gain expected business effects!

8.2. IT as an Enabler – Present Trend

Since the mid 1990ies we have tried to devote a lot of efforts in ISD for designing information in our systems to create new business opportunities for the organisation, and hence strengthen the competitive edge on the market. The IT-systems are regarded as enablers for change. Here the focus is on the potential that a new IT-system represents for the company. The IT-system becomes an enabler for renewing the business. New technological innovations in multimedia, the Internet and electronic commerce become new value-adding enablers to the business of the firm. Instead of a detailed requirements specification we outline a scenario description for analysing the potentials of IT-systems.

8.3. IT as a Business Mission – Future Trend

The future trend will be to further stress that IT-systems are an essential part of the business mission itself for a firm. The mission statement for the company together with a description of management visions of the business operations will govern the development work towards more aggressive use of IT-systems. The perspective of viewing IT as a business mission unites the earlier ISD trends of looking at IT as a support and enabler for change. We need a combined or mixed strategy of supporting and enabling features when designing the IT-systems for professional use in new business mission contexts.

9. SUBJECT MATTER FOCUS

In the ISD area the focus of the subject matter under study has over the time evolved from a computer focus (past situation) to a organisation focus (present situation) and is now moving more to a people focus (future situation). This seems to be a natural progress or growth of the ISD area to comprise both technical, business and human conditions.

9.1. Computer Focus – Past Trend

In the beginning the IT-systems were developed with a computer focus in mind. Such an orientation is focused on an "inside out" way of working for ISD with the computer-based system as the core of interest. The development task is devoted to find out a nice technical solution that could, but not automatically, have good effects for the business activities. It is true that fundamental and strategic technical choices for a company put limitations and constraints for the freedom of action during the ISD process. The lesson learnt from the past trend with a computer focus is that technical considerations are necessary but not sufficient for a successful ISD work.

9.2. Organisation Focus – Present Trend

When developing IT-systems today an organisation focus is prevalent. Such an orientation is focused on the interplay between people and computers in appropriate business settings. The ISD work is concentrated on how to develop the work tasks for individuals and their communication pattern (social system) as well as to construct IT solutions in an efficient manner (technical system). This is done for creating desired business effects in an organisational context. The social interplay in the organisation is put in the foreground and the technology is subordinated to the needs of human beings. The present situation with an organisation focus for ISD is a result of a long tradition of research in socio-technical design.

9.3. People Focus – Future Trend

The future development of IT-systems will take into account more thoroughly a people focus. Such an orientation highlights the persons or individuals as the main focus for the ISD work. ISD will be regarded as a successive formalisation of the people's needs and desires of a new IT-system. The dialogue between different actors, who are affected by the changes, will have a crucial impact on the degree of success of ISD. The dialogue is facilitated if we during the ISD work pay more attention to how we specify and articulate the different professional languages in practical use by the various actors. A people focus puts the human-centred interest of ISD in the foreground with the motive for a better anchorage of and deeper commitment to the IT-systems in organsiations.

10. DEVELOPMENT STRATEGY

By a development strategy we mean the way of implementing and launching IT-systems in organisations. Traditionally a strategy for a full-scale development was the dominating paradigm. Gradually a new strategy for rapid development with a prototyping approach emerged. The future situation will be characterised by more of a strategy for evolutionary development.

10.1. Full-Scale Development – Past Trend

The idea behind the strategy of full-scale development is to deliver and implement the whole IT-system simultaneously in one turn. It is more or less a "big bang" implementation! Characteristic for this situation is a revolutionary approach where the old IT-system is totally discarded at a sharp moment where a wholly new IT solution opens up for use. In other words the new IT-system is implemented in a full scale. This fact implies that our requirements specification needs to be complete and "frozen" before the implementation may begin. In a full scale development we rely on the principle that it is better to carry out large investments in IT-systems through a radical change in the organisation.

10.2. Rapid Development – Present Trend

The idea behind rapid development is the benefit to launch IT-systems rather quick in the organisation in order to gain immediate results for the business. Rapid development is often realised through using a prototyping approach. By making rapid prototypes of a desired future business situation in reality or daily life, the various actors or interest groups have the possibility to discover the effects of introducing new IT-systems. They can react to the prototype solutions and give valuable feedback for further specification of the system. A prototype gives a concrete picture of a business solution and implies a rich learning environment for the actors. In rapid development with prototyping there is interplay between specification and implementation, sometimes in several rounds.

10.3. Evolutionary Development – Future Trend

The idea behind a strategy for evolutionary development is to implement a new IT-system in minor parts which are distributed over a certain period of time. Characteristic for this situation is that the IT-system will be delivered step by step in smaller turns. The launching of the new IT-system is therefore performed in a number of manageable steps. It is regarded as a safer strategy to have a successive renewal of the business operations than to dramatically change the whole organisation. The border lines between development and maintenance would be more or less erased. In an evolutionary development we rely on the principle that it would be better to have a continuous improvement of the business instead of more risky "big bang" solutions.

11. RANGE OF WORK

In the ISD area the range of work has expanded over the years. The development work is in this respect oriented to three various kinds of focus areas labelled systems work, change work and business promotion work.

11.1. Systems Work – Past Trend

Information systems or nowadays IT-systems have been a natural construct to start from when establishing the practice of systems work. The construct of "systems work" was preferred in order to illuminate both systems development and maintenance management. The ISD field is traditionally defined in a rather broad sense to also include relevant issues of software maintenance. A significant feature for setting out a genuine practice of systems work has been the innovative research of useful methodologies for ISD. The profession of ISD has over the years shown to be heavily dependant on practically oriented methodologies such as the ISAC, YSM, SADT, EAR, NIAM, JSD, IE, SSM, UML and RUP approaches for systems work.

11.2. Change Work – Present Trend

The ISD practice has gradually evolved to regard systems work in a larger context as a fraction of more comprehensive change work in today's companies. Development of new IT-systems leads to a major change for affected people and their business operations. Change work implies a purposeful growth and development of organisations and means that we are advancing the business towards some concrete visions or goals. Changes of different kinds should be a starting-point for discussions between different actors in business development. In the ISD field we have learnt the significance to start up development work from a change analysis (cf. the ISAC approach) which builds a platform for further development of e.g. IT-systems.

11.3. Business Promotion Work – Future Trend

The next challenge for the future is to focus the change work in organisations to create IT-systems for really giving improved and sustainable business results. There are many different types of change programs in practice spanning over approaches from radical changes (e.g. BPR) to incremental changes (e.g. TQM). The most likely evolvement for the ISD area is to have approaches with the aim of trying to make business improvements in distinct and manageable steps. The size of changes required in business operations will be dependant on the specific situation. With this kind of change approach we strive for promoting the business on a regular basis in order to achieve our visions and goals. Such a business promotion work will give the potentials for a more positive change attitude together with a constructive development process.

12. CONCLUDING REMARKS ON ISD TRENDS

My main conclusion of the research is that the ISD discipline is going in a right direction towards great challenges in the new economy and for future business potentials. There are some interesting observations from the accomplished study of ISD trends that are worthy to highlight:

- The focus areas for the specific dimensions could remain and be recurring over the time as a general pattern for evolution of the ISD field – learn from history!
- There are some focus areas for the same time perspective but from different dimensions that are concurrent and interacting leading to an extra driving force.
- This review shows that the ISD discipline has more and more broaden up the study field to include various issues of engineering, business and human aspects.

- ISD has as an academic field evolved from regarding IT systems as an isolated phenomenon to a comprehensive/holistic thinking of IT investments in firms.
- It seems that the research view on ISD has changed from efficiency studies (cost focus) to inquiries about effectiveness (benefit focus) of IT systems.

These concluding remarks on ISD trends show the necessity for emphasising even more the need for strong multidisciplinary research on information systems development. Information Systems is a real "relationship" subject trying to integrate knowledge from behavioural science, computer science and business administration. The mission is to investigate how people develop and use IT solutions to support and improve activities in their organisations and social life (cf. Lundeberg et al., 1995; Nilsson and Pettersson, 2001). We will in this sense end up our study by referring to a well-known formula for performing success in business (cf. Likert, 1961, p. 212) by applying it to the ISD area:

$$\text{Degree of success in ISD} = \text{f (Quality} \times \text{Acceptance} \times \text{Value)}$$

The success formula states that to attain a successful result, we must have both sufficient quality in the designed solutions (i.e. the IT-systems) and a good acceptance among the actors (i.e. users) to give them a motivation for using the solution as well as that the designed solutions should create a business value to the ultimate beneficiaries (i.e. the customers to the company). A low figure in either quality, acceptance or value will lead to an unsuccessful result – hence the multiplication signs in the formula.

REFERENCES

Avison, D. E., and Fitzgerald, G., 2003, *Information Systems Development: Methodologies, Techniques and Tools*, 3rd Edition, McGraw-Hill, London.

Burnett, R., Brunstrom, A., and Nilsson, A. G., eds., 2003, *Perspectives on Multimedia: Communication, Media and Information Technology*, Wiley, London.

Davis, G. B., 2003, Building an international academic discipline in Information Systems, in: *Exploring Patterns in Information Management: Concepts and Perspectives for Understanding IT-Related Change, Festschrift in honour of Mats Lundeberg's 60th birthday*, B. Sundgren, P. Mårtensson, M. Mähring, and K. Nilsson, eds., EFI, Stockholm School of Economics, Stockholm, chapter 16, pp. 273–290; electronic version: http://www.hhs.se/im/exploringpatterns.

Iivari, J., and Lyytinen, K., 1998, Research on information systems development in Scandinavia – unity in plurality, *Scandinavian Journal of Information Systems* **10**:135–186.

Langefors, B., 1973, *Theoretical Analysis of Information Systems (THAIS)*, Auerbach, Philadelphia and Studentlitteratur, Lund.

Langefors, B., 1995, *Essays on Infology: Summing Up and Planning for the Future*, Studentlitteratur, Lund.

Likert, R., 1961, *New Patterns of Management*, McGraw-Hill, New York.

Lundeberg, M., Goldkuhl, G., and Nilsson, A. G., 1981, *Information Systems Development: A Systematic Approach*, Prentice Hall, Englewood Cliffs, and New Jersey.

Lundeberg, M., Mårtensson, P., Sannes, R., and Sundgren, B., 1995, Information Management as a field, in: *The Infological Equation: Essays in Honor of Börje Langefors*, B. Dahlbom, ed., Gothenburg Studies in Information Systems, Göteborg University, Gothenburg, pp. 195–209.

Nilsson, A. G., 1990, Information systems development in an application package environment, in: *ISD'1990, Proceedings of The Second International Conference on Information Systems Developers Workbench, Gdansk, Poland, 25–28 September, 1990*, S. Wrycza, ed., University of Gdansk, pp. 444–466.

Nilsson, A. G., 1995, Evolution of methodologies for information systems work: a historical perspective, in: *The Infological Equation: Essays in Honor of Börje Langefors*, B. Dahlbom, ed., Gothenburg Studies in Information Systems, Göteborg University, Gothenburg, pp. 251–285, also in: *ISD'1996, Proceedings of The*

Fifth International Conference on Information Systems Development, Gdansk, Poland, 24–26 September, 1996, S. Wrycza, and J. Zupancic, eds., University of Gdansk, pp. 91–119.

Nilsson, A. G., Tolis, C., and Nellborn, C., eds., 1999, *Perspectives on Business Modelling: Understanding and Changing Organisations*, Springer, Berlin.

Nilsson, A. G., and Pettersson, J. S., eds., 2001, *On Methods for Systems Development in Professional Organisations: The Karlstad University Approach to Information Systems and its Role in Society*, Studentlitteratur, Lund.

Nilsson, A. G., 2003, Change work in organisations: some lessons learned from information systems development, in: *Exploring Patterns in Information Management: Concepts and Perspectives for Understanding IT-Related Change, Festschrift in Honour of Mats Lundeberg's 60th birthday*, B. Sundgren, P. Mårtensson, M. Mähring, and K. Nilsson, eds., EFI, Stockholm School of Economics, Stockholm, chapter 6, pp. 83–99; electronic version: http://www.hhs.se/im/exploringpatterns.

CARS OF THE FUTURE: COMPUTERS AT THE WHEEL

Michel Parent*

We are now observing the very rapid arrival of computer technologies in modern cars with all kinds of functions, which are now handled by computer systems. Nowadays, computer technology can account for close to 30% of the cost of a vehicle. However, one has to realise that computers are now taking over more and more of the driving chores of the driver with numerous assistance such as ABS, ESP, Steering assistance, Parking assistance, ACC, Emergency brake assistance, Lateral guidance, ...

We are now close to hand the driving task fully to computers and the first vehicles with fully automated driving are now on the road in limited applications. To generalise this approach, one has to find an acceptable path to this future. One of the biggest hurdles is the development, certification and maintenance of this complex software. Technologies derived from the aerospace industry are now finding their ways in the automotive industry to reach this goal.

* The French National Institute for Research in Computer Science and Control, BP 105, F78153 Le Chesnay, France, Michel.Parent@inria.fr.

INTEGRATING ENTERPRISE AND IS DEVELOPMENT USING A MODEL DRIVEN APPROACH

John Krogstie*

1. INTRODUCTION

Going back to the beginning of the eighties, there has been numerous accounts of model-generated information systems being touted as the silver bullet to attack problems regarding software productivity and quality (Balzer, 1985). The CASE-tools of the late eighties were oversold on their 'complete' code-generation capabilities. Today similar arguments are found relative to the use of OMG's Model-driven Architecture (MDA) approach (Thomas, 2004), using and integrating UML models on computational independent, platform independent, and platform specific models. Also a number of other areas, including enterprise modeling, workflow modeling, ontologies, and Service Oriented computing are proposed by their proponents as the direction to go to achieve model-generated solutions for tomorrows business applications.

In the newly started EU IST Integrated Project ATHENA (Athena, 2004), we are investigating all of these approaches in parallel, specifically to address problems of business and system interoperability across organizational borders. This paper will outline the different approaches, and illustrating how they can be combined to support integrated and parallel enterprise and IS development, within and across enterprise boundaries, by supporting the development and evolution of appropriate models of high quality

The rest of the paper is organized as follows. Section 2 overviews briefly the different modeling approaches. Section 3 gives an overview of the ATHENA project, whereas Section 4 presents aspects of how to integrate enterprise and IS development by combining these different model-driven approaches. This is not finalized in the project, thus the views on this that is expressed are those of the author, not of the Athena project. Section 5 concludes the paper and describes shortly work ahead on this approach.

* IDI, NTNU and SINTEF, Forskningsveien 1, N-0314 Oslo, Norway, John.Krogstie@sintef.no.

2. BACKGROUND ON MODEL-DRIVEN DEVELOPMENT

In the context of model-driven enterprise and system development, we refer to models developed in languages which have the following characteristics:

- The languages are diagrammatic, with a limited vocabulary (states, processes etc).
- The languages utilize powerful abstraction mechanisms.
- The languages have a formal syntax. The semantics is either operational enabling e.g. generation of other models including executable programs or mathematical enabling advanced analyses.
- The languages are meant to have general applicability across problem domains.

Although most software developers are aware of model-driven methodologies they are seldom followed in great detail in practice, and mostly only in initial development stages. What differentiates model-driven development from using models in this fashion are:

- Models are utilized throughout development, use, and evolution of the systems.
- Models are made both of the system, and of the environment of the system.

To approach this problem area one need to attack and integrate aspects of five different research traditions: Enterprise Modeling (EM), workflow modeling (WM), Ontologies, Service-oriented Architecture (SOA), and Model-Driven Architecture (MDA).

2.1. Enterprise Modeling (EM) and Workflow Modeling (WM)

Enterprise modeling (Fox and Grüninger, 1998) is useful for externalizing, making and sharing enterprise knowledge, which is again vital to support enterprise systems evolution. A crucial problem for the successful evolution of enterprise systems (and thereby of enterprises) is that management have a limited understanding of their own business processes (Dalal et al., 2004), and it is argued that this can be helped by making processes explicit in models. For the support of knowledge workers, who need more flexible support than what can be given by traditional workflow systems, it is important that emergent and interactive work processes can be captured and supported (Jørgensen, 2001; Krogstie and Jørgensen, 2004). The most comprehensive theoretical approach to this field is Peter Wegner's interaction framework (Wegner, 1997; Wegner and Goldin, 1999). The primary characteristic of an *interaction machine* is that it can pose questions to users during its computation. The process can be a multi-step conversation between the user and the machine, each being able to take the initiative. The importance of EM is also evident in the increasing interest for *enterprise architecture* (e.g., Pereira and Sousa, 2004), which has revitalized past research on information systems architecture (Zachman, 1987).

2.2. Ontologies

Though various related ontologies exist (e.g. TOVE (Fox, 1992)), their suitability for application in industry is quite limited. Enterprise ontologies have been proposed as a way of solving the communication problems arising from different interpretative frameworks in different organizations). For many problem areas, this is too limiting, and one has to involve dynamic ontologies (Kahng, 1998) as interactive models as descry bed above.

In general, there is a need for the development of methods and tools for enterprise ontology management, with a focus on supporting enterprise knowledge integration and interoperability for enterprises and software applications. Enterprise knowledge can be divided in two main categories. The first category comprises the knowledge represented by all sorts of documents: from technical reports to emails, from scientific papers to circuit blueprints; this category is mainly conceived to be used by human users. A second category is represented by formal symbolic knowledge, e.g., modeled by using semantic nets or description logics, stored in way that is mainly organized to be exploited by a computer.

Semantic annotation is based on a reference ontology that is used to construct annotation expressions, aimed at giving a clear, agreed, unambiguous meaning to documental (human oriented) knowledge or to enterprise software elements used in co-operative processes. Semantic annotation can be used to associate a formal meaning to enterprise models and in particular to information structures and Business Processes in order to realize a semantic interoperability platform.

Semantic annotation is one of the key enabling technologies to implement a number of solutions for achieving interoperability, e.g., from service discovery to matchmaking, to the linking of Business Processes to computable services.

2.3. Service-Oriented Architecture

Service-Oriented Architectures are being viewed as the next wave of technology to impact business computing by enabling the utilization of distributed components by allowing software vendors to provide not only applications to the market but a suite of services that can be utilized by a wider audience and charged for through an access or usage business model.

Many projects are concerned with the development of service-oriented solutions that can be more easily planned and then later customized when they are being deployed. This is intended to provide better industry focused solutions that can be adapted better for deployment into client environments. This is to directly counter the problem of high costs of customization and integration of highly generic industry solutions.

Research challenges in this field are

- to develop modeling and specification systems that accurately express services and service-oriented architectures and particularly assist in the planning of solutions and the marking of intended customizations for the better deployment of a solution into a wide range of client environments;
- to develop technologies that enable the easier composition of services and the brokering, mediation and ultimately the negotiation of pre-specified but customizable services;
- to develop an execution framework for planned and customizable service-oriented architectures

With respect to supporting dynamically networked organizations, most B2B E-business frameworks including ebXML and BPML (Shim et al., 2000) focus on information exchange and business transactions. They lack support for the dynamic and knowledge-intensive parts of inter-organizational processes, which is address in more detail within enterprise and workflow modeling.

2.4. Model-Driven Architecture (MDA), Based on Design Modeling in UML

Model-driven development is concerned with using the appropriate set of models and modeling techniques, supported by the appropriate tools, to provide sufficient help for reasoning about our systems. Model-driven architecture (MDA^{tm}) has become OMG's notion of doing model-driven development, and has gained a lot of interest recently (Miller et al., 2003). The main idea in MDA is to describe systems in a platform-independent manner in what is called a platform-independent model (PIM) in the MDA (Exertier et al., 2004). From this model, transformations (or mappings) to different platform-specific models (PSM) are defined. A third level, CIM (computational independent model) has recently been introduced, which is meant to be a way to integrate enterprise models with system models. While still a new approach, MDA has begun being used in some practical case studies in industry, even on the enterprise level (Günther and Steenbergen, 2004).

3. OVERVIEW OF ATHENA

Athena is an Integrated Project Sponsored by the European Commission (Athena, 2004). Its objective is to be the most comprehensive and systematic European research initiative in the field of enterprise application interoperability, removing barriers to the exchange of information in and among organizations. It will perform research and apply results in numerous industrial sectors, cultivating and promoting the "networked" business culture. Research and development work will be carried out hand in hand with activities conceived to give sustainability and community relevance to the work done. Research will be guided by business requirements defined by a broad range of industrial sectors and integrated into Piloting and Training. Athena will be a source of technical innovations leading to prototypes, technical specifications, guidelines and best practices, trailblazing new knowledge in this field. Athena will mobilize a critical mass of interoperability stakeholders and lay the foundation for a permanent, world-class hub for interoperability.

The Athena consortium is made up of 19 leading independent research centers, working together as partners in a single venture. The project has an initial estimated duration of 3 years and has an overall horizon of 5 years. Its expected budget is approximately 26, 6 million Euros of which 14, 4 million Euros will be funded by the European Commission.

4. INTEGRATING ENTERPRISE AND IS DEVELOPMENT

The different approaches to model-driven development described in Section 2 are appropriate for supporting different types of processes, from very static, to very dynamic, even emergent processes. The different process types decide the extent to which the underlying technology can be based on hard-coded, predefined, evolving or implicit process models. This gives a number of development approaches as illustrated in Figure 1, on one extreme; systems are manually coded on top of a traditional runtime environment, and on the other enterprise models are used directly to generate solutions. An example of the latter

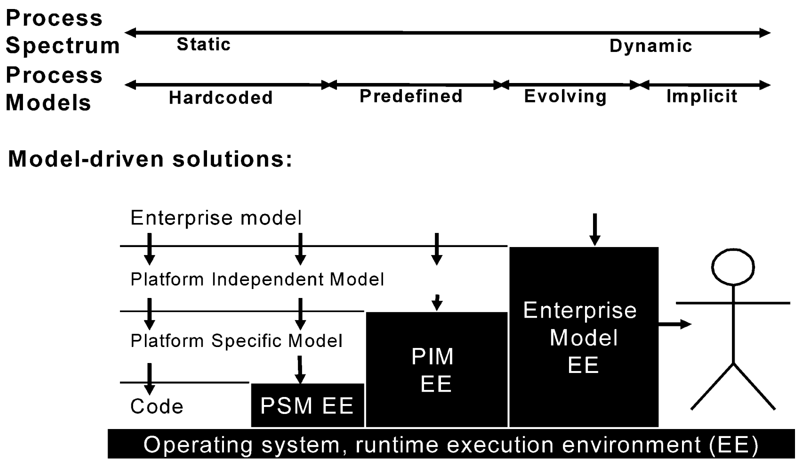


Figure 1. Overview of different execution environment for different process models.

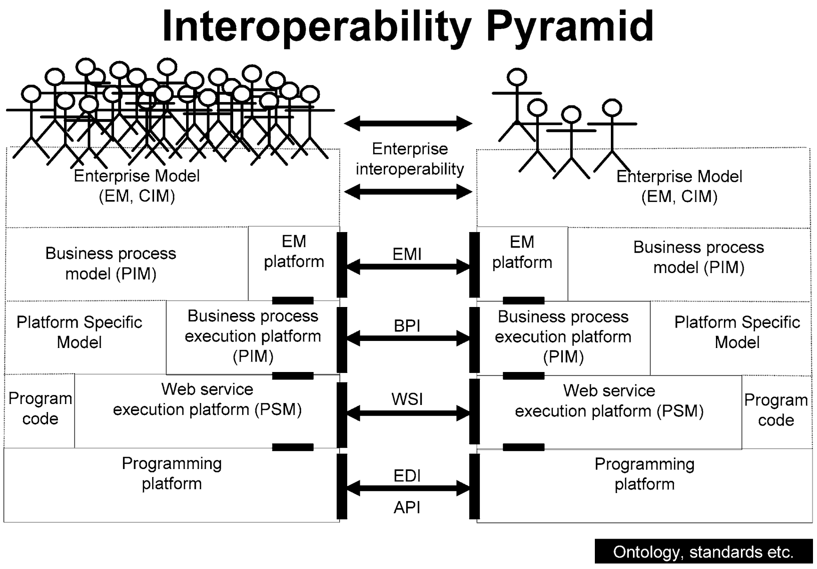

Interoperability Pyramid
Enterprise interoperability
Enterprise Model (EM, CIM)
Business process model (PIM)
EM platform
EMI
Platform Specific Model
Business process execution platform (PIM)
BPI
Program code
Web service execution platform (PSM)
WSI
Programming platform
EDI
API
Ontology, standards etc.

Figure 2. Interoperability between different platforms.

is found later in this section. In between these, we have the approaches typically described in MDA, namely the development of PIMs for code-generation (e.g. on top of a UML Virtual Machine), or for PSMs for more traditional code-generation.

In Figure 2, we outline the different types of interoperability-possibilities across this pyramid. Whereas traditional systems use special APIs and approaches such as EDI for interchange of data, on the next level (PSM), we can identify Web Services Interfaces. Above this level, there is a lot of work being performed on specific business process exe-

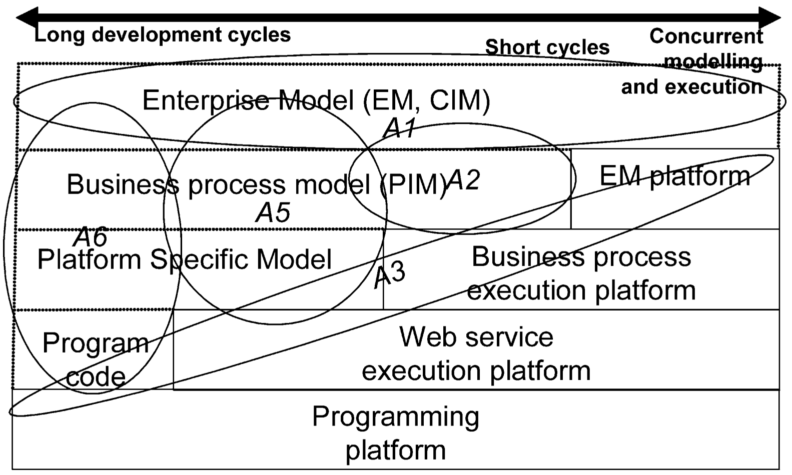

Long development cycles
Short cycles
Concurrent modelling and execution
Enterprise Model (EM, CIM)
A1
EM platform
Business process model (PIM)
A2
A5
A6
Platform Specific Model
A3
Business process execution platform
Program code
Web service execution platform
Programming platform

Figure 3. How modeling approaches in different ATHENA projects cover the framework.

cution platform, with a possibility to exchange directly using a BPI. Finally, projects such as EXTERNAL and UEML (see below) has provided solutions for how to interoperate on the enterprise model level, potentially using different modeling languages and different tools in the process. Standards and ontologies can be used across all levels, and also between levels to make the interoperation happen more smoothly. Finally, Figure 3 illustrates the different research projects with the ATHENA IP, and what modeling techniques they are pursuing to fulfill this picture.

- In A1: Collaborative enterprise modeling, the focus is as the name says on enterprise modeling, but then on models across the spectrum, from models used in system development with a long development cycle (e.g. in connection to development of an ERP system), shorter cycles, and even supporting concurrent modeling and execution using the interactive model approach.
- A2 Cross-organizational business processes focuses on more formal process modeling on top of a business process execution platform, to enable relatively short development cycles.
- A5 Planned and Customized Service Oriented Architecture looks upon models on both the PIM and PSM level to run on top of a web-service execution platform
- A6 Model driven and adaptive Interoperability Architecture is specifically focused on the MDA approach, modeling on all levels, to support a more general system development process when this is appropriate
- A3 Knowledge support and knowledge meditation finally is meant as indicated above to support the interoperability between the different levels and organizations using ontology-oriented techniques

We will try to clarify this picture with an example which originally had focus on the enterprise level and cross-organizational business processes. The infrastructure to support networked organizations developed in the EXTERNAL project (which is one basis technology for the technology for this to be developed in ATHENA) can be described as consisting of three layers. These layers are identified as:

- Layer 1, the *information and communication technology* (ICT) layer: – defining and describing the execution platform, software architectures, tools, software components, connectivity and communication (as outlined in Figure 1).
- Layer 2, the *knowledge representation* layer: – defining and describing constructs and mechanisms for modeling (as outlined in Figure 3).
- Layer 3, the *work performance and management* layer; – modeling and implementing customer solutions, generating work environments as personalized and context-sensitive user interfaces available through portals.

4.1. The ICT Layer

The ICT-infrastructure is an integration of the enterprise and process modeling tools that was brought into the EXTERNAL project by the partners:

- METIS (Lillehagen, 1999), a general purpose enterprise modeling and visualization tool,
- XCHIPS (Haake and Wang, 1997), a cooperative hypermedia tool integrated with process support and synchronous collaboration,
- SimVision (previously Vite) (Kuntz et al., 1998), a project simulator used to analyze resource allocation, highlighting potential sources of delay and backlogs.
- WORKWARE (Jørgensen, 2001; Jørgensen 2004) a web-based emergent workflow management system with to-do-lists, document sharing, process enactment and awareness mechanisms.
- FrameSolutions (Kallåk et al., 1998), a commercially available framework for building automated workflow applications.

Figure 4 depicts the technical infrastructure. The architecture has 3-tiers, clients, application servers, and data servers. The implementation is web-based, utilizing HTTP both for control and data integration, and exchanging data with XML format. The integration work in EXTERNAL proceeded in three steps.

1. Data-centered integration: based on a common EXTERNAL XML DTD, XML importing/exporting utilities are implemented in each of the enterprise modeling tools for data exchange between the tools or between an XML repository and the tools.
2. Control-centered integration: this is done by using the APIs provided by the tools and the repository to be integrated. With the APIs, the tools can call each other and access the shared repository. Some of the APIs may have parameters for denoting content objects and the implementation of them requires the data-centered integration capability as developed in step one.
3. Worktop-based integration: this is a service-oriented integration at the user-interface level which makes use of both data-centered integration and control-centered integration methods to access shared models, information objects, and to invoke individual tools.

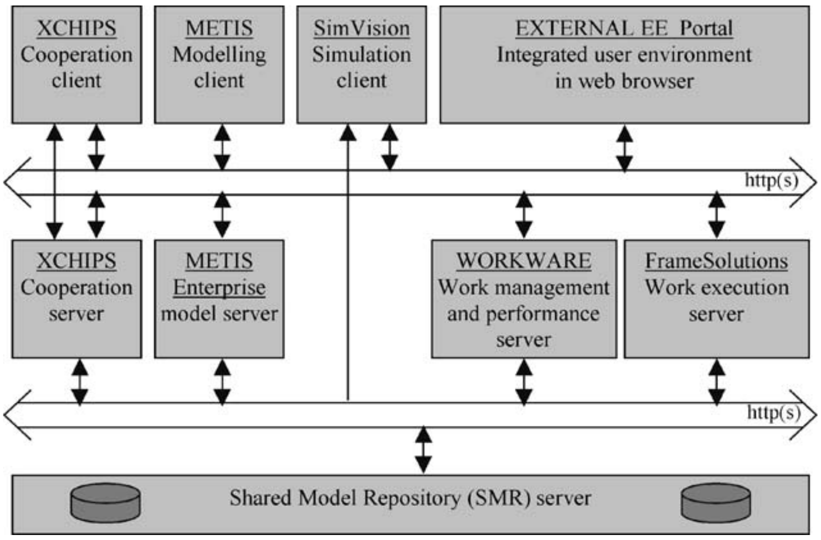


Figure 4. The architecture and components of the EXTERNAL infrastructure, ICT layer.

New components can be developed on as needed basis, using the approaches of service oriented architecture or model-driven architecture. This was not the focus of the EXTERNAL project thought. Thus, although it is not in its present form an example of a full-fledged service-oriented architecture approach, this can be developed on this basis.

4.2. The Knowledge Representation Layer

The knowledge representation layer defines how models, meta-models and meta-data are represented, used and managed. A version of Action Port Modeling (APM) (Carlsen, 1998; Jørgensen, 2004) constitutes the core of EXTERNAL's modeling language (EEML). The kernel concepts are shown in Figure 5 as a simplified logical meta-model of EEML. The process logic is mainly expressed through nested structures of *tasks* and *decision points*. The sequencing of the tasks is expressed by the *flow* relation. *Roles* are used to connect resources of various kinds (people, organizations, information, and tools) to the tasks. Hence, modeling the networked organization in EEML results in models that capture an extensive set of relationships between the goals, organizations, people, processes and resources. This is particularly useful considering the dynamic nature of networked organizations. For new partners joining the network, the rich enterprise models provide a valuable source of knowledge on how to "behave" in the network.

Moreover, the interactive nature of the models, meaning that the users are free to refine them during execution, increases their potential as sources of experience and knowledge. As such they *document* details on how the work was actually done, not only how it was once planned. EEML can be extended to link into for formal process modeling (E.g. BPMN) as well as languages used in the SOA and MDA approaches, particularly

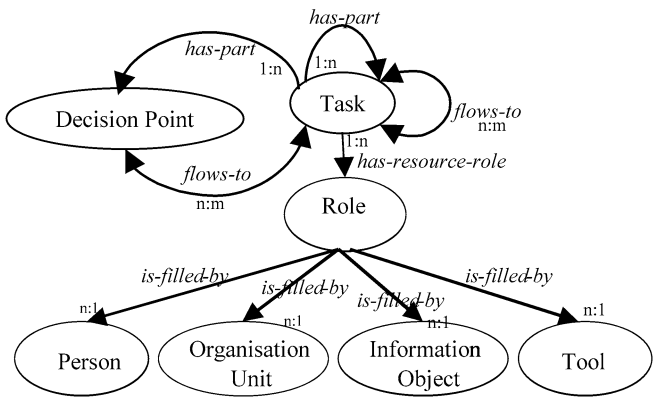


Figure 5. Simplified meta-model of EEML.

UML, through the meta-modeling features of METIS, which is the main implementation platform for the visual language.

From a knowledge management perspective, process models are carriers of process knowledge; knowledge of how to do things. But through the possibility in EEML of attaching information resources to the tasks at any level, such a model also imposes a structure upon the set of information resources relevant for the work described by the process model. That way, the process models themselves form the basis for information management. Further extensions with semantic annotations applying results from the ontology-oriented project A3 would enhance this even further.

4.3. The Work Performance and Management Layer

Users access their solutions through project portals. A project portal for a networked organization must have support for methodology adaptation and for communication, coordination and collaboration in teams. Project work management, reporting and other services must be offered, and finally project work must be performed with possibilities for repetition, providing security and privacy for knowledge workers.

In the EXTERNAL infrastructure, the web-based portal registers and qualifies users, and invokes other tools through WORKWARE. The modeled tasks are also executed through the invocation of tools and applications from the web based user environment comprised of the portal and WORKWARE. WORKWARE sets up the context for each task, giving access to the knowledge and resources needed to perform the task. The actual work performance is done by invoking appropriate services. The task performers may access desktop tools, organizational information systems, web services, or automated processes (in FrameSolutions) through this user environment.

User environments are generated dynamically based on the definition of tasks using EEML. Forms and components for interacting with different model objects are selected and composed based on user interface policies. These policies are also modeled objects. This enables user interface customization and personalization.

The dynamically generated work management interface includes services for work performance, but also for process modeling and meta-modeling. The *worktop* is the main component in this interface. Each task has its own worktop. In addition to the services for performing and managing the task, it contains links to all knowledge in the process models that is relevant for the task. Since the worktop is dynamically generated, subject to personal preferences, the skill levels of task performers can be taken into account, e.g. to provide more detailed guidelines for people who have not previously worked on such tasks. Similarly, customized worktops for project management can support the project management team. The contents may include an overview of the project, adopted management principles, applicable methodologies, project work-break-down structure, results, plans and tasks, technologies and resources, status reporting and calculations. Further details on this approach can be found in (Jørgensen, 2004; Krogstie and Jørgensen, 2004).

5. CONCLUSIONS

For systems and enterprises to evolve in a coordinated manner, there is a need for representing knowledge in a way understandable for both business users and system analysts. Modeling has been touted as an appropriate way of providing the necessary abstraction mechanism to comprehend and analyze complex problems in this regard, but it appears that no modeling technique or approach is applicable across the whole spectrum of process and stakeholder types. This paper has outlined early work on how results from modeling approaches from the fields of Enterprise Modeling, Workflow modeling, Ontologies, Service-oriented Architecture, and Model-Driven Architecture can be combined to provide a more complete coverage of the overall problem area.

In the ATHENA integrated project, we will over the next years work on these problems both standalone and combined, testing solutions out on industrial cases, with a specific focus on achieving both system and business interoperability.

ACKNOWLEDGEMENTS

The material presented in this paper is based on co-operation with a number of partners within the ATHENA-project generally, and at SINTEF specifically. Most notably, Håvard Dingstad Jørgensen has been instrumental in developing the thinking for integrating Enterprise and systems modeling presented in Section 4.

REFERENCES

ATHENA 2004, (October 15, 2004); http://www.athena-ip.org.

Balzer, R., 1985, A 15 year perspective on automatic programming, *IEEE Transactions on Software Engineering* Vol. 11, No. 11 November.

Carlsen, S., 1998, Action port model: A mixed paradigm conceptual workflow modeling language, *Proceedings of Third IFCIS Conference on Cooperative Information Systems (CoopIS'98)*, New York.

Dalal, N. P., Kamath, M., Kolarik, W. J., and Sivaraman, E., 2004, Toward an integrated framework for modeling enterprise processes, *Communications of the ACM* **47**(3):83–87, March 2004.

Exertier, D., et al., 2004, PIM definition and description, *Proc. First European Workshop on Model Driven Architecture with Emphasis on Industrial Application*, Univ. Twente, Netherlands, March 17–18.

Fox, M. S., 1992, The TOVE project: A common-sense model of the enterprise, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*.

Fox, M. S., and Grüninger, M., 1998, Enterprise modeling, *AI Magazine*, AAAI Press, Fall, pp. 109–121.

Günther, J., and Steenbergen, C., 2004, Application of MDA for the development of the DATOS billing and customer care system (Case study on the use of MDA for the development of a larger J2EE System), *Proc. First European Workshop on Model Driven Architecture with Emphasis on Industrial Application,* Univ. Twente, Netherlands, March 17–18.

Jørgensen, H. D., 2001, Interaction as a framework for flexible workflow modeling, *Proc International ACM SIGGROUP Conference on Supporting Group Work*, Boulder CO, September.

Jørgensen, H. D., 2004, *Interactive Process Models*, PhD-thesis, NTNU, Trondheim, Norway, ISBN 82-471-6203-2.

Haake, J. M., and Wang, W., 1997, Flexible support for business processes: Extending cooperative hypermedia with process support, *Proceedings of GROUP '97*, Phoenix, Arizona USA.

Kahng, J., and McLeod, D., 1998, Dynamic classificational ontologies: Mediation of information sharing in cooperative federated database systems, in: *Cooperative Information Systems: Trends and Directions*, M. Papazoglou and G. Schlageter, eds., Academic Press.

Krogstie, J., and Jørgensen, H. D., 2004, Interactive models for supporting networked organizations, *Proceedings of CAiSE'2004*, June 9–11, Latvia, Riga.

Kallåk, B. H., Pettersen, T. B., and Ressem, J. E., 1998, Object-oriented workflow management: Intelligence, flexibility, and real support for business processes, *Proceedings of OOPSLA Workshop on Implementation and Application of Object-Oriented Workflow Management Systems*, Vancouver, Canada.

Kuntz, J. C., Christiansen, T. R., Cohen, G. P., Jin, Y., and Levitt, R. E., 1998, The virtual design team: A computational simulation model of project organizations, *Communications of the ACM*, vol. 41, no. 11.

Lillehagen, F., 1999, Visual extended enterprise engineering embedding knowledge management, systems engineering and work execution, *Proceedings of IEMC '99, IFIP International Enterprise Modelling Conference*, Verdal, Norway.

Miller, G., et al., 2003, Model driven architecture: how far have we come, how far can we go? (Panel at OOPSLA'03, Anaheim CA, October 2003).

Pereira, C. M., and Sousa, P., 2004, A method to define an enterprise architecture using the Zachman framework, *Proc. ACM SAC'04*, Nicosia, Cyprus, March 14–17, 2004.

Shim, S. S. Y., Pendyala, V. S., Sundaram, M., and Gao, J. Z., 2000, Business-to-Business e-Commerce frameworks, *IEEE Computer*, vol. 33, no. 10.

Thomas, D., 2004, MDA: revenge of the modelers or UML utopia? *IEEE Software* May/June 2004, pp. 15–17.

Uschold, M., et al., 1998, The enterprise ontology. In *The Knowledge Engineering Review*, vol 13.

Wegner, P., 1997, Why interaction is more powerful than algorithms, *Communications of the ACM*, vol. 40, no. 5.

Wegner, P., and Goldin, D., 1999, Interaction as a Framework for modeling, in: *Conceptual Modeling. Current Issues and Future Directions*, LNCS 1565, Springer-Verlag.

Zachman, J. A., 1987, A framework for information systems architecture, *IBM Systems Journal* **26**(3):276–292.

ISSUES IN INFORMATION SYSTEMS EDUCATION: CAPITALIZING ON RECENT ADVANCES IN LEARNING THEORY

Geoffrey Black, Wita Wojtkowski, W. Gregory Wojtkowski, and Cristianne Lane*

1. INTRODUCTION

The more we learn about how the brain acquires and stores information, the more efficient and effective we can be in our quest to pass on information to our students. As educators, we are in the business of transmitting information. We can be well-intentioned in that endeavour, but that is not enough. We must be well informed about learning theory and instructional practices. As we increase our knowledge about how information is best received, our teaching practices will evolve in order to match this new knowledge (McCray et al., 2003). We can then more powerfully engage students in the learning process and, most importantly, increase the probability that the acquired information is retained.

This paper examines two critical areas of learning theory. The first area concerns what current research in cognitive neuroscience tells us about how the brain receives, processes, and retains information. We know, for example, that our brains can better assimilate information that is organized and familiar (Yin, et al., 2004). In the same vein, research indicates that linking new information to previous knowledge helps give the new information a "home" (Nicoll, 2001) and that personally relevant information is more meaningful and more likely to be organized, stored, and available to be applied (Georghiades, 2004). Finally, we can better construct meaning when we are able to interact with information (Bransford, et al., 2000).

The second area of learning theory addressed here is how to make the transfer of information more 'brain friendly.' We explore some of the many instructional techniques that can access the types of learning addressed by each of the key findings of brain re-

* Geoffrey Black, Dept. of Economics, Boise State University, Boise, ID 83725-1620, USA, gblack@boisestate.edu. Wita Wojtkowski and W. Gregory Wojtkowski, Dept. of Networking, Operations & Information Systems, Boise State University, Boise, Idaho, 83725-1615, USA, wwojtkow@boisestate.edu, gwojtkow@boisestate.edu. Cristianne Lane, Lee Pesky Learning Center, 345 Bobwhite Court #220, Boise, Idaho, 83706, USA, clane@lplearningcenter.org.

search. We focus on a family of techniques that speak most directly to information systems (IS) instruction – the use of graphic organizers and concept maps. These provide visual representations of concepts and how they are related. They generate a framework for organizing conceptual understanding, linking background knowledge to new information, and facilitating student interaction.

This paper is organized as follows. Section 2 reviews some recent findings about how learning occurs and is facilitated. Section 3 examines how these findings can be used as a lens to ensure that our efforts to transmit information are effective. We conclude the paper reflecting on the issues discussed.

2. NEUROSCIENCE AND LEARNING

Research in cognitive neuroscience is beginning to shed light on questions about how the brain acquires knowledge. A major focus of brain research over the past two decades concerns how and where learning takes place within the brain. Recent technological advances in imaging technology, such as functional Magnetic Resonance Imaging (fMRI), Positron Emission Tomography (PET), and Electroencephalography (EEG) are allowing an examination of the learning process from the molecular level to neural systems and areas. Unlike a computer network, the brain is fundamentally discontinuous. The 'cable' that transmits signals consists of neurons, but the cable is not where learning takes place. Learning happens within the discontinuities. Virtually all of the neurons in the adult brain are extant at birth (de Haan and Johnson, 2002). Brain development consists of a process of increasing synaptic density and complexity in which the volume of the brain increases fourfold from birth to adulthood due to the proliferation of connections (Goswami, 2004). When new information is learned, the biochemistry of the synapses is altered. The biochemical changes at crucial synaptic points in the brain are the foundation of learning (Sebastian-Galles, 2004).

This type of research sheds light on key questions in education research. We know that some information is stored in memory and easily accessed, while other information is not. It is now possible to address questions concerning the types of information, delivery modes, as well as the enhancement of storage and retrieval. Research in neurocognition has shown that at least two aspects of this process are important for effective learning. The first is how the new information relates to an individual's prior knowledge. The second is how relevant the new information is to an individual's self-interest. The biochemical synaptic changes that take place when new information is received create mental pathways that determine how and where learning takes place. These pathways are mapped onto an individual's existing schema (Chamot et al., 1999; Barnhardt, 1997). When the "schemata for a particular topic are well-developed and personally meaningful, new information is easier to retain and recall, and proficient learners initiate and activate their associations between the new and old learning" (Echeverria et al., 2004, p. 81).

For well over a decade, education researchers have stressed the importance of cognitive strategies that incorporate new information with prior knowledge and the critical role of relevance and motivation (eg. Garcia and Pintrich, 1994; Zimmerman, 1990). Strategies that promote self-regulated learning have currently become a major topic in educational research (Paris and Paris, 2001). Section 3 of this paper discusses one of these strategies - the use of concept maps.

Neuroscience research has also shown that learning is enhanced when multiple senses are involved. Certain evidence suggests that brain is highly differentiated and that different systems are specialized to process specific and different kinds of information (Cohen and Tong, 2001; Downing et al., 2001; Neville, et al., 1998). This is the ‘modular’ view of the brain. There is also evidence that the brain is distributive in nature, and that given information is processed by many parts of the brain and specific brain regions are capable of processing many classes of information (Cohen and Tong, 2001). The linkages between different systems within the brain that contribute to distributive nature of the brain appear to be strongest in infancy. This is shown, for example, by research that demonstrates that infants respond to auditory stimulation in multiple regions of the brain, including those pertaining to visual stimulus (Roder and Neville, 2003). It appears that mutual linkages decline during childhood and that the brain becomes increasingly specialized (Goswami, 2004). We posit that the modular and distributive aspects of the brain have significant implications for learning theory and practice. Educational research has already demonstrated that content delivery using a multi-sensory approach facilitates the acquisition and retention of information (e.g. Nolen, 2003). In the next section we describe multi-sensory approach to learning through the use of concept maps.

Cognitive research also shows that the brain does not easily assimilate large amounts of random information. When students attempt to learn through such assimilation, much of the delivered information will not be stored nor will it be transformed into higher levels of learning (Resnick, 2003). Some of the most compelling evidence of this comes from research on dyslexia (e.g. Shaywitz and Shaywitz, 2001, 2004; Shaywitz et al., 2003), a reading disability that plagues an estimated 5–17% of the population and has, until recently, remained a puzzle to educators and neuroscientists. By tracking the brain activity of both successful readers and dyslexic readers through fMRI and other imaging techniques, researchers are able to view the contrast in their brain activity. Dyslexics are unable to decode words and perceive letters and sounds as streams of random information. When this occurs, they rely largely on rote memorization rather than on more efficient learning strategies, and ultimately under-perform.

These findings are relevant for efficient information delivery in higher education. We put forward that the use of isolated reading and lectures by many students results in compensatory strategies similar to those of dyslexics and causes students to rely on the rote memorization of disconnected concepts. The use of concept enables students to organize information, place it in context, and increase retention.

Educational research has also shown that emotions have a profound influence on learning. While much of this research focuses on the learning environment, including sleep, nutrition, home and social context, more research is beginning to focus on the neurological effects of emotions and their impact on memory and learning. These effects occur primarily through the release of different hormones into different parts of the brain. Although some types of emotions positively impact certain types of memory, stress is shown to adversely affect memory formation and learning (Erk, et al., 2003; Rimmele, 2003). In addition to neurological evidence, the negative impact of stress on learning is evidenced by the effects of stress on student performance (e.g. Takahashi, et al., 2004).

In summary, the neuroscience research described above offers several lessons for educational practice. First, we know that the brain acquires information through our senses. The more senses involved, the richer the experience. We also recognize that emotions also play a significant role in learning and that anxiety, for example, inhibits our ability to receive information. Further, we understand that personally relevant information is more meaningful. Finally, we know that linking new information to previous knowledge enables our brains to better assimilate information because it is organized and familiar.

3. EDUCATIONAL PRACTICE AND CONCEPT MAPS

The lecture model as a way of disseminating large amounts of information is still one of the most prevalent instructional techniques in institutions of higher learning. In general, however, the traditional lecture model is not congruent with current learning theory. In this section, we explore the use of concepts maps (also termed graphic organizers), one of several instructional techniques that are consistent with current brain research. This section of the paper offers a brief overview of concept maps and their use in higher education.

3.1. Brief Overview of Concept Maps

Concept maps are visual or spatial portrayals that organize information and display relationships. They serve both as teaching techniques and as aids for student learning. They assist students in discovering key concepts and relationships among them. Concept maps can be used by students before, during, and after the study of a topic or group of topics. Prior to approaching a topic, concept maps can be used as a guide and a means of identifying prior knowledge. Concurrently with a topic's study, they act as a means of note taking, identify key ideas, make connections between concepts, and generate a framework for organizing understanding. Similarly, they can also be used by teachers to deliver material and assess student understanding prior to, during, and after the study of a particular topic.

Concept maps come in a variety of forms to match their functions. Hierarchical concept maps often organize concepts by identifying key ideas and supporting material. Thematic maps (see Figure 1) and network trees are commonly used to display this type of information. Sequential maps are often used to illustrate cause and effect relations or a series of events or concepts. Commonly used concept maps for displaying sequential information are sequential episodic maps and fishbone maps. Comparative concept maps often display similarities and differences and include compare-contrast maps and compare-contrast matrices. Overviews of the types and uses of concept maps are provided by Hall and Strangman (2003), Katayama and Robinson (2000), and Novak (1990).

3.2. Uses and Advantages of Concept Maps

There are several advantages of the use of concept maps. They can help students to make paired-associations and generate analogies, therefore increasing the interaction with information and making it more relevant. They guide students to focus on details, conceptual links, and overarching concepts simultaneously. The use of concept maps enable

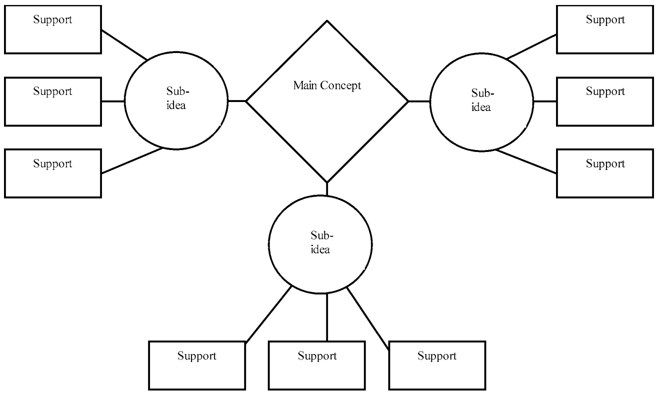


Figure 1. General form of a thematic concept map.

students to better organize material and link new information to previous knowledge. They also guide students by helping them identify critical features, compare concepts, and organize a sequence of ideas.

As educators, we have all witnessed students taking copious notes, attempting to capture every word of lecture. Meanwhile, other students write little, being at a loss for what to record. As experts in our fields, we know the relationship between the concepts we are trying to teach and the supporting information. We posit that through the use of concept maps, ideas and relationships become "visible" to students.

Concept maps are used primarily in elementary education to facilitate vocabulary acquisition and reading comprehension. The advantages of concept maps, however, are broadly applicable. Education research supports the use of concept maps across content areas in higher education. For example, Katayama and Robinson, (2000) show that concept maps improve retention while reading chapter-length text for undergraduate students. Hoffman, (2003) demonstrates that the effectiveness of concept maps increases when students additionally create them as a post-reading strategy. Katayama and Crooks (2003) provide evidence that when providing materials to students, partially completed concept maps are more effective than complete lecture or reading notes. Nilsson and Mayer (2002) show that, in addition to understanding text, comprehension of web-based hypertext material is also enhanced through the use of concept maps. Finally, Moore and Readence (1984) demonstrate that the effectiveness of concept maps is significantly larger for college-level students than for elementary and secondary students. This, and other evidence, argues for the broader use of concept maps in higher education.

The effectiveness of concept maps can be ascribed to the fact that their use is consistent with lessons derived from the neuroscience research. First, concept maps help make explicit the link between new information and prior knowledge. Second, they enable improved assimilation of new information because it is organized and more familiar. Third, concept maps utilize a multi-sensory approach and simultaneously access different neural processes and regions of the brain. Fourth, learner-generated concept maps increase the opportunities to make information personally relevant and meaningful. Finally, all of these effects may reduce the anxiety that inhibits the ability to receive information.

4. CONCLUSION

In this paper we discuss recent research findings on neurocognition and learning theory and capitalize on their relevance to education. We propose that concept maps are congruent with these findings. They increase learning effectiveness primarily by organizing new information and utilizing prior knowledge, multisensory information delivery, and learner interest. This paper is only a start of our inquiry. We are cognizant of the fact that education issues we ponder are complex and that concept maps represent a small part of the panoply of possible approaches that take advantage of recent advances in learning theory.

REFERENCES

Barnhardt, S., 1997, Effective memory strategies, *The NCLRC Language Resource* **1**(6), (July, 1997); http://www.nclrc.org/caidlr16.htm.

Bransford, J., Brown, A., and Cocking, R., 2000, *How People Learn: Brain, Mind, Experience, and School*, National Academic Press.

Chamot, A., Barnhardt, S., El-Dinary, P., and Robbins, J., 1999, *Learning Strategies Handbook*, Pearson ESL.

Cohen, J. and Tong, F., 2001, The face of controversy, *Science* **293**(5539):2405–2407.

Downing, P., Jiang, Y., Shuman, M., and Kanwisher, N., 2001, A cortical area selective for visual processing of the human body, *Science* **293**(5539):2470–2473.

de Haan, M., and Johnson, M., 2003, Mechanisms and Theories of Brain Development, in: *The Cognitive Neuroscience of Development*, M. de Haan and M. Johnson, eds., Psychology Press, pp. 1–18.

Echevarria, J., Vogt, M., and Short, M., 2004, *Making Content Comprehensible for English Learners: The SIOP Model*, Pearson Education.

Erk, S., Kiefer, M., Grothe, J., Wunderlich, A., Spitzer, M., and Walter, H., 2003, Emotional context modulates subsequent memory effect, *NeuroImage* **18**(2):439–448.

Garcia, T., and Pintrich, P., 1994, Regulating Motivation and Cognition in the Classroom: The Role of Self-Schemas and Self-Regulatory Strategies, in: *Self-Regulation of Learning and Performance: Issues and Educational Applications*, D. Schunk and B. Zimmerman, eds., Erlbaum, pp. 127–180.

Georghiades, P., 2004, From the general to the situated: Three decades of metacognition, *International Journal of Science Education* **26**(3):365–383.

Goswami, U., 2004, Neuroscience and education, *British Journal of Educational Psychology* **74**(1):1–14.

Hall, T., and Strangman, N., Graphic Organizers, 2003, National Center on Accessing the General Curriculum (CAST); http://www.cast.org/ncac/GraphicOrganizers3015.cfm.

Hoffman, J., 2003, Student-created graphic organizers bring complex material to life, *College Teaching* **51**(3):105.

Katayama, A., and Crooks, S., 2003, Differential effects of studying complete or partial graphically organized notes, *Journal of Experimental Education* **71**(4):293–302.

Katayama, A., and Robinson, D., 2000, Getting students ‘partially’ involved in note taking using graphic organizers, *Journal of Experimental Education* **68**(2):119–134.

McCray, R., DeHaan, R., and Schuck, J., 2003, Improving undergraduate instruction in science, technology, engineering, and mathematics, Workshop Report, National Research Council; http://www.nap.edu/books/0309089298/html/.

Moore, D., and Readence, J., 2001, A quantitative and qualitative review of graphic organizer research, *Journal of Educational Research* **78**(1):11–17.

Neville, H., Bavelier, D., Corina, D., Rauschecker, J., Karni, A., Lalwani, A., Braun, A., Clark, V., Jezzardi, P., and Turner, R., 1998, Cerebral organization for language in deaf and hearing subjects: Biological constraints and effects of experience, *Proceedings of the National Academy of Sciences* **95**(3):922–929.

Nicoll, G., 2001, A three-tier system for assessing concept maps: A methodological study, *International Journal of Science Education* **23**(8):863–875.

Nilsson, R., and Mayer, R., 2002, The effects of graphic organizers giving cues to the structure or hypertext document on users’ navigation strategies and performance, *International Journal of Human-Computer Studies* **57**(1):1–25.

Paris, S., and Paris, A., 2001, Classroom applications on research on self-regulated learning, *Educational Psychologist* **36**(2):89–101.

Resnick, M., 2003, Thinking like a tree (and other forms of ecological thinking, *International Journal of Computers for Mathematical Learning* **8**(1):43–62.

Rimmele, U., 2003, Cortisol has different effects on human memory for emotional and neural stimuli, *NeuroReport* **14**(18):2485–2488.

Roder, B. and Neville, H., 2003, Developmental Functional Plasticity, in: *Handbook of Neuropsychology*, J. Grafman and I. Robertson, eds., Elsevier, pp. 231–270.

Sebastian-Galles, N., 2004, *A Primer on Learning: A Brief Introduction from the Neurosciences*, Organisation for Economic Co-Operation and Development; http://www.oecd.org/document/57/0,2340,en_2649_14935397_33625337_1_1_1_1,00.html.

Shaywitz, S., and Shaywitz, B., 2004, Reading disability and the brain, *Educational Leadership* **61**(6):6–11.

Shaywitz, S., and Shaywitz, B., 2001, The neurobiology of reading and dyslexia, *Focus on Basics* **5**(A), (Augusts, 2001); http://ncsall.gse.harvard.edu/fob/2001/shaywitz.html.

Shaywitz, S., Shaywitz, B., Fulbright, R., Skudarshi, P., Mencl, W., Constable, R., Pugh, K., Holahan, J., Marchione, K., Fletcher, J., and Lyon, G., and Gore, J., 2003, Neural systems for compensation and persistence: Young adult outcome of childhood reading disability, *Biological Psychiatry* **54**(1):25–34.

Takahashi, T., Ikeda, K., Ishikawa, M., Tsukasaki, T., Nakama, D., Tanida, S., and Kameda, T., 2004, Social stress-induced cortisol elevation acutely impairs social memory in humans, *Neuroscience Letters* **363**(2):125–130.

Yin, Y., Vanides, J., Ruiz-Primo, M., Ayala, C., and Shavelson, R., 2004, A comparison of two construct-a-concept-map science assessments: Created linking phrases and selected linking phrases, CSE Report 624, National Center for Research on Evaluation, Standards, and Student Testing (CRESST)/Stanford University.

Zimmerman, B., 1990, Self-regulated learning and academic achievement: an overview, *Educational Psychologist* **25**(1):3–15.

XML IN THE WORLD OF (OBJECT-)RELATIONAL DATABASE SYSTEMS

Irena Mlynkova and Jaroslav Pokorny*

1. INTRODUCTION

XML[1] is universally recognized as the standard for interchange and device-independent representation of information. On the other hand, XML is recently understood as a new approach to data modelling.[2, 3] A well-formed XML document or a set of documents is an XML database and the associated DTD or schema specified in the language XML Schema[4] is its database schema. Implementation of a system enabling us to store and query XML documents efficiently is developed today in different ways. We do not discuss here native storage solution, i.e. a DBMS dedicated to manage XML data collections. A more practical possible solution can be found in storing XML data in (object-)relational DBMS. Moreover, this approach enables to provide XML with missing database mechanisms (e.g. indexes, transactions, multi-user access, etc.).

Currently there is a relatively large number of works devoted to storing XML data, including the special architectures like PDOM, CMS, XML Servers, XML Query Engines, etc. We refer reader to Bourret[2] for their comprehensive overview. Our contribution is a summarization of recent XML storage techniques based on today's (object-)relational database technologies, their comparing and evaluation. We also present own algorithm for mapping XML Schema structures to object-relational (OR) schema. A more comprehensive discussion can be found in Mlynkova and Pokorny.[5]

For transferring the data between XML documents and (O)R structures so-called mapping methods are of a great importance. A basic classification[6] of existing mapping methods includes the following three classes:

- generic methods, which do not use any schema of stored XML documents,
- schema-driven methods, which are based on existing schema of stored XML documents, and
- user-defined methods, which are based on user-defined mapping.

* Charles University, Faculty of Mathematics and Physics, Department of Software Engineering, Malostranske nam. 25, 118 00 Prague 1, Czech Republic, {mlynkova,pokorny}@ksi.ms.mff.cuni.cz.

```
<person id=1 age=23>
  <name>Irena</name>
  <surname>Mlýnková</surname>
  <address id=2>
    <street>Podlesí 4943</street>
    <city>Zlín</city>
  </address>
</person>
<person id=3 age=30>
  <name>Jim</name>
  <surname>Beam</surname>
</person>
...
```

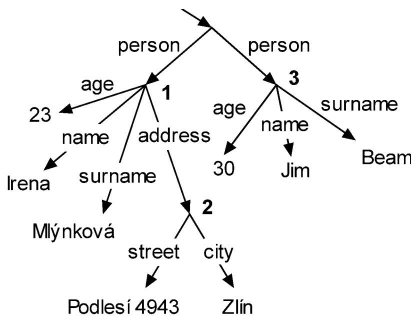


Figure 1. An example of a generic-tree.

Sections 2–4 contain an overview and possible classifications of the respective methods. Their evaluation and discussion is in Section 5. Section 6 provides conclusions.

2. GENERIC MAPPING METHODS

Generic mapping methods do not use (possibly) existing XML schema of stored XML documents. They are usually based on one of two approaches – creating

- a general (O)R schema into whose relations any XML document regardless its structure can be stored, or
- a special kind of (O)R schema into whose relations only a certain collection of XML documents having a similar structure can be stored.

The former methods model an XML document as a tree T according to e.g. the OEM model or the DOM model, while the latter reflect its special "relational" structure.

2.1. Generic-Tree Mapping

A typical representative of generic mapping is a group of methods called *generic-tree mapping*.[7] An example of an XML document and its T is depicted in Figure 1.

There are several methods for storing T, so-called edge, attribute, universal, and normalized universal mapping.

Edge Mapping. This method stores all edges of T in the following table:

`Edge(source, ord, name, flag, target)`

The table contains identifiers of nodes connected by the edge (`source` and `target`), name of the edge (`name`), a flag that indicates whether the edge is internal or points to a leaf (`flag`), and an ordinal number of the edge within sibling edges (`ord`).

Attribute Mapping. In this mapping an extra table for each edge name (so-called *attribute*) is established. The structure of these tables is similar to the previous case:

`Edge`$_{\texttt{name}}$ `(source, ord, flag, target)`

Universal Mapping. This method stores edges of T in so-called *universal table*, which contains columns for all the attribute names described in previous method. In other words, a universal table corresponds to the result of an outer join of all tables from attribute mapping. If $a_1, \ldots, a_k$ are all the attribute names in the XML document, the universal table can have the following structure:

`Uni(source, ord`$_{a1}$`, flag`$_{a1}$`, target`$_{a1}$`,...,ord`$_{ak}$`, flag`$_{ak}$`, target`$_{ak}$`)`

Obviously the universal table contains many NULL values.

Normalized Universal Mapping. This method tries to solve the main disadvantage of universal mapping storing multi-valued attributes in separate, so-called *overflow tables*. An overflow table is established for each attribute name, while its structure is the same as in attribute mapping. The universal table then contains only one row per each attribute name, others are stored in corresponding overflow tables.

There is also a plenty of variations of these methods. First, in all described approaches the values in leaves can be stored either in separate *value tables* (each holds values of a certain type) or in additional columns of existing tables. Other, so-called *hybrid methods* can be created using combinations of the described approaches.

2.2. Structure-Centred Mapping

The structure-centred mapping[8] considers all nodes of the tree T having the same structure defined as a tuple $v = (t, l, c, n)$, where t is the type of the node (e.g. ELEMENT, ATTRIBUTE, TEXT, ...), l is the node label, c is the node content and $n = \{v_1, \ldots, v_n\}$ is the list of successor nodes. The paper[8] considers the problem how to realize mapping of the lists of successor nodes. It proposes three kinds of storage strategies focusing on speeding up the access performance.

Foreign Key Strategy. Each tree node v is simply mapped to a tuple with a unique identifier and a foreign key reference to the parent node. The method is quite simple and the stored tree can easily be modified. Nevertheless, its disadvantage is evident – the retrieval of the data involves many self-join operations.

DF Strategy. In this strategy each node of T is given an index value (a couple of minimum and maximum DF values), which represents its position in T. The DF values are determined when traversing T in a depth first (DF) manner. A counter is increased each time another node is visited. If a node v is visited the first time its minimum DF value v_{min} is set to the current counter value. When all child nodes have been visited, the maximum DF value v_{max} is set to the current counter value (see Figure 2).

Using DF values relationships of nodes (e.g. sibling order, element-subelement relationship, etc.) can easily be determined just by comparisons. For example, a node v is a descendant of node μ, if $v_{min} > \mu_{min}$ and $v_{max} < \mu_{max}$. Moreover, as the nodes can be totally ordered according to DF values, retrieving a part of a document is linear. The weak

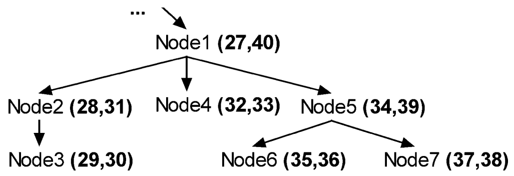


Figure 2. An example of DF indexing.

point of this strategy is document update – in the worst case it requires to update DF values of all nodes of the tree.

SICF Strategy. In this strategy each node of the graph is also given by an identification of its position – in this case so-called *simple continued fraction* (SICF)

$$\sigma = \cfrac{1}{q_k + \cfrac{1}{\cdots \, q_2 + \frac{1}{q_1}}}$$

where $q_i \in N$ $(i = 1, \ldots, k)$ are called *partial quotients* of σ and the expression $<q_1, \ldots, q_k>$ *partial quotient sequence*. Sequences uniquely determine fractions and vice versa. The SICF values are determined in the following way: the root node gets a *seed value* $s \in N, s > 1$ (its SICF value is $<s>$). If a node v has SICF value $<q_1, \ldots, q_m>$ and has n ordered child nodes $v_1, \ldots, v_n$, then the SICF value for i-th child node is $<q_1, \ldots, q_m, i>$.

The advantages and disadvantages of this strategy are similar to the previous one.

2.3. Simple-Path Mapping

This method[9] assumes that queries over the stored XML data are path queries of an XML query language. The main idea is to decompose XML documents into so-called *simple paths* and to store them in the database. Each simple path is based on the relation parent-descendant. Hence, each node in the graph retains its simple path. But as a simple path contains neither position nor order information, these two are stored in the graph too. The position information (called *region*) is a pair of a start and an end value, which are assigned as follows: Each word occurrence is assigned an integer number corresponding to its position within the document. Each tag is assigned a real number – its integer part indicates the position of the preceding word and its decimal part indicates the position of the tag being concerned in the current sequence of tags. The order information is composed of occurrence plus and occurrence minus order information, which expresses the index number of the node within its parent node (see Figure 3).

All the information about T is stored in following four relations:

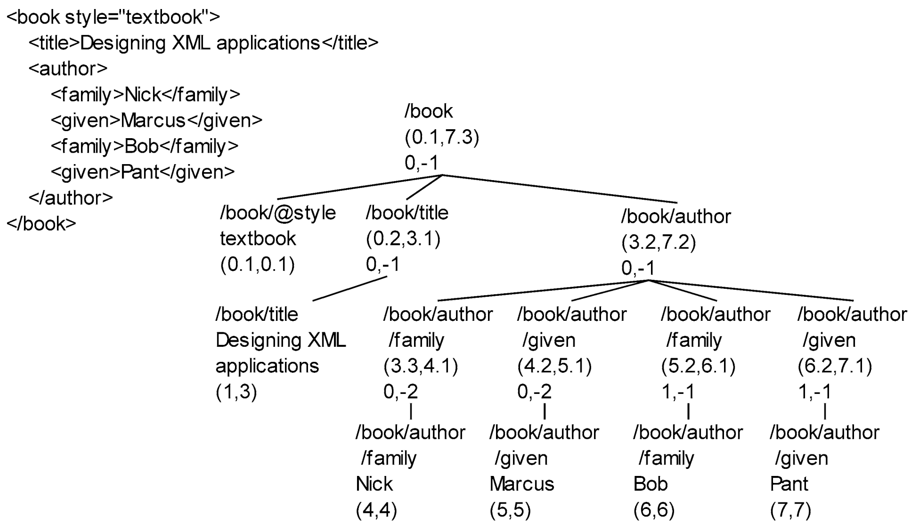
```
Element(docID, pathID, index, reindex, pos)
Attribute(docID, pathID, attvalue, pos)
Text(docID, pathID, textvalue, pos)
Path(pathexp, pathID)
```

First three relations store information about each node type – document identifiers (`docID`), path identifiers (`pathID`), plus and minus occurrence order (`index` and `rein-dex`), regions (`pos`), attribute and text values (`attvalue` and `textvalue`). The relation Path stores simple paths (`pathexp`) and path identifiers (`pathID`).

The main advantage of this method is apparent – storing simple paths of elements and attributes simplifies and speeds up processing path queries.

2.4. Monet Mapping

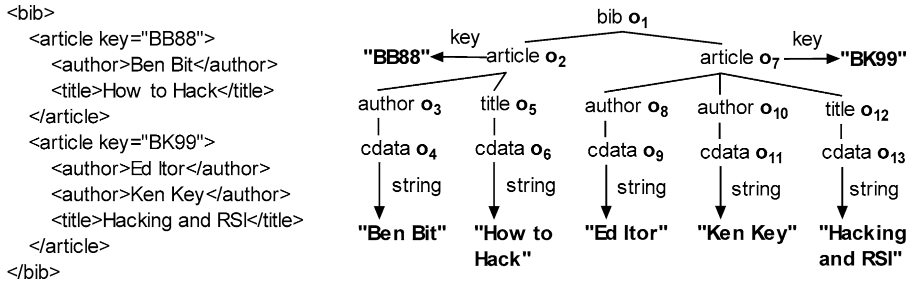
The tree model of XML data in the *Monet mapping*[10] is slightly different than in the previous methods (see Figure 4). The main idea of this method is based on a complete



```
<book style="textbook">
  <title>Designing XML applications</title>
  <author>
    <family>Nick</family>
    <given>Marcus</given>
    <family>Bob</family>
    <given>Pant</given>
  </author>
</book>
```

/book (0.1,7.3) 0,-1
/book/@style textbook (0.1,0.1)
/book/title (0.2,3.1) 0,-1
/book/author (3.2,7.2) 0,-1
/book/title Designing XML applications (1,3)
/book/author /family (3.3,4.1) 0,-2
/book/author /given (4.2,5.1) 0,-2
/book/author /family (5.2,6.1) 1,-1
/book/author /given (6.2,7.1) 1,-1
/book/author /family Nick (4,4)
/book/author /given Marcus (5,5)
/book/author /family Bob (6,6)
/book/author /given Pant (7,7)

Figure 3. An example of a simple-path tree.



```
<bib>
  <article key="BB88">
    <author>Ben Bit</author>
    <title>How to Hack</title>
  </article>
  <article key="BK99">
    <author>Ed Itor</author>
    <author>Ken Key</author>
    <title>Hacking and RSI</title>
  </article>
</bib>
```

Figure 4. An example of a Monet tree.

binary fragmentation of T to binary associations, which describe different parts of the tree (edges, attributes, the topology of the document).

The associations, which bear semantically related information, are stored in relations together. Such information is related to definition of a path(o) as a sequence of (vertex and edge) labels along the path from the root node to o (where $\rightarrow_e$ and $\rightarrow_a$ denotes edge to an element and attribute, respectively), e.g.:

path(o_3) = bib $\rightarrow_e$ article $\rightarrow_e$ author

path("Ben Bit") = bib $\rightarrow_e$ article $\rightarrow_e$ author $\rightarrow_e$ cdata $\rightarrow_a$ string

Each path then describes the position of an element in T relative to the root node. At the same time, path(o) is used to denote the type of binary association (. , o). All associations of the same type are stored in the same binary relation.

The advantage of this method is, that it avoids large and expensive scans over irrelevant data, the disadvantage is the high degree of fragmentation, which can increase efforts to reconstruct the original document or its parts.

2.5. Table-Based Mapping

A typical representative of the approach that enables to store only a certain collection of XML documents having similar structure is called *table-based mapping*.[2] It is based on the assumption, that the stored XML documents have a regular structure reflecting database, tables, rows, and columns. The mapping between elements and relations is exactly defined by the structure of the XML document. Apparently, this method is suitable especially for transferring the data between two relational DMBSs.

3. SCHEMA-DRIVEN MAPPING METHODS

Schema-driven mapping methods are based on existing schema S_1 of stored XML documents, written in DTD or XML Schema, which is mapped to (O)R database schema S_2. The data from XML documents valid against S_1 are then stored into relations of S_2. The purpose of these methods is to create optimal schema S_2, which consists of reasonable amount of relations and whose structure corresponds to the structure of S_1 as much as possible. All of these methods try to improve the basic mapping idea "to create one relation for each element composed of its attributes and to map element-subelement relationships using keys and foreign keys".

3.1. Common Characteristics

Schema-driven mapping methods have several common basic principles[6] resulting from information stored in the XML. The most important ones are:

- Subelements with `maxOccurs = 1` are (instead of to separate tables) mapped to tables of parent elements (so-called *inlining*).
- Elements with `maxOccurs > 1` are mapped to separate tables. Element-subelement relationships are mapped using keys and foreign keys.
- Alternative subelements are mapped to separate tables (analogous to the previous case) or to one universal table (with many nullable fields).
- If it is necessary to preserve the order of sibling elements, the information is mapped to a special column.
- Elements with mixed content are usually not supported.
- A reconstruction of an element requires joining several tables.

3.2. Possible Classifications

The considered methods have several common features according to which they can be classified quite differently.

Source XML Schema. An obvious classification is based on the type of S_1. Most of these methods are based on DTD. The reason for this is, that although the DTD is quite simple, it is still sufficient for most applications. On the other hand, although the XML Schema is much more complex and thus difficult for learning, it contains useful features that DTD lacks and gives users more powerful tool for describing the allowed structure of XML documents. At present, there are also several methods (e.g. XMLSchemaStore mapping or LegoDB mapping), which try to exploit these features.

Target Database Schema. The methods differ also according to the S_2. In this paper two possibilities are concerned – relational or object-relational approach. Most of the methods are based on the former one, since the relational databases and their features managed to gain more focus than others (including OR ones). Despite of this fact there are several methods, which try to take the advantage of OR features, such as NF^2-relations (e.g. Hybrid object-relational mapping) or user defined data types and references (e.g. XMLSchemaStore mapping).

Flexibility. Another classification[6, 11] includes two classes – fixed and flexible methods. Fixed methods (e.g. Basic, Shared, and Hybrid algorithms, etc.) are those, which do not use any other information than S_1 itself and whose mapping algorithm is straightforward. On the other hand, flexible methods (e.g. LegoDB mapping or Hybrid object-relational mapping) use the additional information (e.g. query statistics, element statistics, etc.) and focus on creating an optimal schema for a certain application.

3.3. Algorithms Basic, Shared, Hybrid, and Derived Algorithms

The best-known representative of fixed schema-driven mapping methods is a group of three algorithms for mapping a DTD to relational schema called Basic, Shared, and Hybrid.[12] The main idea is based on a definition of a directed graph, so-called *DTD graph*, which represents the processed DTD. Nodes of the graph are elements (which appear exactly once), attributes, and operators (which appear as many times as in the DTD). Edges of the graph represent element-attribute, element-subelement or element-operator and operator-subelement relationships. Each DTD is also first pre-processed and simplified to contain only ? and * operators and flat expressions (see Figure 5).

These algorithms try to gradually improve the idea "to create one relation for each element". They differ according to the amount of redundancy they may cause.
Basic Algorithm. The Basic algorithm combines two approaches:

- to inline as many descendants of an element as possible and
- to create a relation for each element in the DTD graph.

In the former case only two kinds of element-subelement relationships are solved using keys and foreign keys – subelements with multiple occurrence (indicated by the use of * operator) and recursion (indicated by cycles in the graph). The main disadvantages of this algorithm are obvious – a huge amount of unnecessary relations and a great deal of redundancy since an element node can be represented in several relations.

```
<!ELEMENT author(name?,surname)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT surname(#PCDATA)>
<!ELEMENT book(author*,title)>
<!ATTLIST book published CDATA>
<!ELEMENT title(#PCDATA)>
<!ELEMENT article(author)>
<!ATTLIST article paper CDATA>
```

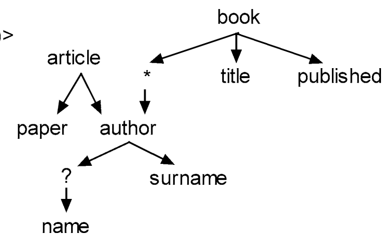


Figure 5. An example of a DTD graph.

Shared Algorithm. The Shared algorithm tries to avoid the drawbacks of Basic. The idea is to identify elements that are represented in multiple relations and to share them by creating separate relations for them. The mapping rules are:

- Nodes with an in-degree of one are inlined to parent relations.
- Nodes with an in-degree of zero are stored in separate relations.
- Repeated elements are stored in separate relations.
- Of all mutually recursive elements having an in-degree one, one of them is stored in a separate relation.
- The problem of inlined elements, which can become roots of an instance XML document, is solved using a flag for each element that indicates this state.

Apparently the main advantage of the Shared algorithm is the reduced amount of relations and redundancy. Its main disadvantage is the number of join operations necessary for restoring an element, which can be worse than in Basic.

Hybrid Algorithm. The Hybrid algorithm tries to combine the join reduction properties of Basic with the sharing features of Shared. The algorithm is similar to Shared except for additional inlining of elements with an in-degree greater than one, that are neither recursive nor reached through a * node.

CPI Algorithm. CPI (Constraints-Preserving Inlining) method[13] can be based e.g. on the mentioned Hybrid algorithm. Its main purpose is to capture not only the structure of the DTD but the semantic constraints as well. The considered constraints are e.g. domain constraints, cardinality constraints (i.e. +, *, ? operators), referential integrity (i.e. ID, IDREF, IDREFS types), etc. These constraints are represented using corresponding SQL constraints e.g. NOT NULL, UNIQUE, PRIMARY/FOREIGN KEY, CHECK, etc.

3.4. Object-Relational Mapping

Object-relational mapping[14] uses the word "object-relational" in a bit confusing way, since it does not denote the type of S_2 but the two steps of the algorithm. S_1 is expressed either in DTD or XML Schema; S_2 is relational in all cases. The two steps are:

1. S_1 is mapped to an object schema expressed in an object-oriented language.
2. The object schema is mapped to S_2.

Obviously, if the object schema is not essential, it can be eliminated.

The object schema models S_1 as a tree of objects. In this step element types with PCDATA-only content and attribute types are considered as *simple types*. Element types with element or mixed content, or element types with attributes are considered as *complex types*. The mapping rules can be summed as follows:

- simple types $\rightarrow$ scalar data types,
- complex types $\rightarrow$ classes with each element type in the content model mapped to a property of the class – the data type of each property is either the scalar data type or a pointer/reference to the corresponding object,
- attributes $\rightarrow$ properties,
- subelements in a sequence or a choice $\rightarrow$ properties (whereas in the latter case the corresponding columns in the relational schema will be nullable),

DTD:	object schema:	relational schema:
<!ELEMENT A(B,C)>	class A {	A(b,c_{fk})
<!ELEMENT B(#PCDATA)>	String b;	
	C c; }	
<!ELEMENT C(D,E)>	class C {	C($\underline{pk}$,d,e,f)
<!ELEMENT D(#PCDATA)>	String d;	
<!ELEMENT E(#PCDATA)>	String e;	
<!ATTLIST E F CDATA>	String f; }	

Figure 6. An example of object-relational mapping for DTD.

```
<schema>
  <element name="E1" type="T1"/>
  <complexType name="T1">
    <sequence>
      <element name="E2" type="string"/>
      <element name="E3" type="int"/>
      <element name="E4" type="decimal minOccurs="0"/>
    </sequence>
    <attribute name="A1" type="string"/>
  </complexType>
</schema>
```

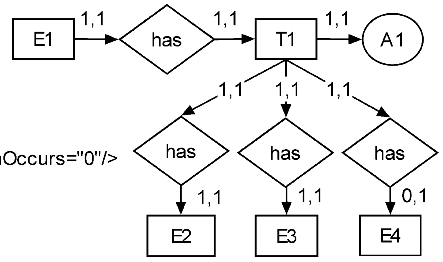


Figure 7. An example of mapping an XML schema to an EER diagram.

- repeated subelements → multi-valued properties of (un)known size,
- mixed content → a multi-valued property for storing PCDATA-values plus additional *order columns* for each property sharing the same order space.

An example of a DTD and associated schemes is in Figure 6.

For XML Schema the transformation is similar, the differences are related to additional features XML Schema has. The step doing object-to-relational transformation does not distinguish from usual approaches used in today's software engineering.

3.5. Constraints Preserving Mapping

Constraints preserving mapping[15] preserves not only the structure of S_1 but also the variety of semantic constraints XML Schema enables to express.

XML Schema structures are formally represented by the regular tree grammar called FD-XML.[15] An extension of ER model, so-called EER model, is proposed and the FD-XML is converted into EER schema. Then, the EER schema is simplified and optimized, preserving both the structure and the semantic constraints and finally, the simplified EER schema is converted to relational schema. The EER model uses (min, max) cardinalities, arrowheads modelling parent-child relationships, and accessories in order to preserve the data constraints (see Figure 7).

The rules for simplification of the EER schema include converting an entity to its parent entity's attribute and removing a subentity from its parent entity if possible. The rules for transforming the simplified EER schema to relational schema are similar to well-known algorithms for design of relational databases.

```
<schema>
  <complexType name="T1">
    <sequence>
      <element ref="E1"/>
    </sequence>
    <attribute name="A1" type="T2"/>
  </complexType>
  <element name="E1" type="string">
  <simpleType name="T2">
    <restriction base="string">
      <length value="5"/>
    </restriction>
  </simpleType>
</schema>
```

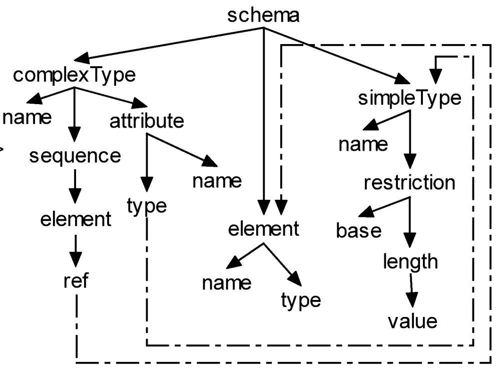


Figure 8. An example of a DOM graph – the solid lines correspond to original edges of the DOM tree; dash-and-dot lines are the additional ones.

3.6. XMLSchemaStore Mapping

XMLSchemaStore mapping[5] maps S_1 expressed in XML Schema to OR schema expressed in SQL:1999* standard.[16] It tries to preserve the structure as well as semantic constraints of the S_1 in the S_2 and to exploit OR features of the SQL:1999 standard.

The mapping rules are as follows:

- built-in and user-defined simple type $\rightarrow$ corresponding database simple type (eventually) together with corresponding integrity constraint(s),
- complex type and model group $\rightarrow$ OR user-defined type (UDT), whereas:
  - XML attributes $\rightarrow$ UDT attributes with corresponding simple types,
  - simple element content $\rightarrow$ UDT attribute with corresponding simple type,
  - element-subelement relationship $\rightarrow$ UDT attribute, whose type is (according to the allowed occurrence and the type of the subelement) either the UDT of the subelement or the REF/ARRAY of REF to the UDT,
- deriving of complex types $\rightarrow$ UDT inheritance,
- element (according to its type and allowed occurrence) $\rightarrow$ own typed table† or a typed column of the table which corresponds to its parent element.

S_2 can be then described as a set of typed tables connected using references.

The mapping algorithm is based on traversing a directed graph called *DOM graph* (see Figure 8), whose edges determine the "order" in which the UDTs and typed tables should be created to follow reference properties.

The DOM graph results from the structure of a DOM tree of the given XML Schema file in the following way:

- The original edges of the DOM tree are directed to express the "direction" of element-subelement or element-attribute relationship.

* Latterly the SQL:2003 standard is at disposal. Its new type MULTISET can be used for unordered XML data.

† A table that is defined based on a UDT, i.e. rows of a typed table are instances of the corresponding UDT.

- New edges expressing the "direction" of the usage of globally defined items (e.g. elements, complex types, etc.) are added.

The mapping is done while traversing the graph starting in `schema` node. First, all descendants of a current node are processed, e.g. the UDTs and typed tables are created. Second, the current node can be processed, since all necessary OR items already exist.

3.7. LegoDB Mapping

A representative of flexible schema-driven mapping methods is an algorithm proposed in LegoDB system.[11] First the method defines fixed mapping of XML Schema structures (for processing simplicity rewritten into syntactically simpler, but semantically equivalent *p-schemas*) to relations. The flexibility is based on the idea to explore a space of possible XML-to-relational mappings and to select the best one according to given statistics including information about a sample set of XML documents and queries.

In order to select the best mapping the system in turns applies the following two steps to the source p-schema, until a good result is achieved:

1. Any possible XML-to-XML transformation is applied to the p-schema.
2. XML-to-relational transformations are applied to the new p-schema and against the resulting relational schema the given queries are estimated.

As the space of possible p-schemas can be large (possibly infinite), the paper[11] also proposes a greedy evaluation strategy that explores only the most interesting subset.

The XML-to-XML transformations used in the algorithm are: *inlining/outlining*, *union factorization/distribution*, *repetition merge/split*, *wildcards rewriting*, and *from union to options*. The XML-to-relational transformations are similar to those described in the previously mentioned methods.

3.8. Hybrid Object-Relational Mapping

Another example of flexible schema-driven mapping methods is a hybrid object-relational mapping.[17] It tries to improve the straightforward mapping of all elements and attributes in a DTD to relations, which can lead to large database schemes, by storing structured parts of the DTD in relations and semistructured parts in so-called *XML data types*, which support path queries and fulltext operations for XML fragments.

The main concern of this approach is to decide which parts of the DTD are structured and which semistructured. The suggested algorithm is as follows:

1. A graph (similar to above-described DTD graph) is built.
2. A *measure of significance* ω is determined for each element/attribute.
3. The resulting database design is derived from the graph.

The measure ω can be expressed as

$$\omega = \frac{1}{2}\omega_S + \frac{1}{4}\omega_D + \frac{1}{4}\omega_Q$$

where the used variables (weights) ω_{S}, ω_{D}, and ω_{Q} are derived from the DTD structure, the existing XML data, and the queries, respectively.

```
<!ELEMENT chapter(ctitle, section+)>       chapter=<id,ctitle,{section}>
<!ATTLIST chapter id ID REQUIRED>
<!ELEMENT ctitle(#PCDATA)>

<!ELEMENT section(stitle, paragraph+)>    section=<id,stitle,{paragraph}>
<!ATTLIST section id ID REQUIRED>
<!ELEMENT stitle(#PCDATA)>
                                           chapter = <id,ctitle,{<id,stitle,{paragraph}>}>
```

Figure 9. An example of hybrid object-relational mapping.

According to a given limit of ω(which influences the level of detail of S_2) the algorithm determines non-leaf nodes ν, each of which fulfils the following conditions:

1. All descendants of ν are below the given limit.
2. There exists no predecessor of ν that fulfils the condition 1.

All subgraphs consisting of these nodes and their descendants are replaced by an XML attribute. The resulting graph is finally mapped using a fixed mapping method to OR schema. The mapping (see Figure 9) focuses on the use of structured or nested attributes in NF^2-relations, assuming the existence of SET* and TUPLE constructors.

4. USER-DEFINED MAPPING METHODS

User-defined mapping methods are most often used in commercial systems. This approach requires that the user first defines S_2 and then expresses required mapping using a system-dependent mechanism, e.g. a special query language, a declarative interface, etc. At present, most of existing systems support some kind of user-defined mapping.

Obviously, this approach is the most flexible one. On the other hand, it requires large development effort and moreover mastering of two distinct technologies (XML and relational DBMS).[11] The description of these methods exceeds the scope of this paper.

5. DISCUSSION OF MAPPING METHODS

Usually XML documents are classified into two groups according to their content, structure, and supposed use – *data-centric* and *document-centric*.[2] The structure of data-centric documents is typically known and is specified in DTD or XML Schema. In document-centric documents the structure is typically specified using "mixed-content models" with arbitrary inter-leaving of text with XML mark-up. The distinction between these two groups is not generally obvious (documents which belong to both groups are called *hybrid* documents). A usability of mapping methods depends on these categories.

Notice, there is probably no reasonable argument for comparing generic and schema-driven methods together. Table 1 summarizes all discussed features of generic methods. Generally speaking, these methods are most suitable in cases when no XML schema exists. All mentioned methods are primarily determined for data-centric XML documents,

* Compare with ARRAY in SQL:1999 or MULTISET in SQL:2003.

Table 1. Summary of generic mapping features

Mapping	Considered features			
	CDATA sections, comments,...	Mixed content elements	Different data types	Preserving the element order
Generic-tree	Not supported	Not supported	Supported	Sibling order
Structure-centred	Not supported	Supported	Not supported	Total ordering
Simple-path	Not supported	Supported	Not supported	Sibling order
Monet	Not supported	Supported	Not supported	Sibling order

Table 2. Summary of schema-driven mapping features

Mapping	Considered features				
	Source schema	Target schema	Mixed content elements	Preserving sibling order	Flexibility
Basic, Shared, Hybrid, CPI	DTD	Relational	Not considered	Can be extended	Fixed
Object-relational	DTD/XML Schema	Relational	Supported using additional fields	Supported	Fixed
Constraints preserving	XML Schema	Relational	Not considered	Not considered	Fixed
XMLSchema-Store	XML Schema	Object-relational	Not supported	Supported	Fixed
LegoDB	XML Schema	Relational	Not considered	Not considered	Flexible
Hybrid object-relational	DTD	Object-relational	Supported using XML-aware type	Can be extended	Flexible

but probably with extensions related to the above-mentioned document-centric items they could be used for document-centric documents as well.

All mentioned features of schema-driven methods are summarized in Table 2. The idea of flexible mapping methods is relatively new. There is no reason for asking whether flexible methods are better than the fixed ones – apparently they are since the resulting schema suits the given statistics at least as well as corresponding fixed method. Indeed these methods can be obviously used only if it is possible to obtain necessary statistic information. The interesting point is how to determine the "best" schema. Just two but nevertheless quite different representatives were mentioned, whereas both are somehow based on a sample set of XML documents and typical queries. Obviously, the most flexible mapping is provided by user-defined mapping methods. Last but not least there is the matter of the S_2 schema. Without any doubts an OR schema solves the problems of multi-valued properties in more natural way than it does a relational schema in the 1NF.

To sum up, schema-driven mapping methods try to exploit the information in the given XML schema as much as possible. Although they can preserve some document-centric features (e.g. document order or mixed content elements), they are usually used for data-centric XML documents.

6. CONCLUSIONS

This paper was trying to offer a general and clear summary of existing strategies for connecting XML and database technologies, especially those related to relational and OR systems. Several possible classifications were mentioned and discussed and the best-known representatives of the classes were briefly described. Finally the general common features, advantages, and disadvantages of the described methods were discussed.

There are several areas, which will probably be in the main focus of future works. The first will apparently concern semantic constraints (especially those expressed using XML Schema) that should be preserved in the target (O)R schemes. Several of the mentioned methods partly focused on this area, but the current features of DBMSs and relational languages still limit these approaches in many ways.

The second interesting point is connected with flexible mapping methods, which try to optimize the fixed schema according to its future use. As there are no rules, which define a "good" XML schema (such as, e.g., normal forms for relations), the fixed mapping of a "bad" one can result in a "bad" relational schema. Thus an important task may be to determine a definition of a "good" XML schema and ways how to establish it.

ACKNOWLEDGMENTS

This research was supported in part by GACR grant 201/02/1553.

REFERENCES

1. Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation (February 4, 2004); www.w3.org/TR/REC-xml/.
2. XML and Databases (2003); www.rpbourret.com.
3. J. Pokorny, XML: a challenge for databases?, in: *Contemporary Trends in Systems Development*, edited by Maung K. Sein (Kluwer Academic Publishers, Boston, 2001), pp. 147–164.
4. XML Schema Part 0: Primer W3C Recommendation (May 2, 2001); www.w3.org/TR/xmlschema-0/.
5. I. Mlynkova and J. Pokorny, XML in the World of (Object-)Relational Database Systems, Technical Report No. 2003-8, Dep. of Software Engineering, Charles University, 2003, p. 28.
6. S. Amer-Yahia and M. Fernandez, Overview of Existing XML Storage Techniques, AT&T Labs, 2001.
7. D. Florescu and D. Kossmann, Storing and querying XML data using an RDBMS, *IEEE Data Engineering Bulletin* **22**(3), 27–34 (1999).
8. A. Kuckelberg and R. Krieger, Efficient structure oriented storage of XML documents using ORDBMS, Springer-Verlag Heidelberg, Vol. 2590, pp. 131–143 (2003).
9. T. Shimura, M. Yoshikawa, and S. Uemura, Storage and retrieval of XML documents using object-relational databases, *Proc. of DESA Conf.*, pp. 206–217 (1999).
10. A. Schmidt, M. Kersten, M. Windhouwer, and F. Waas, Efficient relational storage and retrieval of XML documents, *Proc. of WebDB Conf.*, pp. 47–52 (2000).
11. P. Bohannon, J. Freire, P. Roy, and J. Siméon, From XML schema to relations: a cost-based approach to XML storage, *Proc. of ICDE Conf.*, p. 64 (2002).
12. J. Shanmugasundaram, K. Tufte et al., Relational databases for querying XML documents: limitations and opportunities, *Proc. of VLDB Conf.*, pp. 302–314 (1999).
13. D. Lee and W. W. Chu, CPI: constraints-preserving inlining algorithm for mapping XML DTD to relational schema, *Journal of Data & Knowledge Engineering* **39**(1), 3–25 (2001).
14. R. Bourret, C. Bornhövd, and A. P. Buchmann, A generic load/extract utility for data transfer between XML documents and relational databases, *Proc. of WECWIS Conf.*, p. 134 (2000).
15. H. Sun, S. Zhang, J. Zhou, and J. Wang, Constraints-preserving mapping algorithm from XML-schema to relational schema, Springer-Verlag Heidelberg, Vol. 2480, pp. 193–207 (2002).
16. J. Melton, *Advanced SQL: 1999 – Understanding Object-Relational and Other Advanced Features* (Morgan Kaufmann Publishers, 2003).
17. M. Klettke and H. Meyer, XML and object-relational database systems – enhancing structural mappings based on statistics, *Informal Proc. of WebDB Workshop*, pp. 151–170 (2000).

TOWARDS A PRIVACY FRAMEWORK FOR INFORMATION SYSTEMS DEVELOPMENT

Peter J. Carew and Larry Stapleton*

1. INTRODUCTION

Privacy issues are an increasing concern in our society (Pedersen, 1999). As information and communications technology (ICT) becomes increasingly pervasive, these concerns are being intensified. Privacy is a fundamental human right (UN, 1948) that continues to be violated by intrusive and unethical applications of technology in society and the workplace (cf. Baase, 2003). However, in spite of the ethical concerns and the pivotal role ICT plays in gathering and processing information on people, privacy remains a misunderstood and undervalued concept in ISD.

Although literature addresses many ethical issues associated with intrusive technologies, privacy has received very little attention from ISD researchers, with mainstream literature treating privacy as analogous to data security. Palen and Dourish (2003) note that social and design studies of technology often unknowingly conflate the many functions of privacy and consequently fail to provide sufficient analytical treatment. Current ISD approaches are failing to recognise the significance of privacy issues that affect those involved in the development and deployment of information systems. Privacy violations result in a plethora of negative side effects (e.g. stress, anxiety, resistance) and these may be contributing to the high failure rate of ISD projects.

Although traditional ISD approaches have long recognised the importance of the social element, they continue to focus upon technical issues (Stapleton, 2001). Most methodologies neglect the social interaction and dynamics inherent in the development and deployment process, creating serious problems for the ISD process. Social interaction is a core aspect of ISD, with many processes requiring interaction between various parties. Requirements elicitation (interviews, observation, retrospection, etc.), prototyping, feedback, walkthroughs and numerous other ISD processes require intensive social interaction between analysts, users and other stakeholders. However, there are numerous privacy ramifications pertinent to such interactions, and these ramifications bear heavily upon the

* ISOL Research Centre, Waterford Institute of Technology, Ireland.

success or failure of ISD. For example, individuals will not freely and openly participate in a process, which not only is itself intrusive (examining their work, lives, characters, skills, etc.) but also results in a system that may negatively affect their jobs. Resistance will result, and can include overt or covert behaviours such as conflict, sabotage, coercion, avoidance and withholding or distorting information (Hirschheim and Newman, 1988). Superficially these behaviours can seem irrational and are often treated as such in the literature (see for example ideas of the irrationality of user resistance to change). However, according to a rationality based upon privacy, these behaviours become very rational. The satisfaction of privacy needs leads to effective individual and group functioning (Pedersen, 1999). By implication, ignoring privacy issues during ISD will create serious problems.

Palen and Dourish (2003) advocate the incorporation of privacy rationalities into systems analysis and design. However, while their central concern is with how privacy is conducted in the presence of technology, this can be taken a step further to include the ISD process itself, and not just the product of that process. Palen and Dourish (2003, p. 130) note "fuller treatment of privacy and technology merits a deeper examination of this background [privacy theory]."

This paper expands upon the work of Palen and Dourish, and supplies a fuller treatment of privacy in ISD. It explores the theoretical roots of privacy and a number of related ethical issues posed by technology are highlighted. In particular this paper notes the absence of any coherent framework into which privacy issues relating to ISD are organised. In response to this, a preliminary conceptual framework for interpreting privacy in ISD is presented and subsequently applied to five ISD methodologies in order to provide a reasonably comprehensive assessment of the current state of ISD research in this area.

The next section provides the reader with a brief overview of the key dimensions of privacy as set out in the literature. This provides a basis for the rest of the paper where implications for ISD are set out, and methodologies are assessed.

2. PRIVACY: A REVIEW OF THE LITERATURE

The concept of privacy appears in the literature of several disciplines. There is no universal definition for privacy, and numerous authors have highlighted the difficulties in producing such a definition (cf. Burgoon, 1982; Leino-Kilpi, et al., 2001; Newell, 1998). Theorists argue over whether privacy is a condition, a process or a goal (Newell, 1998). While privacy may be a difficult concept to characterise concisely, the various definitions do have substantial commonalities. One group of definitions emphasise seclusion, withdrawal, and avoidance of interaction with others. The second group puts more emphasis on the control individuals have over their lives.

There are a number of formal models of privacy in the literature, but the theories of Alan Westin (e.g. Westin, 1970) and Irwin Altman (e.g. Altman, 1976) are considered authoritative. Their theories and ideas have stood the test of time and have been the basis of research for many subsequent authors (Margulis, 2003; Pedersen, 1999, 1997; Petronio, 1991). The remainder of this section provides an aggregated overview of some of the core aspects of privacy compiled from the most influential literature. These core aspects set out the theoretical background against which any ISD privacy theory must be constructed.

2.1. Privacy Types, Functions and Mechanisms

People experience and desire several states, or types, of privacy. These include the four identified by Westin (1970): solitude, intimacy, anonymity and reserve. Solitude means to be alone and free from observation by others. Intimacy refers to being alone with a small group to the exclusion of others (e.g. family), and concerns close relationships. Anonymity refers to being unrecognised in a public place – to be inconspicuous and blend into the crowd. Reserve is based on a desire to limit disclosures to others. Pedersen (1997, 1999) extended Westin's model by adding isolation (i.e. using physical distance to be alone) and splitting intimacy into intimacy with family and intimacy with friends. Burgoon (1982) identified the following broad dimensions of privacy: social, physical, informational and psychological.

Privacy functions refer to why individuals seek privacy. Westin (1970) identified four functions of privacy: personal autonomy, emotional release, self-evaluation, and limited and protected communication. Personal autonomy relates to independence and self-identity. It is the desire to avoid being manipulated, dominated or exposed by others. Emotional release refers to freedom from the tensions of social life, and being able to deviate from social norms, roles, rules and customs safely. Self-evaluation refers to integrating experience into meaningful patterns, and the opportunity to plan and assess future actions (i.e. self-reflection and assessment). Limited and protected communication provides the opportunity to share personal information with trusted others. Altman (1976) describes three functions of privacy: interpersonal, the interface to the self and the social world, and self-identity. Pedersen (1997, 1999) empirically identified five basic functions of privacy: contemplation, autonomy, rejuvenation, confiding and creativity. From a systems point of view, Newell (1998) argues that privacy provides an opportunity for restabilisation, system maintenance (i.e. healthy physiological and cognitive functioning) and system development (i.e. towards autonomy and self-actualisation). Individuals may seek to protect their privacy to avoid e.g. embarrassment, harassment, ridicule, shame, scrutiny or discrimination (Shapiro and Baker, 2001).

Behavioural mechanisms are used to achieve a desired level of privacy. These mechanisms include verbal, paraverbal (e.g. tone), non-verbal (e.g. gestures), environmental behaviour (e.g. personal space and territoriality), and cultural norms and customs (Altman, 1976; Pedersen, 1999). Personal space is an invisible zone surrounding the human body, separating people from one another (Leino-Kilpi, et al., 2001). Territoriality refers to a perceived ownership of areas, objects, knowledge or status. These privacy mechanisms function as an integrated system, supporting and substituting each other as appropriate (Altman, 1976).

2.2. Circumstance, Individuality and Culture

Privacy interests vary in both content and magnitude across situations and individuals. What may be trivial to one individual may be significant to another (Shapiro and Baker, 2001). Relevant personal factors include the individual's need for privacy, personal attractiveness, interpersonal skills, personality variables, and ability to use privacy control mechanisms effectively (Pedersen, 1999). Personality variables include extraversion, emotional stability, agreeableness, openness to experience, and conscientiousness (Zweig and Webster, 2003). Gender also can impact on privacy preferences (Newell, 1998; Pedersen, 1999). Some cultures have a stronger preference for privacy and more privacy needs than others (Kaya and Weber, 2003). The need for privacy is universal but manifestations and privacy mechanisms are culturally specific (Margulis, 2003; Newell, 1998). For example, local culture has been shown to affect people's perceptions of crowding (Hall, 1966).

2.3. Intrusion and Privacy Violation

Intrusion essentially is when the desired level for privacy is higher than the actual level being enjoyed (Altman, 1976). Altman's process oriented model for social interaction is useful for further describing what is meant by intrusion or privacy violation. In Altman's theory, privacy has five properties: units of privacy, the dialectic nature of privacy, the non-monotonic nature of privacy, privacy as a boundary control process, and privacy as a bi-directional process (Altman, 1976). Units of privacy refer to the fact that privacy applies at the individual and group levels, and differences exist in privacy dynamics for various social units (Altman, 1976; Margulis, 2003). The units of privacy can be person-to-person, person-to-group, group-to-person or group-to-group (Leino-Kilpi, et al., 2001). The dialectic nature of privacy refers to the fact that individuals continuously change their desire for interpersonal contact. There are two opposing forces at work at all times – one drawing individuals together, and another pushing them apart. Privacy can, thus, be viewed as a dynamic, dialectic process where the need for solitude and the need for interpersonal contact are constantly in opposition. The desired level of privacy depends on which of the two opposing forces is strongest at a given time. The non-monotonic nature of privacy refers to the fact that there is an optimal level of privacy at a given time, and people can have too much privacy (e.g. social isolation) or too little privacy (e.g. crowding). Privacy as a boundary regulation process offers the notion of a flexible barrier between the self and non-self, which can be opened or closed depending on circumstance (Altman, 1976; Petronio, 1991). Finally, privacy can be viewed as a bi-directional process, involving controlling inputs from others and outputs to others.

In terms of Altman's model, intrusion therefore depends on a number of factors. Different social units have different privacy needs (e.g. family, work group, individual), these needs change frequently, and it is possible to have too much or too little privacy. While too much interaction may be experienced as an invasion of privacy, too little may be experienced as loneliness or alienation (Pedersen, 1999). Being forced to interact (i.e. receive input or provide output) beyond the level of interaction desired in a given circumstance is an intrusion as the forced participation implies an attempt to break through the flexible mental barrier (cf. Altman, 1976). The ability to control interactions is essential for privacy management.

Technology has long been recognised as posing a significant threat to the privacy of personal data. The following section looks at some of the privacy related ethical issues in the information society. It shows how ICT is shaping society and the workplace and highlights some dilemmas facing the ethical ISD professional.

3. TECHNOLOGY, PRIVACY VIOLATION AND ETHICAL ISSUES

The human-centred and soft-systems traditions of ISD have an underlying belief that new technologies should be for the benefit of all people and all societies (Gill, 1996; Checkland, 1999). Technology design should, therefore, be concerned not only with technical feasibility (i.e. can we do it?) but also with a social desirability (i.e. do we want to do it?). Privacy is one human factor that must be considered in this context.

Privacy concerns are being fuelled by increasingly intrusive and pervasive technologies in society and the workplace. For example, employers can monitor employees' email, web usage, keystrokes, telephones, transactions, computer screens and location. Monitored employees experience a myriad of negative effects including stress, low morale, anxiety, depression, decreased job satisfaction, lack of involvement, paced work, health problems, lack of control, fear over job loss and a decline in work relationships (Oz, et al., 1999; Ariss, 2002). In addition monitored workers are likely to have higher turnovers, take additional sick days, and work to rule. Furthermore, employees do not passively accept surveillance technologies and may attempt to resist and distort information gathering. Numerous analysts argue that many privacy intrusions are immoral, unnecessary, excessive and self-indulgent voyeurism (cf. Stone and Stone-Romero, 1998).

For many organisations, human-centred issues like privacy are not compatible with organisational life where a competitive, lean rationality and shareholder value are emphasised (Gill, 1996; Brandt and Cernetic, 1998). However, economic calculations frequently fail to place sufficient value on social issues, human capital and the environment. Organisations do have obligations to their stakeholders and shareholders, but they also have ethical responsibilities for protecting the privacy, welfare and dignity of their employees (Stone and Stone-Romero, 1998). Privacy is a fundamental human right, recognised by the United Nations (UN, 1948). Treating it as anything but a human right is globally established as unethical behaviour. A balance should therefore be sought between an employer's legitimate business interests and the employee's legitimate privacy concerns (Ariss, 2002). This includes the ISD process.

The following section presents a developmental framework for analysing privacy dynamics in ISD. This provides a basis for assessing individual and group privacy needs and factors in the context of information systems development and deployment within organisations.

4. A CONCEPTUAL FRAMEWORK FOR PRIVACY IN ISD

This section presents a framework for evaluating and interpreting privacy issues in ISD. The framework includes various privacy factors from the literature that may affect an ISD project. Some factors have been renamed or slighted redefined to correspond more closely to an ISD context (e.g. organisational focus as opposed to a public focus). The framework is structured into four dimensions containing related privacy factors. Table 1 provides a brief overview of all factors in the framework.

The four dimensions have been taken from Burgoon (1982): physical, social, psychological and informational. The physical dimension refers to the environment (e.g. workspace, office, etc.) where an individual may seek physical solitude. Social privacy refers to the freedom to withdraw from, or enter into, interactions with others. Psychological privacy is closely related to the social dimension, but refers only to the individual psyche. Finally, informational privacy refers to an individual's ability to control personal information. These dimensions are not unambiguously discrete, and there is some overlap between them. However, they provide welcome structure for classifying the myriad of privacy factors found in the literature. Each factor has been classified as being: a privacy type (T), a privacy function (F) or a contributing privacy factor (C). Some contributing factors have been identified as (mainly) local to one of the four dimensions whereas others have global significance, affecting all aspects of privacy. The reader should also note how the table anchors each aspect of privacy in the privacy literature, providing details of the sources of each aspect.

Figure 1 provides a graphical representation of the framework. Working from the centre out, the figure shows: the higher order psychological functions of privacy; the four main dimensions of privacy; the types of privacy experienced or desired for each dimension; factors contributing to privacy for each dimension (local); and the outer concentric rings show the global contributing privacy factors. In order to ascertain the levels to which current ISD methodologies consider privacy the following section applies the framework to a selection of ISD methodologies.

Table 1. Overview of privacy framework factors

	Aspect	C	Description
Physical	Environment	T	Individual's physical environment e.g. workplace. Personal space. Crowding may violate. [Refs: Burgoon (1982); Hall (1966); Margulis (2003)]
	Territoriality (Property)	T	Property owned, or perceived to be owned, by an individual. Examples: home, office, equipment, information system, network, etc. [Refs: Burgoon (1982); Altman (1976)]
	Territoriality (Body)	T	A person's body. The most inviolate of territories. Unwelcome contact (tactile, visual, etc.) is a violation. [Refs: Burgoon (1982)]
	Solitude (Physical)	T	Free from direct or remote observation or surveillance. Sanctuary. Seclusion. [Refs: Burgoon (1982); Pedersen (1997, 1999); Westin (1970)]
	Repose	T	Freedom from anything that disturbs or excites. [Refs: Burgoon (1982)]
	Physical Access	C	The ability to physically control access to the self. [Refs: Altman (1976); Burgoon (1982)]
	Sensory and Communication Channels	C	The greater the number of channels of communication (or senses) the less privacy enjoyed. Communications technologies such as email, mobile phones, etc. can also be considered. [Refs: Burgoon (1982)]
	Violator (Humanness and Relationship)	C	The degree of violation depends on the relationship towards the violator. Non-humans (e.g. machines, animals) are not as invasive as humans. Family, friends and those held in low esteem are the least intrusive. Respected humans/acquaintances are the most intrusive. [Refs: Burgoon (1982)]
Social	Intimacy (External)	T	Intimacy with family, friends, etc. external to the organisation. Intimacy requires the exclusion of others to a group. [Refs: Pedersen (1997, 1999); Westin (1970)]
	Intimacy (Internal)	T	Intimacy with colleagues, peers, managers, etc. within the organisation. Intimacy requires the exclusion of others to a group. [Refs: Pedersen (1997, 1999); Westin (1970)]
	Territoriality (Status)	T	Social status or prestige held (or perceived to be held) within the organisation. Dignity. Respect of others. [Refs: Leino-Kilpi et al. (2001)]
	Solitude (Social)	T	Individual freedom from interactions with others. Sanctuary. [Refs: Burgoon (1982); Pedersen (1997, 1999); Westin (1970)]
	Anonymity	T	Not being personally identified/known. To go unnoticed. Blend into the crowd. Freedom from being singled out. [Refs: Pedersen (1997, 1999); Westin (1970)]
	Autonomy	T	Freedom to make own decisions. Protection from interference or coercion by others. Group or individual. [Refs: Pedersen (1997, 1999); Westin (1970); Margulis (2003); Burgoon (1982)]
	Interactions & Communication	C	Control over who to interact with, how frequent these interactions are, how long these interactions are, and the contents/topics of the interactions. [Refs: Altman (1976); Burgoon (1982); Petronio (1991)]
	Formality	C	The degree of formality of the interactions. [Refs: Burgoon (1982)]
	Units	C	Privacy needs exist amongst various social units e.g. individual, group, individual-individual (dyadic), individual-group, group-group, etc. [Refs: Altman (1976); Burgoon (1982); Leino-Kilpi et al. (2001)]
	Personalness of Topic	C	The more personal the content/topic of an interaction, the higher the probability of a privacy violation. [Refs: Burgoon (1982)]

Table 1. (continued)

	Aspect	C	Description
Psychological (Functions)	Self-Identity	F	Development of self-identify/self-ego. Towards self-actualisation. [Refs: Altman (1976); Newell (1998); Westin (1970); Burgoon (1982)]
	Personal Growth	F	Intellectual, emotional and spiritual growth. [Refs: Altman (1976); Burgoon (1982); Newell (1998)]
	Autonomy	F	Freedom to make own decisions. Protection from interference or coercion by others. Independence. [Refs: Pedersen (1997, 1999); Westin (1970); Margulis (2003); Burgoon (1982)]
	Contemplation	F	Self-evaluation and reflection. Learning from and interpreting recent experiences. Planning for future based on experiences. [Refs: Westin (1970); Pedersen (1997, 1999); Altman (1976); Burgoon (1982)]
	Self-Protection	F	Being able to conceal sensitive or potentially harmful information. Security. Diverge from norms/experiment without penalty. [Refs: Burgoon (1982); Shapiro and Baker (2001)]
	Confiding	F	Disclosing information to others in a secure fashion (i.e. to the exclusion of others). Limited and protected communication. [Refs: Pedersen (1997, 1999); Westin (1970); Altman (1976)]
	Emotional Release	F	Relax from social roles and deviate from norms and customs in a protected fashion. Such backstage behaviour allows individuals to escape from conformity and to experiment outside those norms safely. [Refs: Burgoon (1982); Westin (1970)]
	Rejuvenation	F	Recover from life in general (fatigue, stress, etc.) and assaults on self-esteem. Restabilisation. System maintenance. Group or individual. [Refs: Burgoon (1982); Newell (1998); Pedersen (1997, 1999)]
	Creativity	F	Freedom to be creative, innovate, etc. [Refs: Burgoon (1982); Pedersen (1997, 1999)]
Informational	Territoriality (Knowledge)	T	Any action that makes an individual's knowledge/skills less valuable (e.g. codifying, rendering redundant, etc.) may be considered a violation. [Refs: Leino-Kilpi et al. (2001)]
	Reserve	T	Limit knowledge of the self to others. Violations would include any action that exposes sensitive personal information. [Refs: Pedersen (1997, 1999); Westin (1970)]
	Release of Personal Info	C	The extent to which an individual can control the content and amount of personal information being released. [Refs: Burgoon (1982)]
	Distribution of Personal Info	C	The extent to which an individual can control to whom the personal information is being disclosed (individuals, groups, etc.). [Refs: Burgoon (1982)]
	Use of Personal Info	C	The extent to which an individual can control the use of the released personal information (e.g. performance monitoring, direct marketing, etc.). [Refs: Burgoon (1982)]
Global	Control	C	Ability and freedom to alter social interactions and situations. [Refs: Altman (1976); Petronio (1991)]
	Personal Characteristics & Circumstance	C	An individual's personality and other personal traits have a bearing on their need for privacy. Privacy is circumstantial, changing with time and situation. [Refs: Pedersen (1999); Shapiro and Baker (2001); Zweig and Webster (2003)]
	Organisational	C	Organisational culture, factors, management style, stakeholders, etc. affect privacy. [Refs: Stone and Stone-Romero (1998)]
	Cultural	C	Culture affects privacy. Privacy is a universal need, but manifests in different ways in different cultures and societies. [Refs: Hall (1966); Kaya and Weber (2003); Altman (1976); Newell (1998)]
	Societal	C	Society (local and global) has a bearing on privacy. [Refs: Hall (1966); Kaya and Weber (2003); Newell (1998); Shapiro and Baker (2001)]

5. APPLYING THE FRAMEWORK TO FIVE ISD METHODOLOGIES

In this section, the conceptual privacy framework is applied to the following methodologies: UML/UP (OO), SSADM (structured), SSM (soft), Multiview (contingency) and ETHICS (participatory, socio-technical). The methodologies were chosen so as to provide ample coverage of both hard and soft approaches to ISD.

5.1. Method

Jacobson et al. (1999) was chosen as representative for UML/UP. For the remaining methodologies, Avison and Fitzgerald (1995) provide comprehensive overviews. These overviews were supported by one additional authoritative source each: Weaver (1993) for

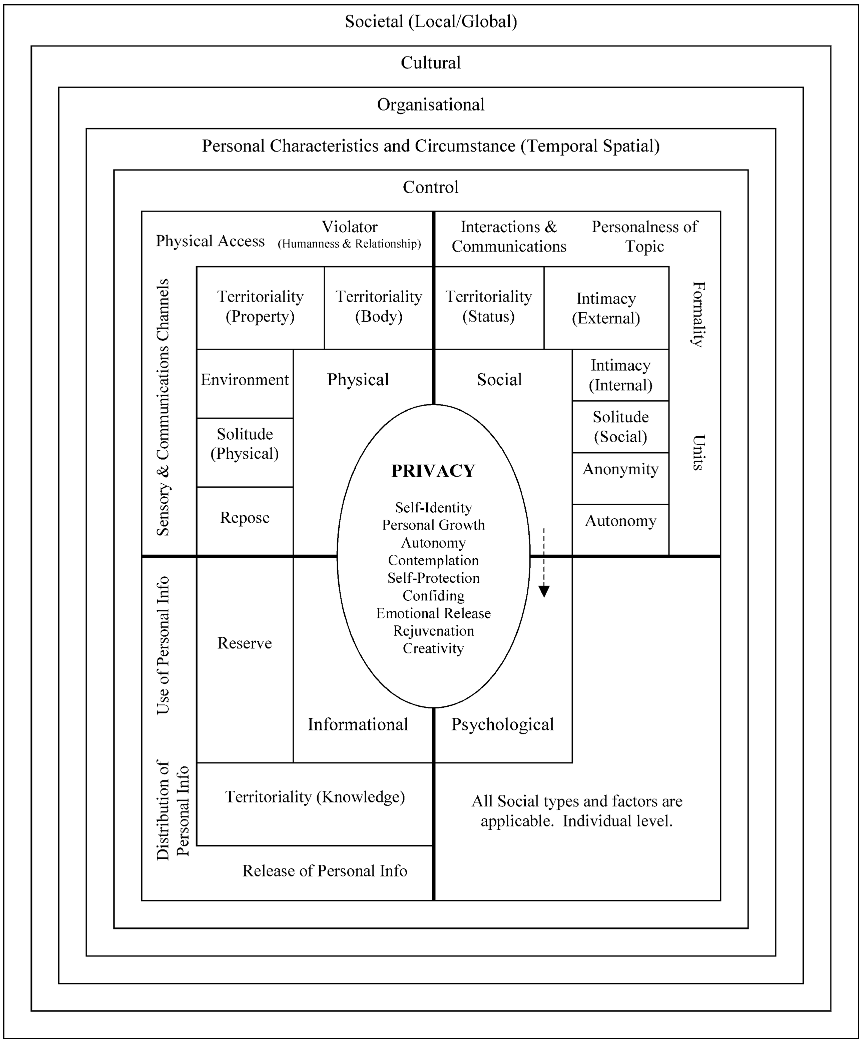


Figure 1. A privacy framework for information systems development.

SSADM, Checkland and Scholes (1990) for SSM, Avison and Wood-Harper (1990) for Multiview, and Mumford (2000) for ETHICS. The study focussed on the main stages of the methodologies, with particular attention to the human-oriented processes. While analysing the methodologies, any stage or description that potentially support or neglect particular privacy factors (types, functions, contributors) in the framework were noted. Note that it is possible for a section to potentially support and neglect a particular privacy factor, depending on how the process described is interpreted or undertaken. As the methodologies are not primarily concerned with privacy, even superficial references to similar or related concepts were noted. The results of the study are summarised in Table 2.

Table 2. Results of privacy analysis

Dimension	Aspect	Class	UML / UP	SSADM	SSM	Multiview	ETHICS
Physical	Environment	T	-	++ / -	+	++	+
	Territoriality (Property)	T	+	--	++	+	++
	Territoriality (Body)	T					
	Solitude (Physical)	T		---			
	Repose	T					
	Physical Access	C		+ / ---			
	Sensory and Communication Channels	C		-			
	Violator (Humanness and Relationship)	C		+			
Social	Intimacy (External)	T					
	Intimacy (Internal)	T	++ / -	+++ / --	++ / -	++ / -	+
	Territoriality (Status)	T		++	++	+++	+++
	Solitude (Social)	T		+ / --			
	Anonymity	T	-	---			
	Autonomy	T		+ / --		++	+++
	Interactions and Communications	C	-	+ / ---		+	+
	Units	C				+	++
	Formality	C	+ / -	++ / -			
	Personalness of Topic	C		+			
Psychological (Functions)	Self-Identity	F				+	++
	Personal Growth	F	+			+	++
	Autonomy	F		--		+++	++
	Contemplation	F					+
	Self-Protection	F		-		+	
	Confiding	F					
	Emotional Release	F					
	Rejuvenation	F					
	Creativity	F	+	+		+	+
Informational	Territoriality (Knowledge)	T	+	--	++		++
	Reserve	T		+ / ---	+		
	Release of Personal Information	C		---		-	
	Distribution of Personal Information	C		---		+ / -	
	Use of Personal Information	C		---		-	
Global	Control	C	-	+ / ---		+++	+++
	Personal Characteristics and Circumstance	C			+	+	++
	Organisational	C	+	+++	+++	++	++
	Cultural	C			+++		
	Societal	C			++		+

Positive Orientation: ++++ (Very Strong), +++ (Strong), ++ (Some), + (Weak)
Negative Orientation: ---- (Very Strong), --- (Strong), -- (Some), - (Weak)

5.2. Results

The Unified Modeling Language (UML) is a graphical modelling language for specifying systems from an object-oriented perspective. It is important to recognise that the UML itself is not a methodology, but simply a modelling notation that provides a variety of modelling diagrams but does prescribe underlying processes for developers to follow. However, the authors of UML have offered the Unified Software Development Process (or simply Unified Process) as a suitable methodology (cf. Jacobson, et al., 1999). As UML is oriented towards specifying technical functionality, it does not consider any issues pertinent to people or privacy. Therefore, the Unified Process (UP) is considered in its stead. In terms of the framework, UML/UP performs poorly, paying no discernable attention to privacy issues. The Unified Process is still technically oriented, and does not consider the social aspects of ISD in any depth. For example, Jacobson et al. (1999, p. 97) state that "management is responsible for non-technical risks." So, according to UML/UP, developers have no responsibility for anything non-technical in the ISD process. Does this, for example, preclude them from considering the ethical ramifications of the systems they are developing or, indeed, how they develop them? This viewpoint is deeply disturbing, especially when presented by a popular, and widely taught, contemporary approach to ISD.

SSADM provides developers with detailed rules, activities, deliverables and guidance for all stages of the project lifecycle. In terms of the privacy framework, SSADM only appears to pay adequate attention to the organisational factors that appear in the global dimension. Many other aspects appear to go against privacy, with users being unable to control access, interactions, anonymity or personal information.

Soft Systems Methodology (SSM) is 7-stage/4-acitvities systems thinking approach to handling real world, unstructured problems. It was originally organised into a seven stage model and later developed into the four activities model (Checkland, 1999). In either case the basic premises and approaches were much the same, focussing as they do upon 'soft' aspects of ISD. Under the privacy framework, SSM did not fare overly well, but this may be partly due to its high-level description. It does address territoriality (conflicts, etc.) to some extent, but its privacy related strengths seem to be in recognising organisational, cultural and societal issues. Given Checkland's deep concern with environmental issues it is not entirely surprising to find that the global dimension of the privacy framework receives most attention in SSM (Checkland, 1999).

Multiview is a 5-stage contingency approach that advocates flexibility in choice of methodology and approach to suit heterogeneous situations. It considers both the human and technical aspects, and is inherently non-prescriptive. Under the privacy framework Multiview performs relatively well, but many factors are not considered. The approach recognises: autonomy, control, territoriality, organisational, and environmental aspects (to varying degrees). Unlike the other methodologies, privacy is addressed explicitly but only in terms of data access and security.

ETHICS is a 6-stage methodology based on the participatory approach to information systems development. It takes a socio-technical view that successful technology should fit closely with social and organisational factors. ETHICS performs respectably under the framework, but many privacy factors are not addressed. The methodology recognises: autonomy, control, territoriality, personal characteristics, organisational factors and the need for individual growth and development.

Figure 2 shows a high-level classification of the five methodologies based on the privacy framework. No methodology appears to consider physical or informational privacy sufficiently (note that data security alone does not sufficiently address informational privacy). This is significant as technology is a major driver of many privacy violations e.g. surveillance systems that gather vast amounts of personal information and remove opportunities for going unnoticed. It is apparent that privacy is a complex, deeply sensitive human-centric issue that has largely been overlooked by ISD methodologies.

6. LIMITATIONS OF STUDY

There are a number of limitations to the critique of the methodologies presented. None of the methodologies reviewed is primarily concerned with the complex notion of privacy

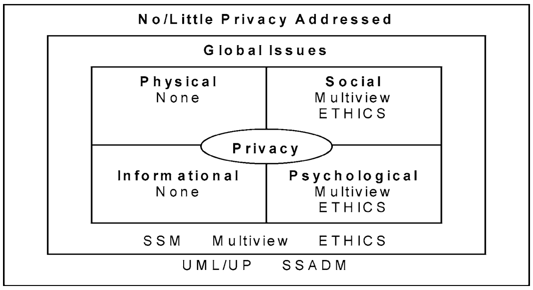


Figure 2. Summarised assessment of methodologies under the privacy framework.

as embodied by the framework. Also, with the exception of SSADM, none of the methodologies are overly prescriptive, allowing for some degree of flexibility. Therefore, although the methodologies don't explicitly include or consider privacy, they do not fundamentally preclude it either. Nevertheless, the fact that privacy is not mentioned or addressed at all (except by Multiview in terms of data security) is of some concern. The results presented in Table 2 are also necessarily subjective and compiled from different texts. However, subjective feature-based analyses do exist in the literature (cf. Galliers, 1992; Avison and Fitzgerald, 1995, p. 465). Also, comment was passed only on factors that showed reasonably strong orientation in the analysis results. Finally, it is readily apparent that the privacy framework presented here is still in a developmental state. In spite of these limitations it is reasonable to say that none of the reviewed methodologies address privacy sufficiently.

7. DISCUSSION AND CONCLUSIONS

Privacy is often misunderstood to be synonymous with secrecy and data security. However, privacy is mainly a social control process that serves a myriad of higher-order social and developmental functions. Organisations should try to balance their need for information against employees' expectations of privacy (Stone and Stone-Romero, 1998). Through doing so, they can realise numerous indirect benefits in terms of ICT (e.g. reduced user resistance and ISD failure) and human capital (e.g. improved creativity).

It is apparent that little is known of privacy dynamics in ICT and especially ISD. The future direction of this research involves a field study using an instrument based on a revised form of the framework presented in this paper. The ultimate goal of this work is to develop a set of privacy-based rationalities for improving the ISD process, making it both more successful and humanitarian. In the meantime, ISD professionals and researchers can use the preliminary framework presented here to assess appropriate methods and practices as regards their privacy orientation.

In conclusion, this paper has highlighted the importance of privacy in the context of ISD. Privacy theory was briefly reviewed and a provisional conceptual framework for interpreting privacy in ISD presented and applied to five ISD methodologies. The results indicate that privacy is not being considered as an important socio-technical factor in ISD, and the serious ramifications of this were highlighted. The ongoing research aims to redress this major gap in the ISD literature.

REFERENCES

Altman, I., 1976, Privacy: A conceptual analysis, *Environment and Behavior* **8**(1):7–29.

Ariss, S. S., 2002, Computer monitoring: benefits and pitfalls facing management, *Information & Management* **39**(7):553–558.

Avison, D. E., and Fitzgerald, G., 1995, *Information Systems Development: Methodologies, Techniques and Tools*, Second Edition, McGraw-Hill, London.

Avison, D. E., and Wood-Harper, A. T., 1990, *Multiview: An Exploration of Information Systems Development*, Blackwell, Oxford.

Baase, S., 2003, *A Gift of Fire: Social, Legal and Ethical Issues for Computers and the Internet*, Second Edition, Prentice Hall, NJ.

Brandt, D., and Cernetic, J., 1998, Human-centred approaches to control and information technology: European experiences, *AI & Society* **12**:2–20.

Burgoon, J., 1982, Privacy and communication, in: *Comms Yearbook 6*, M. Burgoon, ed., Sage, CA, pp. 206–249.

Checkland, P., and Scholes, J., 1990, *Soft Systems Methodology in Action*, Wiley, Chichester.

Checkland, P., 1999, *Soft Systems Methodology: A 30-Year Retrospective*, Wiley, Chichester.

Galliers, R. D., 1992, Choosing information systems research approaches, in: *Information Systems Research*, R. Galliers, ed., Blackwell, Oxford, pp. 144–162.

Gill, K. S., 1996, The human-centred movement: The British context, *AI & Society* **1996**(10):109–126.

Hall, E. T., 1966, *The Hidden Dimension*, Doubleday, New York.

Hirschheim, R., and Newman, M., 1988, Information systems and user resistance: Theory and practice, *The Computer Journal* **31**(5):398–408.

Hirschheim, R., and Newman, M., 1991, Symbolism and information systems development: myth, metaphor and magic, *Information Systems Research* **2**(1):29–62.

Jacobson, I., Booch, G., and Rumbaugh, J., 1999, *The Unified Software Development Process*, Addison-Wesley, MA.

Kaya, N., and Weber, M. J., 2003, Cross-cultural differences in the perception of crowding and privacy regulation: American and Turkish students, *Journal of Environmental Psychology* **23**(3):301–309.

Leino-Kilpi, H., Valimaki, M., Dassen, T., Gasull, M., Lemonidou, C., Scott, A., and Arndt, A., 2001, Privacy: A review of the literature, *International Journal of Nursing Studies* **38**(6):663–671.

Margulis, S. T., 2003, On the status and contribution of Westin's and Altman's theories of privacy, *Journal of Social Issues* **59**(2):411–429.

Mumford, E., 2000, A socio-technical approach to systems design, *Requirements Engineering* **5**:125–133.

Newell, P. B., 1998, A cross-cultural comparison of privacy definitions and functions: A systems approach, *Journal of Environmental Psychology* **18**(4):357–371.

Oz, E., Glass, R., and Behling, R., 1999, Electronic workplace monitoring: What employees think, *Omega* **27**(2):167–177.

Palen, L., and Dourish, P., 2003, Unpacking "privacy" for a networked world, *Proceedings of the conference on Human factors in computing systems*, Ft. Lauderdale, Florida, 129–136.

Pedersen, D. M., 1997, Psychological functions of privacy, *Journal of Envir. Psych.* **17**(2):147–156.

Pedersen, D. M., 1999, Model for types of privacy by privacy functions, *Journ. of Envir. Psych.* **19**(4):397–405.

Petronio, S., 1991, Communication boundary management: A theoretical model of managing discourse of private information between marital couples, *Communication Theory* **1**(4):311–335.

Shapiro, B. and Baker, C. R., 2001, Information technology and the social construction of information privacy, *Journal of Accounting and Public Policy* **20**(4–5):295–322.

Stapleton, L., 2001, *Information Systems Development: An Empirical Study of Irish Manufacturing Firms*, Ph.D Thesis, University College Cork.

Stone, D. L. and Stone-Romero, E. F., 1998, A multiple stakeholder model of privacy in organizations, in: *Managerial Ethics: Moral Mgmt. of People and Processes*, M. Schminke, ed., Erlbaum, NJ, pp. 35–39.

UN, 1948, *United Nations Universal Declaration of Human Rights*; http://www.un.org/Overview/rights.html.

Weaver, P. L., 1993, *Practical SSADM Version 4*, Pitman, London.

Westin, A., 1970, *Privacy and Freedom*, Atheneum, New York.

Zweig, D., and Webster, J., 2003, Personality as a moderator of monitoring acceptance, *Computers in Human Behavior* **19**(4):479–493.

IS DEVELOPMENT AS THE MUTUAL ADAPTATION OF TECHNOLOGY AND BUSINESS PROCESS

Bendik Bygstad*

1. INTRODUCTION

Information systems development is a truly innovative process. Sometimes it is focused solely on constructing a high quality software product, but often it is part of a business change, triggered by external or internal pressures. In the latter case, we may analyse the development as a socio-technical innovation, creating both a software product and a redesigned business process.

How should these two innovation processes be managed? Usually, they are organised in three main steps. First, the business process is analysed and redesigned, and a requirements specification is written to determine the functionality of the software system. Then the software system is designed and constructed, controlled within a software development framework. And lastly, the information system is implemented into the business process, along with the training of users.

This is not always a good strategy. It often leads to misalignments between the organisational context and delivered technology, because both the organisation and the technology may change during the project (Leonard-Barton and Sinha, 1993) and because it is difficult to specify the requirements at an early stage (Jacobson et al., 1999). The real innovation of an IS project is not the software, but the working solution after implementation (Leonard-Barton, 1988). This implies that it is hard to predict the organisational impact of a new information system. It should be developed and implemented in a way that allows for a stepwise learning and adaptation process. Therefore, from a theoretical perspective, a better form is *mutual adaptation* (Leonard-Barton 1988; Majchrzak et al., 2000), where the business process design and the software development are dynamically coupled, and the implementation process is fully integrated.

While theoretically attractive, the mutual adaptation approach is not well understood (Majchrzak et al., 2000) and challenging to achieve in practice (Giaglis 1999). Therefore, there is a need to study this phenomenon over time in order to analyse its temporal and man-

* The Norwegian School of IT, Oslo, Norway, Bendik.Bygstad@nith.no.

agement aspects. This paper explores mutual adaptation in modern, iterative ISD projects, addressing the following three research questions:

- What are the dynamics of mutual adaptation in ISD projects?
- To what extent is mutual adaptation possible to plan and control?
- Should planned mutual adaptation be part of a development project, or is it outside the reasonable scope of an ISD project?

The rest of the paper is structured as follows. In Section 2 the key concepts are discussed. The research method is presented in Section 3. In Section 4 the case is presented. Findings are discussed in Section 5, while Section 6 briefly concludes.

2. KEY CONCEPTS: MUTUAL ADAPTATION AND ORGANISATIONAL MECHANISMS

2.1. Mutual Adaptation

Mutual adaptation between technology and organisation is necessary because there will always be misalignments between a technology and a business environment, which must be solved, either by changing the technology or the environment – or preferably both (Leonard-Barton, 1988). Mutual adaptation happens both in the development and in the implementation of information systems. Changing the organisation means to change its formal structure or its overall work process or work practices. Changing the technology means to change substantial attributes of the software. Some key features of mutual adaptation are:

The mutual adaptation is a dynamic process. Because of the integrated nature of this process, it is a mistake to separate the creation of the product from the implementation. The mutual adaptation is an ongoing process, fuelled by continuous technological and business change (Leonard-Barton, 1988). A change in the organisation may trigger a change in the software, which in turn may trigger a new organisational change. Thus, while a *requirements specification* represents a one-way process (from the organisation to the technology) and the *organisational implementation* of technology represent a one way process in the opposite direction (from technology to the organisation) – mutual adaptation is a two-way process.

Mutual adaptation is part of the power struggle in any organisation, and is connected more to work practices than to formal structures. In IS related change projects, a formal approach of introducing a new organisation chart without changing the work practices, has often proved unsuccessful. Thus, mutual adaptation requires careful analysis of the congruence between existing and desired work practices (Brynjolfson et al., 1997).

Mutual adaptation includes the creation and transfer of knowledge, by establishing strong ties between two different communities of practice (Garrety et al., 2001). In development projects, the IT people need to understand the business issues, and the business people have to understand the capabilities and constraints of the technology. The solution is often negotiated around "boundary objects" (for example user interfaces), which both communities try to influence.

Mutual adaptation is emergent, in the sense that is it difficult to plan in detail. The best solution is not an intellectual construction (like a written specification), but a negotiated situation. Thus, a purely design-oriented approach, as presented by (Jacobson et al., 1995), is less likely to succeed. However, timing is crucial, because there is usually a relatively short "window of opportunity", where both developers and users are willing to invest the effort to change the work process and the technology (Tyre and Orlikowski 1994). Thus, the mutual learning and adjustment must happen before both sides lose resources and interest.

2.2. Different Views on the Question of Controlling Mutual Adaptation

Leonard-Barton (1988) found that mutual adaptation should be controlled through successive large and small cycles of alignment. She recommended that developers took some responsibility for the user adaptations, and that the business people should share some of the uncertainty and risk associated to new technology. She also found that the mutual nature of change required that strategic choices should be driven not only from top management, but also from the knowledge core of the organisation.

In contrast, the concept of *technological drift* was introduced to suggest that the mutual adaptation cannot be planned and managed (Ciborra, 2000). Technological drift describes a discrepancy between plan and outcome, in respect of the implementation of information technology, in which the organisational implementation of technology is basically unpredictable and unmanageable. Ciborra asserts that the solution cannot be more managerial control, which has proven to be part of the problem, because more control will lead to more side-effects.

The software engineering research has not addressed the question directly, but asserts implicitly that mutual adaptation can be planned and managed. The most well-known mechanisms are evolutionary prototyping, and iterative and incremental delivery (Boehm, 1988; McConnell, 1996). These mechanisms are integrated in current software engineering frameworks like Rational Unified Process (Jacobson et al., 1999), OPEN (Henderson-Sellers and Unhelkar, 2000) and Microsoft Solutions Framework (Microsoft, 2001). Although these frameworks are primarily designed to reduce technical risk, they also address the risk of misalignment between the organisation and the software solution. Therefore, iterative and incremental IS projects using these frameworks represent an interesting arena to study this mutual adaptation, because they allow both the developers and the business people to learn, and to act on new learning, during the process.

2.3. Organisational Mechanisms for Mutual Adaptation

If mutual adaptation can be controlled, some kind of management intervention is needed. Several mechanisms have been suggested, like project teams, steering groups, stakeholder webs (Coakes and Elliman, 1999), change agents (Markus and Benjamin, 1997), relationship managers, and organisational mechanisms to support user innovations (Nambisan et al., 1999). To succeed in achieving mutual adaptation an organisational mechanism should at least satisfy three aspects:

- Facilitate learning between different knowledge communities: The emergent nature of mutual adaptation requires that both groups learn from each other (Leonard-Barton, 1988).

- Allow the creation of new knowledge: The dynamic nature of mutual adaptation requires that both sides develop a thorough understanding of each others domain areas and constraints, to be able to create new solutions (Garrety et al., 2001).
- Provide the necessary authority to actually implement the necessary changes in both processes: If this is not the case, the result may be good ideas, but no mutual adaptation, which is dependent on changed work practices (Brynjolfson et al., 1997).

3. RESEARCH APPROACH

The challenge of mutual adaptation was investigated empirically through a longitudinal case study of a software development project, aimed at supporting the financial audit process at the Norwegian Office of the Auditor General. The research approach was Longitudinal Process Research (LPR), which aims to study organizational change over time, through intensive research in the actual context (Pettigrew, 1985, Ngwenyama, 1998). LPR focuses on building theories strongly embedded in the context of study. Important criteria for data collection are (Ngwenyama, 1998):

- Ongoing engagement with the research site, to observe changes over time.
- Participant observation, to contextualize and make sense of observations.
- Multiple data sources, to record different interpretations of events, and to ensure validity of findings.

The case, a software development project using the iterative and incremental software process Microsoft Solutions Framework, was researched for 16 months, using several techniques for data collection: An initial workshop with the most important stakeholders was held to construct a time line for the project. Project managers, developers and users were interviewed at three intervals. Project meetings were observed to understand how problems were conceptualised and how decisions were made, and a vast amount of project documentation (plans, models, reports) was collected. Data was coded following the guidelines of Miles and Huberman (1994). After the videotaped interviews were summarized and registered into an Atlas database, texts were coded with *in vivo* codes, using only domain terms. The large volume of project documentation was coded the same way.

Ngwenyama (1998) suggests three modes for data analysis: Comprehensive analysis helps to identify underlying structures and patterns of the organizational process. Temporal analysis helps contextualizing findings by placing events and situations in a narrative structure. And member verification ensures that the case description and researcher's interpretation are considered correct and meaningful to the organizational actors.

The case was analyzed in the following steps.

A time line was constructed, documenting participants and technology in each iteration. Then each iteration of the project was analyzed in detail, while in parallel looking for repeating patterns and mechanisms. Looking for mutual adaptation, related terms were recoded, and analysed. Mutual adaptation was coded to situations, as described in Section 2, where mutual learning between business people and IS people were observed, where the creation of new knowledge was documented, or where there was a direct or indirect change of process structure.

The principle of *member verification* (Ngwenyama, 1998) was followed. After the initial workshop with important stakeholders, the timeline with important events and stakeholders was sent to the participants for verification. After a year studying the project, a preliminary case description was written, and commented by stakeholders. And lastly, a validation meeting was held with project managers, developers, auditors and managers, assessing and commenting on the final case description.

4. THE CASE STUDY

4.1. Business Context

The Office of the Auditor General (OAG) is instituted in the Norwegian Constitution. Its main tasks are to audit the central government, ministries and their agencies' accounts, and to monitor the ministers' administration of national interests in state-owned companies and banks. The OAG is based in Oslo, where most of the 450 employees are working. The majority of employees are auditors. Auditors perform three types of audit: Financial audit, corporate control and performance audit. The project described in this paper developed a new IT solution for the financial audit, which consists, somewhat simplified, of the following three main steps:

- *Planning the audit*: Receive budget and accounts for the entity. Then classify into account areas, decide materiality limits and assess risks.
- *Execute the audit (visiting the audited entity)*: Produce and follow audit programme, and document the findings.
- *Reporting*: Document conclusions for assessment of entity.

In 1998, a Methodology Group consisting of financial auditors was appointed to improve and standardize the financial audit methodology. Important elements in the methodology were which steps to perform, and defining the areas and scope of the use of individual judgement. New process oriented standards were implemented in 1999, but it was hard to get acceptance in the organisation. The main reasons for this were assumed to be strong local auditing cultures in the different departments, and lack of IS support.

4.2. The PROSIT Project ("Process Oriented IT Audit Support")

A feasibility study conducted by auditors and the IT manager in 1998 produced a requirements specification. In 1999 it was decided to develop a tailor-made system. The main objectives for the project were to:

- Provide a modern and efficient tool to support a standardized audit process
- Support management control of resource use and audit quality
- Support organisational learning through making audit report screens available for the whole organisation

As their process framework the IT department had chosen the Microsoft Solutions Framework (MSF), an iterative and incremental framework (Microsoft, 2001). The project organisation was designed to align the organisation and the development, with a strong management and quality focus. An audit department was appointed to be owner of the system, and the other audit departments were represented both in the steering group and the Methodology Group. In March 2000 the project was organised according to the principles of MSF. Six teams were established; customer, user, release, process, development and QA teams, each of them with a team leader. Following MSF, the team responsibilities and tasks were described in detail, and the teams were empowered to make decisions within their areas. The solution was developed through five iterations, from March 2000 to July 2002 when the system was set into production. Iterations six and seven were carried out from July 2002 to June 2003. The next section analyses these seven iterations, from a mutual adaptation perspective.

4.3. Case Findings and Analysis

Table 1 summarises the case findings, structured according to the MSF iterations. The analysis focuses on the mutual adaptation between the business organisation and the ISD project, and the role (and direction) of the organisational, mediating mechanism. In particular, communication between the two main groups, the creation of new knowledge and the authority of the mechanism is investigated. The content of Table 1 will be described and discussed in the next section.

4.3.1. Iterations 1 and 2

The first iteration concentrated on detailing the specification. Several extensive workshops were held, including up to 40 user representatives, selected by the departments. The specification needed much detailing, and there were long, open workshops. Coordinating the detailing of use cases, design of screens, and programming and testing of these was difficult with many inexperienced participants. The second iteration developed most of the functionality for audit planning, but the project experienced both cooperation and technical problems.

Facilitating mutual learning. The broad workshop approach was chosen to establish a knowledge platform for both the auditors and developers. A member of the Steering Committee commented: "The situation was challenging. We did not really know what we wanted, because we had never seen such a system before. On the other hand, the developers did not understand the audit methodology."

Unfortunately, the approach did not work very well, partly because the complexity of the tasks was too great to handle for the inexperienced participants. Also, some line managers did not prioritise the workshops.

Allowing for knowledge creation. As planned, the functionality for audit planning was programmed and assessed. The quality of the GUIs was regarded as being too low, and the specification needed more detailing

Authority to influence on the processes. At this stage, the focus of the project was mainly on technology related issues. Thus, several decisions concerning the database solution and the design of GUIs were taken. Audit process issues were not really addressed, but it was seen that the audit process description was not detailed enough for the PROSIT design.

Table 1. Adaptation in the PROSIT project

PT = Project Team IT = Implementation Team
UT = User team MG = Methodology Group

Iteration	1 March-June 2000	2 July 2000-March 2001	3 March - Sept 2001	4 June 2001-April 2002	5 January-June 2002	6 July 2002-Jan 2003	7 Jan - June 2003
Business Process	PROSIT approved Strong manage ment support	Line managers not priori-tising project	Line managers and auditors more involved in PROSIT	Audit process changed, as a result of PROSIT design activities	Aware-ness and training All depts made their own impl. plan	Courses, focusing on the use of PROSIT	PROSIT is widely used, as integrate part of the audit process
Organi-sational Mecha-nism - and main direction of adap-tation	PT and UG work shop ↓	PT and UG work shop ↓	↑ MG and PT ↓	↑ MG and PT ↓	↑ PT and IT	↑ PT and IT	↑ PT and line mana-gers
ISD project	Require-ments spec. was detailed	Rel.0.1: Audit planning database	Rel 0.2: Risk and prority	Rel 0.3 Report tool	Rel 1.0. PROSIT set into produc-tion	Rel 2. Mainly change requests	Rel 3. Manage-ment informa-tion
Results of interation	Process unstruc-tured Schedule not followed.	Cost overrun GUIs not satis-fying	On time, on target New GUI standard	On time, on target Align-ment between IS and business process	Success-ful imple-men-tation. Some resis-tance.	Process metho-dology imple-mented in the organi-sation.	PROSIT part of work routine.

At the end of iteration 2 the steering committee was worried. The cost was higher than estimated, productivity and quality lower, and the cooperation between auditors and developers was not tight enough. In a critical evaluation report a number of measures were specified: The process was given more structure, and the budget was increased. Also, the Methodology Group was drawn heavier into the project.

4.3.2. Iterations 3 and 4

The objective for the third iteration was to develop a tool and methodology for risk and priority, and to provide the interface to the audit accounts. The objective in the fourth iteration was to produce tools for report findings. A number of successes appeared.

First, the (MSF) roles worked much closer in this iteration. In the first iteration the user team had defined the requirements, and handed these over to the architect, who then had detailed the architecture for the developers. This created many misunderstandings and delays. Now, the teams worked closer, the auditors often sitting together with developers and prototyping.

Second, the use case descriptions were structured and standardized. From these, the screen designs were solved nicely; showing the steps in the auditing process in the left frame of the screen. This GUI structure allowed for a stronger link between the audit process and the PROSIT design.

Facilitating mutual learning. In these iterations the Methodology Group became heavily involved in the workshops and prototyping. The developers got more detailed feed-back on the process, while the auditors learned more about the complexities and constraints of the technology.

Allowing for knowledge creation. Conceptually, the group was able to connect the two processes on a practical level: They could translate the audit process into small steps which could be modelled. Detailing the use cases, it was often discovered that the auditing process was not sufficiently described. Commented one auditor: "The project has improved the methodology in general, because many issues were not really specified. When this was put into screens and procedural steps, these issues had to be decided. The methodology committee then wrote a proposal, which was later decided by the executive group."

Authority to influence on the processes. The initial role of the Methodology Group was to structure the audit process, and it was later drawn into the project to provide information for the requirements specification. This was indeed an ideal situation for mutual adaptation. Members of the group worked in the project group, and influenced the project greatly. At the same time, they were also in the position to change the rules of the audit process.

4.3.3. Iterations 5 to 7

In the autumn 2001 all departments were asked to make their own implementation plan, focusing on which problems were most important to solve. All users were taken through a two-day course held in the departments. In particular, it was focused on audit planning, risk assessment and accounting requirements. After a two month pilot period PROSIT was set into full production in July 2002. Implementation was successful, despite some resistance, mainly among very experienced auditors who felt the system narrowed their space of individual judgement. During spring 2003 the system and organisation stabilised. After the successful iteration 6, focusing on satisfying change requests, iteration 7 focused on management information.

Facilitating mutual learning. The extensive training activities in iteration 5 were, in spite of some critics (as noted in the section above), a success for the project, creating awareness and acceptance in the audit departments. The project organisation was reduced in size, but it was maintained during the first year of operation, working on iterations 6 and 7. Providing user support and training, the learning space was kept open, because errors and flaws could be corrected quickly.

Allowing for knowledge creation. Since each department made and executed their own implementation plan, there was certainly some space for local adjustments, but mainly on minor issues. After implementation the software change requests had to be assessed by a committee, and prioritised. Thus, knowledge creation was on a smaller scale, producing a large number of change requests. These were handled in iterations 6 and 7.

Authority to influence on the processes. The implementation team was, naturally, focused on implementing PROSIT into the organisation, and aligning the work process to the system. The main responsibility for implementation was laid on line managers. On the other hand, the ability to influence the development team was now reduced, partly because of budget constraints, and partly because of the bureaucratisation of the change request handling. The window of opportunity for mutual adaptation seemed to be closing.

5. DISCUSSION

5.1. What Are the Dynamics of Mutual Adaptation in ISD Projects?

Mutual adaptation implies that it is possible to change structural properties of both the organisation and the information system. Majchzrak et al (2000) suggest that it is not the nature of structures (whether it is technological, political or social) that constrains the adaptation process, but rather the malleability of the structure. As shown by Tyre and Orlikowski (1994) this malleability may vary over time, creating 'windows of opportunity', where mutual adaptation is possible. The PROSIT case illustrates that the malleability of the information system and the business process varies over time. As illustrated in Figure 1, this may be described in three phases.

As documented in the software engineering research (Jacobson et al., 1999), the malleability of the software product is high at the beginning of a development project, becoming gradually lower. At the beginning of the PROSIT project it was fairly easy to change requirements, while at the end of the project it was much harder and more expensive. Thus, the curve in Figure 1 is falling gradually, as more modules of the system are produced. With the iterative and incremental MSF process, the decrease in malleability is not as sharp as in waterfall projects, so there is considerable space for change midway in the project.

The malleability of the business process is generally lower, as it is embedded in institutional practice and culture. During the PROSIT project the malleability was slowly increasing; the resistance against the specified auditing process was diminishing, mainly as a result of the PROSIT project. After implementation malleability is gradually decreasing as the changed business process is routinized by the new information system.

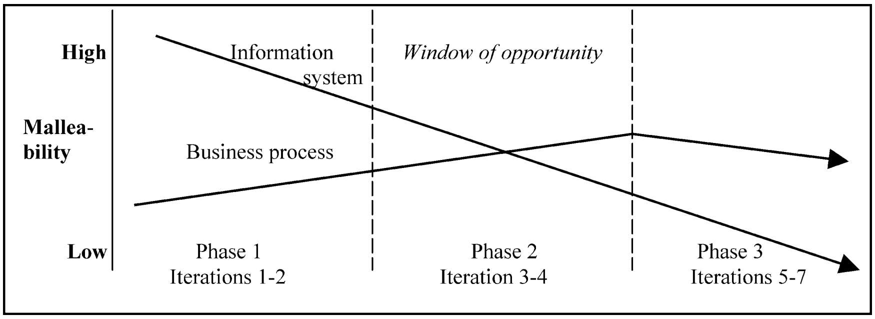


Figure 1. The malleability of the information system and the business process.

In the first phase the direction of adaptation goes from the business process to the IS, in the form of requirements specification: The malleability of the organisation is low, while it is high in the technology. In the last phase the direction is opposite; the finished information system is implemented into the organisation as a change lever. However, in the second phase the adaptation happened both ways. This window of opportunity is created because the PROSIT system is in the process of stabilising, while still being malleable. At the same time the business process is slowly restructuring because of the ISD project.

5.2. To what Extent is Mutual Adaptation Possible to Plan and Control?

A window of opportunity is not sufficient to create mutual adaptation. As argued in Section 2, an organisational mechanism is needed to facilitate mutual learning, allowing for the creation of new knowledge and having the needed authority to implement changes in both processes. The role of the organisational mechanisms in the PROSIT case is summarised in Table 2.

It is important to recognize that participants in a long, socio-technical project like PROSIT are conscious of the emergent nature of such projects, and of the need to establish and observe feed-back loops. MSF iterations are the natural frame for doing this assessment. In the PROSIT project they experimented with different mechanisms, abandoning those that did not work, and reusing those that did. Thus, although user workshops have proven to be successful in many IS projects (Braa and Vidgen, 1997), they were not so in the PROSIT iterations 1 and 2. This mechanism was therefore abandoned, and instead the role of the Methodology Group was expanded.

Table 2 shows that all the organisational mechanisms facilitated mutual learning, but only the Methodology Group allowed for the creation of new knowledge and possessed the authority to change both processes. Thus, in a window of time it was possible to control the mutual adaptation in the PROSIT case.

The extent of this control should not be exaggerated. As has been emphasized in the interpretation of the case, this is not a question of pure design, because the result of the mutual adaptation is basically outside the scope of detailed planning. It is emergent, in the sense that solutions are developed as learning and knowledge creation proceeds in a good project. What is controllable is the organisational mechanism to facilitate this.

Table 2. Attributes of organisational mechanisms in the PROSIT project

Organisational mechanism for mutual adaptation	**Facilitating mutual learning**	**Allowing for the creation of new knowledge in both processes**	**Authority to influence on both processes**
User workshops	Yes	Yes, but mainly focused on how business requirements influence the IS	No, only the ISD process
Methodology Group	Yes	Yes	Yes
Implementation Team	Yes	Yes, but mainly focused on how IS attributes influence the business process	Partly, but mostly on the business process

What would be the area of validity for this organisational mechanism? Of course, one single case study cannot answer this question, but grounded in Longitudinal Process Research (LPR) it presents some interesting findings. First, as the case shows it is possible, in a window of opportunity, to facilitate for successful mutual adaptation. Second, the organisational mechanism (represented by the Methodology Group) could be copied and used by other businesses.

In line with LPR it should be emphasized that the findings are context sensitive. In the PROSIT case the project success was dependent on the Methodology Group, consisting of young and ambitious auditors who realized how the information system could contribute to restructure the business process. They were allowed to spend quite a bit of time in close cooperation with the developers, building a knowledge base. Obviously, this is often not possible. Also, the PROSIT project was run in a well funded public agency, with moderate time pressures. This allowed the project organisation to experiment with different mechanisms and to extend its project period to one year of operations after implementation. Again, in a more competitive business environment, this might not be an option.

On the other hand, research has long established that close cooperation and knowledge sharing between developers and business people are key success factors (Reich and Benbasat, 2000). Even in a more competitive business context there are equal needs to innovate solutions that include changes in both technology and organisation.

5.3. Should Planned Mutual Adaptation be Part of a Development Project, or Is it Outside the Reasonable Scope of an ISD Project?

Should mutual adaptation be part of the ISD project, or be addressed somewhere else (and higher) in the organisation? Some researchers (for example Sauer and Willcocks, 2004) contend that this challenge is impossible to solve in a project context, and have suggested that this should be the responsibility of a new kind of senior manager, an "organisational architect".

Acknowledging that the ISD project manager cannot be solely responsible for mutual adaptation, there are several arguments against this: Organisational change is everybody's responsibility, and cannot successfully be assigned to a single authority (Markus and Benjamin, 1997). Successful mutual adaptation is closely linked to knowledge creation, not only decisions. It is possible to facilitate such knowledge creation, (as with the Methodology Group) but its emergent nature makes it hard to plan in detail. Lastly, as the PROSIT case also shows, modern iterative ISD processes (like MSF) have an iterative structure well suited to support mutual adaptation; the iterative and incremental structure extends the period where the information system is malleable.

6. CONCLUSION

The aim of this paper was to study the dynamics of mutual adaptation between IS development and the business process in modern, iterative ISD projects. The research questions were investigated through a longitudinal case study in a large public auditing agency. There are three major findings. First, the case study identified a window of opportunity for mutual adaptation at a stage in the ISD project where the malleability of both the information system and the business process was sufficient.

Second, an organisational mechanism, a process expert group working closely with the development team, was successful in both structuring the information system and restructuring the business process. This was achieved through mutual learning, knowledge creation and the necessary authority to influence both processes. It is suggested that this organisational mechanism, within the window of opportunity, may be relevant for other ISD projects.

Third, the organisational mechanism integrates well with modern ISD frameworks like MSF: The iterative structure extends the period of time where the information system is malleable, and iterations are a natural frame for assessing the mutual adaptation. Therefore, it represents an interesting option for project managers to increase the scope and business value of ISD projects.

Further research could investigate the dynamics of mutual adaptation in other contexts, for example when implementing business software packages like ERP and CRM. It could also seek to develop an add-in to software engineering frameworks (like MSF, RUP, and OPEN) that gives more support to mutual adaptation.

ACKNOWLEDGEMENTS

A special thank to Liv Dreierstad at the Office of the Auditor General, who provided access to the PROSIT project, and also commented on the case description.

REFERENCES

Boehm, B. W., 1988, A Spiral Model of Software Development and Enhancement, *IEEE Computer*, pp. 61–72 (May).

Brynjolfson, E., van Alstyne, M., Bernstein, A., and Renshaw, A. A., 1997, Tools for teaching Change Management, *The Matrix of Change and Supporting Software*, IAIM'97, Atlanta.

Braa, K., and Vidgen, R., 1997, An Information Systems research framework for the organizational laboratory, in: *Computers and Design in Context*, M. Kyng and L. Mathiasen, eds., MIT Press.

Ciborra, C., 2000, *From Control to Drift*, Oxford University Press, Oxford.

Coakes, E., and Elliman, T., 1999, The role of stakeholders in managing change, *CAIS*, 2.

Giaglis, G., 1999, On the integrated design and evaluation of business process and information systems, *Communications of the AIS*, 2.

Garrety, K., Robertson, P. L., and Badham, R., 2001, Communities of Practice, Actor Networks and Learning in Development Projects, *The Future of Innovation Studies*, Eindhoven; http://www.tm.tue.nl/ecis/papers/iii_4_2.pdf (accessed Jan 15 2004).

Henderson-Sellers, B., and Unhelkar, B., 2000, *OPEN Modelling with UML*, Addison-Wesley.

Jacobson, I., Booch, G., and Rumbaugh, R., 1999, *The Unified Software Development Process*, Reading, Addison Wesley.

Jacobson, I., Ericsson, M., and Jacobson, A., 1995, *The Object Advantage: Business Process Reengineering with Object Technology*, Addison-Wesley, ACM Press.

Leonard-Barton, D., 1988, Implementation as mutual adaptation of technology and organization, *Research Policy* **17**(5):251–267.

Leonard-Barton, D., and Sinha, D. K., 1993, Developer-user interaction and user satisfaction in internal technology transfer, *Academy of Management Journal* **36**(5):1125–1139.

Majchrzak, A., Rice, R., Malhotra, A., King, N., and Ba, S., 2000, Technology adaptation: The case of a computer-supported inter-organizational virtual team, *MIS Quarterly* **24**(4):569–600.

Markus, M. L., Benjamin, R., 1997, The Magic Bullet of IT-Enabled Transformation, *Sloan Mangement Review*, pp. 55–68 (Winter).

McConnell, S., 1996, *Rapid Application Development*, Microsoft Press, Redmond.

Microsoft, 2001, *Microsoft Solutions Framework*, 2001; http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/tandp/innsol/default.asp.

Miles, M. B., and Huberman, A. M., 1994, *Qualitative Data Analysis*, Thousand Oaks, Sage.

Nambisan, S., Agarwal, R., and Tanniru, S., 1999, Organizational mechanisms for enhancing user innovation in information technology, *MIS Quarterly* **23**(3):365–394.

Ngwenyama, O., 1998, Groupware, Social Action and Emergent Organizations: On the process dynamics of computer mediated distributed work, *Accounting, Management and Information Technologies* **8**(2–3):127–146.

Pettigrew, A. M., 1985, Contextualist Research and the Study of Organizational Change Processes, *Research Methods in Information Systems*, Mumford et al., eds., North-Holland.

Reich, B., and I. Benbasat, 2000, Factors that influence the social dimension of alignment between business and technology objectives, *MIS Quarterly* **24**(1):81–113.

Sauer, C., and Willcocks, L. P., 2004, Strategic Alignment Revisited: Connecting Organizational Architecture and IT Infrastructure, *Proceedings of HICSS*, Hawaii.

Tyre, M. J., and Orlikowski, W., 1994, Windows of opportunity: Temporal patterns of technological adaptation in organizations, *Organizational Science* **5**(1):98–118.

RETHINKING ISD METHODS: FITTING PROJECT TEAM MEMBERS PROFILES

Isabelle Mirbel*

1. INTRODUCTION

Information Systems Development (ISD) evolves continually, creating new challenges especially in terms of Method Engineering (ME). Looking at the way ISD methods are used in practice, we notice they are always adapted: steps are added, other removed or skipped and so on. Different factors related to the project, the technology, the team expertize and the business domain lead to method tailoring. One way to answer this need for customization is the assembly of predefined process fragments. Dedicated efforts have been made, in the field of Situational Method Engineering (SME), to decompose ISD methods into process fragments (Brinkkemper et al., 1998; Ralyte, 2001).

Indeed customization of ISD methods have mainly be thought of for the person in charge of building the ISD method, i.e. the method engineer, in order to allow him/her to adapt the method to a corporate or project need. But there is also a need for customizations dedicated to the persons using the method, i.e. project team member, to provide each of them with dedicated guidelines to give them support through their daily tasks. A guideline is a statement or other indication of policy or procedure by which to determine a course of action (Ralyte, 2001).

The approach presented in this paper provides means to customize method 'on the fly' in order to match as well as possible the profile of the project team member job within the project (Whitenack, 1995; Gnatz et al., 2001; Mirbel and de Rivieres, 2002). For this purpose, efficient classification and retrieving means to store and select process fragments have to be provided. Efforts have already been made on this topic in the fields of ME. These classification and retrieving techniques are currently based on structural relationships among process fragments (specialization, composition, alternative,...) and keyword matching (Cauvet and Rosenthal-Sabroux, 2001). From our point of view, these means do not fully exploit the potential of breaking down ISD methods into process fragments and tailoring them. We believe knowledge about organizational, technical and human factors, which are

* Laboratoire I3S, Les Algorithmes, Route des Lucioles, BP 121, 06903 Sophia Antipolis Cedex, FRANCE. Phone: (+33) 4 9294 2760. Fax: (+33) 4 9294 2896, isabelle.mirbel@unice.fr.

critical knowledge about ISD (Cauvet and Rosenthal-Sabroux, 2001), should be taken into consideration in addition to structural knowledge and keywords. It allows to better qualify process fragments when entering them into the repository. It also allows the project team member to better express his/her methodological needs (or profile), improving this way the chance to get adequate and useful process fragments. And finally, it also enables the use of more powerful and suitable retrieving techniques when looking for ISD methodological support. Therefore, we propose a *Reuse Frame* aggregating the different ISD critical aspects meaningful to tailor ISD methods with regards to project team member profiles.

In this paper, we start first by discussing the critical aspects of Information Systems (IS) and we show how to handle them through method customization. Then, the repository of predefined process fragments is described in Section 3. The method customization process is presented in Section 4. Finally, we conclude in Section 5.

2. HANDLING CRITICAL ASPECTS OF INFORMATION SYSTEMS

To better take advantage of customization into SME, we believe IS critical aspects have to be taken into consideration. A process fragment represents a basic block for constructing 'on the fly' methods from the process point of view (by opposition to the product point of view). Process fragments have to be defined with regards to IS critical aspects in order to be more easily reused in similar methodological situations. Methodological support along the development process has to be requested with regards to IS critical aspects. In our approach, the human, organizational and technical IS aspects are grouped into a tree of successively refined aspects. By providing such an ontological structure, we enlarge the panel of means for method engineers to drive project team members on the way to apply a corporate method and to broadcast ways of working which is most of the time reduced to deliverables. In addition, we provide to project team member a structure allowing them to better explain their own need in order to get a customized view of the method they have to use. Our approach provides to project team members means to find the most suitable information with regards to his/her ISD methodological need (or problem).

2.1. The Reuse Frame

ISD critical aspects, meaningful to better store and retrieve process fragments through SME, are described in terms of *aspects*, belonging to *aspect families*, which are successive refinements of the three main factors of IS: human, organizational and technical. In this tree structure, called the *Reuse Frame*, an *aspect* is a leaf node and an aspect *family* is a non-leaf node with at least two sub-nodes (which could be *aspect* or *family* nodes). Families allow a better understanding of aspects by providing a way to group and organize them. With regards to the *organizational* dimension, we started from the work of K. van Slooten and B. Hodes providing elements to characterize ISD projects (van Slooten and Hodes, 1996). With regards to the *technical* dimension, we started from previous work on `JECKO`, a context-driven approach to software development, including a contribution to define software critical aspects in order to get suitable documentation to support the software development process (Mirbel and de Rivieres, 2002). And finally, about the *human* dimension, we currently propose a basic description of the expertize of the project team

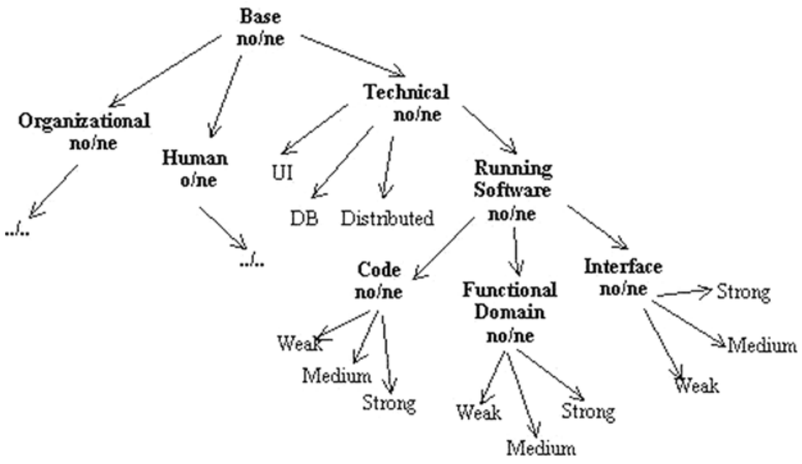


Figure 1. Technical branch of the Reuse Frame.

members. As an example, the technical branch of the *Reuse Frame* is presented in Figure 1. In this branch, we distinguish the following aspects: the application to be developed includes a user-interface (`UI`), a database (`DB`), is distributed (`Distributed`) or is built on top of a running application (`Running software`). This last aspect is presented as a *family* because different features of a running software may be considered: `Functional domain`, `Interface` and `Code` (Mirbel and de Rivieres, 2002). Again a distinction is done among weak, medium and strong reuse of existing code, functional domain and interface.

In the Reuse Frame, nodes close to the root node correspond to general aspects while node close to leaf nodes (including leaf-node) represent precise aspects.

A leaf-node is defined through a *name*. An intermediary node is also defined through a *name* and completed by information about relationships among the different aspects or sub-families belonging to it: the fact that the different aspects may be *exclusive* (or not) and/or *ordered* (or not). The `Software to develop` aspect presented in Figure 1, for instance, is described as *non-ordered* because the different sub-families are non-related aspects. They are described as *non-exclusive* because they may be associated with a same methodological need or process fragment. `Weak`, `Medium` and `Strong` aspects are considered as *non-ordered* because a fragment dedicated to guidelines to weakly keep the code for instance may not necessary be complementary to the one dedicated to keep the code in a medium way. And on the contrary, it is not forbidden also to search for a solution including for instance `Medium` and `Strong` aspects. Therefore, they are specified as *non-exclusive*. For the full definition of *Reuse Frame* elements, refer to (Mirbel, 2004).

2.2. Taking Advantage of the Reuse Frame

The ISD critical aspects defined in the *Reuse Frame* aim at improving meaningful techniques to retrieve process fragments corresponding to a project team member profile.

Therefore, means have to be provided to associate pertinent aspects to (i) process fragment in order to retrieve them and (ii) to project team member profile in order to enable suitable matching techniques. This is done respectively through the notion of *Process Fragment Reuse Context* and *Problem Reuse Situation*.

The Reuse Context associated to each Process Fragment is a set of criteria allowing to situate the fragment with regards to ISD critical aspects. To retrieve process fragments from the repository, the project team member expresses a methodological need (or problem) through keywords and a *Reuse Situation*. The reuse situation is a set of criteria specifying the project team member profile. A criterion is a path in the Reuse Frame.

A **criterion** is a path from the root node `base` to a node n_n of the *Reuse Frame* (if n_n is a *family* node, then its *exclusion* field is different from e).
Two criteria are **compatible** if they do not share in their definition a common family node n_i with an exclusion field equal to e.

The **Process Fragment Reuse Context** is a set of at least one compatible criterion taken from the *Reuse Frame*.

Process fragments providing general guidelines are usually characterized by general criteria, that is to say paths ended by nodes from the *Reuse Frame* close to the root node. On the contrary, specific guidelines are provided in process fragments described through precise criteria, that is to say paths ended by nodes from the *Reuse Frame* close to leaf nodes or leaf nodes themselves.

In the *Problem Reuse Situation*, in addition to the pertinent criteria, called *necessary criteria*, *forbidden criteria* may be given, that is to say aspects the project team member is not interested in. It could be helpful in some cases to be sure the process fragments including these (forbidden) aspects will not appear in the solution answering the methodological need.

A **Problem Reuse Situation** is a set of at least one compatible *necessary criteria* and a set of compatible *forbidden criteria*. All criteria are taken from the *Reuse Frame* and there is no common criterion between necessary and forbidden criteria.

3. BREAKING DOWN METHODS INTO FRAGMENTS

Approaches have been developed to specify methods (Si-said, 1999) and to break them down into fragments (Whitenack, 1995). Efforts have also been made to formalize fragment definition (or pattern definition) (Rolland et al., 1999; Storrle, 2001). In addition to the different specification levels that are provided in the literature about SME, different objectives are targeted by the approaches. A first family of approaches aims at documenting methods through well-defined fragments (Storrle, 2001). These approaches do not provide powerful supports nor to reuse the fragments from one method to another nor to customize the method for a specific project or group of persons. Their strength resides in the effort of specification with regards to the elements a method is made of (tasks, activities, resources, etc.). A second family of approaches groups works which aim is to help in building new methods starting from existing ones (instead of building them from scratch) (Ralyte, 2001; Brinkkemper et al., 1998). In this kind of approaches, the focus is on the operators provided to allow a new combination of existing process fragments and on mechanisms to evaluate the similitude among them. Both combination operators and evaluation mechanisms are dedicated to method engineers. But, project team members also need to benefit, through reuse and adaptation mechanisms, from the experiences acquired during the resolution of previous problems in terms of applying the method in a concrete way (i.e. using it). All along the ISD process, heuristics are elaborated by project team members to deal with their daily development activities. These heuristics may be useful to other teams facing close situations in different projects independently of the functional domain as well as the technical domain. Therefore, a third family of proposals, our work is included in, focuses on method fragmentation for project team members, to provide them with guidelines which are to be reused while performing their daily task (Gnatz et al., 2001; Mirbel and de Rivieres, 2002).

3.1. Process Fragments

Different types of method components have been proposed in the literature: process fragments and product fragments (Brinkkemper et al., 1998; Punter, 1996). Other authors integrate the two dimensions in the same module, called a method chunk (Ralyte, 2001). In our approach, we focus on the process dimension of methods. Our *process fragments* are the result of (i) the capitalization of experiences about ISD directly by project team members, or (ii) the breaking down of known methods by method engineers (Ralyte, 2001).

A **Process Fragment** f_n is a 6-uple, $f_n = \langle nf_n, RC_n, i_n, g_n, Ass_n, Inc_n \rangle$, where

- nf_n is the `name` of the fragment. This name should be meaningful to indicate its purpose. Fragment name is unique in order to identify the fragment.
- RC_n is the `Process Fragment Reuse Context` associated to the fragment as previously defined (cf Section 2.2).
- i_n is the `intention` of the fragment, that is, a textual description of the goal of the fragment and a list of keywords. It is defined as a 2-uple $\langle des_{i_n}, KW_{i_n} \rangle$ where des_{i_n} is the textual description and KW_{i_n} is the set of keywords associated to the fragment.
- g_n is the `guidelines` of the fragment. It is specified through a set of pair $\langle n, h \rangle$ where n is the notation used to express guidelines and h is the textual description of the guidelines.
- Ass_n is the set of fragments **associated** with the current fragment. It is defined as a set of 3-uple $\langle$*fr-ass,re-ass,deg-ass*$\rangle$ where
  - *fr-ass* is a fragment sharing at least part of its *Process Fragment Reuse Context* with the current fragment f_n
  - *re-ass* qualifies the relationship between f_n and*fr-ass*. We distinguish 2 kinds of relationships: *complementarity* and *alternative*.

Name	**Requirement–Out–of–Scope**
Reuse Context	{[base – software – running software]}
Related Fragments	BusinessDomain–Out–of–Scope – complementarity – 0.75
Non–compatible Fragments	–
Intention To document existing parts of the running software useful to understand the purpose of the new software but not directly related to the new development.	
Guidelines	**Notation** UML use–case diagrams
	Description Add use–cases dedicated to running functionalities increasing the understanding of the software. Stereotype them with <<out–of–scope>>. Group all the use–cases stereotyped <<out–of–scope>> in a package (or set of packages) also stereotyped <<out–of–scope>>

Figure 2. An example of process fragment.

  - *deg-ass* quantifies the relationship between f_n and *fr-ass*. *deg-ass* $\in$ [0, 1]. *deg-ass* is a *necessity* degree when associated to the *complementarity* relationship. It is a *clotheness* degree when associated to the *alternative* relationship.

  The *complementarity* relationship is bijective: If a fragment f_1 appears as complementary to f_2, then f_2 also appears as a complementary fragment of f_1 with the same degree.

  The *alternative* relationship is not bijective since a fragment f_1 may include another fragment f_2 and therefore be an alternative to it. But the contrary would be wrong.

  Close intentions are required if *deg-ass* is less than 1, identical ones are requested otherwise.

- *Inc*$_n$ is the set of `not compatible fragments` associated with the current fragment. It is defined as a set of pair $\langle$*fr-inc,deg-inc*$\rangle$ where
  - *fr-inc* is a fragment not compatible with the current fragment f_n
  - *deg-inc* is the degree of *incompatibility* between f_n and *fr-inc*.

An example of process fragment is given in Figure 2, where a process fragment named `Requirement-out-of-scope` is presented. Its *Process Fragment Reuse Context* indicates it shows guidelines for when developping on top of a running software. The process fragment has an associated fragments `DB-out-of-scope`, which is complementary. It does not have incompatible fragments. The intention explains the purpose of the process fragment which is to help in documenting the running part of the application under which the development will take place. Associated guidelines in UML are then given.

To be added to the repository, a new process fragment must be linked to process fragments already stored in the repository. Two process fragments are linked when they can be used inside a same method. The *precedence metric* allows to estimate the sequence in which the process fragments have to be applied.

Let f_a and f_b be 2 process fragments. The *precedence metric* is composed of 3 values:

- *pb* is the probability for f_a before f_b
- *pa* is the probability for f_a after f_b
- *pi* is the probability for f_a and f_b to be in an indefinite sequence

with $pb, pa, pi \in [0, 1]$ and $\sum pb, pa, pi = 1$.

For each process fragment f_b linked to f_a, the project team member introducing f_a gives the *precedence metric* values. Possible values are:

- {*pb=0, pa=0, pi=1*}: the sequence between f_a and f_b is not significant;
- {*pb=1, pa=0, pi=0*}: the project team member indicates f_a should be useful before f_b;
- {*pb=0, pa=1, pi=0*}: symmetrically, f_b should be useful before f_a.

Process fragments answering a methodological need are presented in a sequential way constituting the *road-map* to be followed while applying a method (i.e. applying successively a set of process fragments), as it will be discussed in the following.

3.2. Road-Maps

Different ways may be provided, even inside a same method, to satisfy an engineering goal. Moreover, the sequence through which process fragments have to be applied is not always pre-determined: they may or not be related by a precedence relationships. Therefore, different road-maps among a set of process fragments are possible. A road-map is composed of one or several coherent sequence(s) of process fragments corresponding to the usage of the method by a particular project team member. It is a customized view of the method to match the profile of a project team member and his/her specific methodological problem.

Indeed, process fragments stored in the dedicated repository and organized into road-map allow capitalization of knowledge about the resolution of methodological problems. The *Reuse Frame* provides knowledge *for* reuse, while reusable knowledge is stored in the repository of predefined process fragments. In the next section, we show how to take advantage of this framework through method customization.

4. METHOD CUSTOMIZATION

In our approach for method customization, the project team member looking for a specific view of the ISD method, chosen for the project he/she is involved in, starts by expressing its specific methodological problem, through keywords and problem reuse situation. Then, the process fragments matching the problem are retrieved from the repository of predefined process fragments. Tuning facilities are provided to the project team member to let him/her get the most suitable road-map within the method to answer its specific need. And finally, the selected process fragments are ordered to compose the road-map, that is to say the guidelines to be successively applied, in order to take as much advantage as possible of the method.

In the following, we will discuss more in detail the selection of the right process fragment, the tuning facilities and the road-map building.

4.1. Selecting the Right Process Fragments

Thanks to the *Process fragment Reuse Context* associated to each process fragment stored in the repository and the *Problem Reuse Situation* given by the project team member, matching techniques can be applied to select process fragments corresponding to the problem. Therefore, we introduce the *situational metric* allowing to quantify the matching between a process fragment and a problem. This metric is based on (i) the number of common criteria between the necessary criteria from the *Problem Reuse Situation* and the *Process fragment Reuse Context*, (ii) the number of common criteria between the forbidden criteria from the *Problem Reuse Situation* and the *Process fragment Reuse Context*, (iii) the number of required necessary criteria from the *Problem Reuse Situation*:

$$ms(pb, pf) = card(RC_{pf} \cap CN_{pb}) - card(RC_{pf} \cap CF_{pb})/card(CN_{pb})$$

where pb is a problem, CN_{pb} the necessary criteria of the *Problem Reuse Situation*, CF_{pb} the forbidden criteria of the *Problem Reuse Situation*, pf a process fragment, RC_{pf} the *Process Fragment Reuse Context*.

A positive value of $ms(pb, pf)$ indicates that there are more necessary criteria than forbidden ones in the process fragment pf under consideration with regards to the problem pb. On the contrary, a negative value of $ms(pb, pf)$ indicates that there are less necessary criteria than forbidden ones. The perfect adequation is represented by the value 1 and the worst situation by the value $(-card(RC_{pf} \cap CF_{pb})/(card(CN_{pb}))$.

4.2. Tuning the Selection

The *Reuse Frame* introduced previously (cf Section 2) supports different levels of granularity with regards to criterion definition. Criteria which ending node is close to the *base* node are much more generic than criteria which ending node is close to leaf nodes. Indeed, process fragments providing general guidelines are usually specified through aspects associated to general criteria, while specific guidelines are provided in process fragments specified through aspects associated to precise criteria.

4.2.1. Tuning Necessary Criteria

When searching for predefined process fragments into the repository, a project team member is interested in process fragments which criteria strictly match the necessary criteria from the *Problem Reuse Context*. But process fragments including more specific criteria in their *Reuse Contexts* may also be interesting: process fragments associated to more specific criteria usually provide more specific guidelines. They may better cover part of the methodological problem the project team member has to deal with.

If one looks for instance at process fragments matching the `[ base - technical - running software - functional domain ]` criterion, he/she may be also interested by the process fragments matching the `[ base - technical - running software - functional domain -` **weak** `]`, `[ base - technical - running` `software - functional domain -` **medium** `]` and `[ base - technical - running software - functional domain -` **strong** `]` criteria (cf Figure 1).

In the same way, the project team member may be interested in process fragments associated to more general criteria usually providing more general-purpose guidelines which could also be useful.

4.2.2. Tuning Forbidden Criteria

Taking into consideration process fragments associated to more general and/or more specific criteria may also be interesting with regards to the *forbidden criteria* given in the problem definition. Indeed, enlarging the set of forbidden criteria to more general ones means to forbid full branches of the *Reuse Frame*; and enlarging the set of forbidden criteria to more specific criteria means to forbid process fragments associated to too specific criteria, usually present in reuse contexts of process fragments providing too specific guidelines.

4.2.3. Weighting Criteria

When computing the *situational metric* between a process fragment pf and a problem pb, criteria included as more general or more specific ones have to be weighted with regards to their distance from the criteria given in the problem definition. Therefore, the *situational metric* becomes:

$$ms(pb, pf) = \left(\sum_{card(CNE)}^{i=1} P_{c_i} - \sum_{card(CFE)}^{j=1} P_{c_j}\right) \Big/ \left(\sum_{card(CN)}^{k=1} P_{ck}\right)$$

with CN the necessary criteria, CNE the extended necessary criteria also present in RC_{pf}, CF the forbidden criteria and CFE the extended forbidden criteria of the problem pb also present in RC_{pf}, and with P the criteria weight computed as follows:

$$\text{if } c_i \in CN,\ p_{c_i} = 1 \text{ else } p_{c_i} = 1/(2^{nbn})$$

where nbn is the number of node between cr and c_i excluding cr and including c_i.

4.3. Building the Road-Map

From the selection and tuning steps of our customization process, we get a set of process fragments. Indeed, these process fragments may not be related all together (through the precedence relationship or as associated process fragments). They may for instance cover disjoint stages of the method (at the beginning and the end of the method for instance). So, the road-map corresponding to the project team member methodological problem is indeed made of different subsets. In each subset are reassembled process fragments related to each others as *associated* process fragments or through the precedence relationship.

4.3.1. Adding Complementary Process Fragments

In the repository, process fragments may be linked to each other as *associated* process fragments (cf Section 3.1). Degrees are provided to quantify the complementarity. When building the road-map, the coherency of the process fragments belonging to the same subset has to be enforced by fulfilling the constraints expressed through the complementarity relationship: complementary process fragments which degrees are higher than a predefined threshold have to be added to the solution. Indeed, the project team member tunes the level of coherency he/she wants its roadmap to satisfy by setting the *complementarity* threshold. A high complementarity threshold leads to a solution in which only very complementary process fragments are added, while a low one leads to a solution in which most of the complementary process fragments are included.

4.3.2. Removing not Compatible Fragments

In the same way, process fragments may also be linked to each other as *not compatible* process fragments. In this case also, degrees quantify the incompatibility. When building the road-map corresponding to each subset of the result set, the coherency of the process fragments belonging to the same subset has to be enforced by fulfilling the constraints expressed through these incompatibility relationships: not compatible process fragments which degrees are higher than a predefined threshold have to be removed from the solution. Indeed, the project team member tunes the level of coherency he/she wants its roadmap to satisfy by setting the *incompatibility* threshold. A high incompatibility threshold leads to the removal of very incompatible process fragments from the solution, while a low threshold leads to the removal of most of the incompatible process fragments.

Tuning the road-map building process provides again a way for the project team member to reduce or enlarge the number of process fragments included in the solution to his/her methodological problem.

4.3.3. Removing Still Inconsistent Process Fragments

During the selection step, *Process Fragment Reuse Contexts* are evaluated with the help of the *situational metric* to be or not kept as part of the road-map. Complementary process fragments added to the solution for coherency purposes have to be evaluated with regards to the *situational metric*. Only complementary process fragments which *situational metric* is positive are kept. And if not kept, the process fragments requiring them as complementary process fragment remain still incoherent and have therefore to be removed from the road-map.

Finally, the set of process fragments still kept in each subset constitutes the road-map(s) corresponding to the project team member problem and is presented to him/her as a succession of guidelines to be successively applied.

5. CONCLUSION

In this paper we presented an approach to customize method in order to fit the need of the project team member. Customization is supported by method fragmentation. In addition, we proposed the *Reuse Frame*, a framework representing critical aspects of IS and allowing method engineers to specify process fragments meaningful features and project team member to select the process fragments corresponding to their profile. After discussing the usefulness of the *Reuse Frame* and the way to break down methods into fragments we focused on the customization process and its main 3 steps: selection, tuning and road-map building, allowing project team member to get a customized point of view on the ISD method in order to fit his/her own methodological need.

Attempts have already been made to use such an approach on real projects, especially with regards to the technical dimension of IS (Mirbel and de Rivieres, 2002). A case tool is under development to validate the whole approach.

In the future, our efforts will focus on the exploitation of tracking information about the way project team member reuse the process fragments and the feedback they may give on their reuse experience: some process fragments, for instance, may be more useful than others, fragments may indeed be skipped or removed from road-maps. The *precedence metric* should be updated in consequence to also reflect the experience capitalized through the reuse of the process fragments. In the same way, we would like to weight process fragments with regards to the expertise level of the project team members introducing the process fragments into the repository.

REFERENCES

Brinkkemper, S., Saeki, M., and Harmsen, F., 1998, Assembly techniques for method engineering, in: *10th International Conference on Advanced Information Systems Engineering*, Pisa, Italy.

Cauvet, C., and Rosenthal-Sabroux, C., 2001, *Ingenierie des systemes d'information*, Hermes.

Gnatz, M., Marschall, F., Popp, G., Rausch, A., and Schwerin, W., 2001, Modular process patterns supporting an evolutionary software development process, *Lecture Notes in Computer Science*, 2188.

Punter, H. T., and K. L., 1996, The MEMA model: towards a new approach for method engineering, *Information and Software Technology* **4**:295–305.

Mirbel, I., 2004, A polymorphic context frame to support scalability and evolvability of information system development processes, in: *6th International Conference on Enterprise Information Systems*.

Mirbel, I., and de Rivieres, V., 2002, Adapting Analysis and Design to Software Context: the JECKO Approach, in: *8th International Conference on Object-Oriented Information Systems*.

Ralyte, J., 2001, *Ingenierie des methodes a base de composants*, PhD thesis, Universite Paris I – Sorbonne.

Rolland, C., Prakash, N., and Benjamen, N., 1999, A multi-model view of process modelling, *Requirements Engineering Journal* **4**:169–187.

Si-said, S., 1999, *Proposition pour la modelisation et le guidage des processus d'analyse et de conception*, PhD thesis, Universite Paris I – Sorbonne.

Storrle, H., 2001, Describing process patterns with UML, in: *ESWT*.

van Slooten, K., and Hodes, B., 1996, Characterizing IS development projects, in: *IFIP TC8, WG 8.1/8.2*, S. Brinkkemper and K. Lyytinen, R. W., eds., pp. 29–44.

Whitenack, B., 1995, RAPPeL: a requirement analysis process pattern language for OO development; http://www.bell-labs.com/user/cope/Patterns/Process/RAPPeL/rapel.html.

PRE-REQUISITES FOR SUCCESSFUL ADOPTION OF THE ASP MODEL BY USER ORGANIZATION

George Feuerlicht and Jiri Vorisek*

1. INTRODUCTION

Businesses are increasingly reliant on ICT to support their operations and to maintain competitiveness. At the same time, information systems and their ICT infrastructure are becoming increasingly complex and costly to implement and maintain. Advances in networking, and in particular Internet-related technologies make it possible to implement enterprise applications as services delivered from a remote location potentially in a more predictable and cost-effective manner than in-house applications. Application Service Providing (ASP) emerged towards the end of 90s with claims of extensive advantages for client organizations, in particular for SMEs. Notwithstanding many perceived advantages of the ASP approach, most of the early ASP providers have not been able to establish a viable business model. Factors contributing to the failure of early ASP providers included lack of a suitable technological infrastructure for hosting a large number of complex enterprise applications in a scalable and secure manner, poor customisation capabilities, and almost total lack of integration facilities. As a result of these shortcomings, early ASP providers failed to deliver major cost savings to their customers, resulting in poor acceptance of application servicing by the market place. Recently, however, a number of important ICT vendors have re-confirmed their commitment to application servicing in the context of the new Utility Computing approach, and have made massive investments in infrastructure for the delivery of enterprise application services (Dubie, 2004). However, given earlier experiences with application servicing most user organizations remain skeptical and are waiting to see if the benefits are going to be realized as claimed by the vendors.

In this paper we describe the key differentiators of application servicing when compared to the traditional software-licensing model (Section 2), and then discuss the characteristics of the second generation of application servicing (Section 3). We then discuss end user organizations' strategies for adoption of application servicing and the related critical success factors (Section 4).

* University of Technology, Sydney, Australia; and University of Economics, Prague, Czech Republic, jiri@it.uts.edu.au, vorisek@vse.cz.

2. KEY DIFFERENTIATORS OF APPLICATION SERVICING

Application servicing has evolved into a sophisticated service delivery model over the last two years. Table 1 shows key differentiating features of ASP (software-as-service) when compared to the traditional model of implementing enterprise applications (software-as-license, i.e. when customer purchases the software license and runs the software on their own technology infrastructure). A more comprehensive analysis of the application service model from the perspective of design and technology, business, and ICT management viewpoints is available in elsewhere (Feuerlicht and Vorisek, 2002; Levy, 2004; McCabe, 2004; Vorisek et al., 2003; Wainewright, 2004).

As indicated by the above table (Table 1) the key advantage of the ASP model is the ability of the provider to concentrate ICT resources and take advantage of the resulting economies of scale to deliver scalable services to a large customer base, while increasing utilization and reducing overall costs.

Application Suitability for ASP Delivery

Given the current ASP limitations suitability of applications for ASP delivery needs to be carefully evaluated, as certain types of applications may not be suitable candidates for application servicing. For example, certain types of mission-critical (core) business applications may not be available from external providers. Also the critical nature of these applications dictates in-house implementation and control, irrespective of other considerations (e.g. cost). Another category of applications that may not be suited to ASP delivery are highly customized and specialized applications as the service providers can not gain economies of scale given the relatively small size of the market. Applications with extensive integration requirements have close dependencies on other enterprise applications and cannot be effectively managed externally.

3. SECOND GENERATION ASP

As discussed in Section 1 above the first generation of ASP providers have not been able to significantly reduce the costs associated with delivery of enterprise applications. The lack of a suitable technological infrastructure and a viable business model prevented early ASPs from achieving economies of scale that are essential for this approach to gain wider acceptance. However, a number of important recent ICT trends are likely to shift the balance from in-house implementation of licensed software towards application servicing. The second generation of ASPs (e.g. Salesforce.com, NetSuite, RightNow, Salesnet) have demonstrated that ASP delivery requires a different business model, different technological architecture and possibly even different company culture. They are using unique attributes of the software-as-service model to create applications that are highly customizable, upgradeable, and can be deployed with minimum delays. Established software vendors (e.g. Oracle, PeopleSoft, SAP) offer ASP solutions as an alternative to purchasing software licenses and have achieved varying levels of success with the ASP model. There is some evidence that the rate of growth of software delivered as a service is higher than of traditional software licence sales, in particular in the USA (Newcomb, 2004). The penetration

Table 1. Comparison of the software-as-license vs. software-as-service models for enterprise applications

Differentiator	**ASP (SW as Service)**	**Traditional Approach (SW as License)**
Key characteristic	Provider is responsible for all ICT resources and delivers application services to a large number of customers.	Customer is responsible for the implementation and operation of the application.
Technological architecture	Multi-tenant service-oriented and highly scalable architecture.	Architecture suitable for deployment on a dedicated ICT infrastructure.
Upgrades	Frequent software upgrades - all customers are upgraded simultaneously resulting in significant cost reductions.	Individual customers typically running different versions of software.
Implementation	Short implementation cycle. Typically no requirement for new HW and SW.	Long implementation cycle due to complex implementation of HW and SW.
Customization	Typically limited.	Customization possible, but can be expensive.
Demand on internal resources.	Limited internal resources required; the client company can focus on core business.	Extensive use of internal resources.
ICT costs	Costs are predictable, no large upfront investments is required. Operating costs are correlated with the volume of services.	Both investments and operating costs are involved. Overhead costs can be high and include depreciation and amortizing of investments.
Contractual Arrangements	SLAs are used extensively to define the service (i.e. functionality, volume of service, quality, price, etc)	Contract typically divided into several subcontracts for HW, SW, and services.
ICT resources utilization	ICT resources (i.e. HW, SW, skills, etc.) are used across all customers; provider has advantages of economies of scale.	ICT resources are used only for one organization.
Problem resolution and change management	Short feedback cycle. Support staff can directly identify and fix problems for all customers.	Problem resolution is often indirect via intermediaries (VARs, SIs, etc). Patches and upgrades are implemented at individual customer sites. This process is costly and unreliable.
Main risks	High dependence on the service provider. Client organization can suffer loss of technical expertise. Unsatisfactory customization. Unresolved integration issues. Enhancements not under the control of the customer.	Lagging technological advances. High TCO (Total Cost of Ownership). Low flexibility and scalability.

of the ASP model varies according to type of application with CRM systems being the most popular ASP applications. However, the acceptance of other application types (e.g. ERP and HR) for ASP delivery is growing.

The driving forces for this shift towards software delivered as a service include both business and technological factors.

Business Factors

As a result of the recent ICT downturn the sales of new licenses for enterprise application software have stagnated and in some cases declined. There is some evidence that as the enterprise application software market matures, major ERP vendors are changing their revenue model to decrease their reliance on new software licenses towards income generated from software license upgrades and product support (Oracle, 2004; SAP, 2004). This combined with the fact that most organizations spend as much as 80% of software-related costs on software maintenance and related activities (Haber, 2004), creates a situation where licensed software is de-facto *rented.* It is precisely this high-level of on-going costs that motivate many organizations towards alternatives such as outsourcing and application servicing.

New Computing Models

The ASP model is closely related to recently emerging computing models such as Utility Computing, and On-Demand Computing. The main idea of Utility Computing is that ICT services are supplied on demand (i.e. as required by the end-user organization) via a grid of interconnected, dynamically configurable, highly reliable and scalable computing resources (i.e. servers, storage platforms, and applications). Computing grids provide an ideal infrastructure for application servicing as it can host a large number of ASP applications in a scalable and reliable manner. Resource sharing and improved hardware utilization of grid computing environments provides a more cost effective solution for hosting enterprise applications than a set of independent servers each dedicated to a specific application. Clusters of servers, storage devices and other resources constructed from low-cost (commodity) components create *virtual* resources on-demand as required by enterprise applications. A number of infrastructure vendors (IBM, HP, Oracle) are in the process of building large data centres intended for hosting large number of client applications using the Utility Computing model (Eriksen, 2003). Investment in infrastructure on this scale clearly demonstrates a strategic commitment to Utility Computing and more specifically to application servicing as the new outsourcing model for enterprise applications. Recent efforts to standardize Utility Computing infrastructure in order to facilitate interoperability between vendor solutions led to the creation of Utility Computing Working Group under the auspices of DMTF (Distributed Management Task Force) (DMTF, 2004) and with the participation of major ICT players including Cisco Systems, EDS, EMC, HP, IBM, Oracle, Sun Microsystems and VERITAS. The main objective of the DMTF Utility Computing Working Group is to develop a set of interoperability standards in collaboration with other organizations including OASIS (Organization for the Advancement of Structured Information Standards) and GGF (Global Grid Forum) that will allow the assembly of comprehensive services from components supplied by different vendor platforms.

New Technologies and Application Architectures

Another key trend favoring application servicing over the traditional software-licensing model is the move towards service-oriented architecture (SOA) for enterprise computing. The nature of enterprise applications has changed dramatically over the last five years; most enterprise applications today have requirements to interoperate across enterprise boundaries (i.e. requirements for e-business). Service-oriented computing based on Web services standards and technologies is widely regarded as having the potential to address the requirements for e-business interoperability and are likely to become the dominant enterprise computing architecture in the future. There is a close relationship between application servicing and service-oriented computing. Web services are regarded as the enabling technology for the integration of ASP applications, and for delivery of low-granularity application services (Eisenberg, 2004; Ferguson, 2004). The wide adoptions of Web services standards across the various computing environments (i.e. .Net, enterprise Java) makes Web services an ideal solution for application integration, and for exposing selected business functions of complex enterprise applications.

In summary, business and technological factors discussed in this section have created a situation where delivery of enterprise application in the form of services becomes both technically possible, and economically desirable. This is likely to have major impact on enterprise computing over the next five years, finally tipping the balance from licensed software towards software delivered as a service.

4. PREREQUISITES FOR SUCCESSFUL ADOPTION OF THE ASP MODEL

As described above ASP model can bring a number of significant benefits to end user organizations, and potentially even strategic advantage. The subject of gaining strategic advantage from ICT has been debated extensively recently. Some observers argue that because IT is widely accessible to most organizations it can no longer bring strategic advantages over competitors. According to Carr (Carr, 2003) "IT's potency and ubiquity have increased, so too has its strategic value. It's a reasonable assumption, even an intuitive one. But it's mistaken. What makes a resource truly strategic—what gives it the capacity to be the basis for a sustained competitive advantage—is not ubiquity but scarcity. . . . When a resource becomes essential to competition but inconsequential to strategy, the risks it creates become more important than the advantages it provides." And from this standpoint Carr recommends new rules for ICT management: a) Spend less, b) Follow, don't lead, c) Focus on vulnerabilities, not opportunities. We believe that such arguments are essentially flawed as they imply "a steady state situation" in an ICT environment that is rapidly evolving. Migration towards the software-as-service model presents an opportunity to gain strategic advantage for companies able to take advantage of this new paradigm in a timely and effective manner. While benefits of adopting the ASP model will not follow automatically, we believe that this new model for enterprise application delivery can bring strategic advantages to organizations that are able to re-engineer their business processes to suit service-oriented computing. This view is supported by recent IDC research (IDC, 2002). More specifically, the ASP approach can bring business benefits if the management of the enterprise is able to control the risks associated with the adoption of ASP. We discuss the critical success factors (CSF) for the ASP model in the following sections.

Close Link Between Business, ICT, and Sourcing Strategies

It is recognized today that business and ICT strategies cannot be developed independently, and that ICT strategy cannot be simply derived from the business strategy. Importantly, organizations whose core business is based on ICT (banks, insurance companies, telecoms, etc.) have to develop the main components of both strategies together. The adoption of application servicing introduces a new requirement to define sourcing strategies for core business and for ICT together. Business strategy defines the core business of the organization, separating core business processes from supporting business processes, and defines the partners and their competences and responsibilities as used in the supply chain. When organization decides to outsource a supporting business process to an external partner (so called Business Process Outsourcing – BPO) then in most cases the corresponding ICT services has to be outsourced as well. ICT strategy defines which ICT services are delivered to different groups of users (employees, top management, partners in the supply chain, customers, and public). These services have to be aligned with the products and services produced by core business processes (e.g. internet banking), ICT strategy specifies ICT processes and ICT resources that are used to deliver ICT services, and which ICT resources are owned by the organization and which by its external partners. ASP is one of possible outsourcing options that need to be considered (Feuerlicht and Vorisek, 2003). Outsourcing decisions have impact not only on ICT efficiency but also on flexibility and competitiveness of organizations (Aberbethy, 2004). As a result of increasing importance of sourcing management some authors (Dignan, 2004) recommend new specialized job position – Chief Resource Officer (CRO). The CRO is the technology executive who oversees all of outsourcing agreements and ensures the cooperation of all providers.

Effective Management of Business and ICT Processes and Resources

Another key element for effective ASP deployment is suitable structure of business and ICT processes management. In this respect the SPSPR model is a useful tool (Feuerlicht and Vorisek, 2002). Using SPSPR model enables organization to define ICT services that support business processes effectively and to evaluate which IS/ICT services, processes, and resources to outsource and which maintain in-house. The model supports coexistence of different methods of service delivery (in-house, BPO, ASP etc.). Using this model, business goals are achieved through core business processes. Both the core and supporting business processes are supported by IS/ICT services, which are defined in the form of SLA's. IS/ICT services are delivered and controlled by IS/ICT processes, which consume IS/ICT resources. The purpose of dividing business and IS/ICT management into five layers is to identify the responsibilities of different types of business and ICT managers in a transparent manner that delineates the business goals up to the layer of ICT resources management (Figure 1).

The SPSPR also enables the creation of a set of metrics that can be used for business and ICT evaluation and control. The IS/ICT service layer between "business and ICT domain" enables improved communication between business process managers and ICT

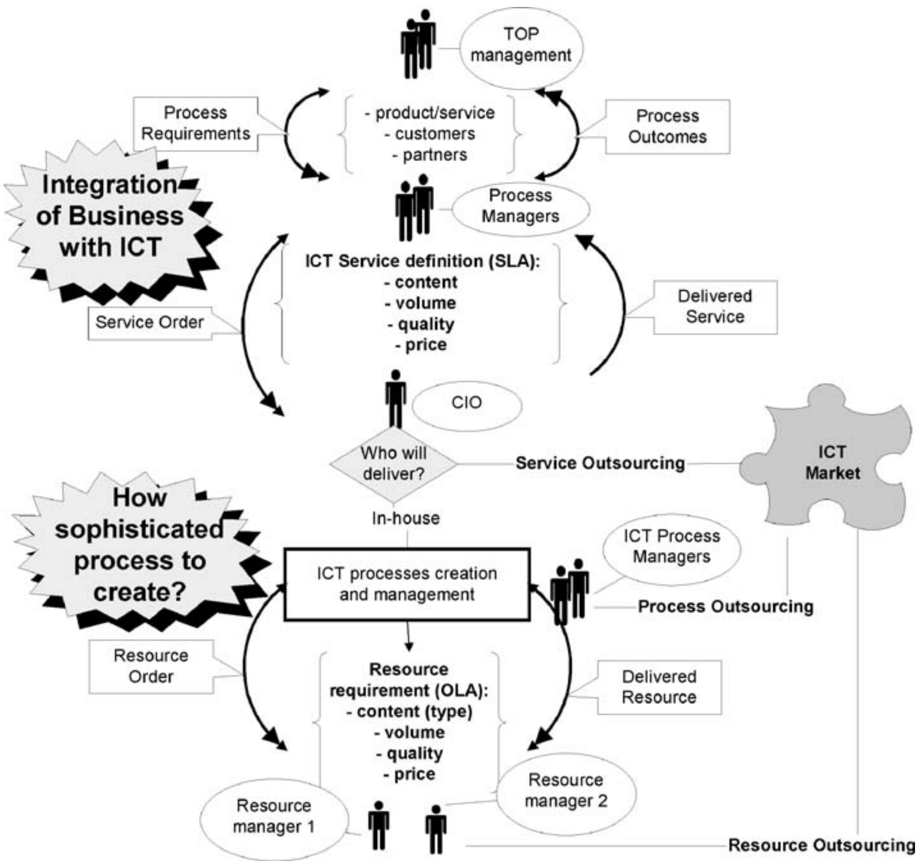


Figure 1. Types of managers and their responsibilities.

managers and clearly distinguishes their responsibilities. Business process manager is responsible for acquisition services so that ICT services support business process in the most effective manner. ICT manager is responsible for delivery of ICT services in accordance to agreed SLAs. In this respect our model is consistent with the Ross and Weill's opinion (Ross and Weill, 2003). IS/ICT processes are incorporated into the model in order to take advantage of the methodologies of IS/ICT processes management such as ITIL or COBIT. Irrespective of the deployed methodology, the IS/ICT processes have to be implemented so that they enable the organization to achieve its goals (Table 2 below).

Technology Architecture Design and Management

Often outsourcing decisions only consider the two extreme variants of outsourcing – "all in" or "all out". This simplistic approach can lead to suboptimal results. In practice, a more sophisticated approach is needed that considers critical success factors for different types of outsourcing as well the benefits associated with alternative outsourcing strategies

Table 2. Business goals and ICT processes outcomes

	Goals	Outcome of the Process and its Priority
Strategic Level	• New business opportunities • Exclusive partner relationships	Complex changes of partner relationships (including ASP) *Priority:* Integration of goals and visions of organization and providers
Tactical Level	• Enhancement of business processes efficiency • Flexible IS/ICT services and resources aligned with business needs • Competitive advantage	Service portfolio management; on-time evaluation of insufficient system performance; incident, problem and change management; changes of SLA's (according to business, technology and service supply development) *Priority:* Flexible SLA's
Operational Level	• IS/ICT services delivery according to SLA's • Services and resources costs evaluation	Service delivery management, user support, reports and costs evaluation *Priority:* Service according to SLA

(Feuerlicht and Vorisek, 2003). Effective deployment of the ASP model relies on the existence of suitable standard-based service-oriented architecture. Mature ICT architecture and consistent use of standards can lead to overall reduction in the total cost of ownership (TCO) and can facilitate outsourcing decisions. Outsourcing decisions need to be made in the context of an architectural framework, rather than ad-hoc.

5. CONCLUSIONS

In this paper we have presented a number of compelling arguments that make the software-as-service model an attractive alternative for the delivery of enterprise applications. Many of the benefits listed above are results of recent technological advances and increasing sophistication of the service providers. The speed of further adoption of the ASP model for delivery of enterprise applications will to a large extend depend on the ability of both providers and the users to minimize the risks and maximize the advantages associated with application servicing.

In conclusion, the ASP model provides a viable alternative to traditional software licensing model and it is likely that the ASP model supported by new computing architectures and technologies (i.e. Utility Computing, Service-Oriented Computing) will become the dominant method for delivery of enterprise applications in not too distant future. This view is supported in the literature, for example Gartner ranked "software as service" as one of the current megatrends and predicts that up to 40 percent of all applications will be delivered over the Internet within the next 2 to 3 years.

There are important consequences of the shift from the software-as-license model to the software-as-service model for the delivery of enterprise applications, creating new opportunities and challenges for both the providers and client organizations. The reduction in the size of the traditional software license market, reduced demand for on-site implementation and the corresponding increase in demand for application services will lead to further rationalization of the ICT vendor market (Cohen, 2004).

REFERENCES

Aberbethy, M., 2004, Outsource tack loses its edge, *Newsweek Bulletin* (February 11, 2004); http://bulletin.ninemsn.com.au/bulletin/EdDesk.nsf/All/FA57D33080F2C482CA256E2900778612.

Boulton, C., 2004, IBM, Veritas lead new utility computing standard (March 15, 2004); http://www.internetnews.com/ent-news/article.php/3311791.

Carr, N. G., 2003, IT doesn't matter, *Harvard Business Review* **81**(5).

Cohen, P., 2004, Twelve technical and business trends shaping the year ahead (March 1, 2004); http://www.babsoninsight.com/contentmgr/showdetails.php/id/687.

Dignan, L., 2004, Outsourcing overseers needed (February 5, 2004); http://www.baselinemag.com/article2/0,3959,1526051,00.asp.

DMTF, 2004, DMTF announces new working group for utility computing (April 12, 2004); http://www.dmtf.org/newsroom/releases/2004_02_17.

Dubie, D., 2004, Vendors make the utility computing grade, Network World Fusion (March 22, 2004); http://www.nwfusion.com/news/2004/0322dellsummit.html.

Eisenberg, R., 2004, Open for business (April 6, 2004); http://www.intelligenteai.com/showArticle.jhtml?articleID=19502159.

Eriksen, L., 2003, Will the real utility computing model please stand up (December 25, 2003); http://www.utilitycomputing.com/news/342.asp.

Ferguson, R. B., 2004, SAP sets its sights on SOAP (May 10, 2004); http://www.eweek.com/print_article/0,1761,a=126512,00.asp.

Feuerlicht, G., and Phipps, T., 2000, Architecting enterprise applications for the internet, *Journal of Applied Systems Studies*, Special Issue on "Applied Cooperative Systems", July, 2000, ISSN 1466–7738.

Feuerlicht, G., and Voříšek, J., 2002, Delivering application services: Who will benefit?, in: *Proceedings of Systems Integration 2002 conference*, J. Pour, ed., VŠE, Praha, pp. 31–41, ISBN 80-245-0300-x.

Feuerlicht, G., and Voříšek, J., 2003, Key success factors for delivering application services, in: *Proceedings of Systems Integration 2003 conference,* VŠE, Praha, pp. 274–282, ISBN 80-245-0522-3.

Haber, L., 2004, ASPs still alive and kicking (January 30, 2004); http://www.aspnews.com/trends/article.php/3306221.

IDC, 2002, IDC's findings reveal ASP implementations yielded an average return of investment of 404% (February 8, 2002); http://www.idcindia.com/pressrel/Show.pressrel.asp?prpath=prt20020038.htm.

Levy, A., 2004, What do enterprises want from ASPs? (February 4, 2004); http://www.aspnews.com/analysis/analyst_cols/article.php/2217411.

McCabe, L., 2004, A winning combination: Software-as-services plus business consulting and process services, Summit Strategies Market Strategy Report (March 25, 2004); http://www.summitstrat.com/store/3ss07detail.

Neel, D., 2002, InfoWorld (April 10, 2002); http://www.infoworld.com/article/02/04/12/020415 feutility_1.html.

Nevens, M., 2002, The real source of the productivity boom, *Harvard Business Review* **80**(3):23–24.

Newcomb, K., 2004, The second coming of ASPs, ASPnews.com (May 5, 2004); http://www.aspnews.com/analysis/aspnews_analysis/article.php/11276_3349851_2.

Oracle, 2004, Oracle Financial Reports (May 20, 2004); http://www.oracle.com/corporate/investor_relations/analysts/.

Ross, W., and Weill, P., 2003, Six IT decisions your IT people shoulnd't make, *Harvard Business Review* **80**(11).

SAP, 2004, SAP financial reports (May 20, 2004); http://www.sap.com/company/investor/reports/.

Voříšek, J., Pavelka, J., and Vít, M., 2003, *Aplikační služby IS/ICT formou ASP – Proč a jak pronajímat informatické služby*, Grada Publishing, Praha, ISBN 80-247-0620-2.

Wainewright, P., 2004, Secret weapons of ASPs (February 27, 2004); http://www.aspnews.com/analysis/analyst_cols/article.php/3090441.

ISD AS KNOWLEDGE WORK – AN ANALYSIS OF HOW A DEVELOPMENT METHOD IS USED IN PRACTICE

Per Backlund*

1. INTRODUCTION

Information systems development (ISD) has recently been characterised as knowledge work which is based on a distinct body of knowledge (Hirschheim and Klein, 2003; Iivari, 2000). Iivari (2000) proposes a framework for ISD as knowledge work which consists of four levels: paradigms, approaches, methods, and techniques. These levels represent an increasingly concrete representation of the body of ISD knowledge; paradigms being the most general and abstract part, and techniques being the most concrete and specific part of the body of knowledge.

The detailed use of ISD techniques, methods, approaches, and paradigms is relatively unexplored (Iivari, 2000). In this paper we pursue an ethnographically inspired approach (Williamson, 2002), to investigate a phenomenon in an empirical setting. We present a case study that evaluates the actual use of a commercial development method, the Rational Unified Process (RUP) (Jacobson et al., 1999; Kruchten, 2000).

There are reports on method adaptation (Russo et al., 1996; Fitzgerald, 1997; Fitzgerald et al., 2002). However, these reports tend to focus on method adaptation and use from an organisational point of view. There is also the field of method engineering (Brinkkemper et al., 1998; Ralyté and Rolland, 2001) with a focus on the method as such. We aim to extend such studies of method use by taking the actual situation of use into account in our analysis.

The aim of this paper is twofold:

- firstly, to extend the framework presented by Iivari (2000) to comprise the perspectives of knowledge presented by Alavi and Leidner (2001);
- secondly, to apply the extended framework in order to characterise and analyse how development process knowledge is utilised in a project setting.

* School of Humanities and Informatics, University of Skövde, P.O. Box 408, SE-541 28 Skövde, Sweden, per.backlund@ida.his.se.

In this paper we analyse method use from a knowledge point of view, i.e. how the knowledge embedded in a method is used. Thus we aim at highlighting the interplay between explicit development process knowledge and the tacit dimension of development process knowledge (the craft-like and professional dimensions).

We focus on the phase subsequent to the introduction and organisational implementation of the development method, i.e. in its situation of use. In order to better understand the adoption of methods we propose that the level of adoption of the knowledge embedded in the new development method depends on the perspective of knowledge that we take. In order to investigate this phenomenon we combine the framework of Iivari (2000) and the knowledge perspectives of Alavi and Leidner (2001).

The rest of the paper is organised as follows. Firstly, we give a brief introduction to knowledge work. Then we present the framework and combine it with the perspectives of knowledge in order to be able to use the expanded framework in our analysis of the empirical case which is presented in Section 4. Finally, we present our case analysis in Section 5 and close the paper with a concluding discussion in Section 6.

2. PERSPECTIVES OF KNOWLEDGE

We view knowledge work in a similar fashion as do for example Nonaka and Takeuchi (1995), Leonard (1995), Davenport and Prusak (1998), and Wiig (1993). This view can be described in terms of the two sub-areas knowledge-building and knowledge-use. Modern organisations spend a large effort in organising their knowledge and its use so that knowledge work can be facilitated.

Davenport and Prusak (1998) offer a working definition in which they state that knowledge is "[. . .] a fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organisations, it often becomes embedded not only in documents or repositories but also in organisational routines, processes, practices, and norms."

Nonaka and Takeuchi (1995) and Wiig (1993) define two types of knowledge: explicit and tacit. Explicit knowledge can be articulated in natural and formal language via e.g. documents and other types of records. Tacit knowledge has to do with personal knowledge that is embedded in personal experience and is therefore not so easy to formalise and record. In this paper we view ISD methods as development process knowledge that has been made explicit by the method developer. Methods may be perceived as materialised bodies of knowledge (Wastell, 1999). The knowledge embedded in the method render the expertise necessary to carry out complex tasks. A main claim by Tolvanen (1998) is that methods should not be viewed as universally applicable. Instead, method knowledge is viewed as situational affected at the different levels: organisation, project, and individual. We also recognize these levels but in addition we claim that they need to be complemented by taking into account the fact that there are different perspectives of knowledge.

Alavi and Leidner (2001), based on an extensive survey of IS literature, describe six different perspectives of knowledge present in the IS community:

- *Access to information*. Knowledge is a condition of access to information.

- *Capability*. Knowledge is the potential to influence action
- *Process*. Knowledge is the process of applying expertise.
- *Object. Knowledge refers to objects, which can be stored and manipulated.*
- *Knowledge vis a vis data and information*. Data is facts. Information is processed/interpreted data. Knowledge is personalised information.
- *State of mind*. Knowledge is the state of knowing and understanding.

The perspectives are motivated by the fact that ISD comprises all these views of knowledge. We propose that the first four perspectives are useful when evaluating the use of ISD methods in practice. According to Alavi and Leidner (2001) the knowledge vis a vis data and information view is common in IS and IT literature and constitutes one of the fundaments for how information systems development is perceived. However, we find this view more useful when participating in the discussion on data vis a vis information in information systems than in the analysis of method use.

The perception of knowledge as objects is relevant in that it implies some sort of representation of knowledge. ISD methods can be viewed as knowledge objects that are used in organisations. The objectified view of knowledge is also present in the area of patterns, see e.g. Fowler (1997). However, since it is not the same thing to actually use a pattern as it is to know of it the process view of knowledge is central. Knowledge work implies the skill of utilising the theoretical subjects of an area in a practical manner, i.e. applying expertise knowledge in order to build information systems. The process view of knowledge is essential in the perception of knowledge as an ability to apply expertise. Hutchins (1995) presents a model that describes the process of becoming skilled in a process in the sense of doing a task without reflecting on every step. In the access to information view of knowledge the main issue is that information must be organised to facilitate access to and retrieval of content. The view of knowledge as potential to influence action may be perceived as an extension to the access view in that it recognises that learning and experience result in an ability to interpret information and judge what information is necessary for decision making.

3. ISD AS KNOWLEDGE WORK

In this section we will give a brief overview of the framework for systems development process knowledge (Iivari, 2000). In short, the framework models ISD work in a hierachy of paradigms, approaches, methods, and techniques.

Iivari (2000) presents the notion of information systems development (ISD) as knowledge and distinguishes three components in information systems development knowledge: knowledge of information technology; knowledge of the application domain and ISD process knowledge. The concept does not refer to codified knowledge only.

The framework consists of four levels (Figure 1). There are four ISD paradigms into which the six ISD approaches, i.e. structured approach, information modelling, object-oriented approach, SSM-based approach, speech act-based approach, and trade unionist approach, are classified. An ISD approach is defined as a class of ISD methods that share a number of common features, i.e. an ISD approach can form a basis for zero or more methods (there are approaches which have no methods). Finally, a specific technique can

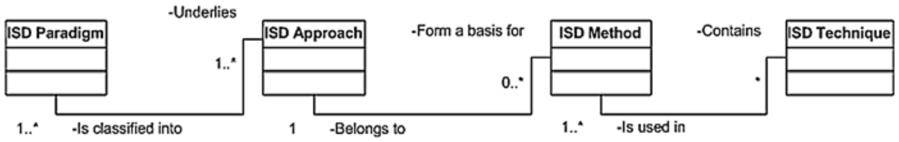


Figure 1. The relationships between ISD paradigms, approaches, methods, and techniques. Based on Iivari (2000).

be used in one or more methods and a method may contain many techniques. We are not concerned with the paradigmatic level per se in this paper since we wish to focus our analysis on the levels that are more explicitly present in the organisation. However, the epistemological dimension of the paradigmatic level is represented in the views of knowledge.

Iivari (2000) also provides a categorisation of ISD work into: routine, craft-like, professional, and creative knowledge work. Craft-like knowledge work is essentially skill-based and can only be learned through apprenticeship and practical experience. Skilful operation is characterised by an ability to recognise problems, diagnosing their cause and applying appropriate corrective procedures (Keller and Dixon Keller, 1996). Professional knowledge work is characterised by judgement and adaptation and creative knowledge work is characterised by intuition and imagination; and is therefore harder to understand and analyse. The difference between craft-like and professional knowledge work is subtle and some of the things characterising a skilled craftsman are also applicable for professional knowledge work. One distinguishing characteristic of professional knowledge work is its reflective qualities. Hence we may view this in terms of a sliding scale. Craft-like and professional knowledge work is further characterised by three features (Hirschheim and Klein, 2003): its close relationship to a person's identity which require hard work and mistakes to acquire; its connection to personal emotions and interests which makes it dependent on social interaction and socialisation, and; its holistic nature which makes hard to split into goals and means.

A combination of these theories results in a view of ISD as being not only influenced by the hierarchical relationship between paradigms, approaches, methods, and techniques. What we mean by utilising an approach/method/technique is also affected by the perspective of knowledge that we take.

4. THE CASE

The study was made at the IT department of a car manufacturer. In order to study how development process knowledge was utilised in an actual ISD project we followed a development project for eight months. The general aim of the project is to replace the numerous system registers in use with one general register. The new system is intended to provide a more consistent record of the various systems in use, which will aid in making system maintenance more efficient. Apart from the *product* goal the project also has two *process* goals. The first one: to give the team members an opportunity to *use* RUP in a real project setting; and the second one: to *introduce* new technology and a new development

Table 1. The situations in which data was collected

Object of Observation	Number of Instances
Development team meeting	15
Meeting with stakeholders	3
Tool workshop	2
Project documentation	2 (versions)
Interview	4

tool. The major risks of the project have been identified as the lack of resources in relation to the scope of the project and the introduction of the new tool.

The project is staffed by the following roles: one project manager, one architect, four analysts, four developers, five implementers, and one test designer; with a total number of eight people involved.

Data was collected by observing project meetings and work sessions taking detailed field notes, see Table 1. Furthermore, the observation data was complemented by informal discussions and interviews with project members. We also had access to internal project documentation in different versions which has given us an opportunity to review how the different RUP artefacts have evolved over time. The different objects of observation provide different views of how the development method was used in the project. Moreover, the different sources cater for source triangulation (Williamson, 2002).

The field notes and the project documentation were analysed using a combination of qualitative analysis (Patton, 1990) and inductive analysis (Hartman, 1998). The first part of the analysis work is the *content analysis*, which aims at identifying, coding and categorising the primary patterns in the data. This has been done by reading through field notes and project documentation in order to organise the data for further analysis.

5. AN EMPIRICAL INVESTIGATION OF ISD AS KNOWLEDGE WORK

In this section we will apply the combined framework to analyse an empirical case in which RUP was used. The analysis was conducted by finding indicators of routine knowledge work, professional knowledge work, and craft-like knowledge work respectively in the observations made. These indicators will then be discussed in relation to the levels technique, method, and approach. The aim of the analysis is to show how work was affected by the factors identified and to describe how the different perspectives of knowledge characterises what it means to use a technique, a method, and an approach respectively. We refer to this in terms of explicit and implicit method use (Backlund, 2004). Explicit method use refers to situations when parts of the development method were explicitly used, e.g. using templates. The explicit dimension of method use is present in the routine aspect of knowledge work. Implicit method use refers to the tacit knowledge of the developers. This comprises situations in which the developers carry on their work in the smooth fashion characterising a craftsman.

The indications of routine knowledge work are closely connected to the technique level whereas craft-like and professional knowledge work rather characterises implicit method use since they presuppose that the method has been learned and internalised. Hence we can characterise the interplay between implicit and explicit method use.

5.1. The Technique Level

The technique level typically supports routine tasks. An analysis from the object point of view gives at hand that we can view the documented techniques and templates in terms of knowledge objects that can be stored and manipulated. We found several indicators of this in the routine aspect. The function of the *development case* is to facilitate process tailoring. There is a significant amount of duplication from other projects involved in this task. The duplication is an example of reuse from other projects. However there is a risk that the new process is not sufficiently tailored if the degree of duplication becomes too high. The project did not utilise the process as described in the development case, which led to rework of the development case throughout the project, i.e. the method was continuously adapted during the project.

There is also the issue of whether a technique is truly implemented. This may for example be illustrated by a discussion between two developers about whether two *use case* descriptions were essentially describing the same thing; where one of the developers claimed that

> 'This is easier to code than to write in a use case description'. [Quotation 1, Developer]

One reason for this may well be the good domain knowledge possessed by the developers in the project, as they are developing a system internal to their own organisation. This is indicated by several occasions when the developers stated that

> 'We already know what this will look like.' [Quotation 2, Developer]

In these situations some of the developers experienced parts of the analysis work as superfluous. Hence a problem occurs when the process goal is to carry out an analysis using the new method.

In one situation a team member had problems understanding the concept of *extension points* in use cases. The document templates have a pre-specified heading for the purpose but the team members could not agree on whether it was necessary to do that work in the particular situation. We interpret this as a situation in which routine and craft like work counteract, thus leading to a shift of focus from the problem to be solved. This is hence an indication of *a technique not being completely implemented from a process point of view*. From a process view of knowledge we find that the skill to apply a new technique is relevant, e.g. how to carry out use case modelling in an efficient way. We found indicators of successful use in the use of templates as a basis for carrying out use case modelling. However, we also saw examples of problems in the organisation of use cases, which led to problems in ensuring consistency between use cases as well as in remembering the relations between use cases. This shows that most requirements do not come as neat use cases. Instead it is a question of modelling different alternatives in order to find the relevant use cases. The process of re-analysing use cases was going on throughout the project and the team members had to remind each other to update the documentation.

Taking an access to information view in our analysis we find that the new method being implemented is perceived as large and complex; a fact that overshadows the efforts in the project. For example, the project manager did not perceive any support from the new method in her management of the project. This is supported by the fact that only a very limited part of the RUP *project management workflow* was actually used during the project. This may be interpreted in terms of a lack of capability in the knowledge as a capability to influence action perspective, i.e. *the knowledge embedded in the method did not help to change the actions taken by the project manager.*

5.2. The Method Level

The method level is, to a large extent, mapped to craft-like knowledge work. We find work habits and the explicit use of routines at this level. It is typically at the method level that we discover old work habits overriding new ones even though new types of artefacts are produced.

At this level we find the process view of using the new method most important. We view this in terms of internalising the new method in order to make it a part of the individual and collective knowledge. Never the less, methods are often viewed as commodities that can be purchased, which essentially is an objectified view of knowledge. However, the knowledge embedded in an artefact is not simply re-extracted when the artefact is used (Hildreth and Kimble, 2002). One symptom of this is the situation described by Fitzgerald (2001) where the output of a work process is adapted to give the impression that a specific method has been used.

The *use case driven way of work* versus the data driven way of work in database modelling has rendered the developers in the team problems. Most of them have a genuine database background and according to some of them it is hard to work in a fashion where they should develop the system by modelling the way that users are supposed to interact with the system and then identify relevant data; as opposed to identifying relevant data in the domain. The developers state that this is problematic since they have good domain knowledge, as they are developing a system internal to their own organisation.

> 'We tend to know where we are going. [...] You are geared towards a solution. [...] You think in code, rather than discarding that from start.' [Quotation 3, Developer]

Due to this situation they have preconceived notions of the domain which are problematic when trying to adopt a use case driven way of work.

Work is typically carried out in a craft-like manner utilising already existing skills. This is especially apparent when it comes to situations of problem solving. Skilful (craft-like) operation is characterised by an ability to recognise problems, diagnosing their cause and applying appropriate corrective procedures (Keller and Dixon Keller, 1996). These types of actions are typically illustrated by situations in which the team utilised database modelling techniques in an informal way to understand problems.

Risk management was put forward as a positive aspect of the new method. It was generally stated among the team members that managing risks had become more in focus. There was an awareness of such issues before the new method was introduced as well but the general opinion is that the awareness has been made more explicit, i.e.

'This is the way we are supposed to work according to company standards' [Quotation 4, Developer].

We interpret this as an *indication of a move towards a more professional attitude* in that it includes a valuation of current work habits and routines.

The major conflict between the new method and old work habits was detected in the *project management* aspect of the project. The project manager perceived little value added by the project management workflow and used the in-house method to manage the project.

'I haven't really seen the benefit of RUP project management. I rather turn to my PCM [the internal project management method] management habits.' [Quotation 5, Project manager]

This statement was made by the project manager who has about five years of experience of the in-hose project management method. This is also emphasised by the fact that only a limited part of the RUP project management workflow was utilised in the project. However, this is, to some extent, in accordance with the company's view of how the new method should be used. The organisation has described a 'method landscape' where RUP is one method in conjunction with a number of in-house methods.

5.3. The Approach Level

Object orientation is perceived as problematic by the developers since they tend to think ahead, focusing on the fact that they know that there is going to be a relational database solution underlying the future system. This is seen as a problem when trying to adopt an object oriented approach as it obstructs the modelling efforts. We may view this in terms of a conflict between the object oriented model and the relational model. Such a conflict, on the approach level, may be hard to resolve as it involves a change of the personal mindset and problem solving strategies. The fact that the developers turn to data modelling techniques when they encounter problems is an indication of this. Furthermore, it is also a fact that technical problems in connection to database connectivity had to be solved in the design phase of the project. Finally, some of the developers put forth the fact that they could not foresee object oriented databases in the organisation due to company policies and legacy systems in use, something which is also obstructing the new approach.

One of the developers also expressed doubts about the organisation's ability to take in the new approach, present in RUP, in terms of how the six *best practices* (develop iteratively, manage requirements, use component-based architectures, use visual modelling, continuously verify software quality, and control changes to software) are used. The developer in question claimed that there is a focus on visual modelling only, at the expense of the other practices. This, in turn, is an indication of the fact that the new approach is not fully understood, according to the same person.

Iterative work is cumbersome for many reasons since it is hard to estimate both the extent and time consumption of an iteration. The problems that occurred in the project were associated with the small project scope that did not actually call for advanced iteration planning. Hence, the need and motivation for iteration planning was low and therefore iterative work was not applied as described by RUP. Never the less, much of the work carried out was iterative in that the team returned to artefacts and developed them further, even though this was not explicitly planned. We interpret this in terms of the developers being used to working iteratively in some sense (the process view of knowledge) but it is not that easy to make it explicit in planning and documentation.

> 'It is not easy for a project manager to control iterative work. [. . .] you tend to go on where you stopped the last time.' [Quotation 6, Developer with project management experience]

This illustrates the need to focus on releases in iterative planning. We did not find any indications of iterative work causing problems due to fixed contracts as reported in (Madsen and Kautz, 2002). We attribute some of this effect to the low priority of the project. We also see that iterative work makes sense from a software engineering perspective.

The use *case driven approach* was frequently discussed in various situations. One problematic issue is the order in which to design and implement the use cases. RUP proposes an approach where the most complicated use cases should be attended to first. This was perceived as problematic since the functionality of and output from some of the less complex use cases was considered useful in building the more complex ones. The problem was resolved by a risk management approach that was based on a more thorough analysis of those critical use cases. However, the use cases were built in a sequence that allowed the developers to use and reuse as much code and output as possible. The strategy to *code for reuse* also led to more sophisticated solutions than necessary, since the team saw future use for solutions designed in a certain way. Such a strategy may be counterproductive for the specific project but productive for the organisation as a whole. One major reason for adopting the strategy is the fact that the project is to be a reference project for the rest of the organisation.

6. DISCUSSION AND CONCLUSION

We have demonstrated how the framework presented by Iivari (2000) can be extended to comprise the different knowledge perspectives (Alavi and Leidner, 2001) and then be applied on an empirical case to analyse the incorporation of a new development method. By doing this, we have been able to highlight the interaction between explicit and tacit development process knowledge. These two contributions extend the existing theory and highlight the knowledge aspects of information systems development respectively. The main contributions of the paper are the extension of the framework for ISD as knowledge work and an application of the extended framework to analyse an empirical case.

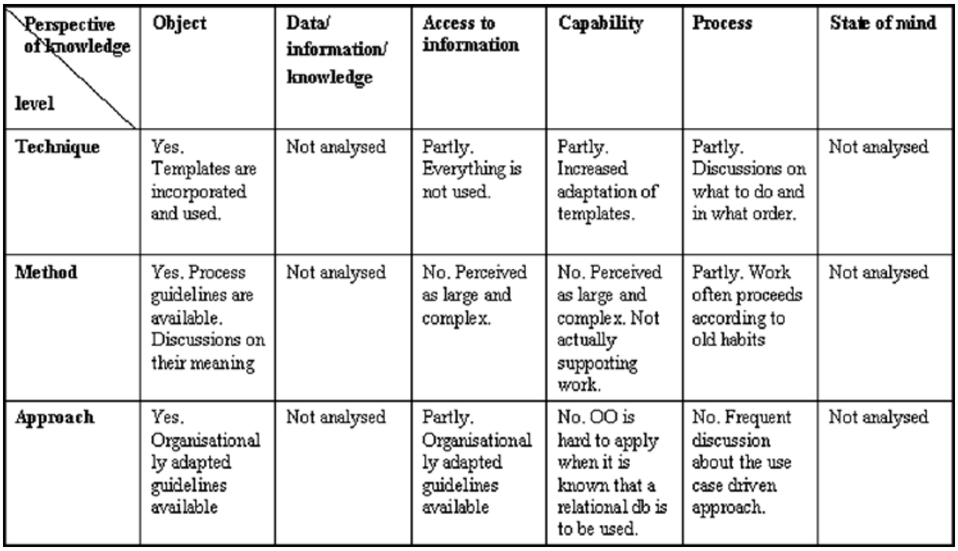
The indications of routine knowledge work are closely connected to an explicit use of the ISD method whereas craft-like, professional, and creative knowledge work rather characterises implicit method use at the method and approach levels respectively, since they presuppose that the method has been learned and internalised. Hence we can describe the interplay between implicit and explicit method use. The indicators are summarised in Table 2.

When comparing our findings with Madsen and Kautz (2002) we find that the adoption of the underlying development perspective in the organisation has been changed to some extent, whereas Madsen and Kautz (2002) report on a waterfall model supplemented with new tools and techniques. Moreover, we find the explanation for this phenomenon in the

Table 2. Summary of the case findings

Technique level	Method level	Approach level
Use templates	Work habit	Strategy
Use artefacts	Fluent work	Valuation of routine
Update artefacts	Explicit use of routine	Valuation of work habits
Create artefacts	Old habits overriding new ones	Valuation of approaches
Organise artefacts		

Table 3. Applying the extended framework. The level of method incorporation depends on the perspective of knowledge



Perspective of knowledge / level	Object	Data/ information/ knowledge	Access to information	Capability	Process	State of mind
Technique	Yes. Templates are incorporated and used.	Not analysed	Partly. Everything is not used.	Partly. Increased adaptation of templates.	Partly. Discussions on what to do and in what order.	Not analysed
Method	Yes. Process guidelines are available. Discussions on their meaning	Not analysed	No. Perceived as large and complex.	No. Perceived as large and complex. Not actually supporting work.	Partly. Work often proceeds according to old habits	Not analysed
Approach	Yes. Organisational ly adapted guidelines available	Not analysed	Partly. Organisational ly adapted guidelines available	No. OO is hard to apply when it is known that a relational db is to be used.	No. Frequent discussion about the use case driven approach.	Not analysed

fact that there are differences in analysing the work process depending on what perspective of knowledge we adopt.

We have shown how the different indicators of knowledge work can be mapped to the framework and connected to the different perspectives of knowledge. An overview of the findings is presented in Table 3. The extent to which the method can be said to be implemented in the organisation decreases when the capability and process perspectives of knowledge are used. Our interpretation of this is that it is harder to change the process dimension of knowledge since it involves the internalisation of new knowledge as opposed to an objectified view which does not acknowledge such problems. By adopting a knowledge based view of ISD it becomes possible to refine the analysis of method use in organisations. For example, it can provide a part of the explanation to why the time it takes to introduce a new method is often underestimated.

We can also interpret Table 3 horizontally in order to highlight the explicit/tacit dimensions of knowledge. The method and approach levels are more geared towards the tacit dimension of ISD. The further down we move the harder it is to change a work habit, since it involves changing already internalised knowledge and essentially skilled based abilities. We view this as an explanation to the problem of introducing new methods and approaches, as reported by e.g. Kautz and McMaster (1994) and Middleton (1999).

The view of knowledge is important when we analyse if and to what extent a new method has been implemented in an organisation. There is no straight forward answer to the question whether the new method has been implemented or not. The answer to a large extent depends on the perspective of knowledge that we take when conducting the analysis. In that sense we can use *the combined framework as a diagnosis tool for analysing method adoption*.

Obviously there are limitations to a single case study made in one organisation. Hence we do not aim at making statistical generalisations from our material. We rather aim at a contribution of rich insight, as described by Darke, Shanks, and Broadbent (1998). As no organisations are alike, the combined framework is useful as a provider of general knowledge about method use in organisations. Its strength thus lies in pointing out the complexity of method use. Furthermore, we argue that the empirical study of information systems development as knowledge work adds to the understanding of the area. The largest limitation is perhaps the low priority of the project since it may have an effect on the involvement of the developers. However, we do not judge this effect as severe since the motivation for using new tools and learning the new process have been high. Furthermore, the application under development is the object of great interest in the organisation, which should serve as a motivator.

ACKNOWLEDGEMENTS

The author wishes to thank Benkt Wangler and Anne Persson for their valuable advice. This research is sponsored by the Swedish Knowledge Foundation (KK-stiftelsen).

REFERENCES

Alavi, M., and Leidner, D. E., 2001, Review: Knowledge management and knowledge management systems: conceptual foundations and research issues, *MIS Quarterly* **25**:107–133.

Backlund, P., 2004, Adopting the Knowledge Embedded in Development Methods – The Challenge of Aligning Old and New Practices, in: *The 12th European Conference on Information Systems (ECIS 2004)* T. Leino, T. Saarinen, and S. Klein, eds., Turkku, Finland.

Brinkkemper, S., Saeki, M., and Harmsen, F., 1998, Assembly Techniques for Method Engineering, in: *CAiSE'98*, Vol. LNCS 1413, B. Pernici and C. Thanos, eds., Springer, Pisa.

Darke, P., Shanks, G., and Broadbent, M., 1998, Sucessfully completing case study research: combining rigour, relevance, and pragmatism *Information Systems Journal* **8**:273–289.

Davenport, T. H., and Prusak, L., 1998, *Working Knowledge: How Organisations Manage What they Know*, Harvard Business school, Boston, Mass.

Fitzgerald, B., 1997, The use of systems development methodologies in practice: A field study, *The Information Systems Journal* **7**:201–212.

Fitzgerald, B., 2001, Method-in-Action: A Framework for IS Development, Lecture notes, University of Skövde.

Fitzgerald, B., Russo, N. L., and O'Kane, T., 2002, Software development method tailoring in Motorola, *Communications of the ACM* **46**:64–70.

Fowler, M., 1997, *Analysis Patterns Reusable Object Models*, Addison-Wesley, Menlo Park, California.

Hartman, J., 1998, *Vetenskapligt tänkande Från kunskapsteori till metodteori*, Studentlitteratur, Lund.

Hildreth, P. M., and Kimble, C., 2002, The duality of knowledge, *Information Research* **8**.

Hirschheim, R., and Klein, H., 2003, Crisis in the IS field? A critical reflection on the state of the discipline, *Journal of the Association for Information Systems* **4**:237–293.

Hutchins, E., 1995, *Cognition in the Wild*, MIT Press, Cambridge, Massachusetts.

Iivari, J., 2000, Information Systems Development as Knowledge Work: The body of systems development process knowledge, in: *Information Modelling and Knowledge Bases XI*, E. Kawaguchi, I. A. Hamid, H. Jaakkola, and H. Kangassalo, eds., IOS Press.

Jacobson, I., Booch, G., and Rumbaugh, J., 1999, The unified process, *IEEE Software*, pp. 96–102.

Kautz, K., and McMaster, T., 1994, Introducing structured methods: An undelivered promise? – A case study, *Scandinavian Journal of Information Systems* **6**:59–78.

Keller, C. M., and Dixon Keller, J., 1996, *Cognition and tool use*, Camebridge University Press, Cambridge.

Kruchten, P., 2000, *The Rational Unified Process An Introduction Second Edition*, Adison-Wesley, Reading, Massachusetts.

Leonard, D., 1995, *Wellsprings of Knowledge Building and Sustaining the Source of Innovation*, Harvard Business School Press, Boston.

Madsen, S., and Kautz, K., 2002, Applying System Development Methods in Practice – The RUP Example, in: *Information Systems Development (ISD)*, J. Grundspenkis, M. Kirikova, W. Wojtkowski, G. Wojtkowski, S. Wrycza, and J. Zupancic, eds., Kluwer Press, Riga.

Middleton, P., 1999, Managing information system development in bureaucracies, *Information and Software Technology* **41**:473–482.

Nonaka, I., and Takeuchi, H., 1995, *The knowledge-creating company: How Japanese companies create the dynamics of innovation*, Oxford University Press, New York.

Patton, Q. M., 1990, *Qualitative Evaluation and Research Methods*, SAGE Publications, London.

Ralyté, J., and Rolland, C., 2001, An Assembly Process for Method Engineering, in: *CAiSE'01*, K. Dittrich, A. Geppert, and M. Norrie, eds., Springer, Interlaken, Switzerland.

Russo, N. L., Hightower, R., and Pearson, M., 1996, The Failure of Methodologies to Meet the Needs of Current Development Environments, in: *The Fourth Conference of the British Computer Society Information Systems Methodologies Group*, N. Jayaratna and B. Fitzgerald, eds., BCS Publications, Cork, Ireland.

Tolvanen, J.-P., 1998, Incremental Method Engineering with Modeling Tools Doctoral Thesis, Department of Computer Science and Information Systems, Universtiy of Jyväskylä.

Wastell, D. G., 1999, Learning dysfunctions in information systems development: Overcoming the social defenses with transitional objects, *MIS Quarterly* **23**:581–600.

Wiig, K. M., 1993, *Knowledge Management Foundations*, Schema Press, Arlington.

Williamson, K., 2002, *Research methods for students, academics and professionals Information management and systems*, Centre for Information Studies, Wagga Wagga.

CUSTOMIZING TRACEABILITY IN A SOFTWARE DEVELOPMENT PROCESS

Patricio Letelier, Elena Navarro, and Víctor Anaya*

1. INTRODUCTION

Requirements management is a recognized key practice that deals with the definition and change of software requirements, and should be properly integrated as a subprocess in the software development process (Corriveau, 1996). The success of this subprocess depends on how well defined the relationships among requirements and other kinds of specifications generated by the software process are. Requirements traceability is defined as the ability to describe and follow the life of a requirement in both directions, towards its origin or towards its implementation, passing through all the related specifications. Requirements management and especially requirements traceability can be expensive activities. The detail level in these activities and the collected information must be configured according to the particular project needs, in order to obtain a positive cost-benefit ratio.

Nowadays, the effectiveness in traceability practices differs considerably among development teams. Some problems that can explain this situation are: there are not detailed guidelines regarding the kinds of information that must be gathered for traceability, the context in which such an information must be used, and the lack of consensus about the semantic for the links between specifications (Ramesh et al., 1998; 2001).

Requirements have been traditionally specified using textual forms of specification above all, mainly using natural language. Consequently, tools supporting requirements management have been focused on the manipulation of text pieces. These textual-expressed requirements are linked forming a traceability graph which is used to manage the requirements and their traceability. In this approach, the specifications generated in other activities of the development process can also be added to the traceability graph, representing them as text (normally using the name of the specification, for instance: the name of the class, attribute or operation). Test specifications are also mainly textual, therefore they can be

* Patricio Letelier and Víctor Anaya, Department of Information Systems and Computation, Polytechnic University of Valencia, Camino de Vera s/n, Valencia - 46022 (Spain), Phone: +34 96 387 7007, ext. 73589, Fax: +34 96 387 7359, {letelier | vanaya}@dsic.upv.es. Elena Navarro, Computer Science Department, University of Castilla-La Mancha, Avda. España s/n, Albacete - 02071 (Spain), Phone: +34 967 599200 – ext 2461, Fax: +34 967 599224, enavarro@info-ab.uclm.es.

handled in a similar way. Even though several CASE tool vendors claim that their products offer a good integration among requirements management, modeling and test, the solution is usually based on import/export mechanisms. Another suggested alternative is to specify explicitly in additional diagrams the traceability links between model elements. But apart from not covering all kinds of specifications, due to the complexity of the traceability graph, this is not viable, even for small systems.

In this work, we show an approach to face these issues by providing a set of guidelines to configure requirements traceability. In this way, traceability can be customized according to the specific needs of the project.

On the other hand, regarding the software modeling, UML (OMG, 2002) has quickly become the most popular notation for object-oriented modeling. Thanks to the definition of its metamodel and the included extension mechanism by defining profiles, UML offers an excellent opportunity to establish a common framework for representing specification of requirements, design and test. We use this extension mechanism to develop a framework based on a traceability metamodel. This is defined as a UML profile, integrating different kinds of traceability information. In this way, all involved artifacts have an homogenous representation compliant with UML. Our approach can be applied to any software process based on UML.

This paper is organized in seven sections. Following this introduction, section two describes our metamodel for requirements traceability. In section three we present the definition of the metamodel by means of a UML profile. In section four we describe the tasks for configuring the traceability in a project and the concepts of implicit and explicit traceability. The fifth section illustrates the application of our approach using a small project based on Rational Unified Process (RUP) and using SharpTrace, our tool which extends Rational Rose's tool. The sixth section describes some related works and specific tools for requirements management, from the perspective of frameworks for requirements traceability. Eventually, the seventh section presents our conclusions.

2. A METAMODEL FOR REQUIREMENTS TRACEABILITY

Before presenting our metamodel for requirements traceability we will summarize the information needs for requirements management. Next we indicate the kinds of information associated to requirements traceability and their possible uses (adapted from Dömges et al., (1998)):

- Traceability links between different specifications allow validating that the system functionality covers the stakeholders expectations, that there is not superfluous functionality implemented, and performing impact analysis when requirements change.
- Contribution structures (Gotel et al., 1997), that is, the links between stakeholders and specifications allow improving the communication and cooperation among stakeholders, and guaranteeing that the contribution of every individual is considered and registered.
- Rationale associated to specifications, including alternatives, decisions, assumptions, etc. contribute to improve the understanding and acceptance of the sys-

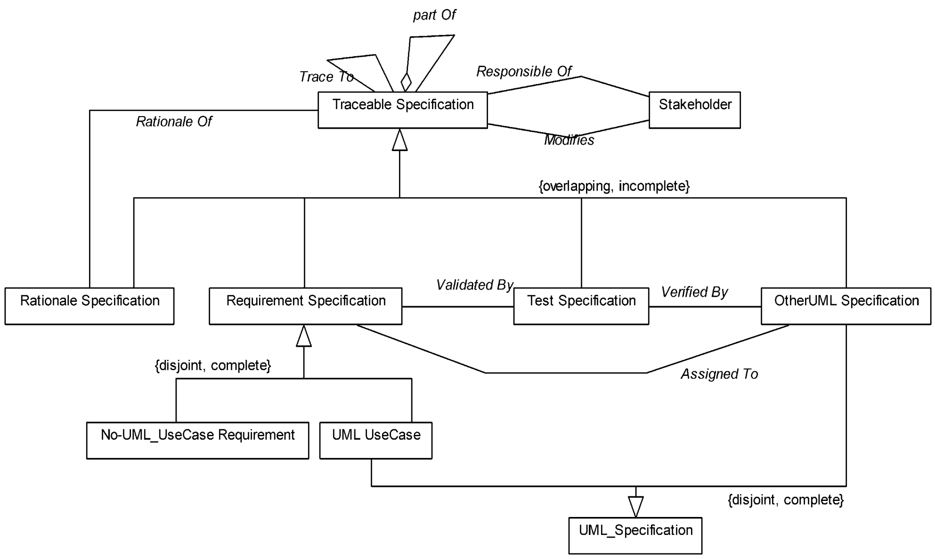


Figure 1. Metamodel for requirements traceability.

tem from the stakeholders, and to improve change management avoiding studying again considerations already excluded. This is possible thanks to making accessible the solutions, their foundations, and the excluded alternatives.

In Figure 1, by means of a class diagram, we present a core metamodel for requirements traceability. Classes represent entity types and associations represent types of traceability links.

Globally speaking, we are concerned with two types of entities: *TraceableSpecification* and *Stakeholders*. *Stakeholders* are responsible for creating and modifying specifications. A *TraceableSpecification* could be any software specification with a certain granularity level, that is, it can be a document, a model, a diagram, a section in a document, a text specifying a non-functional requirement, a use case, a class, an attribute, etc. The granularity for a *TraceableSpecification* is defined by means of the aggregation with the role name *partOf.*

The metamodel showed in Figure 1 covers the four perspectives of traceability information included in the works by (Ramesh and Jarke, 2001): requirements, rationale, allocation of requirements to model and implementation elements, and finally, test. Moreover, our metamodel incorporates *pre-traceabilit*y and *post-traceability* aspects (Jarke, 1998) and (Pohl, 1996). Pre-traceability allows going from the origins of the requirements until their explicit specification in the Software Requirements Specification (SRS) document or vice versa. Post-traceability allows going from the SRS to the subsequent software and test specifications or vice versa. In both kinds of traceability our metamodel provides the types of links *responsibleOf* and *modifies* to determine the *Stakeholders* involved. For pre-traceability the type of link *traceTo* is available between requirements expressed in different abstraction levels and *rationaleOf* for rationale associated to such requirements specifications. The post-traceability is supported by the types of links *traceTo*, *validatedBy*, *verifiedBy*, *assignedTo* and *rationaleOf*.

3. UML CONTEXT FOR THE METAMODEL

To make simple and practical the application of our metamodel it is convenient to integrate all types of entities and links in a common context. Considering that: (a) UML specifications are more precisely defined and accepted than other specifications, (b) UML provides extension mechanisms (stereotypes, tagged values and constraints) to incorporate new types of specifications, and (c) UML specifications are supported by most of the CASE tools, it is obvious that it would be appropriate to integrate all types of specifications of our metamodel in the context of UML. Thus, for each type of entity and type of link a correspondence with a UML model element will be established. To do this, UML metaclasses will be chosen as base classes to establish new stereotypes. When the type of entity or type of link matches up semantically with a UML metaclass, this metaclass will be used directly without defining a new stereotype. The result of this analysis is a UML profile for our traceability metamodel. Next we give details about how such integration is performed.

Traceability entities in the UML context. For the entity *Stakeholder* the choice is simple; the model element *Actor* is the metaclass used as a base class to define the corresponding stereotype. Other types of entities should have the possibility of permitting associations in order to establish aggregation relationships between them. According to this, the selected metaclass should be among the UML elements that are child classes of *Classifier*. For entities corresponding to non-UML standard specifications, the *Classifier* named *Artifact* (added in UML version 1.4) has been chosen. *Artifact* has some predefined stereotypes, among them «document», which is the one we will use to represent documents and document sections. For types of entities that match up directly with UML model elements (*UML_UseCase* and *OtherUML_Specification*) we will use the UML model element itself. On the other hand, we will use the UML *Package* to group and organize UML artifacts and *Stakeholders*. Optionally we will add the predefined stereotypes model or subsystem, depending on whether we are defining a model of a system/subsystem or dividing the system in subsystems, respectively.

Traceability links in the UML context. Types of links will be represented as UML model elements of *Abstraction* type, except the relationship *partOf*, which is represented by aggregation or composition between specifications using the metaclass *Association* as the base class. Although different types of links are modeled by different associations in our traceability metamodel, they are not independent, in fact, the type of link *traceTo* is a generalization of all other types of links. The type of link *traceTo* will be coincident with the stereotype «trace», predefined in UML. In UML a trace dependency indicates a historical or process relationship between two elements that represent the same concept without specifying derivation rules between them (OMG, 2002). Excepting the type of link *partOf*, other types of links will be child stereotypes of the predefined stereotype «trace».

Figure 2 and Figure 3 show the UML representation for types of entities and types of links included in our traceability metamodel. This representation constitutes an essential UML profile for requirements traceability.

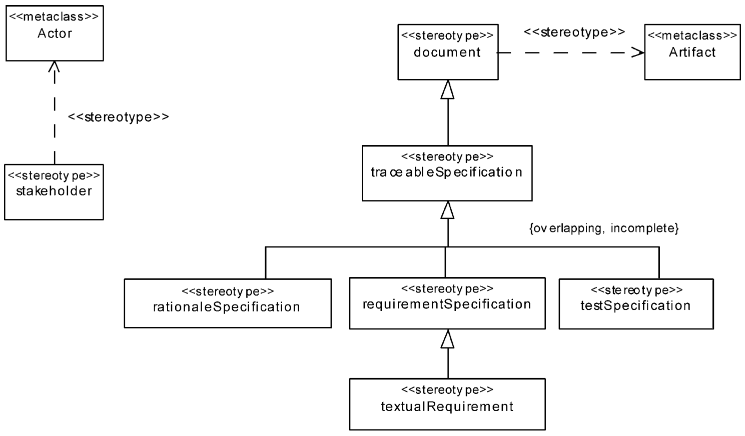

<<metaclass>> Actor
<<stereotype>>
<<stereotype>> stakeholder
<<stereotype>> document
<<stereotype>>
<<metaclass>> Artifact
<<stereotype>> traceableSpecification
{overlapping, incomplete}
<<stereotype>> rationaleSpecification
<<stereotype>> requirementSpecification
<<stereotype>> testSpecification
<<stereotype>> textualRequirement

Figure 2. Stereotypes for stakeholder and core textual specifications.

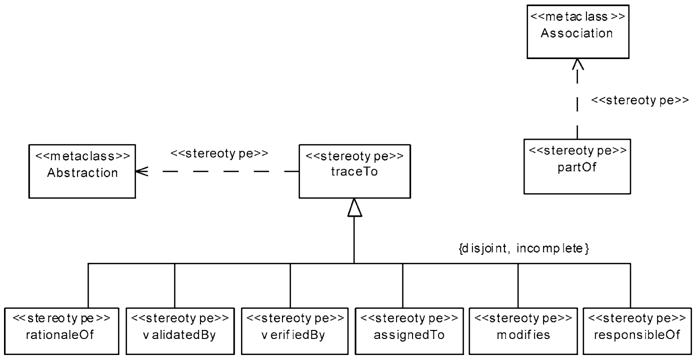

<<metaclass>> Association
<<stereotype>>
<<stereotype>> partOf
<<metaclass>> Abstraction
<<stereotype>>
<<stereotype>> traceTo
{disjoint, incomplete}
<<stereotype>> rationaleOf
<<stereotype>> validatedBy
<<stereotype>> verifiedBy
<<stereotype>> assignedTo
<<stereotype>> modifies
<<stereotype>> responsibleOf

Figure 3. Stereotypes for traceability links.

4. CONFIGURING TRACEABILITY

In requirements traceability we can identify two activities: (a) configuration of traceability according to the project needs, and (b) specifying and exploiting traceability information during the software development and maintenance. We will focus on the configuration activity by applying our UML profile for traceability. We will use the term artifact in a wider sense, beyond the definition provided in UML, and in the same way as most software processes do (for instance RUP). Thus, we will consider as artifacts all documents, files and other physical elements generated or used during the software development process, in addition, we will call artifacts any UML model element. The profile will act as a framework for establishing the types of artifacts relevant for traceability and the types of links between them. Consequently, traceability links established during the software development or maintenance are verified against the traceability configuration (for instance, for a particular project, certain types of links are valid only between certain types of artifacts).

Configuring traceability in a project includes the following tasks:

1. Extend the framework with the definition of the artifact types of the software development process used in the project and that are not considered in the profile.
2. Define aggregation relationships between artifacts. This task may not be necessary for all types of artifacts if such relationships are predefined and included in the description of types of artifacts.
3. Select the artifact types that will be traced. These are a subset of the artifact types used in the project.
4. Establish types of traceability links that are relevant to the project. The types of links are established between pairs of types of artifacts selected in task 3. In this case, it may also be necessary to extend the traceability profile including new types of links as stereotypes specializations.
5. Define criteria to implicitly derive traceability links and what types of links (established in task 4) will use these criteria. In the next subsection the concepts "explicit traceability" and "implicit traceability" are explained.

4.1. Explicit and Implicit Traceability

During the configuration of requirements traceability only the types of links important for the project are established (this is performed in task 4). However, even with this restriction, the effort associated to gather the information of traceability links can be considerable. Thus it is important to provide mechanisms that allow deriving automatically part of those traceability links. We will refer to "explicit traceability" when talking about those traceability links that are manually specified. Consequently, we will use "implicit traceability" when referring to those traceability links that are automatically derived according to some established criteria.

Next we describe an example to illustrate explicit and implicit traceability. Supposing the following traceability configuration (obtained as a result of tasks 1, 2, 3 and 4):

- Types of artifacts: A, B, C, D, E and F.
- Aggregation relationships: A ◇− B, D ◇− C.
- Type of links that are important to the project:

$$B \xrightarrow{\text{«assignedTo»}} C,\ B \xrightarrow{\text{«assignedTo»}} E,$$
$$B \xrightarrow{\text{«assignedTo»}} D,\ D \xrightarrow{\text{«assignedTo»}} E \text{ and}$$
$$E \xrightarrow{\text{«traceTo»}} F$$

In addition, we have the following artifacts instances (with their corresponding types): $a : A$, $b : B$, $c : C$, $d : D$, $e : E$ and $f : F$. Figure 4 shows a traceability graph associated to our example, where the following explicit traceability has been intro-duced: aggregation relationships $a \Diamond\!- b$ and $d \Diamond\!- c$, and the traceability links $b \xrightarrow{\text{«assignedTo»}} c$ and $d \xrightarrow{\text{«assignedTo»}} e$. Using transitivity as a simple criterion for implicit traceability, links $b \xrightarrow{\text{«assignedTo»}} d$ and $b \xrightarrow{\text{«assignedTo»}} e$ in Figure 4 could be derived. Others criteria for implicit traceability are described in (Letelier, 2002).

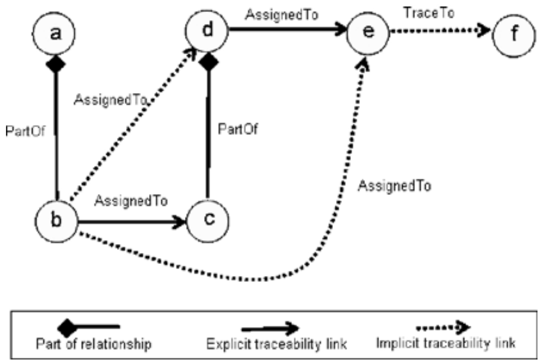


Figure 4. Example of a traceability graph.

5. CONFIGURING TRACEABILITY WITH SHARPTRACE

The presented metamodel and the corresponding profile are independent from the software development process, the only assumption is that the process is based on UML. Therefore, for illustrating the application of our approach we have chosen RUP as a process, mainly because it offers enough details and variety regarding the available artifacts. RUP is a Rational Software product based on the Unified Software Development Process (Jacobson et al., 1999).

We have developed a prototype tool named SharpTrace (Anaya et al., 2002), which implements the UML profile for traceability defined in (Letelier, 2002). It has been developed as an add-in, which extends Rational Rose so that it is able to integrate UML and no-UML specifications by defining new artifacts and link types of traceability. In addition, SharpTrace includes a traceability configuration process that provides guidelines for this adaptation. Thus, the extended CASE is able to work with all kinds of artifacts in the same context, all of them are saved in the same Rational Rose repository (.mdl file).

We have specified an example based on an information system for a virtual shop. According to the traceability configuration process, the first step is the definition of the artifact types that will be used in the project. For each type of artifacts the user of the tool must check whether the UML basic metamodel has a corresponding construct that represents that concept. If it does not exist, the user of the tool checks if there is a stereotype in the UML profile that represents that concept. If both of the previous checks fail, the UML profile should be extended with a new stereotype. In order to do this, we find the stereotype in the UML profile that has the closest meaning to the concept to be represented. After that, we define a subclass from this stereotype.

Next table shows the list of artifact types used in the example and the stereotypes from which they are defined. When the base stereotype cell is empty, the RUP artifact can be directly represented by the corresponding metaclass in the UML metamodel.

In order to improve requirements management, each artifact type has a set of attributes. For instance, a software feature can have attributes such as: *state* (proposed, approved or incorporated), *benefit* (critical, important or useful), *estimated effort*, *risk* and *stability* (for

Table 1. List of Artifacts

List of RUP artifacts	Base Stereotype
User Need	*<<No UML-UseCase requirement>>*
Software Feature	*<<No UML-UseCase requirement>>*
Actor	
Use Case	
Step	*<<traceable specification>>*
Precondition	*<<No UML-UseCase requirement>>*
Non-Functional Requirement	*<<traceable specification>>*
Change Request	*<<textual requirement>>*
Class	
Table	*<<traceable specification>>*
Component	
Node	
Test Case	*<<test specification>>*
Justification	*<<rationale specification>>*
Assumption	*<<rationale specification>>*

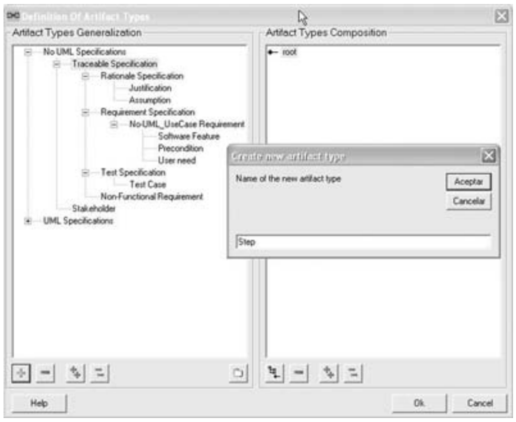


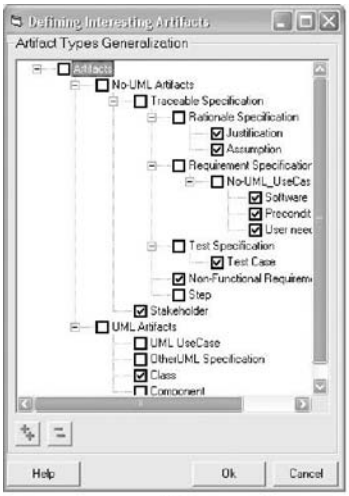
Figure 5. Modifying the UML Profile for Traceability.

these last attributes, the usual values are: high, medium or low). SharpTrace allows easily adapting the profile for traceability (see Figure 5).

Afterwards, we select the artifacts types that we want to trace. These are a subset of those defined in task 1. Figure 6(a) shows how this task is done with SharpTrace.

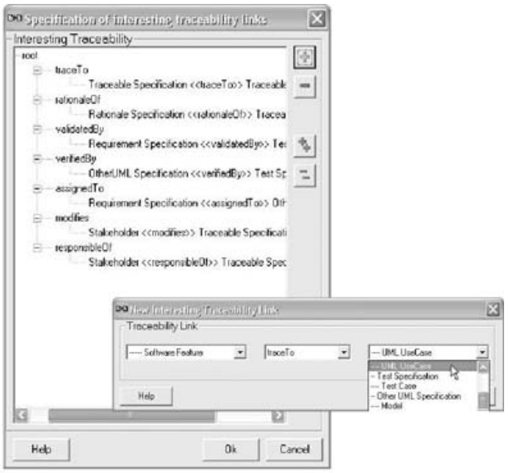
In the next step, we define the traceability link types. We could define what artifact types can be linked using a specific traceability link. This information establishes restrictions about what links can be applied to artifacts. SharpTrace provides by default the traceability link types shown in Figure 6 (b) (those specified in the core metamodel).

SharpTrace allows defining new traceability link types. These new links are subclasses from those provided by the UML profile for traceability. Thus, a new link type inherits constraints from its parent. Additionally, the inherited constraints can be restricted even more.



```
Defining Interesting Artifacts
Artifact Types Generalization
☐ Artifacts
    ☐ No-UML Artifacts
        ☐ Traceable Specification
            ☐ Rationale Specification
                ☑ Justification
                ☑ Assumption
            ☐ Requirement Specificatior
                ☐ No-UML_UseCas
                    ☑ Software
                    ☑ Precondit
                    ☑ User neec
            ☐ Test Specification
                ☑ Test Case
            ☑ Non-Functional Requirem
            ☐ Step
        ☑ Stakeholder
    ☐ UML Artifacts
        ☐ UML UseCase
        ☐ OtherUML Specification
        ☑ Class
        ☐ Component
Help    Ok    Cancel
```

(a)



```
Specification of interesting traceability links
Interesting Traceability
root
    traceTo
        Traceable Specification <<traceTo>> Traceabl
    rationaleOf
        Rationale Specification <<rationaleOf>> Tracea
    validatedBy
        Requirement Specification <<validatedBy>> Te
    verifiedBy
        OtherUML Specification <<verifiedBy>> Test Sp
    assignedTo
        Requirement Specification <<assignedTo>> Oth
    modifies
        Stakeholder <<modifies>> Traceable Specificati
    responsibleOf
        Stakeholder <<responsibleOf>> Traceable Spec
Help    Ok    Cancel

New Interesting Traceability Link
Traceability Link
---- Software Feature    traceTo    -- UML UseCase
                                    -- UML UseCase
                                    - Test Specification
                                    -- Test Case
                                    - Other UML Specification
                                    --- Model
Help
```

(b)

Figure 6. (a) Relevant Artifact Types for Traceability (Virtual Shop) **(b)** Establishing interesting Trace Link Types (Virtual Shop).

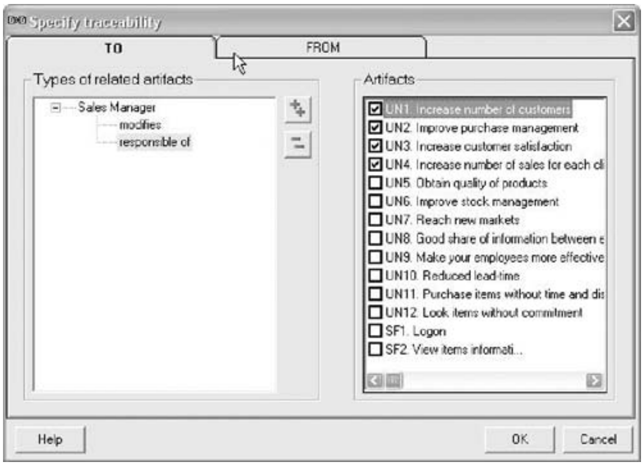
The existence of various link types for traceability allows richer analysis of traceability information.

When the traceability configuration process is concluded, SharpTrace enables the specification of traceability links only for traceable artifacts (those defined in traceability configuration). When selecting a traceable artifact SharpTrace supplies a list of link types according to the configuration (see Figure 6 (b)). When selecting a link type in this list only the reachable artifacts are displayed.

Traceability links have direction. Therefore, the traceability links of an artifact can be classified as incoming or outgoing links ("from" or "to" links, respectively). The incoming links are those coming from a certain artifact to the selected one. The outgoing links are those established from the selected artifact to other artifacts. Taking into account the list of traceable artifacts, the user marks the artifacts among he wants to record traceability links. Task 4 provides us with guidelines to the specification of these traceability links. In the example, the "sales manager" proposes the user needs 1, 2, 3 and 4, therefore he is responsible for them (see Figure 7).

6. RELATED WORKS

Ramesh and Jarke (2001) offer a wide vision about the information needed in requirements traceability. Their study is based on the analysis of industrial software development projects. They identify two segments of traceability users and suggest two corresponding traceability metamodels (one is a simplification of the other). The most complete metamodel has 31 types of entities (metaclasses in the metamodel) and about 50 types of links. In addition to the complexity associated to the diversity of types of entities and links, a



Sales Manager
- modifies
- responsible of

Artifacts

- [✓] UN1. Increase number of customers
- [✓] UN2. Improve purchase management
- [✓] UN3. Increase customer satisfaction
- [✓] UN4. Increase number of sales for each cl
- [ ] UN5. Obtain quality of products
- [ ] UN6. Improve stock management
- [ ] UN7. Reach new markets
- [ ] UN8. Good share of information between e
- [ ] UN9. Make your employees more effective
- [ ] UN10. Reduced lead-time
- [ ] UN11. Purchase items without time and dis
- [ ] UN12. Look items without commitment
- [ ] SF1. Logon
- [ ] SF2. View items informati...

Figure 7. Outgoing “responsible of” links from Sales Manager.

precise definition for those elements is not provided, which makes the application of the metamodel a difficult task. Furthermore, all analysis and design specifications are only represented by *System_Subsystem_Component*, that is, there is no more granularity or connection with any specific modeling notation. Eventually, the only suggested mechanism to configure the metamodel according to the project needs is to cut or to add parts of the metamodel

Toranzo and Castro (1999) present a traceability metamodel defined by multiple views, each of them associated to a certain kind of user for the traceability information (Project Manager, Requirement Engineer or Software Engineer). Currently there is no configuration mechanism to tailor the traceability to the project needs. Furthermore, the granularity level of the artifacts is too thick, working with documents and diagrams.

Spence and Probasco (1998) present several strategies for traceability when a Use Case driven process is used (as the case of RUP). Each strategy is described with a simple traceability metamodel, establishing the types of artifacts and links. All the strategies suggested only consider links between artifacts to requirements (User Needs, Software Features, Use Cases, etc.). The connection with artifacts for modeling and test is left implicit according to what a Use Case driven process establishes (Use Case analysis or design realization, functional test for each use case, etc.). Furthermore, the only type of link they use is our equivalent *traceTo*.

Similarly, Leite et al., (1995; 1997) provide a framework for elicitation and organization of requirements expressed in natural language. They establish traceability links between requirements but they do not include traceability to other subsequent artifacts.

On the other hand, requirements management tools offer a satisfactory treatment for textual specifications but they have inconveniences when integrating those specifications with others not expressed textually. This integration is based on import mechanisms connecting with a CASE tool. Usually, in this approach the names of the modeling elements are handled in the context of the requirements management tool. A tool frequently cited is

TOOR (Traceability of Object-Oriented Requirements), presented by Pinheiro and Goguen in (Pinheiro et al., 1996), it is based on FOOPS, a formal object-oriented language. Curiously, there are not any works about TOOR after this paper. However, its formal approach and the functionality described remain interesting.

RequisitePro is a Rational Software tool for requirements management. RequisitePro provides a set of templates adapted to different kind of projects. In RequisitePro all the artifacts are called requirements (even those defined in design and implementation phases), which is a bit confusing. The provided templates can be adapted, but the supplied solution lacks of guidelines. RequisitePro only provides one type of traceability link, the traditional "trace-to". The tool has a poor integration with Rational Rose, due to it only allows traceability links between requirements defined in RequisitePro and Use Cases defined in Rational Rose. Consequently, it is not possible to establish links with other types of analysis or design artifacts.

Another well known requirements management tool is Telelogic DOORS. This tool can be connected with most popular CASE tools, using similar import mechanisms to Rational RequisitePro ones, but providing more functionality and facilities to change from the DOOR context to the CASE tool context. Anyway, the user must work with two separated environments, and depending on whether he/she wants to do requirement management or software modeling, he/she must switch the environment.

All the mentioned tools have inconveniences as far as the configuration of traceability to the project needs is a concern. They are not oriented to a specific software process and although some of them allow defining types of requirements, they do not offer a framework for configuring requirement traceability. Eventually, all the definitions and interpretation about the traceability information is left to the user of the tool.

7. CONCLUSIONS

Requirements traceability is the key to achieve a successful requirements management process. However, there is no consensus about the more suitable strategies to perform effective requirements traceability. Thus in practice, requirements traceability presents different levels of satisfaction and acceptance in software development projects. Consequently, support provided by tools is not the most appropriate.

In this work we have presented a traceability metamodel integrating textual specifications (for requirements, rationale and tests) with standard UML specifications, using the UML context itself. Thus, from the point of view of requirements traceability, our metamodel offers a core framework for types of entities and types of traceability links that can be customized to a particular project using the extension mechanisms provided by UML. The traceability metamodel has been translated to a UML profile which allows an easier application in a CASE tool supporting UML.

Additionally, we have presented a configuration process for requirements traceability based on our UML profile for requirements traceability. Our approach including the metamodel, the corresponding UML profile and the configuration process only have the assumption of using a UML-based process, but it is independent of any particular process. However, to illustrate our approach we have presented an example using RUP as a development process. We have developed a tool called SharpTrace that implements the UML profile for traceability, extending Rational Rose. SharpTrace provides Rational Rose with mechanisms to work with non-UML artifact types. It is also possible to define and follow the artifacts all over their life with any level of granularity, thus a more precise analysis of the traceability information can be carried out.

We are working on the implementation of mechanisms to define criteria for automatic derivation of traceability links. We are also validating our approach with some projects. In addition, we want to extend SharpTrace with mechanisms to analyze the traceability information.

ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish CICYT project DYNAMICA TIC2003-07776-C02-02.

REFERENCES

Anaya, V., and Letelier, P., 2002, SmarTTrace: A tool for requirements traceability in UML-based projects, in: *Proceedings of the V Workshop on Requirements Engineering*, O. Pastor and J. Sánchez Díaz, eds., pp. 210–224.

Corriveau, J.-P., 1996, Traceability Process for large OO Projects, *IEEE Computer* **29**(9):63.

Dömges, R., and Pohl, K., 1998, Adapting traceability environments to project-specific needs, *Communications of ACM* **41**(21):54.

Gotel, O., and Finkelstein, A., 1997, Extended requirements traceability: results of an industrial case study, in: *Proceedings of 3rd International Symposium on Requirements Engineering*, IEEE Computer Society Press, Los Alamitos, pp. 169–178.

Jacobson, I., Booch, G., and Rumbaugh, J., 1999, *The Unified Software Development Process*, Addison-Wesley, Boston.

Jarke, M., 1998, Requirements tracing. *Communications of the ACM* **41**(12):32.

Leite, J. C., and Oliveira, A., 1995, A client oriented requirements baseline, in: *Proceedings of the Second IEEE International Symposium on Requirements Engineering, 1995*, IEEE Computer Society Press, Los Alamitos, pp. 108–115.

Leite, J. C., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G., and Oliveros, A., 1997, Enhancing a requirements baseline with scenarios, *Requirements Engineering* **2**(4):184.

Letelier, P., 2002, A Framework for requirements traceability in UML-based Projects, 1st International Workshop on Traceability in Emerging Forms of Software Engineering; http://ase.cs.ucl.ac.uk/.

OMG, 2002, Unified Modeling Language Specification. UML 1.4 with Action Semantics, (January, 2002); http://www.omg.org.

Pinheiro, F., and Goguen, J., 1996, An object-oriented tool for tracing requirements, *IEEE Software* **13**(2):52.

Pohl, K., 1996, Enabling requirements pre-traceability, in: *Proceedings of the 2nd International Conference on Requirements Engineering*, IEEE Computer Society Press, Los Alamitos, pp. 76–44.

Ramesh, B., 1998, Factors influencing requirements traceability practice, *Communication of the ACM* **41**(12):37.

Ramesh, B., and Jarke, M., 2001, Towards reference models for requirements traceability, *IEEE Transactions on Software Engineering* **27**(1):58.

Rational RequisitePro, 2002; http://www-306.ibm.com/software/awdtools/reqpro/.

Spence, I., and Probasco, L., 1998, Traceability studies for managing requirements with use cases, Rational Software White Paper No. 22701; www-106.ibm.com/developerworks/rational/library/content/03July/getstart/RP/ReqPro_PM.pdf.

Telelogic DOORS, 2002; http://www.telelogic.com/products/doorsers/doors/index.cfm.

Toranzo, M., and Castro, J., 1999, A comprehensive traceability model to support the design of interactive systems, in: *Lecture Notes in Computer Science 1743*, A. M. D. Moreira and S. Demeyer, eds., Springer-Verlag, Heidelberg, pp. 283–284.

AGENT-ORIENTED INFORMATION SYSTEMS DEVELOPMENT USING OPEN AND THE AGENT FACTORY

B. Henderson-Sellers, Q.-N. N. Tran, J. Debenham, and C. Gonzalez-Perez*

1. INTRODUCTION

Information systems development (ISD) requires the underpinning of a high quality methodology (which includes elements to describe both the process of development and the work products which are the consumables used and produced by the process). However, each ISD project is different and the best-fit methodology is also consequently different. This means that a one-size-fits-all methodology will only rarely give ideal results (Cockburn, 2000), when the tenets of the methodology designer coincidentally coincide with those of the particular project.

Rather than seeking an all-encompassing methodology, we advocate here the use of method engineering (Brinkkemper, 1996) or, preferably, situational method engineering or SME (Ter Hofstede and Verhoef, 1997). SME involves defining a repository of method fragments together with techniques for assembling these method fragments or method chunks (Rolland and Prakash, 1996) into site-specific methodologies specifically tuned to the situation of the project at hand (Brinkkemper, 1996) i.e. one that meets the requirements of a particular project. Thus, selection of method fragments is individualized and "tailored" to the specific requirements of the organization and project using construction guidelines supplied with the repository (Brinkkemper et al., 1998; Ralyté and Rolland, 2001). Many papers describing situational method engineering tend to focus on the process engineering element rather than the combination of process and product viz. the "methodology". Since "process" is therefore a subset of "methodology", when discussing only the "process" component of a methodology, the term *process engineering* is often substituted for the broader term "method engineering".

For commercial adoption, the first choice is a widely used methodology framework with an existing extensive catalogue of method fragments. OPEN (Object-oriented Process,

* University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia.

Environment and Notation) is both described in an extensive set of books (e.g. Graham et al., 1997; Firesmith and Henderson-Sellers, 2002) and used in industrial applications as well as having (at least embryonic) tool support (Nguyen and Henderson-Sellers, 2003). While OPEN was originally developed for ISD in an object-oriented context, in recent years method fragments have begun to be developed for application to agent-oriented ISD (Debenham and Henderson-Sellers, 2003). In our current project, prior to a formalized extension of OPEN to fully support agent-oriented (AO) information systems development, we are analyzing each stand-alone AO methodology. For each such methodology, we seek to isolate method fragments identifiable in the methodology and then compare these with existing method fragments in OPEN/Agent OPEN. When there is no match, we suggest the development and incorporation in the repository of a method fragment to support the concept detailed in the AO methodology under analysis. Following successful analysis of Tropos (Henderson-Sellers et al., 2003, 2004b), MASE (Tran et al., 2004), Gaia (Henderson-Sellers et al., 2004a), Prometheus (Henderson-Sellers et al., 2004c) and Cassiopeia (Henderson-Sellers et al., 2004d), in this paper we look at the details of the Agent Factory (Collier et al., 2004).

In the following sections, we introduce situational method engineering (Section 2) as an effective approach for constructing an organizational method that may be tailored or customized for individual projects in the context of the OPEN process (Section 3). In Section 4 we outline briefly the Agent Factory approach and then analyze in detail in the following section (Section 5) in order to identify existing OPEN support for the Agent Factory set of concepts and to derive any necessary new method fragments.

2. BRIEF OVERVIEW OF SITUATIONAL METHOD ENGINEERING

Over recent years, there has been significant research into the field of method engineering and situational method engineering (Brinkkemper, 1996; Ter Hofstede and Verhoef, 1997; Rupprecht et al., 2000; Ralyté and Rolland, 2001). With SME, a method is constructed based on a methodological requirements statement made by the organization that requires methodological support for their software development. This requirements statement helps the method engineer to identify appropriate method fragments stored in the repository. Such an SME approach offers advantages to the use of a single “off-the-shelf” methodology since the method fragments in an SME repository not only support the methodology “as is” but also offer additional support that allow the methodology to be extended beyond its original intentions and scope. Alternatively, rather than extending an existing methodology, the more usual application of SME is to construct a methodology *ab initio*, specifically for the target situation.

In terms of SME research, the work by Brinkkemper et al. (1998) and Klooster et al. (1997) is also worth mentioning. These authors focus on the techniques for assembling available method fragments into situational methods/processes – an approach where a sound mathematical basis for process construction is described formally in a number of rules. The use of these rules allows the creation of a knowledge base, where knowledge can be stored and disseminated.

Ideally, the elements in an SME repository should be compliant with (in fact generated from) a set of concepts described by a metamodel (Henderson-Sellers, 2003). One such example is the metamodel+repository-based OPEN process framework (Firesmith and Henderson-Sellers, 2002). The OPEN metamodel contains a number of conceptual entities modelled by object-oriented "classes", typically described using the UML notational concepts. Those concepts most relevant to the process aspects of a methodology (as to be discussed here) are (i) Task, (ii) Technique and (iii) Work Product. Each of these metaclasses can be instantiated to create numerous instances of Task, Technique and Work Product respectively, all of which are stored in the OPEN repository. It is the elements of this repository that form the focus of our analysis here, in which we analyze existing process elements in the OPEN repository for their potential support for the Agent Factory approach to ISD.

3. BRIEF OVERVIEW OF OPEN

Unlike other OO ISD processes, OPEN (Graham et al., 1997; Firesmith and Henderson-Sellers, 2002) is defined as a process framework encompassing a metamodel. It is thus highly compatible with the ideas of method engineering and process construction as described above. From this framework, OPEN-compliant processes can be instantiated to be used in actual organization and software projects. In other words, the process engineer has to configure OPEN, creating an instance of OPEN that is suitable for use on a specific project. Instantiating OPEN is one of the more difficult and time-consuming jobs in adopting OPEN, since the process engineer has to understand the methodology, the organization, the environment and the software project itself in order to select the appropriate components in the OPEN repository to use on the project. Traditionally, this process is carried out using predefined organizational requirements and the experience and knowledge of the process engineer although tool support is likely in the near future (Saeki, 2003).

An important part of OPEN is the repository of process components, which can be used in different software projects. This repository provides an almost complete set of components and can be extended to support changes in technology.

The OPEN metamodel defines five main classes of process components as shown in Figure 1. From these (and their subclasses) are generated a significant number of instances, together with a number of guidelines (Firesmith and Henderson-Sellers, 2002) for helping organizations to adopt OPEN as a standard for their information system development projects. These guidelines offer advice on the selection of specific components based on the notion of deontic matrices. A deontic matrix is a two dimensional matrix of values that represent the possible relationship between each pair of process components in OPEN. For example, the possibility value for using the OPEN Task: Evaluate quality to help fulfil the Activity: Verification and Validation (V&V) might be assessed as being "Recommended" (one of the five prescribed values). Tool support for completing these matrices is currently under development (Nguyen and Henderson-Sellers, 2003).

In addition, the idea that business culture should be adapted to fit a specific methodology is not good business sense, despite its prevalence in many of the marketed methodologies to date. When using IT and its dependent parts, such as methodologies, it is critical that they fit the business and not the other way around.

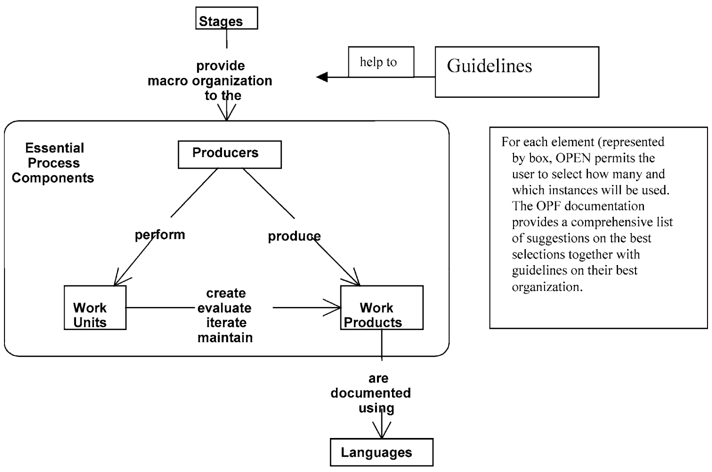

Stages
provide macro organization to the
help to
Guidelines
Essential Process Components
Producers
perform
produce
Work Units
create evaluate iterate maintain
Work Products
are documented using
Languages
For each element (represented by box, OPEN permits the user to select how many and which instances will be used. The OPF documentation provides a comprehensive list of suggestions on the best selections together with guidelines on their best organization.

Figure 1. The five major metaclasses of OPEN's metamodel (after Firesmith and Henderson-Sellers, 2002) © Addison-Wesley.

4. BRIEF OVERVIEW OF THE AGENT FACTORY

Agent Factory (Collier et al., 2003, 2004) is a four-layer framework for designing, implementing, and deploying multi-agent systems. It contains (i) an agent-oriented software engineering methodology, (ii) a development environment, (iii) a FIPA-compliant runtime environment and (iv) an agent programming language (AF-APL); with a stated preference for the BDI agent architecture according to the analysis of (Luck et al., 2004). Here, we only evaluate the methodology components.

By employing UML and Agent UML, the Agent Factory methodology provides a visual, industry-recognized notation for its models - regarded by its authors as a major advantage over other approaches, such as Gaia (Wooldridge et al., 2000) and Tropos (Bresciani et al., 2004), which have non-standard (i.e. non-UML compliant) notations. These models are capable of promoting design reuse (via the central notion of *role*), and being directly implemented by automated code generation (Collier et al., 2004).

The Agent Factory describes three phases and an overall lifecycle approach. These, and their relationship to OPEN stages, are as follows:

Cycle: Agent Factory is iterative across the first three tasks of the development process (i.e. developing System Behaviour Model, Activity Model, and Interaction Model), but is sequential regarding the remaining steps.

Support from OPEN: A combination of "Iterative, incremental, parallel life cycle" (for iterative refinement) and "Waterfall life cycle" (for sequential development).

Life cycle: Agent Factory offers three phases, covering design, implementation and deployment. However the first three tasks of the Design phase (i.e. developing System Behaviour Model, Activity Model, and Interaction Model) are more appropriate to analysis rather than design activities. In other words, it appears that Agent Factory In fact more realistically can be said to cover four phases: analysis, design, implementation and deployment.

Support from OPEN: the three Agent Factory phases correspond to the Initiation, Construction, and Delivery phases of OPEN.

5. TASKS, TECHNIQUES AND WORK PRODUCTS IN THE AGENT FACTORY AND THEIR SUPPORT IN OPEN

In this section, we identify process component descriptions within the Agent Factory documentation, captured here as instances of elements in the OPEN metamodel. In particular, we seek Tasks, Techniques and Work Products. For each of these three elements, we analyze the Agent Factory descriptions and then recast them into the OPEN software engineering approach. This leads us to propose three new subtasks for addition to the OPEN repository as we extend this repository to encompass not only an object-oriented approach to software development but, increasingly, an agent-oriented approach. These new process components in the OPEN repository add to those already proposed to support agent-orientation in e.g. (Henderson-Sellers and Debenham, 2003; Henderson-Sellers et al., 2004b; Tran et al., 2004).

5.1. Tasks in the Agent Factory and Their Support in OPEN

For each Agent Factory task identified, we first describe it and then create a parallel OPEN method fragment.

5.1.1. Developing System Behaviour Model

Description: This task involves identifying the *key system behaviors* (i.e. *sets* of activities and/or interactions that occur during the operation of the system), and the *roles* that the agents will play while engaged in these behaviours.

Support from OPEN: The identification of system behaviours is supported by standard Requirements Engineering tasks in OPEN, particularly task “Use case modeling” (because Agent Factory captures system behaviors through use cases). Roles can be identified via the task “Model agent’s roles” in Agent OPEN (Debenham and Henderson-Sellers, 2003).

5.1.2. Developing Interaction Model

Description: This task investigates the system behaviours that involve interactions between two or more roles, and identifies all the potential “interaction scenarios” that may occur in each of these behaviours.

Support from OPEN: Agent OPEN offers tasks “Construct agent interaction protocol” and “Construct agent communication protocol” that can be extended to model interactions at a higher-level of abstraction (i.e. inter-role interactions).

5.1.3. Developing Activity Model

Description: This task investigates each system behaviour and identifies all potential “activity scenarios” that may occur within each behaviour. An activity scenario describes what activities need to be performed by the participant role(s) in order to realize a particular system behaviour.

Support from OPEN: The identification and modelling of activities within roles are addressed by Agent OPEN task "Model agent's roles", particularly by its sub-task "Model roles' responsibilities" (Henderson-Sellers et al., 2004a).

5.1.4. Developing Protocol Model

Description: This task formalizes the interaction scenarios identified in the Interaction Model with interaction protocols. Each protocol elaborately defines the inter-role interactions in a particular system behaviour, encapsulating all possible variations.

Support from OPEN: This task can be supported by Agent OPEN's tasks "Determine agent interaction protocol" and "Determine agent communication protocol", although at the "role" level of abstraction rather than at the "agent" level.

5.1.5. Developing Agent Model

Description: This task identifies agent classes from roles, and models agent classes in terms of their activities, roles, and roles' protocols.

Support from OPEN: Agent OPEN task "Construct the Agent Model" (Tran et al., 2004) directly supports this activity.

5.1.6. Defining Application-Specific Ontologies

Description: This task specifies the conceptualization of the target application domain via a (set of) ontologies. These ontologies are needed to form the beliefs of individual agents and the information exchanged between interacting agents.

Support from OPEN: The issue of ontology specification to be used in system operation is currently not addressed in OPEN. A new task is thus desirable

TASK NAME: Define ontologies
Focus: Domain conceptualization
Typical supportive techniques: Domain analysis
Explanation: A conceptualization of the target application domain needs to be specified. This conceptualization should contain all the concepts, entities, and relationships that exist in the domain, and which are relevant to the needs of agents in the system (e.g. communication and internal processing needs).

5.1.7. Building Agent Components

Description: This task generates agent components that are required by the final system, particularly perceptor and actuator units.

Support from OPEN: Agent OPEN recently introduced a task "Design agent internal structure" (Tran et al., 2004) that addresses the design of agent internal modules. We suggest adding two new sub-tasks to explicitly address the specification of perceptor and actuator modules. These are subtasks to the existing Task: Design agent internal structure.

SUBTASK NAME: Define perceptor module
Typical supportive techniques: Environmental evaluation

Explanation: Define for each agent its required sensing abilities, and mechanisms needed to convert raw data/percepts to beliefs. To promote reusability, sensing abilities and mechanisms can be packaged into perceptor components that are later associated to agents.

SUBTASK NAME: Define actuator module
Typical supportive techniques: Environmental evaluation
Explanation: Define for each agent the primitive actions that can be directly executed by the agent on the environment. To promote reusability, sets of primitive actions can be packaged into actuator components that are later bound to agents.

5.1.8. Building Platform Services

Description: This task identifies and constructs services that are required to be deployed on the agent platform, e.g. message transport, migration, persistence services.

Support from OPEN: The OPEN task "Create a system architecture" can be extended to include a new sub-task "Determine MAS infrastructure facilities".

SUBTASK NAME: Determine MAS infrastructure facilities
Typical supportive techniques: Environmental evaluation
Explanation: Facilities required to support the operation of MAS as a whole and of agents should be identified, as well as how they are managed (e.g. by agents).

5.1.9. Implementing Agent Classes

Description: This task generates AF-APL code to implement agents.

Support from OPEN: The OPEN task "Code" can be extended to cover the coding of agent classes

5.1.10. Testing

Description: This task performs "protocol tests" on agent interaction protocols, and "behaviour tests" on agent behaviours, in order to evaluate their correctness.

Support from OPEN: A range of OPEN Testing tasks, including "Design test suite", "Execute tests", and "Report on test results", can be extended to cater for agent-oriented testing.

5.1.11. Deployment

Description: This task involves configuring the agent development platform and deploying the application.

Support from OPEN: OPEN's set of Deployment tasks, although initially intended to support the deployment of OO applications, can be equally applicable to AO application deployment.

5.2. Agent Factory Techniques and Their Existing Support in OPEN

For each Agent Factory technique identified, we first describe it and then create a parallel OPEN method fragment.

5.2.1. For Developing System Behaviour Model

Description: Regarding the identification of system behaviours, the developer should investigate both activity-oriented behaviours (i.e. those associated with a single role), and interaction-oriented behaviours (i.e. those associated with two or more roles). Regarding the identification of roles, Agent Factory offers no techniques.

Support from OPEN: Various standard OPEN techniques can be useful for system behaviours identification, including “Scenario development”, “Activity grid construction”, and “Service identification”. Role identification can be assisted by the conventional OPEN technique “Role modeling” (although this technique is still weak in guidance for role elicitation), and the newly-added technique “Environmental evaluation” in Agent OPEN (Henderson-Sellers and Debenham, 2003).

5.2.2. For Developing Interaction Model

Description: For each interaction-oriented system behaviour in the System Behaviour Model, the developer should define a number of “interaction scenarios”, each specifying a potential set of interactions that may incur within the behaviour. Each scenario should describe the types of messages sent among roles, and the order in which they are sent. There typically exist one “standard” scenario and multiple alternate scenarios for each interaction-oriented behaviour.

Support from OPEN: Various conventional OO techniques can be useful for the identification and specification of interaction scenarios, including “Scenario development”, “Collaboration analysis”, and “Interaction modeling”.

5.2.3. For Developing Activity Model

Description: The developer should specify at least one “activity scenario” for each activity-oriented system behaviour, and zero or more “activity scenarios” for each interaction-oriented behaviour. In each scenario, the activities to be performed by each participant role should be specified. Multiple activity scenarios exist when there are different ways to fulfil a behavior.

Support from OPEN: The identification and specification of activity scenarios can be supported by conventional OPEN techniques “Scenario development”, “Responsibility identification”, and “State modeling”.

5.2.4. For Developing Protocol Model

Description: Each interaction-oriented system behaviour typically requires one protocol to be specified. If multiple interaction scenarios have been identified for the behaviour in the Interaction Model, they can be integrated into a single protocol. The developer may make use of existing protocol templates, and/or formulate new templates by identifying and extracting common interactions within the defined protocols.

Support from OPEN: Conventional OPEN technique “Interaction modeling” and Agent OPEN techniques “Contract net”, “Market mechanisms”, and “FIPA-KIF compliant language” can be applied to specify the protocols and exchanged messages.

5.2.5. For Developing Agent Model

Description: Agent classes can be derived from roles via a many-to-many correspondence, i.e. each role can be mapped onto many agent classes, while each agent class can be mapped to multiple roles. Agents are associated with activities, which are determined by examining the potential "activity scenarios" for each system behaviour in the Activity Model, and selecting a scenario the agent should employ when realising a particular behavior.

Support from OPEN: Regarding the identification of agents, OPEN technique "Intelligent agent identification" can be applied (although it still requires enhancement). The determination of agents' activities can be assisted by various Agent OPEN techniques "Commitment management", "Activity scheduling", "Task selection by agents", "Deliberative reasoning", and "Reactive reasoning".

5.2.6. For Defining Application-Specific Ontologies

Description: An ontology can be formed by mapping logical predicates to domain relations, for example, predicate *position(?lat, ?long)* can be used to represent a user's position in latitude and longitude. No techniques are provided on how to identify domain concepts/relations

Support from OPEN: Technique "Domain analysis" of OPEN can be used to support domain concepts identification.

5.2.7. For Building Agent Components

Description: Perceptor and actuator units can be identified by reviewing activities specified in the Activity Model. Perceptor units can be implemented as Java classes that encapsulate sensing abilities and convert raw data into beliefs. Actuator units can be realized as Java classes that contain actions directly executable by the agents.

Support from OPEN: Agent OPEN technique "Environmental evaluation" can be applied to determine how, and what sensing/affecting abilities are required for, the agents to interact with the environment.

5.2.8. For Building Platform Services

Description: FIPA-standards can be investigated to identify and build necessary platform services.

Support from OPEN: No techniques are found from OPEN that explicitly support the identification and design of infrastructure facilities. This is a topic for future research.

5.2.9. For Implementing Agent Classes

Description: Each agent class is implemented as a mental entity with beliefs (i.e. knowledge about the current state of itself and its environment), commitments (i.e. current and future activities that the agent has decided to perform), and commitment rules (i.e. mappings between beliefs and commitments).

Support from OPEN: Various Agent OPEN techniques can be used to facilitate the transformation of agent class design to implementation, including "3-layer BDI model", "deliberative reasoning: plans", "reactive reasoning: ECA rules", "Commitment management", "activity scheduling", "task selection", and "belief revision".

5.2.10. For Testing

Description: No techniques are provided for the formulation and execution of protocol and behaviour tests.

Support from OPEN: Conventional testing techniques "Unit testing" and "Integration testing" of OPEN can be extended to cater for agent behaviors testing and agent interactions testing respectively.

5.2.11. For Deployment

Description: Platform configuration file(s) need to be generated, specifying which agents should initiated, which resources should be connected/created, and which facilities need to be used.

Support from OPEN: This is a topic for future research.

5.3. Work Products of the Agent Factory

The Agent Factory specifically aims, where possible, to use pre-existing design notations (Collier et al., 2004). Thus, Agent Factory adapts UML and Agent UML diagrams for its work products. The adaptations are outlined as follows:

5.3.1. System Behaviour Model

The system behaviour model is documented by using UML Use Case Diagrams where actors represent the roles to be played by agents (denoted as «role» stereotyped entities), and use cases represent behaviours associated with roles (denoted as "role-use-case" stereotyped entities).

5.3.2. Interaction Model

An interaction model uses UML Collaboration Diagrams to model "interaction scenarios". Interacting objects represent roles (denoted with "role" stereotype), and message types are FIPA-ACL performatives (denoted with "fipa-acl" stereotype).

5.3.3. Activity Model

An activity model in the Agent Factory is depicted with regular UML Activity Diagrams to model "activity scenarios", with each swimlane representing the processing of a role involved in the scenario.

5.3.4. Protocol Model

The Agent Factory protocol model is depicted with an Agent UML Sequence Diagram, one for each protocol (Figure 2). There may be secondary sequence diagrams which model protocol templates.

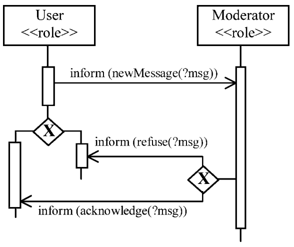


Figure 2. Example protocol model (adapted from Collier et al., 2004).

5.3.5. Agent Model

An agent model in Agent Factory is depicted using a UML Class Diagram that shows all agent classes in the system and their associated roles. Each role class is characterized by its associated protocols, while each agent class is characterized by a list of protocols (not those specified in roles) and activities.

6. SUMMARY AND ACKNOWLEDGEMENTS

A method engineering-based approach is, by its nature, advantageous to a single methodology "by itself", since the ME-based approach encompasses the specific methodology *plus* an arbitrarily wide range of additional method fragments that can be combined with the original value, thus adding value to it.

As part of an extensive research programme to combine the benefits of method engineering and to extend an existing object-oriented framework (OPEN) to create a highly supportive methodological environment for the construction of agent-oriented information systems, we have analysed here contributions from the Agent Factory AO methodology. We have identified three new subtasks for pre-existing Tasks only. All other aspects of Agent Factory can be satisfactorily simulated using method engineering with OPEN. This means that a process engineer or project manager wishing to use the style of development advocated by the Agent Factory need only take the OPEN repository, as enhanced here, and select from it the appropriate method fragments from which to "method engineer" this particular agent-oriented approach to information systems development.

We wish to acknowledge financial support from the University of Technology, Sydney under their Research Excellence Grants Scheme. This is Contribution number 04/21 of the Centre for Object Technology Applications and Research.

REFERENCES

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopolous, J., and Perini, A., 2004, Tropos: an agent-oriented software development methodology, *Autonomous Agents and Multi-Agent Systems* **8**(3):203–236.

Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.* **38**(4):275–280.

Brinkkemper, S., Saeki, M., and Harmsen, F., 1998, Assembly techniques for method engineering, *Procs. CAISE 1998*, LNCS1413, B. Pernici and C. Thanos, eds., Springer Verlag, pp. 381–400.

Cockburn, A. S., 2000, Selecting a project's methodology, *IEEE Software* **17**(4):64–71.

Collier, R., et al., 2003, Beyond prototyping in the factory of agents, in: *Multi-Agent Systems and Applications III,* LNCS 2691, V. Marik, J. Muller, and M. Pechoucek, eds., Springer-Verlag, New York, pp. 383–393.

Collier, R., O'Hare, G., and Rooney, C., 2004, A UML-based software engineering methodology for Agent Factory, *Procs. SEKE 2004*, Knowledge Systems Institute, Skokie, IL, USA (in press).

Debenham, J., and Henderson-Sellers, B., 2003, Designing agent-based process systems – extending the OPEN Process Framework, Chapter VIII in: *Intelligent Agent Software Engineering*, V. Plekhanova, ed., Idea Group Publishing, Hershey, USA, pp. 160–190.

Firesmith, D. G., and Henderson-Sellers, B., 2002, *The OPEN Process Framework. AN Introduction*, Addison-Wesley, Harlow, Herts, UK.

Graham, I., Henderson-Sellers, B., and Younessi, H., 1997, *The OPEN Process Specification*, Addison-Wesley, Harlow, UK.

Henderson-Sellers, B., 2003, Method engineering for OO system development, *Comm. ACM* **46**(10):73–78.

Henderson-Sellers, B., and Debenham, J., 2003, Towards OPEN methodological support for agent-oriented systems development, in: *Procs. First International Conference on Agent-Based Technologies and Systems*, B. H. Far, S. Rochefort, and M. Moussavia, eds., University of Calgary, Canada, pp. 14–24.

Henderson-Sellers, B., Giorgini, P., and Bresciani, P., 2003, Evaluating the potential for integrating the OPEN and Tropos metamodels, in: *Procs. SERP '03,* B. Al-Ani, H. R. Arabnia, and Y. Mun, eds., CSREA Press, Las Vegas, USA, pp. 992–995.

Henderson-Sellers, B., Debenham, J., and Tran, Q. N. 2004a, Adding agent-oriented concepts derived from Gaia to Agent OPEN, in: *Advanced Information Systems Engineering. 16th International Conference, CAiSE 2004, Riga, Latvia, June 2004 Proceedings*, LNCS 3084, A. Persson and J. Stirna, eds., Springer-Verlag, Berlin, pp. 98–111.

Henderson-Sellers, B., Giorgini, P., and Bresciani, P., 2004b, Enhancing Agent OPEN with concepts used in the Tropos methodology, in: *Procs. ESAW'03 (Engineering Societies in the Agents World)*, LNAI Volume 3071, A. Omicini, P. Pettra, and J. Pitt, eds., Springer-Verlag, Berlin.

Henderson-Sellers, B., Debenham, J., and Tran, N., 2004c, Incorporating elements from the Prometheus agent-oriented methodology in the OPEN Process Framework, in: *Procs. AOIS@CAiSE*04*, Faculty of Computer Science and Information, Riga Technical University, Latvia, pp. 370–385.

Henderson-Sellers, B., Tran, Q.-N. N., and Debenham, J., 2004d, Method engineering, the OPEN Process Framework and Cassiopeia, in: *The Symposium on Professional Practice in AI*, E. Mercier-Laurent and J. Debenham, IFIP, pp. 263–272.

Klooster, M., Brinkkemper, S., Harmsen, F., and Wijers, G., 1997, Intranet facilitated knowledge management: A theory and tool for defining situational methods, in: *Procs. CAISE 1997*, LNCS1250, A. Olive and J. A. Pastor, eds., Springer Verlag, pp. 303–317.

Luck M., Ashri, R., and D'Inverno, M., 2004, *Agent-Based Software Development*, Artech House, Boston, pp. 208.

Nguyen, V. P., and Henderson-Sellers, B., 2003, OPENPC: a tool to automate aspects of method engineering, in: *Procs. ICSSEA 2003*, Paris, France, Volume 5, pp. 7.

Ralyté, J., and Rolland, C., 2001, An assembly process model for method engineering, in: *Advanced Information Systems Engineering*, LNCS2068, K. R. Dittrich, A. Geppert, and M. C. Norrie, eds., Springer, Berlin, pp. 267–283.

Rolland, C., and Prakash, N., 1996, A proposal for context-specific method engineering, in: *Procs. IFIP WG8.1 Conf. on Method Engineering*, S. Brinkkemper, K. Lyytinen, and R. Welke, eds., Chapman and Hall, pp. 191–208.

Rupprecht, C., Funffinger, M., Knublauch, H., and Rose, T., 2000, Capture and dissemination of experience about the construction of engineering processes, in: *Procs. 12th Conference on Advanced Information Systems Engineering (CAISE)*, LNCS 1789, B. Wangler and L. Bergman, eds., Springer-Verlag, Berlin, pp. 294–308.

Saeki, M., 2003, CAME: the first step to automated software engineering, Process Engineering for Object-Oriented and Component-Based Development, in: *Procs. OOPSLA 2003 Workshop*, C. Gonzalez-Perez, B. Henderson-Sellers, and D. Rawsthorne, eds., Centre for Object Technology Applications and Research, Sydney, Australia, pp. 7–18.

Ter Hofstede, A. H. M., and Verhoef, T. F., 1997, On the feasibility of situational method engineering, *Information Systems* **22**:401–422.

Tran, Q. N., Henderson-Sellers, and B., Debenham, J. 2004, Incorporating the elements of the MASE methodology into Agent OPEN, in: *Procs. ICEIS2004*, I. Seruca, J. Cordeiro, S. Hammoudi, and J. Filipe, eds., INSTICC Press, Volume 4, pp. 380–388.

Wooldridge, M., Jennings, N. R., and Kinny, D., 2000, The Gaia methodology for agent-oriented analysis and design, *Autonomous Agents and Multi-Agent Systems* **3**:285–312.

MOTIVATION AND JOB SATISFACTION AMONG INFORMATION SYSTEMS DEVELOPERS – PERSPECTIVES FROM FINLAND, NIGERIA AND ESTONIA: A PRELIMINARY STUDY

Princely Ifinedo*

1. INTRODUCTION

Information Systems (IS) have become one of the most important assets in organisations over the past decades. Adopting and using IS bestows a variety of advantages to organisations that are adept at it (Senker and Senker, 1992). Developing IS within organisations, unarguably serves the objectives of management. To that end, information systems development (ISD) professionals are pivotal in the implementations of such systems and their motivation and job satisfaction equally paramount. Some researchers have also commented on economic value of IS professionals to organisations (Niederman and Crosetto, 1996) Apparently, there is a growing body of literature on the issue of motivation and job satisfaction both in other fields (Locke, 1983; Herzberg et al., 1959; Scarpello and Campbell, 1983) and in the Information Systems domain (Mumford, 1972, Hackman and Oldham, 1978; Bartol and Martin, 1982; Couger, 1988, 1989; Eldon and Abraham, 1991;Goldstein and Rockart, 1984; Baroudi and Ginzberg, 1985; McMurtrey et al., 2002). Overall, this attention can be attributed to the increased interest in the human resources (HR) management of IS professionals (Niederman, et al., 1991, Champy, 1992) and the socio-technical systems approaches to job design (Bostrom and Heinen, 1977; Pasmore, 1988).

Past research on the motivation of IS personnel (Couger and Zawacki, 1980) and job satisfaction (Baroudi and Ginzberg, 1985; Goldstein and Rockart; Griesser, 1993) have been carried out using IS personnel within a single country; namely, the US, with the exception of a handful of cases that straddle national boundaries. Such few cases include the work of the following researchers (Couger, 1986; Couger et al., 1990; Bryan et al., 1995; Couger and Ishikawa, 1995). Consequently, this present study aims to fill this research gaps with its contribution to the discourse by using three new countries that have hitherto not

* Department of Computer Science and Information Systems, University of Jyväskylä, FIN - 40351, Jyväskylä, Finland, premifinl@cc.jyu.fi.

been investigated. Moreover, cross-cultural IS research in similar areas is becoming popular and relevant. For example, differences were found among systems designers' values across nations (Kumar and Bjørn-Andersen, 1990). Also, Shore and Venkatachalam (1994) indicate that dimensions of national cultures could impart systems analysis and design of information systems development (ISD) projects, to mention but a few. Admittedly, the study of ISD professional's motivation and job satisfaction is pertinent in light of the increasing number of organisations developing IS applications across national boundaries (Carmel, 1999). In this era of globalisation of resources, useful insights emerging through cross-cultural/national information systems research on the levels of motivation of ISD professionals would, in turn, present information as to how to best manage differences or similarities that may be uncovered in this area of human resource management of IS personnel. Also, no study in the IS domain, to our knowledge has studied the nature of motivation and job satisfaction of ISD professionals among the selected three countries despite the increasing relevance of IS/IT in each of the countries. For example, Finland is a force to reckon with in IS/IT (Lyytinen and Goodman, 1999); Estonia is making remarkable progress in the use of information and communication technologies (ICT) and Nigeria is embracing and using new ICT (IMG, 2003; Anakwe et al., 1999; Ifinedo, 2004). The justification for the choice of the three countries is discussed below.

That said, the objectives of the paper are twofold: First, to find out if similarities/differences exist in the motivation, job satisfaction and job dissatisfaction factors for ISD professionals in the three countries, in view of any computer subculture (explained below) that may exist. Second, to determine the relative importance of each job satisfaction/dissatisfaction factor for each country. Comparisons are then made among the three, and relevant HR implications for each country discussed. This article is organised as follows: The vignette looks at introduction of the paper. The next section covers background of the study. Then, the research hypotheses, methodology and analysis of results are discussed. The last sections of the paper deal with the discussions and conclusion.

2. BACKGROUND OF THE STUDY

Job satisfaction defines condition where an individual has a perception that work is not a necessity imposed upon him or her. It shows the degree to which an individual feels positively or negatively affected by his/her job. It is an emotional response to one's tasks, as well as the physical and social conditions of the workplace (Herzberg et al., 1959; Locke, 1983; Iaffaldano, and Muchinsky, 1985). The impact of motivation on jobs has been a recurring theme in the behavioural sciences since the beginning of modern HR management, and in the IS/IT domain studies as well (Bartol and Martin, 1982; Scarpello and Campbell, 1983). It is known that management tends to devote substantive finance to peopleware costs than to any other department within organisations (Griesser, 1993; Lucas and Turner, 1982). Likewise, Couger and Amoroso (1990) found that motivation is a major issue among chief information officers (CIOs) in organisations.

Clearly, motivation describes the intensity of a person's desire to engage in some activity (Dessler, 2001). The themes of motivation and job satisfaction are couched in organisational behaviour (OB); therein, differences between the terminologies are made. Herzberg et al.'s, (1959) highlighted in their Motivation-Hygiene Theory, the sorts of factors that

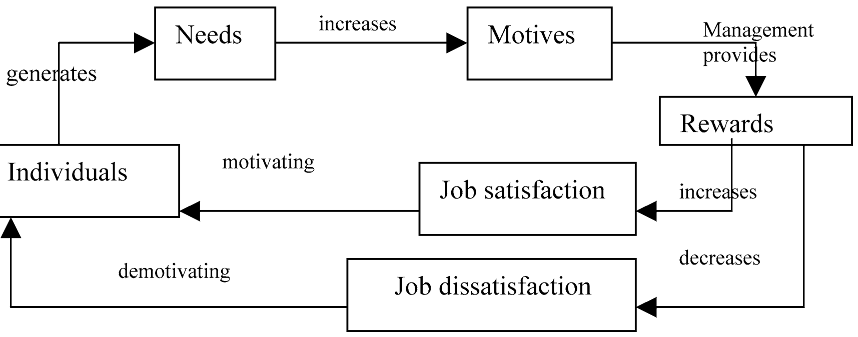


Figure 1. Interaction between needs, motivation, job satisfaction and the environment.

can result in job satisfaction, which they termed "motivators" and the ones that merely prevents job dissatisfaction (called hygiene factors). This paper took cognizant of such knowledge by formulating two different measures for the concepts; namely, "motivators" is seen as being distinct from "job dissatisfaction factors" in this paper. The interaction between the concepts is further explained by (Rosenfeld and Wilson, 1999, p.75), where a similar but different illustration to Figure 1 was utilised. Thus, the work of Herzberg and colleagues, as well those of Hackman and Oldham's (1978) who realistically extended Herzberg et al.'s, (1959) concepts to the IS domain informed the approach used herein. Similarly, the questionnaire used by another researcher (Hofstede, 2001) was incorporated in so far as it avail us the opportunity to investigate motivation and job satisfaction of ISD professionals from the three countries, on the "culture" front. The findings of that research will be published elsewhere. Essentially, job satisfaction is often measured and studied using several constructs. Cheney and Scarpello (1985) presents a variety of some of those constructs in the IS domain. Many factors affect job satisfaction and motivation (Lawler, 1994; Locke, 1983). And, the "sources of job satisfaction vary from one individual to the other" (Eldon and Abraham, 1991). Notwithstanding, we argued in the foregoing, that this particular effort will rest primarily on the study of motivation and job satisfaction from the work of (Herzberg et al.'s, 1959), though some items from other constructs such as the Job Characteristics Model (JCM) developed by Hackman and Oldham (1978) are also incorporated for participants to rank.

3. RESEARCH HYPOTHESES

As the world gets *globalised*, a variety of activities, concepts, and so on, arguably move towards uniformity (Levitt, 1983). Interestingly, researchers in IS have commented on the diffusion of ISD methods and attempts have been made at comparing ISD methods used regionally, in order to assess the degree to which uniformity holds (Pavlia and Hunter, 1996). Such research might have arisen as conformance to the same methods may be expected in a field that technical-oriented. This argument may be illuminated by the comment that "professionals worldwide belong to the computer subculture" (Carmel, 1999, p. 73). Camel cites an earlier publication by Constantine (2001) where the latter argues that “the computer subculture is stronger than national culture and that the programmer in Moscow is more similar to his American programming peer than to other Russians” (ibid). Carmel, further added that De Meyer and Mizushima (1989) made identical argument to Constantine’s by stating that in the area of R&D, the scientific culture dominates national culture.

Specifically, in extending the foregoing to the crux of this discourse - motivation and job satisfaction of ISD professionals – we believe that there are in fact differing perspectives to be garnered as one move from nation to the other. This is the basis upon which the thesis of this paper rests. In this regard, we formulated the hypothesis below to test our perception in regard of motivation and job satisfaction of ISD workers for the countries. Of note, we distinguish between motivators and dissatisfiers; hence, the layers of hypotheses presented. There are reasons why our viewpoint might be true. Our scan of relevant literature reveal that there are suggestions that workers from economically poorer nations tend to place more value on extrinsic job reward – fringe benefit, physical working conditions, pay and so forth – than counterparts from wealthier nations (Elizur et al., 1991; Hofstede, 1980, 2001). In the same vein, evidence indicate that intrinsic job reward (such as achievement, advancement, and so on) rate highly amongst workers from more advanced nations (Elizur et al., 1991; Hofstede, 1980; Harpaz, 1990; Robinson, 1983; Couger et al., 1990). Yet, others believe that no differences exist in how workers from developing and developed countries view variables or items of job (dis)satisfaction and motivation (see, for example; Adigun and Stephenson, 1992; Goldthorpe et al., 1968). This fact is corroborated by Rubin and Hernandez (1988) who cites the work of Mata et al. (1985), and writes that “Computer professionals, however, have traditionally been unimpressed and unaffected by such standard extrinsic motivators as pay, fringe benefit, job security, chances for promotion, and son on”.

As was stated, the objective of this study is to assess the similarities/differences that may exist for motivation/job satisfaction - amongst the ISD professionals - across nations. The relative rank ordering for each country is also presented. In view of the above, we intend to elicit information from a few countries – three. The choice of the three countries is informed by two reasoning: 1) the conclusions reached by work of Pavlia and Hunter (1996) where they noted that, “when comparing [the] use of methods across different countries, organizational profile especially in term of size, *revenue level*... may be an important intervening variable, 2) there are commonalities in motivation for IS workers, where similar economic conditions prevails; as gleaned from the publication of Couger and colleagues (Couger et al., 1990). Therefore, attempts were made at selecting three different countries at varying levels of economic performance, as our differentiator, we hope that better insight could emerge through this delineation. The geographic proximity of two of the selected countries is incidental. Thus, Finland is a rich country; Estonia an emerging economy and Nigeria a developing country are chosen (see, CIA world Factbook (2004), for relevant information). More importantly, the researcher has contacts in these three countries, which may be helpful in the data gathering phase. The following hypotheses are formulated to test our claims in line with our foregoing discussion.

H1a: *In light of the emerging computer subculture globally, ISD professionals from the different nations will attach the same importance to the factors of motivation.*

H1b: *No significant statistical differences exist in the factors that motivate ISD personnel for the three countries.*

H2: *No significant differences exist in the factors of job satisfaction for ISD professionals for the three countries*

H3a: *ISD professionals from different nations will attach the same importance to the factors of job dissatisfaction.*

H3b: *There will be no significant differences in the job dissatisfaction factors for ISD professionals in the three countries.*

3.1. Research Method – Participants and Procedures

The samples consists of systems analysts (42%), programmers/developers (47%) and other IS personnel (11%) averaged across the three countries. The sample consists of 71 participants (mostly mid-level IS professionals) representing a convenience sample, which is appropriate for a preliminary study of this nature. See Table 1 below for a summary. The data for Finland was collected in February, 2002 while the data from Nigeria and Estonia were collected between July – August, 2003. The respondents came from mainly software development, telecommunications and network engineering firms, IT departments of educational, and finance organisations. Prior to distributing the questionnaires a pilot-test was carried out using students in a polytechnic in Finland that has students from the chosen countries, as well as faculty members of the same institution. Useful insight emerged that assisted in the refinement of the final questionnaires adopted and distributed. The participants filled-out, a three-page long self-administered questionnaire. The service of *contacts* within the sampled organisations was employed to distribute and collect the questionnaires, which were numbered. The researcher then collected the returned questioners at specified periods. The research took place in Oulu (Finland), Tallinn (Estonia) and Lagos, Benin City (Nigeria).

The response rate of the returned questionnaires excluding the unusable questionnaires is 68%, 43% and 76% for the Finland, Estonian and Nigerian sub-samples respectively.

Table 1. Profile of respondents for the three countries

Total (N)	= 71		
Age			
	Nigeria (18) Frequency	Estonia (21) Frequency	Finland (32) Frequency
Below 25	2	10	6
26 – 39	13	9	21
40 – 55	3	1	5
56 – 67	0	1	0

<u>*Summary for the three* (3) *samples*</u>

Gender: Nigeria [female (f)- 28%, male (m)- 72%]; Finland [f- 34%, m - 66%];
Estonia [f - 19%, m - 81%]

Education: 65% of 3 samples have college degrees and 35% have other qualifications

Year of working experience: 7.5 years is the mean for the 3 samples, with s.d.= 0.9

These figures together represent an average of 62%, which is appropriate for this type of study. The participating organisations were selected at random. Non response error was checked for, by dividing each sample in two parts in order to see if there are differences in the pattern of responses – a common method used in statistics – none was found, indicating a fairly representative sample. Low non-response rate was tackled with the assurance of confidentiality and the usage of local languages in the questionnaire, except the ones for the Nigeria sample in which English was used. Back-translation method (Brislin, 1986) from English to Finnish and Estonian that converged at the second trail was employed. A short cover-letter explaining the purpose of the research was also provided.

The questionnaire used has a couple of parts. The section that deals with motivation utilises items whose construct and content validity have been verified and established (see, for example, Couger and Zawacki, 1980). Also, the job dissatisfaction part consists of items classified as "dissatisfiers" in Herzberg et al.'s, (1959) work. Other items on motivation and job satisfaction considered important in these countries were incorporated (Adigun and Stephenson; Hietanen, 2001). The ranking of motivation factors has been used by several researchers; see for example, (Couger et al., 1990). This part of the questionnaire has 13 determinants of motivation. Participants were asked to rank the listed items from the *most important* (1) to their motivation at their work place, to the *least important* (13). The second part of the questionnaire uses a 4-item Likert-type scale to determine job dissatisfaction, ranging from (strongly dissatisfied = 4, to not dissatisfied, at all = 1). Further, an item was used to assess the influence of personality types on the participants. It uses a 3-item Likert-type scale to measure the personality types (Eysenck and Wilson, 1975); namely, *introvert* (quiet and reserved) = 1, in-between = 2 and *extrovert* (lively and confident) = 3. Lastly, the job satisfaction measure developed by Hofstede (2001, p. 467) was incorporated without any modification. It comprises 14 items and uses the 5-item Likert scale ranging from (*not at satisfied* = 1 to *extremely satisfied* = 5).

Descriptive statistics were used to determine the relative order and importance of the items of motivation and job satisfaction, whilst ANOVA One way test was used to test for differences among the groups. Controlling tests involving age, personality type, work contract, education, income (parity), years of working experience and current tenure of employment were conducted to check for statistical significance differences in these variables, though not shown in the paper because of space restrictions. These tests were performed in order to assess if variations exist among the groups of ISD professionals on those variables. No differences were found, and the tests relating to the hypotheses began.

4. DATA ANALYSIS AND RESULTS

Hypothesis **H1a** predicted that ISD professionals from the three countries would attach the same importance to the motivation factors. Table 2 below shows the results. The results indicate differences between the two Western nations on the one hand, and the developing country on the other. ISD personnel being subsets of their corresponding nations and cultures; may attach different importance to such factors, as this study seems to suggest. Research in other fields have shown that job satisfaction needs and work-goals vary from countries to countries (Elizur et al., 1991; Harpaz, 1990). Apparently, this finding does not support the view of a global computer subculture for software professionals

Table 2. Descriptive Statistics for motivating factors the ISD professionals

	Finland			Estonia			Nigeria		
Factors	Rank	Mean	s.d.	Rank	Mean	s.d.	Rank	Mean	s.d.
Interesting job	**1**	2.3	2.0	**1**	4.7	3.9	**4**	5.8	3.6
Task Variety	**2**	5.0	3.0	**2**	5.3	2.7	**12**	8.8	2.4
Use of ability/ Fit with job	**3**	5.1	3.0	**3**	6.0	3.6	**6**	6.3	3.3
Autonomy and Independence	**4**	6.0	3.1	**4**	6.1	3.7	**11**	8.4	3.2
Creativity	**5**	6.2	3.5	**6**	6.5	3.5	**7**	6.5	4.3
Growth	**6**	6.3	3.2	**5**	6.3	2.9	**2**	4.8	3.3
Responsibility	**7**	6.4	2.8	**12**	8.6	3.8	**8**	7.1	2.8
Opportunity for advancement	**8**	6.8	3.6	**8**	7.4	4.2	**1**	4.6	2.9
Work Itself	**9**	7.4	4.4	**10**	7.7	4.6	**5**	5.9	4.5
Contribution to the org.	**10**	8.8	3.1	**7**	7.4	3.5	**9**	8.2	3.6
Recognition	**11**	9.3	2.5	**11**	8.1	3.8	**3**	5.1	3.6
Feedback	**12**	10.0	2.0	**9**	7.6	2.9	**13**	9.9	3.6
Status	**13**	10.1	3.2	**13**	9.2	3.5	**12**	9.6	2.7

across the globe (Carmel, 1999); at least, on the issue of job satisfaction and motivation. However, the ranking orders for the two Europeans countries appear more agreeable and uniform. In short, similarities are seen for the two economically advanced nations than for the developing nation. Nonetheless, the hypothesis is rejected.

H1b predicted that there would be no significant statistical differences in the factors that motivate ISD professionals from the three countries. The ANOVA One way test conducted indicates that 4 factors exhibit significant statistical differences. They are as follows: Opportunity for advancement: F= 4.65, p = .014; Task variety: F = 10.13, p = 0.000; Recognition: F = 4.56, p = 0.015; and Feedback: F = 4.02, p = .024 (all significant at 0.01 level). A strict test was employed by using the same number of subjects – 18 for each country. This finding is consistent with expectations. Moreover, it has been suggested that IS professionals from developed countries accept task variety, feedback, interesting job and autonomy as sources of motivation (Couger and Zawacki, 1980; Baroudi and Ginzberg, 1985; Couger and Cotler, 1983). Surprisingly, though, in this study, it is the ISD professionals from the less economically endowed nation that seemed to place more importance on such intrinsic factors as growth and advancement. This somewhat contradicts the work of (Elizur at al., 1991; Couger et al., 1990). Regardless, the hypothesis is rejected as no uniformity was observed here.

Also, **H2** predicted no significant differences would exist in the job satisfaction factors for the ISD professionals across board. This measure is the Hofstede's job satisfaction construct – not shown here. The ANOVA One way test (using 54 subjects) indicate there are differences on four items on this factor, as well. The items include, working with people who co-operate well with one another, having training opportunities, having opportunity

Table 3. Ranking of job dissatisfaction factors among ISD professionals

	FINLAND		NIGERIA		ESTONIA	
Factors	Rank	Mean/s.d.	Rank	Mean/s.d.	Rank	Mean/s.d.
Job security and stability	**1**	2.33/1.14	**5**	2.72/0.96	**17**	1.50/0.71
Failure to achieve	**13**	1.68/0.69	**12**	2.44/1.04	**7**	2.22/0.94
Company policy and administration	**12**	1.68/0.59	**13**	2.44/0.98	**9**	2.11/1.02
Pay	**9**	1.72/0.46	**1**	3.06/0.80	**3**	2.39/1.14
Training opportunity	**6**	1.94/1.00	**11**	2.44/0.78	**8**	2.17/0.99
Inter-personal relations/ Co worker support	**8**	1.83/0.86	**17**	1.89/0.76	**16**	1.61/0.92
Time for family and hobbies	**5**	1.94/0.73	**9**	2.50/1.04	**14**	1.72/0.83
Supervision	**11**	1.67/0.69	**15**	2.11/0.68	**12**	1.94/0.94
Promotion opportunities	**2**	2.11/0.90	**4**	2.72/0.83	**5**	2.28/1.27
Lack of fringe benefits	**7**	1.89/0.68	**2**	2.89/0.76	**1**	2.56/1.20
Urgent work (haste)	**4**	1.94/0.64	**8**	2.56/1.10	**11**	1.94/0.94
Lack of status	**10**	1.67/0.49	**14**	2.39/0.85	**13**	1.83/0.79
Too much work	**3**	2.06/0.80	**10**	2.50/0.92	**10**	2.06/0.94
Lazy/incompetent workers	**17**	1.39/0.61	**16**	1.94/0.94	**4**	2.39/1.09
Too little work	**16**	1.39/0.61	**7**	2.56/1.25	**15**	1.61/0.59
Negative feedback	**15**	1.39/0.50	**3**	2.83/1.10	**6**	2.28/1.02
Hassle	**14**	1.61/0.61	**6**	2.56/0.98	**2**	2.44/1.25

4-item Likert-type scale (strongly dissatisfied = 4; to not dissatisfied, at all = 1)

for advancement and having opportunity for advancement. This means that the ISD professionals from the three countries hold differing views on these items/factors. This hypothesis is rejected.

H3a hypothesized that the ranking order for the job dissatisfaction factors for ISD professionals would be same. Table 3 shows the summary. Pay (extrinsic factor) ranks among the highest sources of job dissatisfaction factors for the Nigerian and Estonian ISD professionals; on the other hand, it occupies a moderate position for Finnish ISD professionals. This is consistent with other studies (Harpaz, 1990; Elizur et al., 1991). Speaking about pay, McLean et al. (1996) concludes that pay does not lead to long-term satisfaction among IS professionals, in accordance with Herzberg et al., (1959) classic; however, it is noted as the most mentioned source of dissatisfaction for many IS employees. (See, also Smits et al., 1995). Moreover, job security ranked highest for Finnish ISD professionals. Incidentally, this finding reflects realities in Finland where job security concerns have consistently ranked highest as a de-motivator according to studies in the country (see, Hietanen, 2001). The lack of fringe benefits ranks highly amongst the Nigerian and Estonian samples and moderately with their Finnish counterparts. Promotion opportunities and training opportunity appear equally important for all three samples. Nonetheless, this hypothesis is rejected in so far as no same order of ranking is observed.

Hypothesis **H3b** predicted that there would be no significant statistical differences in job dissatisfaction factors for the groups. The ANOVA One way tests (using 54 subjects) indicate there are no significant statistical differences for the items except for company policy and administration, inter-personal relations/co worker support, supervision, promotion opportunities and haste. This hypothesis is also rejected.

5. DISCUSSIONS AND IMPLICATIONS

This paper addressed two main objectives: First, to determine whether similarities or differences exist on motivation, job satisfaction and dissatisfaction for ISD professionals from three selected countries, so as to see if any global professional computer subculture (on those themes) can be supported; second, how do these factors rate for each country? Overall, there were more marked differences than similarities regarding perception of motivational needs, job satisfaction and dissatisfaction factors. Generally speaking, our findings support evidence from other fields that workers from different nations hold differing viewpoints on the subject (Elizur et al., 1991; Harpaz, 1990). The findings of this paper contradict the notion of commonalities in motivational needs at a global level (Couger et al., 1990); although, there were more similarities between the Finnish and Estonian ISD professionals in some instances than with their Nigerian counterparts. This led us to suggest that such commonalities may be observable for nations with similar socio-economic endowments, or perhaps cultural similarities. Nevertheless, this present work is not aimed at providing a definitive answer in regard of that proposition. Our observation is that ISD professionals in each country attach different importance to items of motivation and job satisfaction despite the similarity in profession. To that end, we found no support for the claim of "computer professional subculture", at least, in regard of motivation and job satisfaction factors.

The implications of this study are threefold: First, for IS research, it does not depart from findings in extant literature that IS workers place less value on intrinsic motivators (Couger and Zawacki, 1980) even when economic indicators may differ markedly. Similarly, the results of this study highlights the import accorded such items as pay, job security and so on that recur in the discourse of job dissatisfiers among IS professionals (Rubin and Hernandez, 1988; McMurtrey, et al. 2002). Secondly, as per the usefulness for each country, this work may help practitioners to identify and better manage their valuable asset – the ISD professionals. For example, in Nigeria, IS workers may be motivated with the provision opportunity for growth, advancement and recognition; in Estonia and Finland, ISD workers may be motivated by the availability of an interesting job with variety that allows such workers to use their abilities/skills accordingly. Likewise, in managing "dissatisfiers" for a Finnish ISD professional, efforts should be made towards attenuating the discomfort that may arise from lack of job security and promotion opportunities, whilst for Estonian and Nigerian ISD professionals, monetary rewards appear more important. As was illustrated in Figure 1, management in each country can bring the best out of their ISD professionals only if they know which factors are important, and which ones are not. Thirdly, this paper adds knowledge that may be valuable in the wider HR management with its findings, which could be useful as human resources are increasingly being sought and engaged from all parts of the globe. One would expect that specific underpinnings (on motivation and job satisfaction) of each nation are taken into consideration, when the need for such arises. In general, this paper adds to the important discourse in HR management, as it relates to IS professionals (Brancheau et al., 1996; Bostrom and Heinen, 1977; Couger, 1989).

The limitations in this study include the following: The small-sized sample used, and the bias from the usage of a convenience sample. Larger samples may yield different results. Thus, the generalisability and validity of the findings, is limited by the foregoing reasoning. Another limitation is the heterogeneity among the participants studied, majority were analysts and programmers, while others (11%) were Network Engineers and Administrators. Also, the incidence of bias on the parts of the respondents cannot be ruled out in so far as some had to give back the filled-in questionnaire with their organisations.

6. CONCLUSION

Motivation and job satisfaction of ISD personnel are key issues in IS human resources that needs to be seamlessly addressed by management (Brancheau et al., 1996; Couger, 1989; Bartol and Martin, 1982; Couger and Amoroso, 1990). This paper used empirical data to investigate motivation and job satisfaction among ISD professionals in three economically different countries. This study found differences and similarities on the issue of motivation, job satisfaction and dissatisfaction among ISD professionals for the three countries. The evidence uncovered does not allow us to support the claim of a global professional computer subculture; at least on the themes of this work. Instead, it is seen that ISD professionals from different nation have differing viewpoints. It is worthwhile to note that commonalities in perspectives were noticeable for motivation for the two countries that are geographically and economically similar. The contribution of this paper thus, is that it enhances our knowledge regarding the discourse of IS professionals' job satisfaction and motivation within the global economy characterised by cross-national collaborations. The specific challenges to each of the countries studied were also discussed. Importantly, it is concluded that factors of motivation - extrinsic and intrinsic - that may impact upon ISD professionals may in fact, be specific to each nation. Further research using larger sample size is needed to deepen our knowledge.

REFERENCES

Adigun, I. O., and Stephenson, G. M., 1992, Sources of job motivation and satisfaction among British and Nigerian employees, *Journal of Social Psychology* **132**(3):369–377.

Anakwe, U. P., Anandarajan, M., and Igbaria, M., 1999, Information technology usage dynamics in Nigeria: an empirical study, *Journal of Global Information Management* **7**(2):13–21

Bartol, K., and Martin, P., 1982, Managing information systems personnel: A review of the literature and managerial implications, *MIS Quarterly*, Special Issue, pp. 49–70.

Baroudi, J. J., and Ginzberg, M. J., 1985, Impact of the technological environment on programmers and analysts job outcomes, *Communication of the ACM* **29**(6):546–555.

Bostrom, R., and Heinen, J. S., 1977, MIS problems and failures. A socio-technical perspective. Causes, *MIS Quarterly* **3**.

Brancheau, J. C., Janz, B. D., and Wetherbe, J. C., 1996, Key issues in information systems management: 1994–1995, SIM Delphi results, *MIS Quarterly* **20**(2).

Brislin, R., 1986, The wording and translation of research instruments, in: *Fields Methods in Cross-Cultural Research*, W. J. Lonner and J. W. Berry, eds., Sage, CA.

Bryan, N. B., McLean, E. R., Smits, S. J., and Burn, J. M., 1995, Work perceptions among Hong Kong and the US I/S workers: a cross-cultural comparison, *Journal of End User Computing* **7**(4):22–29.

Carmel, E., 1999, Global Software Teams: Collaborating Across Borders and Time Zones, Prentice Hall, NJ.

Champy, J. A., 1992, Mission Critical, *CIO*, January 1992.

Constantine, L., 2001, The Peopleware Papers: Notes on the Human Side of Software, Prentice Hall, NJ.

Couger, J. D., and Cotler, M. A., 1983, The Effects of maintenance assignments on goal congruence for programmers and developers, in: *Proceedings of ICIS*, pp. 83–100.

Couger, J., 1989, New challenges in motivating information systems personnel, *Journal of Information Systems Management*, Fall, pp. 36–41.

Couger, J. D., 1986, Effects of cultural differences on motivation of analysts and programmers: Singapore vs. the United States, *MIS Quarterly* **10**(2):188–196.

Couger, J. D., 1988, Motivators vs. demotivators in IS environment, *Journal of Systems Management* **39**(6).

Couger, J. D., and Zawacki, R. A., 1980, Motivating and managing computer personnel, John and Wiley and Sons, New York.

Couger, J. D., Adelsberger, I., Borovits M., Zviran, M., and Motiwalla, J., 1990, Commonalities in motivating environments for programmers/analysts in Austria, Israel, Singapore, and the U.S., *Information and Management* **18**:41–46.

Couger, J., and Amoroso, J., 1990, Elevating the Top IS Positions to CIO, *Proceedings: International Conference on Systems Science*, January, pp. 324–330.

De Meyer, A., and Mizushima, A., 1989, Global R & D Management, *R & D Management* **19**(2).

Dessler, G., 2001, Management: Leading People and Organizations in the 21st Century, Prentice Hall.

Eldon, Y. L., and Abraham, B. S., 1991, Stress dynamics of Information Systems Manager: A contingency Model, *Journal of Management Information Systems* **7**(4):107–130.

Elizur, D., Ingwer, B., Raymond, H., and Istvan, M. B., 1991, The structure of work values: A cross cultural comparison, *Journal of Organizational Behavior* **12**:21–38.

Eysenck, H. J., and Wilson, G., 1975, Know Your Own Personality, Harmondsworth, Penguin.

Goldstein, D. K., and Rockart, J. F., 1984, An examination of work-related correlates of job satisfaction in programmers/analysts, *MIS Quarterly* **8**(2):103–115.

Goldthorpe, J. H., Lockwood, D., Bechhofer, F., and Platt, J., 1968, The Affluent Worker: Industrial Attitudes and Behaviour, Cambridge University Press, Cambridge.

Griesser, J. W., 1993, Motivation and information system professional, *Journal of Managerial Psychology* **8**(3):21–30.

Hackman, J. R., and Oldham, G. R., 1978, Development of the job diagnostics survey, *Journal of Applied Psychology* **60**(2):159–170.

Harpaz, I., 1990, The importance of work-goal: an international perspective, *Journal of International Business Studies*, First Quarter, pp. 75–93.

Herzberg, F., Mausner, B., and Snyderman, B. B., 1959, The Motivation to Work, Wiley, New York.

Hietanen, J., 2001, Increase in atypical work weakens employees' motivation, Finnish Ministry of Labour press [id: FI0101173F], referenced in *eiro* (European industrial relation online, Dec. 2003).

Hofstede, G., 2001, Culture's consequences: comparing values, behaviours, institutions, and organizations across nations, Sage Publications.

Iaffaldano, M., and Muchinsky, P., 1985, Job satisfaction and job performance: A meta analysis', *Psychological Bulletin* **97**(2):251–273.

Ifinedo, P., 2004, E-government – Precursors, Problems, Practices and Prospects: A Case of Nigeria, *Proceedings of the 2004 IBIM Conference*, Amman, Jordan, 4–6 July 2004.

IMJ, 2003, Estonia Embraces Cyberspace, *Information Management Journal*, July/August, 2003.

Kumar, K., and Bjørn-Andersen, N., 1990, A cross-cultural comparison of IS designer values, *Communication of the ACM* **33**(5):528–538.

Lawler, E. E., 1994, Reviews of 'Motivation in Work Organizations', *Worklife Report* **9**(3):20.

Levitt, T., 1983, The Globalisation of Markets, HBR, 1983, May/June.

Locke, E. A., 1983, The nature and causes of job satisfaction, in: *Handbook of Industrial and Organisational Psychology*, M. D. Dunnette, ed., John Wiley and Sons, New York.

Lucas, H., and Turner, J., 1982, A corporate strategy for the control of information processing, *SMR* **23**(3).

Lyytinen, K., and Goodman, S., 1999, Finland: the unknown soldier on the IT front, *CACM* **42**(3).

Mata, T., Ramon, A., and Unger, E. A., 1985, Another look at motivating DP professionals, Comp. Personnel, 10.

McLean, E. R., Smits, S. J., and Tanner, J. R., 1996, The importance of salary on job and career attitudes of information systems professionals, *Information and Management* **31**(3):291–299.

McMurtrey, M. E., Grover, V., Teng, J. T. C., and Lightner, N. J., 2002, Job satisfaction of Information Technology workers: The impact of career orientation and task automation in a CASE environment, *Journal of Management Information Systems* **19**(2):273–302.

Mumford, E., 1972, Job Satisfaction: A study of Computer Specialists, Longman, London.

Niederman, F., Brancheau, J. C., and Wetherbe, J. C., 1991, Information Systems Management Issues for the 1990s, *MIS Quarterly* **15**(4):475–495.

Niederman, F., and Crosetto, G., 1996, Valuing the IT workforce as intellectual capital, *Proceedings of the 1999 ACM SIGCPR conference on Computer personnel research*.

Palvia, S. C., and Gordon Hunter, M. G., 1996, Information systems development: a conceptual model and a comparison of methods used in Singapore, USA, and Europe, *JG IM* **4**(3).

Robinson, R. V., 1983, Review of Culture's consequences: International differences in work-related values, *Work and Occupations* **10**:110–115.

Rosenfeld, R. H., and Wilson, D. C., 1999, Managing Organizations: Texts, Readings & Cases, MGH, London.

Rubin, H. I., and Hernandez, E. F., 1988, Motivations and behaviors of software professionals, *Proceedings of the ACM SIGCPR conference on Management of information systems personnel*, pp. 62–71.

Senker, J., and Senker, P., 1992, Gain Competitive Advantage from Information Technology, *JGM* **17**(3).

Scarpello, V., and Campbell, J. P., 1983, Job satisfaction: are all parts there?, *Personnel Psychology* **36**.

Shore, B., and Venkatachalam, A. R., 1994, The role of national culture in systems and design, *JGIM* **3**(3).

Smits, S. J., Tanner, J. R., and McLean, E. R., 1995, Herzberg Revisited: The Impact of Salary on the Job and Career Attitudes of I/S Professionals, *1995 ACM SIGCPR Conference*, Comp. Personnel Research.

A COMBINED NEURAL NETWORK AND DECISION TREE APPROACH FOR TEXT CATEGORIZATION

Nerijus Remeikis, Ignas Skucas, and Vida Melninkaite*

1. INTRODUCTION

As the volume of information continues to increase, there is growing interest in helping people better find, filter, and manage these resources. Text categorization - the assignment of natural language documents to one or more predefined categories based on their semantic content - is an important component in many information organization and management tasks. Automatic text categorization task can play an important role in a wide variety of more flexible, dynamic and personalized tasks as well: real-time sorting of email or files, document management systems, search engines, digital libraries, decision support systems for design. Text categorization is now being applied in many contexts, ranging from document indexing based on a controlled vocabulary, to document filtering, automated metadata generation, word sense disambiguation, population of hierarchical catalogues of Web resources, and in general any application requiring document organization or selective and adaptive document dispatching.

A number of statistical classification methods and machine learning techniques have been applied to text categorization, including techniques based on decision trees (Lewis and Ringuette, 1994), neural networks (Wiener et al., 1995), Bayes probabilistic approaches (Lewis and Ringuette, 1994). However there is still need more accurate text classifiers based on new learning machine learning approaches.

The purpose of the current work is to describe ways in which hybrid machine learning method can be applied to the problem of text categorization, and to test its performance relative to a number of other text categorization algorithms. In this paper, we introduce the use of a hybrid decision tree and neural network technique to the problem of text categorization, because hybrid approaches can simulate human reasoning in a way that a decision tree learning is used to do qualitative analysis and neural learning is used to do subsequent quantitative analysis. Our approach is based on hybrid algorithm, which

* Vytautas Magnus University, Vileikos str. 8, Lt-3035 Kaunas, Lithuania, Nerijus_Remeikis@fc.vdu.lt, Ignas_Skucas@fc.vdu.lt, Vida_Melninkaite@fc.vdu.lt.

was described in paper (Banerji, 1997) and has been shown to perform well on standard machine learning tasks. The main idea of this method is to transform decision trees to neural networks. Our proposed hybrid machine learning method for text categorization task constructs the networks by directly mapping decision nodes or rules to the neural units and compresses the network by removing unimportant and redundant units and connections.

2. PROBLEM DEFINITION

Automatic text categorization has always been an important application and research topic since the inception of digital documents. The text classification task can be defined as assigning category labels to new documents based on the knowledge gained in a classification system at the training stage. A wide range of supervised machine learning algorithms has been applied to this area using a training data set of categorized documents. Although text classification performance results has been quite encouraging, but there is still need more accurate text classifies based on new learning machine learning approaches (Sebastiani, 2002).

Performance of neural networks learning is known to be sensitive to the initial weights and architecture – number of hidden layers and neurons in these layers. Traditionally, the initial values of weights are determined randomly in the backpropagation neural network (BPN) and modified by the generalized delta rule. Although the BPN has been implemented in many applications, it still has some major drawbacks; namely, its convergence tends to be very slow, the optimal number of hidden nodes is difficult to determine, and usually it does not yield optimal solutions. Usefulness of good initialization, effect of initial values, prior weights are discussed in (Raudys, 2001). Recently, pattern recognition techniques have been used to initialize weights. (Raudys and Skurichina, 1992) used a piece-wise linear classiffier to initialize hidden layer weights of the three-layer neural network. In (Sethi, 1991; Banerji, 1997) decision tree was applied to initialize the neural network. These methods typically construct the networks by directly mapping decision nodes or rules to the neural units. Text categorization datasets are large and increases very rapidly. Growing decision trees with increasingly larger amounts of training data will result in larger decision tree sizes. As a result, the neural networks constructed from these decision trees are often larger and more complex than necessary.

Error based pruning can be used to prune a decision tree. It uses a parameter, the certainty factor, which affects the size of the pruned tree. Here, we show that varying the certainty factor allows significantly smaller trees to be obtained with minimal or no accuracy loss. Also, appropriate choice of certainty factor is able to produce trees that are essentially constant in size in the face of increasingly larger training sets. Experimental results support the conclusion that error based pruning can be used to produce appropriately sized trees, which are directly mapped to optimal neural network architecture with good accuracy.

This paper presents our attempt to improve text classification accuracy by neural network initialized with decision tree classifier and to compare classification performance to previous researches results.

3. RELATED WORK

Combination of decision trees and neural networks can be classified into two groups:

1. In the first group neural networks may be used in the implementation of various aspects of decision trees. For instance, (Gelfand and Guo, 1992) use neural networks in the internal nodes of a decision tree to perform the task of feature selection and decision boundary construction. In this approach, one is constructing a tree of neural networks.
2. The second group uses decision trees to form the structure of the neural network. The key idea is to construct a decision tree and then convert the tree into a neural network. This is the direction (Sethi, 1991; Banerji, 1997) have taken. The main idea here is that in the design of multilayer feedforward networks, the structure of the network, i.e., the number of hidden layers and the number of neurons in each hidden layer is not known in advance, and is often chosen rather heuristically and by trial and error. The method described in (Sethi, 1991) offers one way to uncover the structure of the network That is, they offer a method to implement a pre-designed decision tree via a multilayer feed forward neural network. Perhaps one of the main advantages of this approach, other than possibly eliminating guess work in the design of multilayer neural networks, is that the network designed this way usually has fewer connections.

Our method presented in this paper has some advantages over (Sethi, 1991; Banerji, 1997) method. Decision tree size grows approximately linearly with increasingly larger amounts training set size. Neural networks constructed from these trees are often larger and complex than necessary. When the certainty factor value is appropriately tuned for the data set, error-based pruning can give trees that are essentially constant in size regardless of the increasingly larger amount of training data.

4. DECISION TREE MAPPING TO NEURAL NETWORK

4.1. Decision Tree Construction Algorithm

An algorithm constructs the decision tree with a divide and conquer strategy. Each node in a tree is *associated* with a set of cases. At the beginning, only the root is present, with associated the whole training set and with all case weights equal to 1. At each node the following *divide and conquer* algorithm is executed, trying to exploit the locally best choice, with no backtracking allowed.

Let T be the set of cases associated at the node. The weighted frequency $\mathit{freq}(C_i, T)$ is computed of cases in T whose class is C_i, $i \in [1, N]$. If all cases in T belong to a same class C_j (or the number of cases in T is less than a certain value) then the node is a leaf, with associated class C_j.

If T contains cases belonging to two or more classes, then the *information gain* of each attribute is calculated:

$$I = H(T) - \sum_{i=1}^{s} \frac{|T_i|}{|T|} \times H(T_i), \tag{1}$$

where

$$H(T) = -\sum_{j=1}^{n} \frac{freq(C_j, T)}{|T|} \times \log_2 \left(\frac{freq(C_j, T)}{|T|} \right), \tag{2}$$

is the entropy function.

While having an option to select information gain, by default, however, C4.5 (Quinlan, 1993) considers the *information gain ratio* of the splitting $T_1, \ldots, T_s$ which is the ratio of information gain to its split information:

$$S(T) = -\sum_{i=1}^{s} \frac{|T_i|}{|T|} \times \log_2 \left(\frac{|T_i|}{|T|} \right). \tag{3}$$

4.2. Training Multilayer Neural Network

Neurons in the input layer only act as buffers for distributing the input signals x_i to neurons in the hidden layer. Each neuron j in the hidden layer sums up its input signals x_i after weighting them with the strengths of the respective connections w_{ji} from the input layer and computes its output y_j as a function f of the sum as follows

$$y_j = f\left(\sum w_{ji} x_i\right), \tag{4}$$

where f can be a simple threshold function, a linear or a sigmoid function.

There are many available learning algorithms in the literature (Rumelhart and Mcclelland, 1986; Haykin, 1994). Backpropagation with momentum is the most commonly adopted neural network training algorithm (Rumelhart and Mcclelland, 1986). The backpropagation algorithm employs a gradient descent technique to adopt the neural network weights to minimise the mean squared difference between the ANN output and the desired output. The change in weight $\Delta w_{ji}(k)$ between neurons i and j is as follows

$$\Delta w_{ji}(k) = \eta \delta_j x_i + \alpha \Delta w_{ji}(k-1), \tag{5}$$

where η is a parameter called the learning coefficient, α is the momentum coefficient, and δ_j is a factor depending on output neuron or a hidden neuron. For output neurons

$$\delta_j = \frac{\partial f}{\partial net_j} (y_j - y_{net-j}), \tag{6}$$

where $net_j = \sum_i x_i w_{ji}$, y_j, y_{net-j} are the target and the neural outputs for neuron j, respectively. For hidden neurons

$$\delta_j = \frac{\partial f}{\partial net_j} \sum_q w_{qj} \delta_q. \tag{7}$$

As there are no target outputs for hidden neurons in (4), the difference between the target and the actual neural output of a hidden neuron j is replaced by the weighted sum of the δ_q terms already obtained for neurons q connected to the output of j. Thus, iteratively, beginning with the output layer, the δ term is computed for neurons in all layers and weight updates determined for all connections according to (5).

Training a neural network by backpropagation with momentum training algorithm to compute y involves presenting it sequentially with different training sets. Differences between the target output and the actual output of the neural network are backpropagated through the network to adapt its weights using (5)–(7). The adaptation is carried out after the presentation of each set. Each training epoch is completed after all patterns in the training set have been applied to the networks.

4.3. Decision Tree Error-Based Pruning

In general, a decision tree can be grown so as to have zero error on the training set. Also, in general, over-fitting occurs and the tree needs to be pruned in order to generalize well on the test set.

Error-based pruning considers the E errors among the N training examples at a leaf of the tree to give an estimate of the error probability for that node. The assumption is that these are E events in N independent trials which is, of course, not perfectly true. We want to know what the observed result tells us about the probability of an error over the entire population of examples that will end up at the leaf. Using the binomial theorem, confidence limits can be calculated for the probability of error for a given confidence level. The confidence level is the certainty factor parameter CF. The upper limit of the probability is found. Given this value the predicted number of errors for each leaf of a test node being considered for pruning can be calculated by multiplying the number of examples at the leaf by the upper limit of the probability confidence limit. The predicted number of errors if a node was a leaf can be calculated from the observed number of errors after its leaves are collapsed. Error estimate for a node is:

$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \bigg/ \left(1 + \frac{z^2}{N} \right), \tag{8}$$

where f is the error on the training data, N is the number of instances covered by the leaf and z is computed from normal distribution. The leaves are pruned if the number of predicted errors after pruning is less than the sum of predicted errors across the leaves. The smaller the CF becomes the more certain we are that the confidence interval contains the true probability of error. That is, the confidence interval is wider, and the upper limit on the probability that a particular example is in error is higher, making an example more likely to be incorrect and hence more pruning will be done. With a $CF = 100$ we have no confidence that the true error is in the interval and would simply take the observed error rate at the leaf.

The first thing to recognize is that a tree pruned at CF_2 can be obtained by pruning the tree pruned at CF_1 when $CF_1 > CF_2$. So, the search for the appropriate certainty factor would consist of choosing an initial certainty factor, CF_i, for pruning and then evaluating the resultant tree on the validation set to determine its accuracy. Next, choose a new certainty factor lower than the last and prune the pruned tree. Evaluate the resultant tree on the validation set. Continue creating new pruned trees until the stopping criterion is met.

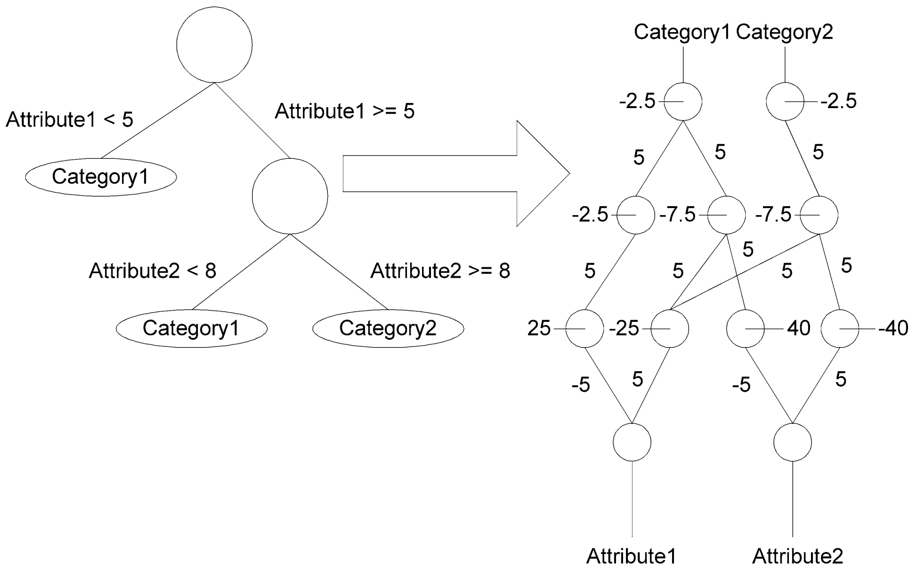


Figure 1. Transformation of decision tree to neural network.

4.4. Neural Network Initialization with Decision Tree Classifier

If we compare decision trees and neural networks we can see that their advantages and drawbacks are almost complementary. For instance humans easily understand knowledge representation of decision trees, which is not the case for neural networks. Decision trees have trouble dealing with noise in training data, which is again not the case for neural networks, decision trees learn very fast and neural networks learn relatively slow, etc. Our idea was to combine decision trees and neural networks in order to combine their advantages. That is why we developed a combined approach for building decision trees.

First we build a decision tree that is then used to initialize a neural network. Such a network is then trained using the same training objects.

The source decision tree is converted to a disjunctive normal form, which is a set of normalized rules. Then the disjunctive normal form serves as source for determining the neural network's topology and weights. The neural network has two hidden layers. The number of neurons on each hidden layer depends on rules in the disjunctive normal form. The number of neurons in the output layer depends on how many outcomes are possible in the training set. The conversion is described in the next steps (see Figure 1):

1. Build a decision tree, using any available approach.
2. Every path from the root of the tree to every single leaf is presented as a rule.
3. The set of rules if transformed into the disjunctive normal form, which is minimal representation of original set of rules.
4. In the input layer create as many neurons as there are attributes in the training set.
5. For each literal in the disjunctive normal form there is a neuron created in the first hidden layer (literal layer) of a neural network.

6. Set weights for each neuron in the literal layer, that represents a literal in the form (attribute $\geq$ value) to $w_0 = -\sigma * value$ for each literal in the form (attribute $\geq$ value) to $w_0 = \sigma * value$. Set all the remaining weights to $+\beta$ or $-\beta$ with equal probability. Constant σ is usually a number larger then 1 (usually 5) and constant β is a number close to 0 (usually 0.05).
7. For every conjunction of literals create a neuron in the second hidden layer (conjunctive layer).
8. Set weights that link each neuron in the conjunctive layer with the appropriate neuron in the literal layer to $w_0 = \sigma * (2n - 1)/2$, where n is a number of literals in the conjunct. Set all the remaining weights to $+\beta$ or $-\beta$ with equal probability.
9. For every possible class create a neuron in the output layer (disjunctive layer).
10. Set weights that link each neuron in the disjunctive layer with the appropriate neuron in the conjunctive layer to $w_0 = -\sigma * (1/2)$ Set all remaining weights to $+\beta$ or $-\beta$ with equal probability.
11. Train the neural network using the same training objects as were used for training the decision tree.

Such network is then trained using backpropagation. Mean square error of such network converges toward 0 much faster than it would in the case of randomly set weights in the network. Even if we would use neural network before the backpropagation stage, it would already give good results.

5. EXPERIMENT AND RESULTS

5.1. Text Corpora

It is hard to find standard benchmark sets for text classification, where each method can be tested and its performance compared reliably with other methods. The Reuters sets are a notable exception. This collection consists a set of newswire stories classified under categories related to economics. Although different versions are available, many researchers use it for benchmarking. We will use the ApteMod version of Reuters-21578 (Yang and Liu, 1999). The ApteMod set has 7769 documents for training and 3019 for testing, after stemming and stop word removal 24240 unique terms remain. The Aptemod version has an average of 1.3 categories per document, with a total of 90 categories that occur in both sets.

5.2. Feature Selection and Extraction

In text categorization, features are often measures of frequencies of words appearing in a document. Feature selection chooses which features to be used in classification. It is preferable to use less features than the raw measurements (say, frequency of each word), so that classification will be performed in a feature space of a lower dimensionality. By reducing the dimensions of the feature space, it not only increases the efficiency of the training and test processes, but also reduces the risk of overfitting the model to data. Feature extraction computes the chosen features from an input document. In statistical classification, features are represented in a numerical vector, which is subsequently used by the classifiers. Feature selection involves stop word removal, stemming, and term selection:

- **Stop Word Removal.** Words used in text indexing and retrieval are called terms. According to the term discrimination model, moderate frequency terms discriminate the best. High frequency words, which are called *stop words*, have low information content, and therefore have weak discriminating power. They are removed according to a list of common stop words.
- **Stemming.** Stemming reduces morphological variants to the root word. For example, “asks”, “asked”, and ”asking” are all reduced to “ask” after stemming. This relates the same word in different morphological forms and reduces the number of distinctive words. The *Porter stemmer* is a commonly used stemmer (Frakes and Baeza-Yate, 1992).
- **Term Selection.** Even after the removal of stop words and stemming, the number of distinct words in a document set may still be too large, and most of them appear only occasionally. In addition to removing high frequency words, the term discrimination model suggests that low frequency words are hard to learn about and therefore do not help much. They should be removed to reduce the dimensions of the vector space as well. We used information gain selection method (Yang and Pedersen, 1997).
- **Feature Extraction.** After the terms are selected, for each document a feature vector is generated whose elements are the feature values of each term. A commonly used feature value is the *term frequency* (number of occurrence of a term in a document).

5.3. Classification

A number of classifiers have been tried on text categorization. In our experiment, we focused on the evaluation of the neural network initialized with decision tree classifier on text categorization. We compare its accuracy to those of classical decision tree C4.5 (Quinlan, 1993) and feedforward backpropagation neural network initialized with random initial weights. Other the error back-propagation neural networks parameters that have been used in the experiments:

- Learning rate – 0.6.
- Number of iteration – 500.

In our research we used Weka 3 (Witten and Frank, 2000), a machine learning software in Java developed at the University of Waikato in New Zealand. Weka is a collection of machine learning algorithms (neural networks, decisions trees) for solving real-world data mining problems. Weka is also well suited for developing new machine learning schemes. We implemented combined neural network and decision tree algorithm in this package. Multilayer neural networks, decisions trees are also implemented.

5.4. Evaluation

The performance of category ranking can be evaluated in terms of *precision* and *recall*, computed at any threshold on the ranked list of categories of each document. The category

Table 1. The decision matrix for calculating the classification accuracy

	Expert YES	Expert NO
System YES	a	b
System NO	c	d

assignment of a binary classifier can be evaluated using a two-way contingency table (see Table 1).

The relationship between the system classification and the expert judgment is expressed using four values as shown in Table 1. Precision is defined as a/(a+b), and recall is defined as a/(a+c).

For evaluating performance average across categories, there are two conventional methods, namely macro-averaging and micro-averaging. Macro-averaged performance scores are computed by first computing the scores for the per-category contingency tables and then averaging these per-category scores to compute the global means. Micro-averaged performance scores are computed by first creating a global contingency table whose cell values are the sums of the corresponding cells in the per-category contingency tables, and then use this global contingency table to compute the micro-averaged performance scores. There is an important distinction between macro-averaging and micro-averaging. Micro-averaging performance scores give equal weight to every document, and is therefore considered a per-document average. Likewise, macro-average performance scores give equal weight to every category, regardless of its frequency, and is therefore a per-category average.

5.5. Results

Decision tree size grows approximately linearly with increasingly larger amounts training set size. Neural networks constructed from these trees are often larger than necessary. Large networks take a long time to learn, and tend to give accurate classification results for training data, but not for unknown test data. There are various approaches to pruning decision trees, including error-based pruning, reduced error pruning, minimum description length pruning, and others (Quinlan, 1993). One well-known element of machine learning folklore is that decision tree pruning methods generally do not prune hard enough. Here we will show that, in general, an appropriate setting of the certainty factor for error-based pruning will cause decision tree size to plateau.

Figure 2 shows results obtained with the Reuters-21578 data set. The training set size is varied from 1000 to 9584. Data is plotted as the error-based pruning certainty parameter is varied across values of 25 (the default), 10, 1, 0.1, 0.01, 0.001, 0.0001, 0.00001. The curve for the default certainty factor grows linearly. However, the family of curves clearly shows that the behavior depends on the value of the certainty factor. If the certainty factor is set as low as 0.001, then the average tree size varies between 61 and 373 over all training set sizes. That is, the tree size is minimal and optimal, just as desired.

For evaluating the effectiveness of our system, we used the breakeven points. The breakeven point is the point at which precision equals recall. Various background conditions, often extraneous to the learning algorithm itself, may influence the results. These may include, among others, different choices in pre-processing (stemming, etc.), indexing,

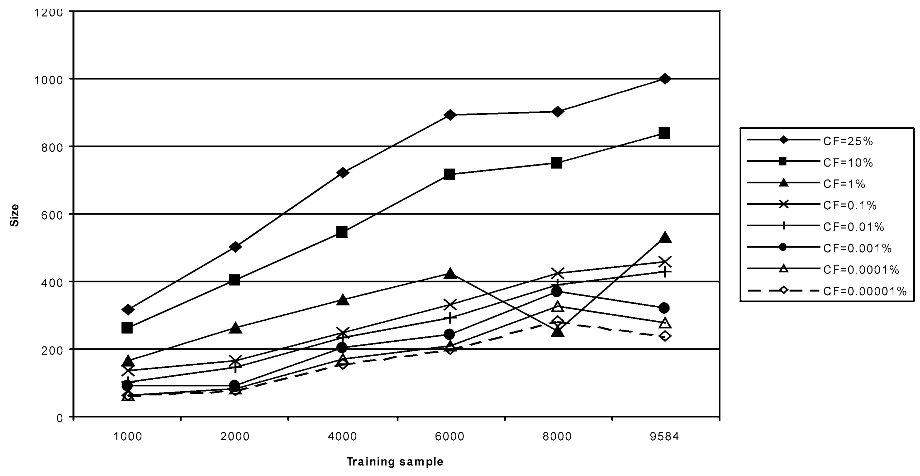


Figure 2. Low certainty factor can give constant tree size with added training data.

Table 2. Precision/recall-breakeven point for ten most populated Reuters categories

Category	Hybrid approach	Neural networks	Decision trees
Acq	90.9%	84.2%	85.3%
Corn	82.3%	62.7%	87.7%
Crude	77.9%	58.4%	75.5%
Earn	92.8%	85.6%	96.1%
Grain	72.1%	56.4%	89.1%
Interest	70.5%	60.8%	49.1%
Money-fx	72.4%	62.3%	69.4%
Ship	73.6%	67.6%	80.9%
Trade	69.7%	59.2%	59.2%
Wheat	86.5%	46.9%	85.5%
Micro-avg	82.1%	68.2%	79.4%
Macro- avg	78.24%	64.40%	77.78%

dimensionality reduction, classifier parameter values. Table 2 summarizes micro-averaged breakeven performance for our hybrid classifier and to previous researches results achieved by neural networks and decision trees classifiers (Sebastiani, 2002). The measures used are precision/recall-breakeven point, micro-average and macro-average on ten most populated Reuters categories. Our system proves to perform well as compared to the neural networks and decision trees methods.

6. CONCLUSIONS

This paper discussed a neural network initialization technique for text categorization. It employs the use of neural networks initialized with decision tree classifier. Our study provides evidence that hybrid machine learning approach can be used for the construction of effective classifiers for automatic text categorization. We have presented a hybrid decision tree and neural network algorithm for building the classifier.

Decision tree size grows approximately linearly with increasingly larger amounts training set size. Neural networks constructed from these trees are often larger and complex than necessary. When the certainty factor value is appropriately tuned for the data set, error-based pruning can give trees that are essentially constant in size regardless of the increasingly larger amount of training data. The solution for choosing the certainty factor is given in this paper. This generally requires values of the certainty factor much smaller than the default value. Appropriate choice of certainty factor is able to produce trees that are essentially constant in size in the face of increasingly larger training sets. Experimental results support the conclusion that error based pruning can be used to produce appropriately sized trees, which are directly mapped to optimal neural network architecture with good accuracy.

This paper showed that hybrid decision tree and neural network approach improved accuracy in text classification task and are substantially better than single decision tree or neural network initialized randomly text classifiers performance comparable to previous researches results.

Although encouraging results have been obtained using hybrid approach based classifier, there is still much work remaining to be investigated. They include to create new decision tree construction algorithm for textual data and to determine how much it has an effect on hybrid classifier accuracy. These issues are left for our future works.

REFERENCES

Banerji, A., 1997, Initializing neural networks using decision trees, *Computational Learning Theory and Natural Learning Systems*, MIT Press, IV, 3–15.

Frakes, W., and Baeza-Yates, R., 1992, *Information Retrieval: Data Structures & Algorithms*, Prentice Hall.

Guo, H., and Gelfand, S. B., 1992, Classification trees with neural-network feature extraction, *IEEE Trans. Neural Networks* **3**:923–933.

Haykin, S., 1994, *Neural Networks: A comprehensive foundation*, Macmillan College Publishing Comp., New York.

Lewis, D. D., and Ringuette, M., 1994, A comparison of two learning algorithms for text categorization, in: *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, pp. 81–93.

Quinlan, J. R., 1993, *C4-5: Programs for machine learning*, Morgan Kaufmann, San Mateo, CA.

Raudys, S., 2001, *Statistical and Neural Classifiers: An integrated approach to design*, Springer-Verlag, NY, pp. 169–173.

Raudys, S., and Skurichina, M., 1992, The role of the number of training samples on weight initialization of artificial neural net classifier, *Neuroinformatics and Neurocomputers. Proc. RNNS/IEEE Symposium*, Rostov-on-Don, Russia, pp. 343–353.

Rumelhart, D. E., and Mcclelland, J. L., 1986, *Parallel distributed processing 1*, MIT Press, Cambridge, MA.

Sebastiani, F., 2002, Machine learning in automated text categorization, *ACM Computing Surveys* **34**(1):1–47.

Sethi, I. K., 1991, Decision tree performance enhancement using an artificial neural network implementation, in: *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*, I. K. Sethi and A. K. Jain, eds., Elsevier, Amsterdam, the Netherlands, pp. 71–88.

Wiener, E. D., Pedersen, J. O., and Weigend, A. S., 1995, A neural network approach to topic spotting, in: *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, pp. 317–332.

Witten, I. H., and Frank E., 2000, *Data mining: Practical Machine Learning Tool sand Techniques With Java Implementation*, Morgan Kaufmann Publishers, San Francisco, CA.

Yang, Y., and Pedersen, J., 1997, A comparative study on feature selection in text categorization, in: *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, US, pp. 412–420.

Yang, Y., and Liu, X., 1999, A re-examination of text categorization methods, in: *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, Berkeley, US, pp. 42–49.

WORKING WITH METHODS: OBSERVATIONS ON THE ROLE OF METHODS IN SYSTEMS DEVELOPMENT

Päivi Ovaska*

1. INTRODUCTION

Studies on large software projects' failures show that projects are frequently not completed in time, are of heterogonous quality and often more money is needed than planned.[1] System development practitioners and researchers agree that these failures result from irrational application of development practices by system development practitioners.[2] As a solution to these problems, researchers have developed hundreds of new system development methods[3]. The general status of methods has been described as "method jungle",[4, 5] an unorganized collection of methods more or less similar to each other.[6, 7] When we include the in-house method development, the number of methods used rises to thousands.[2, 8] Paradoxically, despite the efforts devoted to method development, no agreement on their usefulness in organizations exists.[9, 10] The two main reasons cited for this are that the, mostly academic, method developers have no profound knowledge of the practical development work[3, 11–13] and that no consensus on what constitutes a method has so far been reached.[10, 14–16]

This study deepens our understanding of earlier studies concerning the role of methods and summarizes four case studies performed in an international ICT company, reported in[17–20]. These in-depth studies analyze two systems development projects within a company, which develops mobile services for the global and domestic telecommunication markets. The observations of our studies suggest that working with methods is more about social interaction and mutual understanding between project participants than progressing according to milestones and strictly following the prescribed phases of the method. In our case studies, successful use of system development methods required good communication between project participants to adopt the method to the development situation at hand. Without this communication the methods could not be used. When communication

* South Carelia Polytechnics, Koulukatu 5 B, FIN-55120 Imatra, Finland, paivi.ovaska@scp.fi.

was successful, the methods were used for learning to understand the system and its requirements. Without a common understanding of the system between project participants, the estimation of project timetable and resources became unrealistic and caused schedule overruns and wrong timing of resources.

The paper is structured as follows. Section 2 describes related research on the role of methods in the literature and the terminology used in this context. Section 3 presents the organization and the projects studied. In the next section (Section 4) a summary of the case study narratives is presented. Section 5 presents the results of our study. In Section 6, we discuss the findings of our study as well as its implications for research and practice. Finally, we offer conclusions.

2. DEVELOPMENT METHODS IN LITERATURE

2.1. The Role and Use of Methods

The majority of system development methods have a common set of concepts, which originate in the structured techniques of the 60's and 70's.[15] These concepts include life-cycle thinking,[11] design strategies such as functional decomposition[21] and information hiding.[22] It can be argued that even object-orientation and the definition of the Simula programming language[15, 23] can be traced to this period. Development practitioners tend to be quite conservative in their selection of methods, with methods from the 70's and 80's still dominating in practice.[14] Some studies argue that later methods, such as the object-oriented ones are little used in practice.[14] In contrast, some studies suggest that object-oriented methods are widely used in practical organizations and they are becoming more common.[14] Recently, agile methods in general and Extreme Programming (XP) in particular have gained much following among system development practitioners. Agile methods were developed as an alternative for the complicated traditional methods. Agile methods emphasize communication, development speed, lighter documentation and team effectiveness.[24, 25] However, no empirical evidence of their practicability or effectiveness exists.[25]

There are only few reports on the use of methods in practice.[8, 10, 14–16, 26] Some system developers claim to use no methods, others state that they tend to use a part of a method rather than following all the steps required by a particular method.[3] Many organizations have developed their own methods[10] to better meets the needs of the organizational environment. One of the better known examples of this kind of in-house method is Nokia's OMT++, which is an enhanced version of OMT for designing network management systems for mobile phones.[27] No evidence exists whether these in-house methods are used more successfully than textbook methods.[2] Studies of in-house method development indicate that the selection of methods, their development and their introduction is done in an ad-hoc manner, with no control over the adaptation.[8, 28] Nandakumar and Avison[16] go as far as to say that methods are "treated primarily as a necessary fiction to present an image of control or to provide a symbolic status in today's organization". In contrast, Undhelkas and Mandapur[29] propose a metaphor of a "road map" for development methods, suggesting that methods may not be able to recognize all situational factors and are more useful for a "foreigner" than for a "seasoned" practitioner.

As a reason for the limited use of methods it has been suggested that they do not address the most troublesome aspects of development, especially the thin spread of application domain knowledge, and conflicting requirements and breakdowns of the communication process.[11] Further, they are too mechanistic and detailed,[16] do not take different development situations into account,[26] and do not consider individual creativity, intuition and learning over time.[26] Also, by using methods one might lose sight of the fact that the real objective is the development of an actual system.[26]

Previous studies have reported method usage rates from 62 to 87 percents.[8, 26] These studies assume that if a method is not followed in detail, it is not used at all.[26] This does not mean that the method does not have any influence on the development. Some studies suggest that IS development in practice is not done according to any formalized methods, but rather providing learning methods and guidelines for participants on how to organize their work.[30, 31] Cockburn argues that methods reside in the tacit understanding shared by the participants, and in their habits of conversation.[24] Obviously, researchers and practitioners interpret the use methods differently.[10, 14, 32] There is also unfortunate terminological disagreement on method and methodology, which are discussed in the next section.

2.2. Terminology and Concepts of Methods and Their Use

The simplest definition of a method is "the way of doing things",[33] which is a popular definition among system development practitioners.[10] Divergent opinions on what constitutes a method, what is methodology and how these concepts are to be used, is very confusing.

First, there is the method versus methodology debate. The use of these terms has been confused.[2, 10] Methodology is used as the study of methods, but also as a synonym for method.[2, 10]

Second, various definitions of method exist. Dictionaries define the term "method" as "the procedure of obtaining an object"[34] and emphasize the process rather than the product. In contrast, Wijers[35] notes that text-book methods in IS focus on feasible specifications of products rather than on the process of developing such specifications. Table 1 shows the most widely used definitions of method.

Third, there is confusion about what constitutes a system development method and a software development method. Generally, software development method refers to the design and production of software while systems development method involves more widely people and organizational aspects.[10]

Fourth, the use of methods has been regarded from a number of points of view.[14] The use of methods can be seen from the users' perspective: a number of people in different roles use a method, users can be developers, project managers and quality managers.[14, 36] Method use can be seen also from the perspective of the explicitness of use varying from defined use (e.g. company standard enacted by a CASE tool) to implicit use as an influential determinant of the way of working.[14]

Use of methods may also play several different roles in systems development. Table 2 shows examples of the roles methods can play according to the existing literature.

Table 1. Examples of widely used definitions of systems development method

Source	*Method definition*
Hirschheim et al. [37]	An organized collection of concepts, methods, beliefs, values and normative principles supported by material resources
Avison and Fitzgerald [38]	A collection of procedures, techniques, tools and documentation aids which will help the systems developer in their efforts to implement a new information system. A method will consist of phases, themselves consisting of subphases, which will guide the systems developers in their choice of the techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate information systems project. It is usually base on some 'philosophical' view.
Wynekoop and Russo [10]	A systematic approach to conducting at least one complete phase (e.g. requirement analysis, design) of systems development, consisting of a set of guidelines, activities, techniques and tools, based on a particular philosophy of systems development an the target system
Rumbaugh [39]	1) a set of fundamental modeling concepts to capture semantic knowledge about a problem and its solution 2) a set of view and notions for presenting the underlying modeling information 3) a step-by-step iterative process for constructing model and implementation of them 4) a collection of hints and rules of thumb for performing development
Smolander, Tahvanainen and Lyytinen [2, 28]	A predefined and organized collection of techniques (product aspect) and a set of rules which state by whom, in what order, and in what way the techniques are used (process aspect)
Jayaratna [5]	An explicit way of structuring one's thinking and actions. Methods contain model(s) and reflect particular perspectives of 'reality' based on set of philosophical paradigms. A methods should tell you 'what' steps to take and 'how' to perform those steps but most importantly the reasons 'why' those steps should be taken, in that particular order.

Table 2. Examples of systems development methods roles. Adopted from[14]

Purpose to serve as a	*Definition*	*General examples*	*Systems development*
Constitutive rule [14, 40]	To determine action	Rules of games	Essence of a specific systems development approach (e.g.OO)
Regulative rule [14, 40]	To regulate action	Traffic rules	When including standards of systems development
Resource [14, 41]	To support action	A plan of canoeing a rapids	Method as a resource in project planning and execution
Reminder [14, 42]	To remind about the action	A shopping list	Method as a list of systems development tasks
Model for the ideal process [14, 43]	To serve as a model of the ideal process even though it cannot be followed in practice	After-the-fact reconstruction of the argumentation	Systems documentation as the reconstruction of the ideal development process
Vehicle of learning [14, 44]	To serve as a model on which the practice can be contrasted	Scientific theories and hypotheses to be tested	A method as a hypothesis of effective systems development process

3. THE CASE STUDIES

In this section we describe the organization studied and two case projects chosen to this study.

3.1. Case Organization

This study was performed in an international ICT company. The development of new applications and services were the responsibility of in-house software development unit or outsourced companies. The in-house software development unit was divided into three departments at different geographical locations in Finland. The coordination between the different sites was planned to be carried out by using common processes and written specifications. In order to support informal communication across the sites, computer-mediated communication devices such as video conferencing, Internet Relay Chat channels, shared calendars and electronic mail were in place.

The use of an in-house software development unit for development of services was mandated by the company's top management. The in-house software development unit employed approx. 150 persons. The unit's work had previously focused on R&D work in the company. During the past few years, it had developed its software development skills and its own in-house methods to make the systems development more effective and also to prove their capability to the company's business units. Despite in-house software development unit's effort on method development, the business units of the company were nor convinced of its competencies and methods. Quite often business units still preferred outsourcing instead of developing in-house.

3.2. Case Projects

The customers of both the projects were internal ones. The projects took place during the year 2001 and were planned and organized traditionally according to a waterfall model with distinct requirement elicitation, analysis, software design, implementation and testing phases. Both of these projects were developed based on an object-oriented approach and followed principles of a well known object-oriented system development method, namely UML.[45] The methods were intended to be followed mainly as an ideal process (Table 2), although they were intended to be adapted to the need of each individual project. The methods did not give any guidance for the adaptation, the same phenomenon found in[8]. The object-oriented method was selected by choosing one of the well known "text-book" methods and combining this with a waterfall process model. In both projects, at least parts of the systems were developed at three different sites in Finland.

The projects (EC project and DS project) developed mobile services to the global and domestic telecommunication markets. A goal of both projects was the renewal of old platform architecture to allow the services to be better and easier modifiable, maintainable and scalable. The EC project developed an Electronic Commerce mobile service platform. The system was intended to enable organizers or their sponsors to promote their products in all kinds of events, such as ice hockey and football games. The system was composed of two subsystems: the platform in which the services are run (Platform subsystem) and the toolbox (Tool subsystem). The Tool subsystem allowed adding, configuring and simulating

Table 3. Characteristics of the studied projects

Characteristic	*DS Project*	*EC project*
Project start and end dates	Start date 3.4.2001 End date 5.12.2001	Start date 23.3.2001 End date 16.10.2001
Software size (Line Of Codes, LOC)	138 000 LOC	32 000 LOC
Project cost	In total 3500 Man Days, design and implementation phase 2900 Man Days	In totally 1217 Man Days, Execution Phase 780 Man Days
Architecture	Distributed Server and centralized Client	Platform and Tool
Projecting	Divided into two subprojects, same-site (Server) subproject and multi-site (Client) subproject	One project, but two subsystems developed quite separately
Projet goal	The renewal of old architecture	Platform enhancement, content administrator
Targeted markets	International	International

the services and was intended to run in a Windows PC and the service platform in an UNIX environment.

The DS project developed a directory service platform for international markets. The project was partitioned into two subprojects to facilitate easier management. Partitioning was carried out on the basis of the architecture and technology: one subsystem had a highly distributed, component-based architecture (Server) and the other was a centralized subsystem (Client), which handled authentication, authorization and user interfaces. The functionality of the services required subsystems to communicate only through an extensible and configurable interface. Table 3 represents some characteristics of these projects.

4. SUMMARY OF THE CASE NARRATIVES

In this section we highlight issues and problems of method use observed in both case study projects. These descriptions are summary of the narratives from the project, described in more detail in[19, 20]. These narratives were formulated based on the qualitative analysis using grounded theory approach.[46]

4.1. DS Project: Communication and Coordination[18, 19]

At the beginning of the DS project, software architecture was regarded as an important method in the coordination of distributed software development. The allocation of the development work was performed according to the software architecture. The software development method in the company was also thought to guarantee coordination. During the project, it became apparent that the actual allocation of work did not follow the plans and

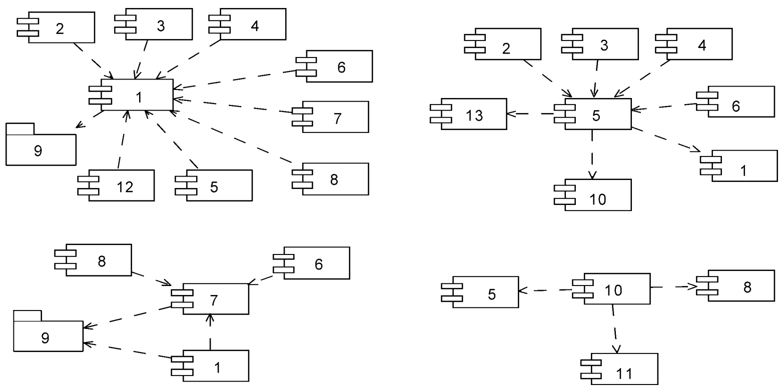


Figure 1. Four different interpretations of the architecture in DS project. The same components are marked with the same number.

the method did not guide the work at all. Every designer saw the architecture of the system and its component dependencies differently (Figure 1).

Communication barriers and cultural differences caused by the architect's one-sided approach to project activities in the multi-site project hindered a common understanding of the architecture of the other subsystem. In the Server subproject, which was completely carried out at one site, the informal communication between project participants allowed a shared understanding of the architecture to be developed.

In the later phases of development, the lack of common understanding of the architecture caused many coordination problems in the multi-site subproject. These problems complicated the realization of the actual. The developers had to do many changes to the interfaces and components before the integration of the components into a working system succeeded. These problems surfaced in the testing phase, as the integration problems made running the tests impossible.

4.2. EC Project: Communication and Understanding Requirements[17, 20]

In the beginning of the EC project, many communication problems and a number of disagreements about project goals, used development methods and strategies emerged. A small user interface part was neglected in the beginning, but grew bigger in during the course of the project. The project participants followed the in-house development methods strictly because they needed guidance for their development decisions. The line organization managers based their resource decisions also on the methods. The strong reliance on methods, the disagreements between participants and the communication problems seemed to inhibit the understanding of the user interface part of the system and its requirements. The understanding of user interfaces improved when the two technical persons involved in the project discovered that this former small user interface part was not only comprised of few user interfaces, but it was also a separate subsystem with its own functionality. This understanding increased radically when the user interface designer and another designer joined the project and the project moved towards more iterative development and conver-

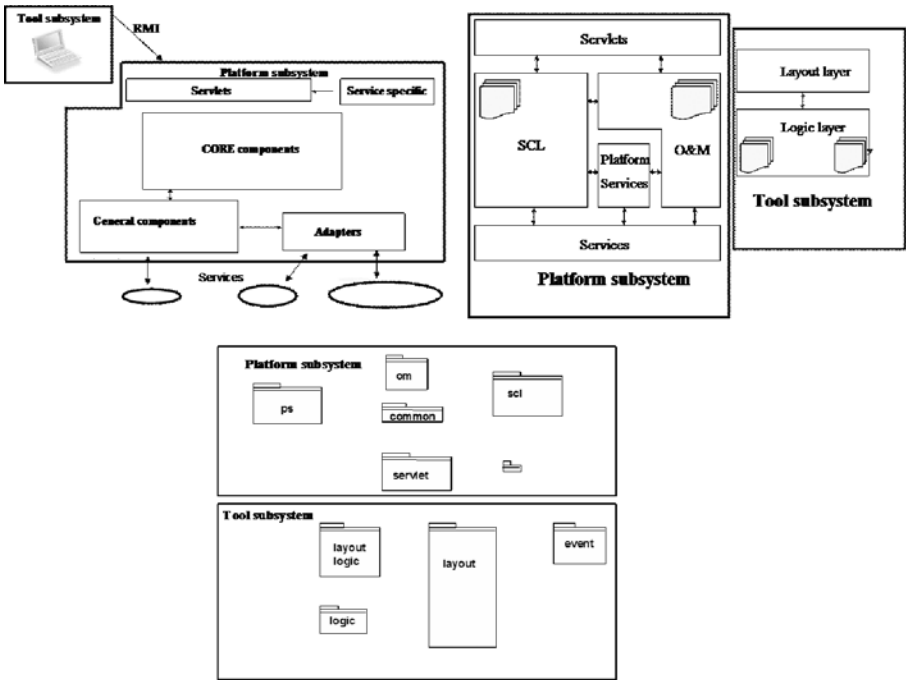


Figure 2. Three models of the system in the EC project. Analysis model resides in the left upper corner, design model in the right upper corner and implementation model in the middle down. The increased understanding of Tool subsystem requirements can be seen as crowing size of the Tool subsystem 'box'.

sations between the customer and the developers to better satisfy the customer needs and expectations.

Figure 2 shows the requirement evolution process during the project in a form of three conceptual models of the system (analysis, design and implementation models).

Although there were problems in the beginning of the project, in the later phases the project group's ability to solve the disagreements and turn the development towards a more iterative way made the actual system a success. The final system met customer requirements well and the customer was satisfied with the result. The development still required more time than initially estimated and the schedule of the project exceeded the original plan by more than 40%. The requirements in the other subsystem changed repeatedly throughout the process causing delays. The original time schedule had been planned according to the initial requirement specification. Therefore, project managers were unable to estimate the new time budget during the development phase. Nobody involved in the project could see the current state of the project not having a good enough understanding of the system and obvious changes to the project schedule came always as a surprise to the majority of the participants. Changes were made eight times during the second half of the development phase, and every time two weeks were added to the schedule. As a reason it was always stated that the Tool subsystem was not ready.

5. OBSERVATIONS

5.1. The Use of Methods

The case study descriptions above indicate that the use of methods in the company required a lot of communication between participants. Despite the development of in-house methods, the methods did not guide the participants in the various project situations, nor was there training on identifying situations where deviations from the method are necessary. Without this communication the method failed. This happened in the multi-site DS subproject causing coordination problems in the later phases of development. In the same-site DS subproject, the designers were experienced and "knew what to do". Still the methods were used for communicating architectural decisions between designers. When in the EC project communication succeeded, the understanding of the system improved. The understanding of the system arose mainly from the communication between participants, not from the system itself, nor from the descriptions alone. This understanding was a result of a learning process, where system development participants were able to change their understanding of the system according to changing development situations. Estimation difficulties arose in both projects because the participants misunderstood the system and development situation.

5.2. The Role of Methods

During the analysis, it became evident that the role of methods in the practical work varied according to the position and background of the participant. For example, managers and designers used methods in their work for different purposes. Managers used methods for project planning purposes, estimating project resources and timetable, whereas less experienced designers used methods more to learn the system, its requirements and development practices. The more experienced designers used methods mainly for communicating architectural decision. The role of methods from project stakeholders perspective are summarized in the next table (Table 4).

6. DISCUSSSION

Our observations have several implications for the systems development research by suggesting that methods are not followed strictly, but used more as resource in project plan-

Table 4. Role of methods from method user's perspective

Methods user environment	*Role of method*
Emphasized by managers	Estimating project resources and timetable
Emphasized by designers, with less experience	Learning to understand the system, it's requirements and development practices
with more experience	Communicating to understand the system and it's requirements
with less experience, unable to communicate	Mainly not using methods
Strongest among customer, important for all	Communicating to adopt to the development situation Understanding the system and it's requirements

ning and a vehicle of communication and learning. Our observations support the findings of the limited method use.[8, 10, 14–16, 26] We extend this by suggesting that working with methods is more social interaction and mutual understanding than progressing according the milestones and following method stages ín a strict manner. Iivari and Maansaari[14] summarize the method use besides strict reliance of methods also as rules, resource, reminders and vehicle for learning. Some of these are in line with our observations, namely resource in project planning[41] and vehicle of learning,[44] also observed in[30, 31]. Unhelkas and Mandapur[29] propose that methods are more useful for inexperienced practitioners than experienced ones, but our findings do not support this. In our case studies, more experienced designers used methods to communicate and understand the architecture of the system in the same way. This observation supports Cockburn's[24] argumentation that although methods are not explicitly used, they still reside in the tacit understanding held between participants, and in their habits of conversation.

As illustrated in Section 4, the methods' role in development work is slightly different than many of the currently available methods assume. Our study emphasizes the communication and social interaction between project stakeholders in adopting the methods to the development situation at hand. Systems development is not a rational process but depends on the circumstances and is more an 'ad hoc' improvisation process towards a working system. Methods serve as resources for systems development but do not determine its course. Suchman[41] calls this kind of activity situated action in the area of human - machine communication. Thus, there is a need to update the situation by developing methods better suited for the needs of the current development environment, also recommended in[15]. The following highlights some issues that should be taken into account in the derivation of new development methods:

- Our study emphasizes the communication and social interaction between project participants. The methods should support communication, not be mere tools and techniques for communication, but also serve as a boundary object to which several stakeholders associate their particular meanings.[11, 47]
- As our studies indicate, the development lifecycle model and development strategy is situation dependent. Neither waterfall model nor prototyping and iterative development are suitable for every situation. Methods should also guide the development in multi-site environments with geographically and culturally dispersed teams. This means that methods should rather be guiding practices in different situations than phases determining the course of systems development.
- As we observed from the EC project, factors can emerge in business or development environment that keep the participants from understanding the system and its requirements. The important role of methods is to guide how these factors can be identified.

7. CONCLUSIONS

In this paper, we have sought to illustrate the use of system development methods by drawing on the insights gained from two case studies on system development method use and adaptation in an international ICT company.

The findings extend the earlier understanding of method usage by suggesting that working with methods is complex social interaction and a mutual understanding process between project participants. The methods were used in our case study as a *vehicle of communication* and *learning* and as a *resource in project planning* among system development participants, not as a list of phases to be followed in detail. Method use requires communication between project participants to adopt the method to the development situation. Without communication, methods use failed. When participants gained common understanding of the appropriate development strategy and method, the method use became a learning process. In this learning process, system development participants changed their understanding of the system, according to changing development situations. Without the common understanding of the system between project participants the estimation of project timetable and resources did not succeed. Based on the observations of our study, we argue that methods are used more as support and guide the development than strict phases to be followed in detail.

While these in-depth studies enable us to gain insights into the complex social interactions in working with methods, it also means that we must be cautious about generalizing from these studies concerning only one company. The understanding gained in this study provides a basis for understanding similar phenomena in similar settings, rather than enabling us to understand phenomena in different contexts. It also provides a basis from which we can continue the study of the methods' role in systems development.

REFERENCES

1. K. Lyytinen and D. Robey, Learning Failure in Information System Development, *Information Systems Journal* **9**(2), 85–101 (1999).
2. J.-P. Tolvanen, Incremental Method Engineering with Modeling Tools – Theoretical Principles and Empirical Evidence, in: *Department of Computer Science and Information Systems* (University of Jyväskylä, Jyväskylä, 1998), p. 301.
3. B. Fitzgerald, Formalized systems development methodologies: a critical perspective, *Information Systems Journal* **6**(1), 3–23 (1996).
4. D. Avison and B. Fitzgerald, *Information Systems Development Methodologies: Techniques and Tools* (Blackwell, Oxford, McGraw-Hill, 1995).
5. N. Jayaratna, *Understanding and Evaluating Methodologies* (McGraw-Hill Book Company, 1994).
6. R. Hirschheim, J. Iivari, and H. Klein, A Comparison of Five Alternative Approaches to Information Systems Development, *Australian Journal of Information Systems* **5**(1), 3–29 (1997).
7. D. Avison, Information systems development methodologies: a broader perspective, in: *Method Engineering: Principles and method construction and tool support*, edited by S. Brinkkemper, K. Lyytinen, and R. Welke (Chapman & Hall, 1996), pp. 263–277.
8. N. Russo and J. Wynekoop, The Use and Adaptation of System Development Methodologies, in: *Managing Information & Communications in a Changing Global Environment: Proceedings of the Information Resources Management Association International Conference*, edited by M. Krosrowpour (Idea Group Publishing, Atlanta, 1995), p. 162.
9. K. Lyytinen, A taxonomic perspective of information systems development: theoretical constructs and recommendations, in: *Critical issues in information systems research*, edited by H. Boland (John Wiley & Sons Ltd., 1987), pp. 3–41.
10. J. Wynekoop and N. Russo, Systems development methodologies: unanswered questions, *Journal of Information Technologies* **10**, 65–73 (1995).
11. B. Curtis, H. Krasner, and N. Iscoe, A Field Study of the Software Design Process for Large Systems, *Communications of the ACM* **31**(11), 1268–1287 (1988).

12. D. G. Wastell, The fetish of technique: methodology as a social defence, *Information Systems Journal* **6**, 25–49 (1996).
13. L. Introna and E. Whitley, Against method-ism:exploring the limits of method, *Logistics Information Management* **10**(5), 235–245 (1997).
14. J. Iivari and J. Maansaari, The usage of systems development methods: are we stuck to old practices?, *Information and Software Technology* **40**, 501–510 (1998).
15. B. Fitzgerald, Systems development methodologies: the problem of tenses, *Information Technology & People* **13**(3), 174–185 (2000).
16. J. Nandhakumar and D. Avison, The fiction of methodology development: a field study of information systems development, *Information Technology & People* **12**(2), 176–191 (1999).
17. P. Ovaska, Measuring Requirement Evolution – A Case Study in the E-commerce Domain, in: *International Conference in Enterprise Information Systems, ICEIS 2004* (Porto, Portugal, INSTICC-Institute for Systems and Technologies of Information, Control and Communication, 2004), pp. 669–673.
18. P. Ovaska and A. Bern, Architecture as a Predictor of System Size – A Metaphor from Construction Industry, in: *The 16th International Conference on Advanced Information Systems Engineering, CAISE' 04 Forum* (Latvia, Riga, Riga Technical University, Faculty of Computer Science and Information Technolgy, 2004), pp. 193–203.
19. P. Ovaska, M. Rossi, and P. Marttiin, *Architecture as a Coordination Tool in Multi-Site Software Development*, accepted to Software Process: Improvement and Practice, 2004.
20. P. Ovaska, On the Organizational Factors in the Understanding of IS Requirements, in: *26th Information Systems Research Seminar in Scandinavia* (Porvoo Finland, 2003).
21. C. P. Gane and T. Sarson, *Structured Systems Analysis: Tools and Techniques* (Prentice Hall, 1979).
22. D. L. Parnas, On the Criteria To Be Used in Decomposing Systems into Modules, *Communications of the ACM* **15**(5), 330–336 (1972).
23. C. Nygaard and O.-J. Dahl, Simula: an ALGOL-based simulation language, *Communications of the ACM* **9**(9), 671–678 (1966).
24. A. Cockburn, Agile Software Development: A Cooperative Game, in: *The Agile Software Development Series*, edited by Cockburn and Hicksmith, 2000.
25. P. Abrahamsson, et al., New Directions on Agile Methods: A Comparative Analysis, in: *25 th International Conference on Software Engineering* (Portland, Oregon, 2003), p. 244.
26. B. Fitzgerald, An empirical investigation into the adoption of systems development methodologies, *Information & Management* **34**, 317–328 (1998).
27. J.-M. Aalto and A. Jaaksi, Object-Oriented Development of Interactive Systems with OMT++, in: *Technology of Object-Oriented Languages and Systems (TOOLS 14)* (Prentice-Hall, 1994), pp. 205–218.
28. K. Smolander, V.-P. Tahvanainen, and K. Lyytinen, How to Combine Tools and Methods in Practice – a Field Study, in: *Second Nordic Conference CAISE '90* (Stockholm, Sweden, Lecture Notes in Computer Science, 1990).
29. B. Unhelkas and G. Mandapur, Practical aspects of using methodology: A road map approach, *Report of Object Analysis and Desing* **2**(2), 34–36, 54 (1995).
30. L. Mathiassen, et al., Method Engineering: Who's the Customer?, in: *Method Engineering. Principles of Method Construction and Tool Support*, edited by S. Brinkkemper, K. Lyytinen, and R. Welke (Chapman & Hall, London, 1996), pp. 232–245.
31. R. Baskerville, J. Travis, and D. P. Truex, Systems without method: The impact of new technologies on information systems development projects, in: *Transactions on the impact of computer supported technologies in information systems development*, edited by K. E. Kendall, K. Lyytinen, and J. I. DeGross, (1992), pp. 241–260.
32. J. Bubenko, Information system methodologies – a research view, in: *Information Systems Design Methodologies: Improving the practice*, edited by T. W. Olle, H. G. Sol, and V.-S. A. A (Elsevier Science Publishers B.V., 1986), pp. 289–312.
33. I. O. Angell and B. H. Straub, Though this be madness, yet there is method in't, *Journal of Stratetic Information Systems* **2**(1), 5–14 (1993).
34. R. Baskerville, Structural Artifacts in Method Engineering: The Security Imperative, in: *IFIP TC8 Working Conference on Method Engineering: Principles of method construction and tool support* (Great Britain, Chapman & Hall, 1996), pp. 8–28.
35. G. Wijers, A. ter Hofstede, and N. van Oosterom, Representation of information modeling knowledge, in: *Next Generation of CASE tools*, edited by K. Lyytinen and V.-P. Tahvanainen (IOS Press, Amsterdam, the Netherlands, 1992), pp. 167–223.

36. B. Henderson-Sellers, Who need an object-oriented methodology anyway?, *Journal of Object-Oriented Programming* **8**(6), 6–8 (1995).
37. R. Hirschheim, H. Klein, and K. Lyytinen, *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations* (Cambridge University Press, Cambridge, 1995).
38. D. Avison and B. Fitzgerald, *Information Systems Development: Methodologies, Techniques and Tools*, 2nd edition (McGraw-Hill, New York, 1995).
39. J. Rumbaugh, What is a method?, *Journal of Object-Oriented Programming* **8**(6), 10–16 (1995).
40. R. Searle, *Speech Acts, An Essay in the Philosophy of Language* (Cambridge University Press, Cambridge, 1969).
41. L. Suchman, *Plans and Situated Action* (Cambridge University Press, Cambridge, 1987).
42. P. Ehn and M. Kyng, The Collective Resource Approach to Systems Design, in: *Computers and Democracy*, edited by G. Bjerkenes, P. Ehn, and M. Kyng (Avebury, Aldershot, 1987), pp. 17–57.
43. D. L. Parnas and P. C. Clements, A rational desing process: how and why to fake it?, *IEEE Transactionson Software Engineering* **2**, 251–257 (1986).
44. P. Checkland and J. Scholes, *Soft System Methodology in Action* (Wiley, Chichester, 1990).
45. I. Jacobsson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process* (Addison-Wesley Professional, 1999).
46. B. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research* (Adline, Chigago, 1967).
47. G. C. Bowker and S. L. Star, *Sorting Things Out: Classification and Its Consequences* (MIT Press, Cambridge, MA, 1999).

JMINING – INFORMATION DELIVERY WEB PORTAL ARCHITECTURE AND OPEN SOURCE IMPLEMENTATION

Algirdas Laukaitis, Olegas Vasilecas, and Raimondas Berniunas*

Abstract In this paper we present a new framework and its Java based open source implementation for building web information delivery portals (IDP). The framework provides the necessary infrastructure for development and management of web portal using only web browser interface. Presented framework enables speedup of information delivery portal development compared with some commercial available solutions. All presented components are implemented as separate open source project JMining (VGTU Java open source project). The contribution of this paper is threefold: Firstly, we introduce a IDP framework based on atomic application objects container. Proposed framework is characterized by its simplicity and a possibility to build complex information delivery web portals. Secondly, we show new way of deploying applications into portal using only web browsers. We discuss new requirements for web browsers and show that up today only few of them meets them. Thirdly, all presented concepts are implemented as Java open source project. We present discussion about open source projects and importance for support for such projects from academic environment. Our research shows that until now there was no open source project in information delivery portals domain and we think that our project can fulfil such gap.

Keywords: Information delivery web portals, data analysis visualization objects, XML, OLAP, databases web interface, Java, open source software

1. INTRODUCTION

Data environments are becoming more and more complex as the amount of information a company manages continues to grow. Information delivery web portals have emerged

* Vilnius Gediminas Technical University (VGTU), Sauletekio av. 11 LT-2040, Vilnius, Lithuania, algirdas@isl.vtu.lt, olegas@fm.vtu.lt, r.berniunas@one.lt.

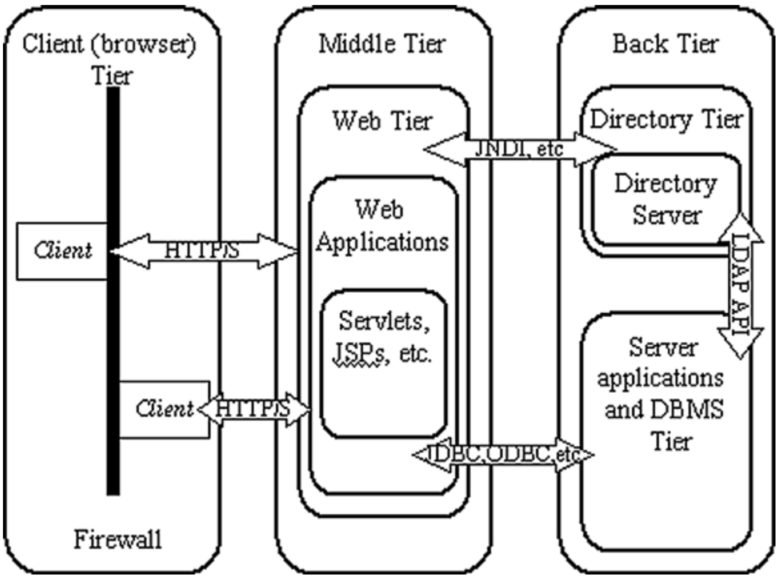


Figure 1. Three tier JMining architecture.

as the preferred way to bring together information resources. Using information delivery web portal, your organization's employees, customers, suppliers, business partners, and other interested parties can have a customized, integrated, personalized, and secure view of all information with which they need to interact.

There are many commercially successful information delivery web portal products that are available in the market. Figure 1 presents architecture of IDP implemented by our project JMining[8] and similar tree tier architecture is implemented by many IDP providers. We have no intention to describe this architecture in details and for details about our implementation as open source project JMining we refer to[8] or SAS,[12] Oracle,[10] Microsoft,[9] Information builders,[7] etc for details of some commercial implementations. Instead, in Section 2 of this paper we describe architecture of middle tier that is based on atomic applications container. We show that for some category of users our approach can be more preferential than some commercial alternatives. A fundamental idea behind proposed architecture is the way atomic applications are deployed into web server environment. In this section we discuss this novel way of web applications deployment. We show that our method can be more suited for some middle size IDP implementations compared with existing commercial alternatives.

In the past ten years, open source software has become one of the most discussed topics among software users and practitioners. The increasing interest in open source software has been motivated by several factors:

1. The success of products such as Linux (operating systems), Apache[1] (http servers, etc.), MySQL[4] (DBMS), GATE[3] (NL processing), Weka[13] (machine learning), etc.

2. The uneasiness about the Microsoft or Oracle monopoly in the software industry.

3. The increasingly strong opinion that "classical" approaches to software development are failing to provide a satisfactory answer to the increasing demand for effective and reliable software applications.[5]

The lack of open source project for information delivery web portal was one of the reason that triggered our project JMining. In Section 3 we describe some aspects of open source software development and the way we see our contribution in this software domain.

2. ATOMIC APPLICATIONS ARCHITECTURE AND DEPLOYMENT

The system for building information delivery portals that we developed we call JMining. Hereafter, we will use this name to describe the architecture of the system as well as the main objects.

JMining is implemented as database and platform independent. Data base system accessed by one of the following protocols (ODBC, JDBC or XML). The JMining is server-based application written completely in Suns Java programming language. Because the JMining modules are written in Java, they can run on any server platform that supports a Java Virtual Machine. Data used by the portal: account credentials, access controls, demographics, personalization parameters, and configuration information can be stored within an X500 directory services database accessible through LDAP (Light-weight Directory Access Protocol) or a simple user access control modules can be used as alternative way to control accounts and the other users personal settings. In Figure 1 can see three tiers of portal. For detailed description for each tier we refer to.[8]

In this section we will concentrate mainly on a fundamental idea in our architecture which we call atomic applications container. There where two main problems that we challenged by creating our framework JMining and its open source implementation:

1. Creating small business applications without programmers involved has been an increasing trend between knowledge workers. Such applications has been created with some PC based product from which Microsoft Access is the most popular. On the other hand when we dealing with sharing knowledge between corporate workers with the use of web portals, programmers and other IT stuff must be involved. If the knowledge worker want to share knowledge with other corporate employees in the form of software application it can specify business requirements and to acquire IT resources to build necessary web applications. The biggest drawback of such approach is a time of building necessary applications and the development cost involved. The problem is well understood by information delivery portal vendors and more sophisticated solutions are constantly provided. But most solutions provided requires PC based software components. After application is written by analyst on its PC he can involve some mechanisms of deploying application to web server. We have put the requirement in our architecture to remove PC based components and to achieve the way of doing analysis and sharing knowledge directly on the web server with the use only web browser.

2. The second problem is flexibility related of reporting objects templates. In most commercial products there is a set of generated templates and user has no possibility to extend this set. In the context of information delivery portals we separate two kind of templates. One is for knowledge worker and the second one for software developers. Templates

generated as the script of atomic application solution can be used by others knowledge workers. Programmers can write new Java objects that has a purpose of data visualization.

To solve the problems mentioned above we propose IDP framework based on so called atomic applications container. By atomic application we understand the small web application which contains following components: database script, user interface HTML page, data representation script (XML, XSL, etc.) and documentation page (additionally there is connection to DBMS parameters, name of the application, and parent name of the application to organise all atomic applications in one single directory structure). Atomic application structure in some way resemblance to well know web applications developing technologies like Servelets, JavaServer Pages (JSP) and Active Server Pages (ASP). With such technologies like JSP you can have the full power of general programming language like Java. But on the other hand it is unlikely that such technology can be handed by non-programmer or person without Java knowledge. On the other hand by putting more constraint on the web applications structure we achieved that nonprogrammer can successfully develop web applications. Surely that doesn't mean that no IT skills required. The user of our software actually is the user who previously used such products like Microsoft Access to develop some local based database applications. Such user mostly has a good understanding of a database model as well as some basic SQL knowledge (sure most often that is no need for the user to write SQL sentences, instead it is done by interactive software wizards).

Next we provide basic structure for atomic application.

Atomic application

Atomic application represents one of the basic classes. Object derived form the class (like a brick in the house) is used to build an enterprise information delivery web solution. As mentioned above the set of such atomic applications can bring full portal solution to some business subject. The goal for us was to find minimum number of components needed for such a class. Below we describe in details these components. Figure 2 presents the main components of the atomic application. We like to mention that the biggest implementation of information delivery portal based on proposed architecture at one of Lithuania companies has more than 1000 atomic applications that represents the information delivery portal for the 2000 employees.

The basic structure in JMining portal is the object with the following attributes.

SQL – set of SQL statements that are send to DBMS. There unlimited number of SQL statements that can be send to SQL server within one request but the last one must be SELECT type SQL statement. The reason for this is that system always tries to represent the last statement to user. Figure 3 shows the syntax of this module.

These statements are then executed in the selected database to retrieve information and to display it to the user through selected reporting template which can have graphical or textual formats. Also the users have the choice of modifying these SQL statements as well as reporting templates to create their own applications.

User interface HTML page – html document used to set user request parameters which can be used later to form dynamic SQL statements. Even if the primary intention of this parameter was to support dynamic SQL statements, it can be used as an independent HTML page for other web portal need. User has choice to keep parameter values perma-

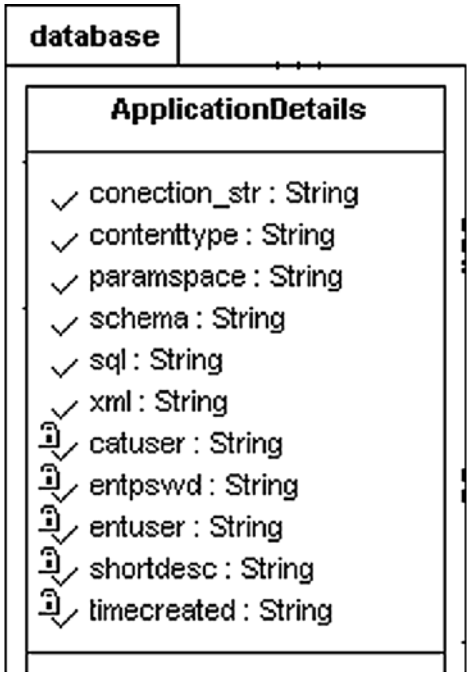


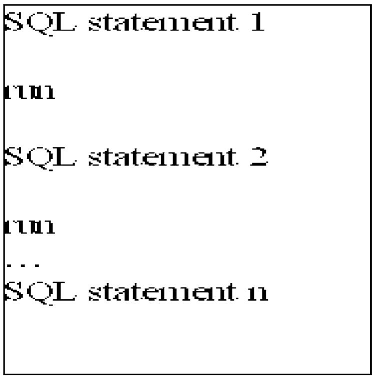
Figure 2. Part UML class diagram for the atomic application attributes.

nently to the end of internet session or just to the end of request implementation by web server.

Type of visualization object – used to choose selected data representation object from web server (e.g., graphic, bar char, some form of text (XML, HTML, TXT) layout, etc.).

XML (XSL). Extensible Markup Language (XML)[14] offers its users many advantages, including: simplicity, extensibility, openness. XML as the atomic application component is used as some script for data visualisation(e.g., it can say which column forms x or y axis in a graphic or which field represents grouping, total variables and how they must be presented in the HTML document, etc.). From DBMS selected data are parsed with statements that are extracted from XML document. If the data comes from XML document (it is common situation in organizations that some data now can be received from XML documents instead traditional of DBMS) XSL[15] document can be used to transform data to HTML format.

1. The proposed structure of atomic application is optimal in the following way: it contains the minimum number of components that are required for building complex web portal. Before building JMining system the project team has been involved into several projects to build web based applications using traditional Java techniques (servlets,JSP, Java beans). After analysis of developed solutions we found that separating data base scripts (SQL), HTML document and XML document can speedup web applications development. Proposed architecture is well suited when developer or analyst must introduce



```
SQL statement 1
run
SQL statement 2
run
...
SQL statement n
```

Figure 3. SQL module syntax.

some small changes into application. Changes made in one atomic application has no effect on the rest of atomic applications. The speedup of development is achieved by XML repository library as an developer can search for ready to use atomic application templates. We found that the big effectiveness can be achieved if some development process is done not by programmers but by data analysts. Proposed architecture is robust to some faults done by non professional programmers (bugs can effect only one atomic application but the whole system is unaffected).

2. One exclusive property of proposed architecture is that the whole development and deployment is done only through web browser interface. Developers has a huge feasibility and mobility by choice of the platform. Security level is the same as in most web based e-commerce applications.

3. One of the problems that faces data-intensive web-based information systems is that generating web pages on the fly can lead to severe performance problems.[2] We propose pre-generation of web pages, OLAP cubes and graphs. They are stored in server RAM area on the base of individual atomic applications. Developer of an individual atomic application can set the schedule which shows at what frequencies the results generated by atomic applications are stored in servers main memory. Such approach is well suited for pages or OLAP cubes that changes on the daily bases.

Note concerning web browsers

We must to mention that we have tested three browsers with the JMining: Netscape Navigator 7.1, Microsoft Internet Explorer and Opera. Until now Netscape Navigator has one bug and that makes Netscape Navigator not suitable for updating the atomic application management container. The rest of the JMining portal operations can be handled without problems with Netscape Navigator. Opera and Microsoft Internet Explore handle JMining interfaces without errors.

The problem with Netscape Navigator is that this browser loses some information that was entered in browser scrolling text box as another complex HTML document. That means that if you entered HTML document into HTML page scrolling text box, Netscape Navigator may lose some information.

3. JMINING AS OPEN SOURCE SOFTWARE

At the initial stage of our project we understood that to be successive in promoting our ideas the code of our project must be open source. One of the reason that computer scientist ignored open source as the way to develop information systems is a cost involved and lack of resources to backup new ideas and concepts with softwares realizations. On the other hand the success of open source software has led a number of researchers and experts to believe that open source might really be the answer to the software crisis.[5]

There are two groups of system developers that can participate by their delivery for JMining evolution. One group are system users themselves. By doing analysis and reporting and by changing generated templates in JMining system they can contribute they ideas for other users of the system in the for of atomic application templates.

The second one is Java programmers which can contribute to the system by writing additional components for data visualisation or data analysis. Such module like OLAP has been proposed and written by programmers outside initial project group. There is significant number of sites were professionals can contribute their modules and solutions for review and evaluation.[6] But until now there was no fully developed product for information delivered portals as free and open source software.

We hope that our paper will stimulate new research in this new software area. Finally we recommend for the readers of this paper constantly to check JMining official page[8] for changes and new research results.

REFERENCES

1. Apache Software Foundation; http://www.apache.org/.
2. P. Atzeni, G. Mecca, and P. Merialdo, Design and Maintenance of Data-Intensive Web Sites, in: *Proc. EDBT'98* (1998).
3. H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, and Y. Wilks, Experience of using GATE for NLP R/D, in: *Proceedings of the Work-shop on Using Toolsets References 200 and Architectures To Build NLP Systems at COLING-2000* (Luxembourg, 2000); http://gate.ac.uk/.
4. MySQL AB; http://www.mysql.com/products/mysql/.
5. A. Fuggetta, Open source software – an evaluation, *Journal of Systems and Software* **66**(1), 77–90 (2003).
6. Java review service; http://www.jars.com/.
7. Information Builders, Leveraging Your Data Architecture for Enterprise Business Intelligence, White Paper, (2004); http://www.informationbuilders.com.
8. Vilnius Gedimino Technical University, JMining project; http://193.219.146.140:8080/j_mining_info/index.html.
9. Microsoft corporation, Building a Corporate Portal using Microsoft Office XP and Microsoft SharePoint Portal Server, White Paper (2001).
10. Oracle corporation, Oracle9iAS Portal 3.0.9.8.2 Architecture and Scalability, White Paper (2002).
11. M. Schrefl, E. Kapsammer, W. Retschitzegger, and B. Proll, Self-maintaining web pagesan overview, *Proceedings of the 12th Australasian Database Conference (ADC)* (Queensland, Australia, January/February, 2001).
12. SAS corporation, SAS Information Delivery Portal, White paper (2000).
13. I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques with Java implementation* (Morgan Kaufmann, San Francisco, 2000).
14. World Wide Web Consortium, Extensible Markup Language; http://www.w3.org/XML/.
15. World Wide Web Consortium, Extensible Stylesheet Language; http://www.w3.org/Style/XSL/.

CONCEPTUAL FRAMEWORK FOR INTEGRATION OF MULTIAGENT AND KNOWLEDGE MANAGEMENT TECHNIQUES IN INTELLIGENT TUTORING SYSTEMS

Janis Grundspenkis*

1. INTRODUCTION

At present many approaches, methods, and technologies for education and training support already exist, and new ones appear very rapidly. Education and training is very broad area of research and development where ideas and solutions from different fields like pedagogy, psychology, multimedia, and information technology, etc. amalgamate. May be it is the reason why so wide spectrum of terminology is used. The following are only few more frequently used terms out of many others: online education, training or tutoring, computer based education, Internet augmented teaching, Internet or Web aided education, instruction or learning. The large number of terms confuses not only novices but also those who are working in the field. In (Anohina, 2003) an attempt has been carried out to clarify the terminology used in the field. The classification has been proposed that is based on two criteria: technological concepts (computer, distance, E-, Internet, online, resource based, technology, Web) and concepts taken over from pedagogy (education, instruction, learning, teaching, training, and tutoring). It is shown that boundaries between possible subfields marked with different terms are fuzzy and sometimes disappear at all. At the same time the described situation reflects great interest and activities in this field, manifests that it is relatively new and permanently changeable research and development topic, and shows that many relevant problems have not been solved yet.

Although today's learning settings are quite distinct from those of recent past and more distance education environments have been developed and new distance learning techniques and systems used, the experience obtained till now shows that learning effectiveness is still behind the desired level. One of the reasons is that intelligent support provided by these systems is far behind of that demonstrated by the human teacher who is able to adapt to each learner individually. In other words, distance education systems (even

* Department of Systems Theory and Design, Riga Technical University, 1 Kalku Street, LV1658, Riga, Latvia.

those that include, so called, *student model*), as a rule, are insufficiently adaptive to requirements of individual learners or their groups. Faster changes started with the development of WWW technology, when Web-based intelligent tutoring systems become the mainstream area of research and development (Yang et al., 2002). In result, new methodologies appear, for instance, (Traxler, 2002), and agent technologies are used to improve the quality of Web-based education (Johnson, 2003). In (Johnson, 2003) several animated agents, so called, guidebots are described, while in (Duh, 2001) a multiagent virtual seminar system (MAVIS) is proposed. More information about intelligent tutoring systems may be acquired by visiting several Web sites (see list at the end).

In this paper a novel conceptual framework for the development of intelligent tutoring systems based on integration of multiagent and knowledge management techniques is proposed. The implementation of the proposed conceptual model is going on now, and at the present moment only a prototype of two components has been implemented and tested.

The paper is organized as follows. In the second section a potential role of knowledge management in development of intelligent tutoring systems is described. The section contains discussion on notions of intellectual capital, corporate memory, its different forms (knowledge attic, knowledge sponge, and knowledge pump), and multilevel enterprise memory. The third section is devoted to the proposed conceptual model of intelligent tutoring system that has three layers – systems, multiagent, and knowledge worker's layers. Knowledge worker's layer is described in more details. Conclusions include the outline of future work and short overview of future intelligent agents and their potential impact on the evolution of intelligent tutoring systems.

2. THE ROLE OF KNOWLEDGE MANAGEMENT IN INTELLIGENT TUTORING SYSTEM DEVELOPMENT

In all education systems regardless of their kind (face-to-face, distance learning or hybrid) there are two groups of actors, namely, supervisors and students who are working with knowledge. These actors of the intelligent tutoring system are considered to be the *knowledge workers* supported by groupware technology to assist interacting group. Communities of knowledge workers are one of the four components of knowledge management architecture (Borghoff and Pareschi, 1998). According to the proposed conceptual model the group of students and a supervisor being part of intelligent tutoring system is embedded into a knowledge management system as it is shown in Figure 1.

There are several aspects that should be taken into account considering a potential role of knowledge management in distance education. *Knowledge management* is defined as a process through which organizations create, store, and utilize their collective knowledge (Sarvary, 1999). In other words, knowledge management is the formalization of and access to experience, knowledge, and expertise that create new capabilities and enable more effective performance of organizations. The main objective of knowledge management system's architecture is to provide an effective knowledge flow. Effective knowledge flow is strongly connected with knowledge sharing that enhances the learning capability both at individual and organizational levels.

The *knowledge management system* enables to turn information into action, that is, enables an effective learning process. It is obvious, that enhancing of the learning capability is

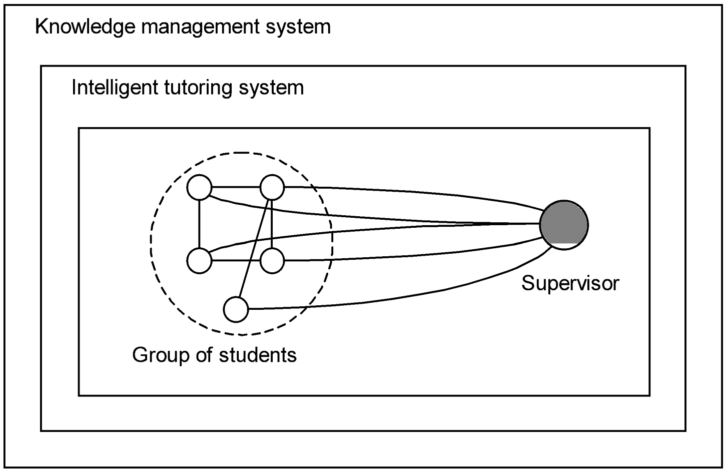


Figure 1. Intelligent tutoring system and its environment: knowledge management system.

one of the main goals of any tutoring system as well. The most relevant aspects of effective learning process are construction of knowledge, co-operation, and teamwork in learning and learning through problem solving. In other words, the knowledge management system supports expansion of individual's personal knowledge to the knowledge of the group as a whole. To achieve this goal, the intelligent tutoring system must become a learning organization that requires the capacity to expand individual's (supervisor's) personal knowledge and the ability to work in teams (student group). Knowledge management tools and techniques (knowledge environment) are effective technological solutions for knowledge creation (development, acquisition, inference, generation), knowledge storage (representation, preservation), knowledge aggregation (creation of meta-knowledge), use/reuse (access, analysis, application), and transfer (distribution, sharing). Moreover, knowledge management environment must contribute both personal knowledge and organizational knowledge as well. It is quite obvious that practically all mentioned aspects are important in teaching and learning process.

A closer look at the nature of knowledge helps to clarify why knowledge management plays so important role in the proposed conceptual framework. First, each educational organization to be competitive permanently must enhance its knowledge assets or at least must keep them at the needed level. Unfortunately, education organization very easily may loose its knowledge assets when some teachers are leaving the organization. To avoid loses (at least to the certain extent), organization must extend its *intellectual capital.* According to (Stewart, 1994) intellectual capital is intellectual material that has been formalized in some useful order, captured in a way that allows it to be described, shared, distributed, and leveraged to produce a higher valued asset. Intellectual capital has two major components (Koenig and Srikantaiah, 2000): information/knowledge capital and structural capital. Information and knowledge capital is the organization's information and knowledge that can be informal and unstructured as well as formal. The structural capital is mechanism to capture, store, retrieve, and communicate that information and knowledge, i.e., to take advantage of the information and knowledge capital. Knowledge capital, in turn, includes all the organization's tacit and explicit knowledge.

Different types of knowledge that education organization possesses and various possessors is the second factor why knowledge management may play an important role in the context of intelligent tutoring systems. The most popular is distinction between *tacit knowledge* and *explicit knowledge* proposed in (Nonaka and Takeuchi, 1995). Tacit knowledge is personal knowledge embedded in individual experience. It is shared and exchanged through direct, face-to-face contact and can be communicated in a direct and effective way. Explicit knowledge is formal knowledge that can be found in textbooks, documents, data and knowledge bases, etc. It is straightforward that users of intelligent tutoring system have tacit knowledge while the system itself captures explicit knowledge. From the learning perspective the distinction is between *built-in knowledge* and *self-acquired knowledge* (Kirikova and Grundspenkis, 2002). All knowledge possessors are divided into natural knowledge possessors (only human beings) and artificial knowledge possessors (Kirikova and Grundspenkis, 2000). In the context of this paper intelligent tutoring system is regarded to be an artificial knowledge possessor.

The third aspect showing the potential role of knowledge management in intelligent tutoring system development is the mode in which collection and retrieval of knowledge is performed. This aspect is closely connected with the notion of *corporate memory* defined in (van Heijst et al., 1998): a corporate memory is an explicit, disembodied, persistent representation of the knowledge and information in an organization. The simplest form of corporate memory management is called the *knowledge attic* that is characterized by passive collection and passive distribution of knowledge from the corporate memory. This type of corporate memory often is the most feasible in practice because it is not intrusive and emphasizes the bottom-up nature of learning. At the same time it requires a high discipline of the knowledge worker but rather frequently it is not a case even at the university. *Knowledge sponge* corporate memory is featured by active collection and passive distribution of knowledge. This case is very similar of that implemented in traditional distance learning systems during their evolution. The *knowledge pump* corporate memory is characterized by active collection and active distribution of knowledge. The ultimate goal of any intelligent tutoring system and also of any academic organization should be development, implementation, and maintenance of this type of corporate memory.

Typically knowledge management tools and techniques practice converting information to knowledge and connecting people to knowledge (Borghoff and Pareschi, 1998). The concept of the *multilevel enterprise memory* has been developed as a structure that can support the creation, preservation, storage, aggregation, use, reuse, and transfer (distribution, sharing) of the individual knowledge worker's knowledge, experience, and lessons learnt as well as intelligent tutoring system's knowledge (Grundspenkis, 2003). Due to the scope of this paper it is impossible to give details neither about the multilevel enterprise memory nor about well known technological components of knowledge management systems that are applicable for the intelligent tutoring system development. Let only point out that the multilevel enterprise memory is modelled as a sequence of seven phases of knowledge life cycle, namely, identification of knowledge sources, knowledge acquisition, formalization, representation, processing, application, and use.

3. CONCEPTUAL MODEL OF INTELLIGENT TUTORING SYSTEM

A conceptual model has three layers as it is shown in Figure 2.

At the system's layer traditional components of intelligent education systems are included, namely, the student model, the expert model, the domain knowledge base, the interface, and the tutoring module (Figure 3).

In fact, these components correspond to the learning agent architecture (Russell and Norvig, 2003). If one compares the learning agent architecture and the components of the system's layer it is easy to see that the interface plays the role of an agent's sensors and effectors, the domain knowledge base corresponds to the agent's built-in knowledge base, the expert model corresponds to the critic and the learning element, and the student model is equivalent with the learning and performance elements. The tutoring module to the certain extent corresponds to the problem generator but in the proposed conceptual model the planning and searching agents play an important role, too. The purpose of the planning agent (the domain planner) is to provide the appropriate learning plan. The searching agents are needed in case if several plans exist.

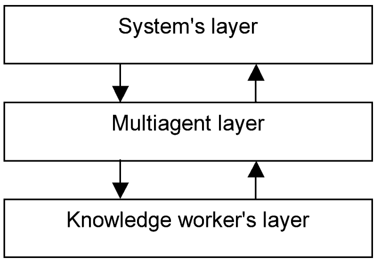


Figure 2. Three layers of the conceptual model.

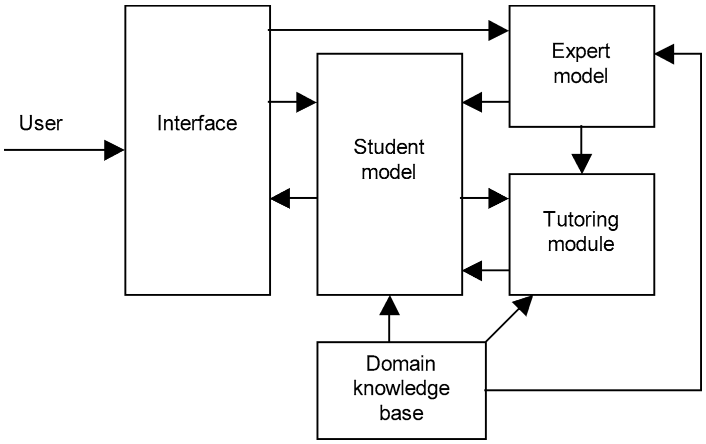


Figure 3. The architecture of system's layer.

At the *multiagent layer* the group of students and the supervisor are modelled as a multiagent system (Wooldridge, 2002). The teaching and learning process at the multiagent layer is considered to be cooperative work and communication of agents. Each actor (student) is modelled as a learning agent with reinforcement learning capability. The intelligent tutoring system offers a sequence of subtasks for students to solve. Teacher (critic) provides each learning agent with the reward (positive or negative) at the end of the learning session. One interesting future research perspective may be the swarm intelligence approach (Engelbrecht, 2002) to simulate evolution of interacting student group from the wheel structure that represents the student group when it starts the learning process to the star structure where each student communicates with all others using the intelligent tutoring system embedded in the knowledge management environment.

The *knowledge worker's layer* is the support layer of students and the supervisor involved in the teaching and learning process. Students and supervisors are supported by a plethora of intelligent agents. The knowledge worker is supported by agents that conceptually may be located in three circles (Grundspenkis and Kirikova, 2004). The internal circle is shown in Figure 4. Let discuss the potential role of different intelligent agents in the intelligent tutoring system.

The knowledge worker is surrounded with personal agents. Personal agents belong to humans, support human computer interaction, and help knowledge workers to acquire, process, and use knowledge. The most appropriate agents at this circle are *search*, *assistant*, *filtering*, and *work-flow agents* described in (Knapik and Johnson, 1998). The most commonly used are search agents and they work in different ways. Some agents search titles of documents or documents themselves, others search directories on the Web. Filtering agents may monitor the data stream searching the text for knowledge and phrases as well as the list of synonyms, and try to forward only the information that the user really needs. More advanced filtering agents can be trained by proving the sets of examples illustrating articles that users choose to read. Assistant agents are designed to wait for events such as E-mail messages to occur, then to sort them by sender, priority, subject, time, etc. Workflow agents are useful for task coordination, appointment, and meeting scheduling. In

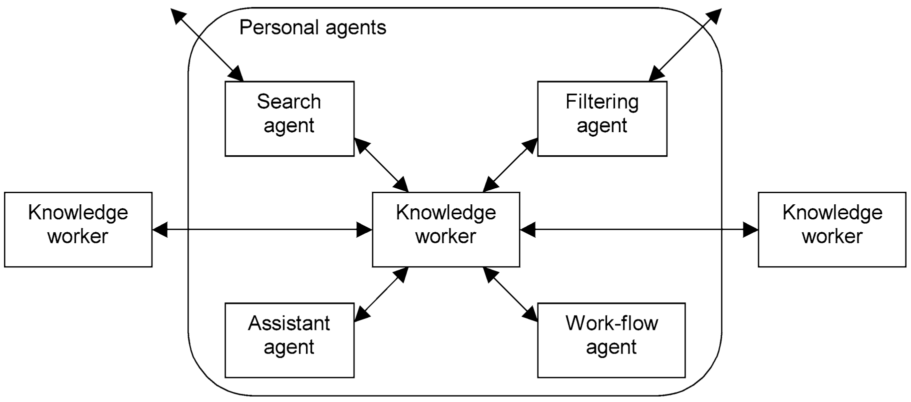


Figure 4. The internal circle of agents.

near future smart agents (Case et al., 2001) will appear that will be able to acquire, store, generate, and distribute knowledge. Search and filtering agents definitely must be included in any intelligent tutoring system based on agent paradigm. Search agents may search the domain knowledge base for knowledge needed at the particular learning stage (knowledge contents are defined by the tutoring module). Filtering agents may monitor the data stream and try to forward only information that the student really needs. Personal assistants and work-flow agents also may be included to extent intelligent support of teaching and learning process.

The medium circle in shown in Figure 5. Communications between individuals of the multiagent community is the most relevant issue for effective knowledge acquisition, sharing, and distribution. In the medium circle such communication management agents as *messaging*, *team*, *collaborative*, and *cooperative agents* are included. Messaging agents can connect students within the group and with the supervisor no matter where they are and what communication medium is used. Team agents facilitate communication in the group of students, while cooperative and collaborative agents are able to cooperate and to collaborate with filtering agents in the interior circle.

The external circle of agents is shown in Figure 6. There are agents for communication with external systems, for instance, *network agents*, *network software distribution agents*, *database agents*, *connection and access agents*, and *intelligent Web agents*.

Without any doubt, the most important role may play intelligent Web agents because nowadays the Web is the richest source of data, information, and knowledge that is needed in learning and is accessible for any user. Unfortunately, currently the Web contains a lot

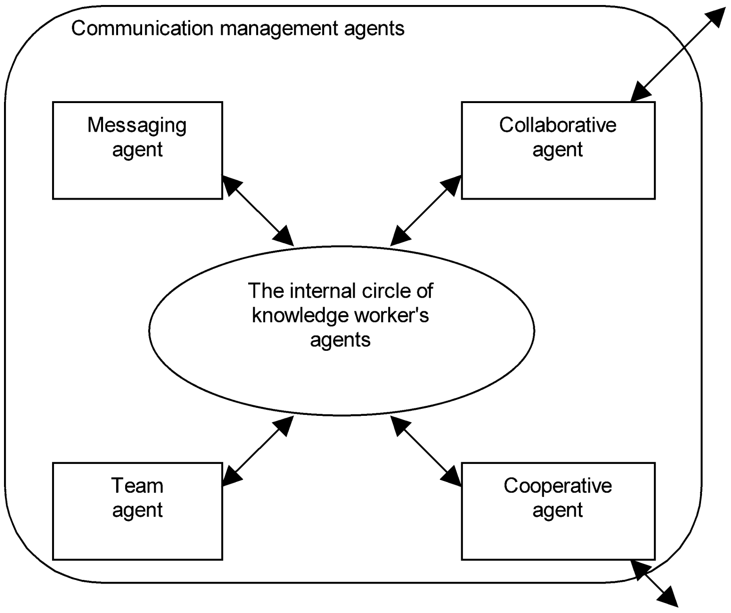

Communication management agents
Messaging agent
Collaborative agent
The internal circle of knowledge worker's agents
Team agent
Cooperative agent

Figure 5. The medium circle of agents.

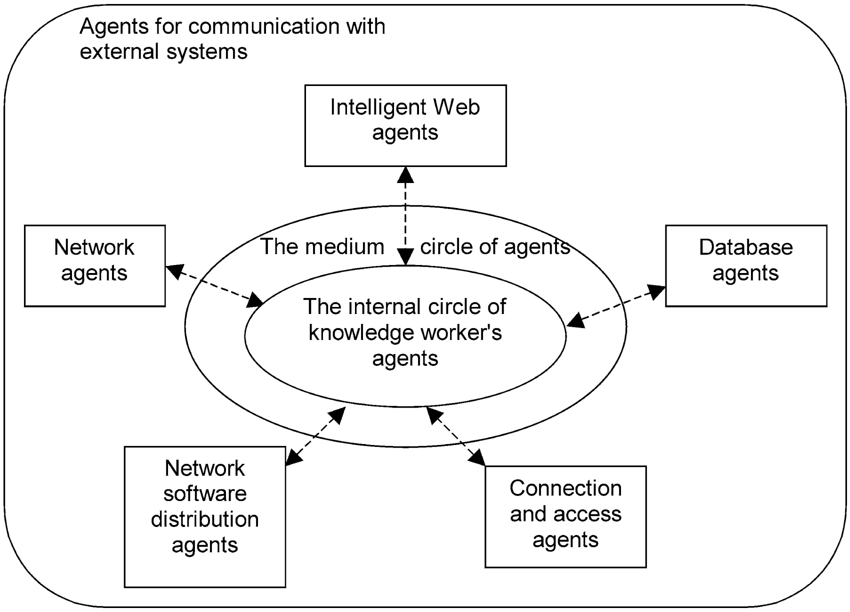

Agents for communication with external systems
Intelligent Web agents
Network agents
The medium circle of agents
Database agents
The internal circle of knowledge worker's agents
Network software distribution agents
Connection and access agents

Figure 6. The external circle of agents.

of data, more and more structured data (structured documents, online databases), simple metadata but very little knowledge, i.e., very few formal knowledge representations (Zhong at al., 2003). The reason is that the knowledge is encoded using various languages and practically unconnected ontologies. As a consequence, each knowledge source requires the development of a special wrapper for its knowledge to be interpreted and hence retrieved, combined, and used. Efforts to solve this problem resulted in the appearance of a new paradigm, so called Web intelligence for developing the Web-supported social network intelligence. Many details on developed approaches and tools in this hot research topic may be found in (Zhong et al., 2003) where intelligent Web agents, Web mining and farming for Web intelligence, intelligent Web information retrieval, Web knowledge management, etc. are discussed.

4. CONCLUSIONS

In this paper the system approach is used for the development of the conceptual framework for the intelligent tutoring system. The tutoring system is based on the intelligent agent and multiagent paradigm while the knowledge management system plays the role of its environment. A synergy effect is expected if agent technologies are integrated with knowledge management (Grundspenkis, 2003), especially in a hybrid course development where part of contents is taught in the traditional face-to-face manner, and another part using distance learning facilities.

This conceptual framework till now is only implemented for the prototypes of the student and the expert models. Further work is required to add more implemented models and agents in the system and to evaluate them in practice.

The potential of intelligent agents in development of intelligent tutoring systems is rather high. Today one can notice only the first steps towards intensive use of agent technologies. The future evolution of suitable agents for intelligent tutoring systems is connected with *information agents* and their extension – *knowledge agents* that will be able to learn from their environments and from each other as well as to cooperate with each other (Knapik and Johnson, 1998). They will have access to many types of information and knowledge sources and will be able to manipulate information and knowledge in order to answer queries posed by students and their knowledge agents. Teams of agents will be able to search Web sites, heterogeneous databases, and knowledge bases, and work together to answer queries that are outside the scope of any individual intelligent agent. These agents will execute searching in parallel, showing a considerable degree of natural language understanding using sophisticated pattern extraction, graphical pattern matching, and context-sensitive search. Coordination of agents will be handled either by supervising agents or via communications between agents. As the result, more and more activities performed by humans in teaching and learning process will be automated. This, in turn, will crucially impact the evolution of intelligent tutoring systems making them more and more intelligent.

REFERENCES

Anohina, A., 2003, Clarification of the terminology used in the field of virtual learning, in: *Scientific Proceedings of Riga Technical University, 5th series, Computer Science, Applied Computer Systems, Vol. 17*, Riga, RTU, pp. 94–102.

Borghoff, U. M., and Pareschi, R., 1998, Introduction, in: *Information Technology for Knowledge Management*, U. M. Borghoff and R. Pareschi, eds., Springer Verlag, Berlin, Heidelberg, New York, pp. 3–16.

Case, S., et al., 2001, Enhancing e-communities with agent-based systems, *Computer*, July, 2001, pp. 64–69.

Duh, Ch. M., et al., 2001, A multi-agent virtual seminar system (MAVIS) in collaborative distance learning, in: *New Perspectives on Information Systems Development*, G. Harindranath, et al., eds., Kluwer Academic/Plenum Publishers, New York, pp. 355–366.

Engelbrecht, A. P., 2002, *Computational Intelligence*, John Wiley & Sons, The Atrium, Southern Gate, Chichester, England.

Grundspenkis, J., 2003, Development of hybrid intelligent systems: integration of structural modelling, intelligent agents and knowledge management techniques, in: *Scientific Proceedings of Riga Technical University, 5th series, Computer Science, Applied Computer Systems, Vol. 17*, RTU Publishing, Riga, pp. 7–30.

Grundspenkis, J., and Kirikova, M., 2004, Impact of intelligent agent paradigm on knowledge management, in: *Intelligent Knowledge-Based Systems*, C. T. Leondes, ed., Vol. 1., Kluwer Academic Press, NY, pp. 164–206 (to appear).

Johnson, W. L., 2003, Using agent technology to improve the quality of web-based education, in: *Web Intelligence*, N. Zhong, J. Liu, and Y. Y. Yao, eds., Springer-Verlag, Berlin, Heidelberg, pp. 77–101.

Kirikova, M., and Grundspenkis, J., 2000, Using knowledge distribution in requirements engineering, in: *Knowledge Based Systems, Vol. 1*, C. T. Leondes, ed., Academic Press, San Diego, pp. 149–184.

Kirikova, M., and Grundspenkis, J., 2002, Types of knowledge and knowledge sources, in: *Scientific Proceedings of Riga Technical University, 5th series, Computer Science, Applied Computer Systems, Vol. 13*, Riga, RTU, pp. 109–119.

Knapik, M., and Johnson, J., 1998, *Developing Intelligent Agents for Distributed Systems*, McGraw Hill, NY.

Koenig, M. E. D., and Srikantaiah, T. K., 2000, The evolution of knowledge management, in: *Knowledge Management for the Information Professional*, T. K. Srikantaiah and M. E. D. Koenig, eds., ASIS Monograph Series, Medford, New Jersey, pp. 23–36.

Nonaka, I., and Takeuchi, H., 1995, *Knowledge Creating Organizations*, Oxford University Press, New York.

Russell, S., and Norvig, P., 2003, *Artificial Intelligence. A Modern Approach*, Pearson Education International, NJ.

Sarvary, M., 1999, Knowledge management and competition in the consulting industry, *California Management Review* **41**:95–108.

Stewart, T. A., 1994, Your company's most valuable asset: intellectual capital, *Fortune* **130**(68):68–74.

Traxler, J., 2002, Developing web-based education using information systems methodologies, in: *Information Systems Development. Advances in Methodologies, Components, and Management*, M. Kirikova, et al., eds., Kluwer Academic/Plenum Publishers, New York, pp. 69–77.

van Heijst, G., van der Spek, R., and Kruizinga, E., 1998, The lessons learned cycle, in: *Information Technology of Knowledge Management*, U. M. Borghoff and R. Pareschi, eds., Springer Verlag, Berlin, Heidelberg, New York, pp. 17–34.

Web Intelligence, N. Zhong, J. Liu, and Y. Y. Yao, eds., Springer-Verlag, Berlin, Heidelberg, 2003.

Wooldridge, M., 2002, *Introduction to Multiagent Systems*, John Wiley and Sons, The Atrium, Southern Gate, Chichester, England.

Yang, A., Kinshuk, and Patel, A., 2002, A plug-able web-based intelligent tutoring system, in: *Proceedings of the Xth European Conference on Information Systems, ECIS 2002*, S. Wrycza, ed., Vol. 2, Wydawnictwo Uniwersytetu Gdanskiego, Gdansk, Poland, pp. 1422–1429.

LIST OF WEB SITES

http://www.info.uqam.ca/~nkambou/DIC9340/cheikes95gia.pdf

http://www.dcs.napier.ac.uk/~dbenyon/IITpaper.pdf

http://www.sts.tu-harburg.de/~st.ziemer/its.pdf

http://www.iitg.ernet.in/engfac/sbnair/public_html/ai/batch1997/intelligent_tutoring_system_learnPro/intelligent_tutoring_system_learnPro.pdf

http://www.ssgrr.it/en/ssgrr2000/papers/capuano.pdf

http://www.aect.org/intranet/publications/edtech/19/index.html

http://www.sacla.org.za/SACLA2002/Proceedings/Papers/Padayachee.pdf

COMBINING SIMULATION MODELS WITH THE INFORMATION SYSTEM FOR AN OPERATIVE CONTROL OF THE OIL TERMINAL

Henrikas Pranevicius and Vytautas Pilkauskas*

1. INTRODUCTION

Nowadays, interoperability becomes vitally important for modern information technology solutions. The users would like to get the ability to allow software applications to use other applications. This effort to expose functionality has resulted the plans of Web services creating. These plans originated as companies tried to figure out distributed computing. The advantage is heavy computational tasks over multiple computers in a network. Distributed computing system makes the results appear to the end-user as if they were the product of running an application on a single, supper powerful machine. Definitions of Web services may change but the core concept is that all applications are made of self-describing components that can be activated on the fly over the network. The interoperability of a simulation model is the ability to provide services to and accept services from other simulation models or simulation model related components with the important goal to make an effectively operating environment.

Interoperability, in this meaning, requires two or more different simulation models which can exchange data and are able to interpret it. This must include effective data sharing and consistent data interpretation.

The High Level Architecture (HLA) is the most famous technology being used for realization of interoperability of simulation models. Recently Web services technologies have been created for integration of separate program components (Short, 2002). This paper presents subsystem for creation of tankers loading schedule. This subsystem integrates two simulation models (oil terminal and railway) and the database of oil products terminal information system.

The integration of simulation models and database has been done using service-oriented architecture and component based development approach.

* Kaunas University of Technology, Studentu 50 LT-Lithuania, hepran@if.ktu.lt, vytpilk@ktu.lt.

The paper is structured as follows: a short description of the aggregate method is presented the second section; the model of logistics processes of oil transportation through Klaipeda oil terminal and the research results are presented in the third section; possible technologies for integrating simulation models into information systems are reviewed in the fourth section; the architecture of system for creation of tankers loading schedule is presented in the fifth section.

2. THE USE OF PLA FORMALISM FOR HARBOUR PROCESSES ANALYSIS

Object orientation has become the dominant approach to the analysis and design computerised systems. OOA&D method has been well tested. At the same time, formal methods are becoming more popular for designing complicated information systems. There are a number of motivations behind using formal methods (Pranevicius, 1992):

- permits prepare formal description of analysed system having one meaning;
- properties of model may be analysed using mathematical proof techniques;
- formal description approach is a theoretical background developing software tools for computerised analysis (validation verification, simulation) of formal specifications.

In this paper piece-linear formalism (PLA) (Pranevicius, 1992) will be used for creation the dynamical model of business processes in Klaipeda oil terminal, which will be used in terminal operative information system.

Piece-linear aggregate formalism. In the application of the aggregate approach for system specification represents the system as a set of interacting piece – linear aggregates (PLA). The PLA is taken as an object defined by a set of states Z, input signals X and output signals Y. The aggregate function is considered in a set of moments in time t $\in$ T. The state z $\in$ Z, the input signals x $\in$X, and the output signals y $\in$ Y are considered to be time functions. Along with these sets, transition H and output G operators must be known as well.

The state $z \in Z$ of the piece-linear aggregate is the same as the state of a piece-linear Markov process, i.e.:

$$z\left(t\right) = \left(\upsilon(t), z_{\upsilon}\left(t\right)\right),$$

where $\upsilon(t)$ is a discrete state component taking values on a countable set of values; and $z_{\upsilon}(t)$ is a continuous component comprised of $z_{\upsilon 1}(t), z_{\upsilon 2}(t), \ldots, z_{\upsilon k}(t)$ co-ordinates.

When there are no inputs, the state of the aggregate changes in the following manner:

$$\upsilon\left(t\right) = \text{const}, \frac{dz_{\upsilon}\left(t\right)}{dt} = -\alpha_{\upsilon},$$

where $\alpha_{\upsilon} = (\alpha_{\upsilon 1}, \alpha_{\upsilon 2}, \ldots, \alpha_{\upsilon k})$ is a constant vector.

The state of the aggregate can change in only two cases: when an input signal arrives at the aggregate or when a continuous component acquires a definite value. The theoretical basis of piece-linear aggregates is their representation as piece-linear Markov processes.

Aggregate functioning is examined on a set of moments time $T = \{t_0, t_1, \ldots, t_m, \ldots\}$ at which one or several events take place, resulting in aggregate state alternation. The set of events E which may take place in the aggregate is divided into two non-intersecting subsets $E' = E' \cup E''$. The subset $E' = \{e'_1, e'_2, \ldots, e'_N\}$ comprises classes of events (or simply events) e'_i, $i = \overline{1, N}$ resulting from the arrival of input signals from the set $X = \{x_1, x_2, \ldots, x_N\}$. The class of events $e''_i = \{e''_{ij}, j = 1, 2, 3, \ldots\}$, where e''_{ij} is an event from the class of events e''_i takes place the j-th time since the moment t_0. The events from the subset E' are called external events. A set of aggregate input signals is unambiguously reflected in the subset E' i.e., $X \to E'$. The events from the subset $E'' = \{e''_1, e''_2, \ldots, e''_f\}$ are called internal events where $e''_i = \{e''_{ij}, j = 1, 2, 3, \ldots\}$, $i = \overline{1, f}$ being the classes of the aggregate internal events. Here, f determines the number of operations taking place in the aggregate. The events in the set E'' indicate the end of the operations taking place in the aggregate.

For every class of events e''_i from the subset E'', control sequences are specified $\{\xi_j^{(i)}\}$, where $\xi_j^{(i)}$ is the duration of the operation which is followed by the event e''_{ij} as well as event counters $\{r(e''_i, t_m)\}$, where $r(e''_1, t_m)$, $i = \overline{1, f}$ is the number of events from the class e''_i that have taken place in the time interval $[t_0, t_m]$.

In order to determine start and end moments of operation, taking place in the aggregate, the so-called control sums $\{s(e''_i, t_m)\}$, $\{w(e''_i, t_m)\}$, and $i = \overline{1, f}$ are introduced, where $s(e''_i, t_m)$ is the moment in time of the start of operation followed by an event from the class e''_i. This moment in time is indeterminate if the operation was not started; $w(e''_i, t_m)$ is the moment in time of the end of the operation followed by the event from the class e''_i. In the case there are no prioritised operations, the control sum $w(e''_i, t_m)$ is determined in the following way:

$$w\left(e''_i, t_m\right) = \begin{cases} s'\left(e''_i, t_m\right) + \xi_{r(e''_i, t_m)+1}, & \text{if at the moment } t_m \text{ an operation is taking place, which is followed by the event } e_i; \\ \infty, & \text{in the opposite case.} \end{cases}$$

The meaning of the aggregate state coordinates can be specified. The discrete component of the state, $\nu(t_m) = \{\nu_1(t_m), \nu_2(t_m), \ldots, \nu_p(t_m)\}$, presents the system state, i.e. counters of transmitted and received information packets, readiness for information transmission etc.

$$z_\nu(t_m) = \left\{w\left(e''_1, t_m\right), w\left(e''_2, t_m\right), \ldots, w\left(e''_f, t_m\right)\right\}$$

are control coordinates specifying the moment when evolutionary events occur.

When the state of the system $z(t_m)$, $m = 0, 1, 2, \ldots$, is known, the moment t_{m+1} of the following event is determined by a moment of input signal arrival to the aggregate or by the equation:

$$t_{m+1} = \min\left\{w\left(e''_i, t_m\right)\right\}, \quad 1 \le i \le f.$$

3. LOGISTICS PROCESSES IN KLAIPEDA OIL TERMINAL

Modern and especially international industry is closely linked with sea transportation. For this reason, harbours play a very important role in transportation chains.

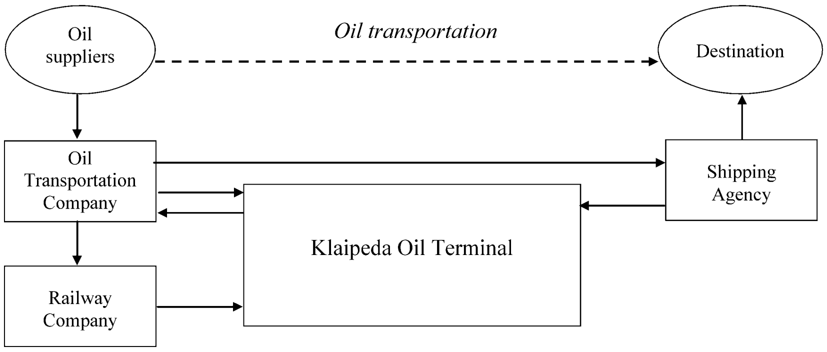


Figure 1. Main components of the logistics process for transporting oil through Klaipeda.

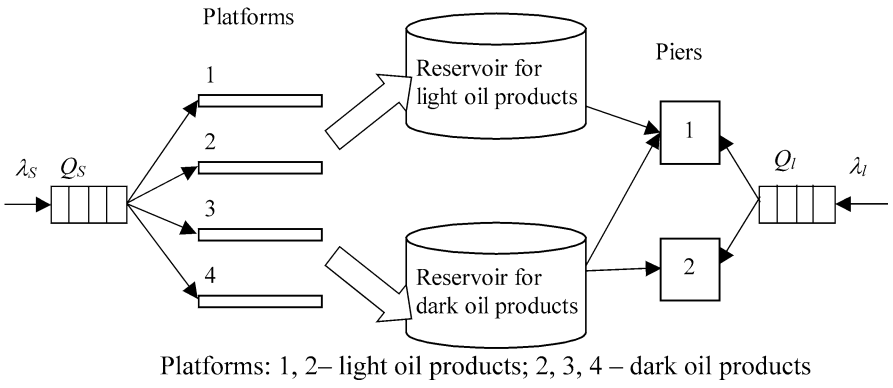

Platforms
1
2
3
4
λ_S Q_S
Reservoir for light oil products
Reservoir for dark oil products
Piers
1
2
Q_l λ_l
Platforms: 1, 2– light oil products; 2, 3, 4 – dark oil products

Figure 2. Structural scheme of queuing system, which represents transportation of oil products through Klaipeda.

Solving logistic tasks in modern harbours worldwide is extremely important not only as place for transshipment and storage but also as places of partial manufacturing and especially as distribution centers.

For harbours as the most significant links in transport chains, logistics supply is very important. The essence of this supply is a preparing rhythmic work of all types of transport; aggregating and assigning loads according to a means of transportation (ship, train, etc); spending as little as possible to execute necessary tasks.

Figure 1 presents the main components of logistic processes for transporting oil through Klaipeda.

Oil transportation creates the following flows:

- Flows of orders for oil products transported through terminal;
- Flows of trains;
- Flows of tankers.

Figure 2 presented the structural scheme of Klaipeda oil terminal (Pranevicius et al., 1999).

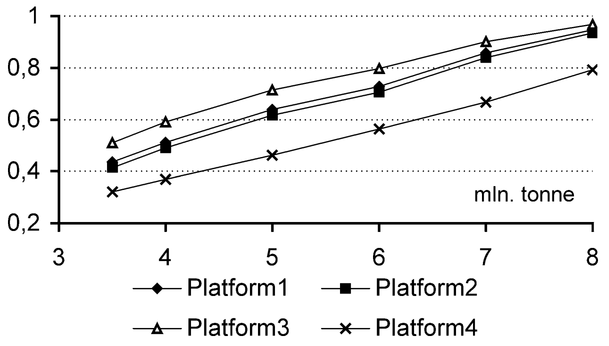


Figure 3. Dependencies of platforms occupation with respect to transported amount of oil.

Analyses of Klaipeda oil terminal were performed considering different infrastructures (number of platforms, embankments, reservoirs and their capacities, etc). Analyses performed with the simulation model established the main factors influencing transportation times through Klaipeda terminal are the amount of transported oil annually and the operative control algorithms used.

In order to fulfill user requirements to deliver oil to destination within a specified time, the incoming flow of orders needs to restrict in some cases.

Simulation experiments carried out defined functional requirements for reengineering the Klaipeda oil terminal information system. Thus, the use of simulation models for loading processes in the terminal and transportation of oil products by rail in the operative control system.

Simulation Results.

Technological parameters of terminal:

a) Time during which oil products are poured from wagons to reservoirs for each kind of oil.
b) Rate of pouring oil from reservoirs to tanker.

Parameters characterizing control of terminal:

a) Order performance decision-making algorithm, which evaluates the number of wagons, which are in railway station.
b) Order performance decision-making algorithm, which does not evaluate the number of wagons, which are in railway station.
c) Algorithm carrying orders of tankers and evaluating only needed amount empty reservoir for realization of order.
d) Algorithm carrying orders of tankers and evaluating needed amount empty reservoir for realization of order and number of wagons in railway station.

Figure 3 depicts dependencies of platforms occupation with respect to transported amount of oil. Figure 4 depicts similar dependencies for embankments. Figure 5 depicts distribution functions of service time of orders for transportation oil.

Simulation results show that up to 8 megatons per year of oil products can be transported through Klaipeda oil terminal.

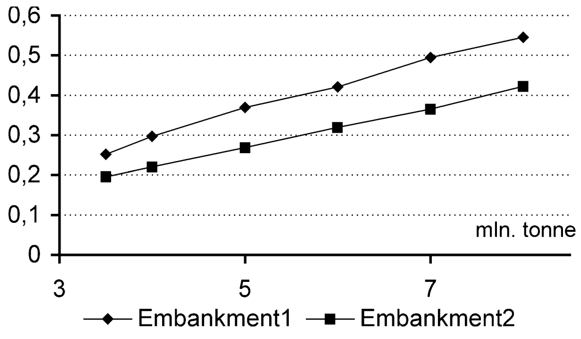


Figure 4. Dependencies of embankments occupation with respect to transported amount of oil.

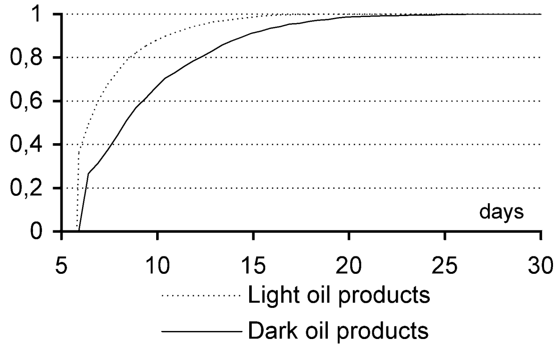


Figure 5. Distribution function service time for order for different kinds of oil product (transportation amount is 6 megatons).

4. POSSIBLE TECHNOLOGIES FOR INTEGRATING SIMULATION MODELS IN INFORMATION SYSTEMS

Service-oriented architecture and component-based development (Brown et al., 2003) were used to develop a prototype of the Tanker Loading Operative Control System (TLOCS) at the Klaipeda oil terminal.

Service-oriented architecture is not a new idea and has taken on importance because of emerging web services technology. A service is generally implemented as a coarse-grained, distributed software entity that exists as a single instance and interacts with applications and other services through a loosely coupled (often asynchronous), message-based communication model (Matthew, 2003).

While the services encapsulate the business functionality, some form of inter-service infrastructure is required to facilitate service interactions and communication. Different infrastructure forms are possible because services may be implemented on a single machine, distributed throughout a set of computers in a local area network or distributed more widely throughout several companies' networks. A particularly interesting case is when the services use the Internet as the communication mechanism. The resulting web services share the characteristics of more general services, but they require special consideration as the result of using a public, insecure, low-fidelity mechanism for inter-service interactions.

The World Wide Web Consortium (W3C) Web Services Architecture Working Group: has formulated a definition: "A web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols."

Web Service is a collection of functions packaged together and published on a network for use by other client programs. At a very high level of abstraction, we might think of a Web Service as a type of Remote Procedure Call (RPC) or messaging system. The idea of a client application requesting a service from a server application is not new; the difference is all in the plumbing. A Web Service, as its name implies, is based on common, Web-related technologies: TCP/IP, HTTP, FTP, SMTP, and XML. In addition, three new specifications – Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI) – together form the foundation set of technologies that define Web Services.

"Any Web Service can interact with any other Web Service. Web services can also aggregate to provide higher-level functions. After all, they are just software components, and all software components have the potential to invoke Web Services. By implication, some Web Services will have dependencies on other services that are either invisible to the client or require client participation. For example, a Web Service might require that the client use another Web Service to create some token or value that it must then pass on to the next service. Or, a Web Service might internally invoke other services before returning a result to the client."

Because of the additional transport overhead associated with Web services and the expectation that services will, by their nature, be remote, it is important to reduce the time a requestor spends waiting for responses. By making a service call asynchronous, with a separate return message, we allow the requestor to continue execution while the provider has a chance to respond. For services that expect a very high load, we would need to decouple the part that listens to the requestor and the part that services the request itself. This is already a well known pattern (Figure 6), in which a message queue is used to decouple a service facade from the service implementation (Brown et al., 2003).

Such a pattern can be easily implemented in both .NET and J2EE using services provided by those platforms:

- MSMQ for .NET and Java Message Queue Service (JMS) or message-driven beans for J2EE.
- This provides the developer with a simpler scalability model; rather than handling a set of threads with synchronization for the requests, the implementation can simply add additional queue listeners to pick messages from the queue, even across multiple machines.

Integrating the simulation model in an information system requires data access in database management system. Microsoft's NET has a new mechanism for accessing data: ADO.NET. One of the general ADO.NET element is the DataSet. The DataSet can be filled either from a data source (using DataAdapter object) or dynamically from a program code

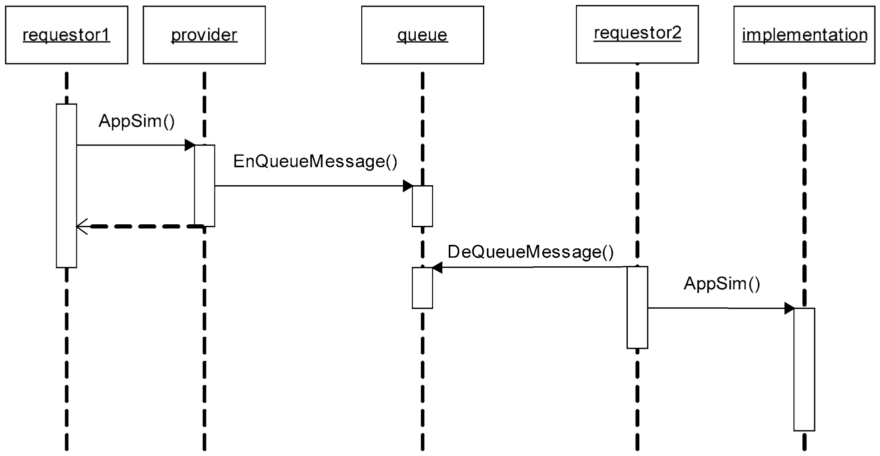


Figure 6. Decoupling a service facade from the service implementation.

(simulation model). Using these DataSet properties makes integrating simulation model program code with an information system database.

PLA formalism is used to create the simulation model. Then the discrete coordinates of the aggregate state are entered in the elements of the DataSet table.

5. TANKERS LOADING OPERATIVE CONTROL SYSTEM ARCHITECTURE

The service-oriented architecture and component-based development approaches (Short, 2002; Matthew, 2003) have been used for developing Loading Operative Control System (TLOCS) in Klaipeda oil terminal (Pranevicius et al., 2003). The created TLOCS system consists of (Figure 7):

- The terminal simulation model of loading processes in the terminal;
- The railway simulation model of oil transportation through the railway;
- The terminal Web service;
- The railway Web service;

The terminal web service provides:

- Web based simulation interface for terminal simulation model;
- Combined terminal simulation model and terminal IS;
- Interoperability between terminal simulation model and railway simulation model.
- Visualization tanker loading processes.

The railway web service provides:

- Interoperability between terminal simulation model and railway simulation model.

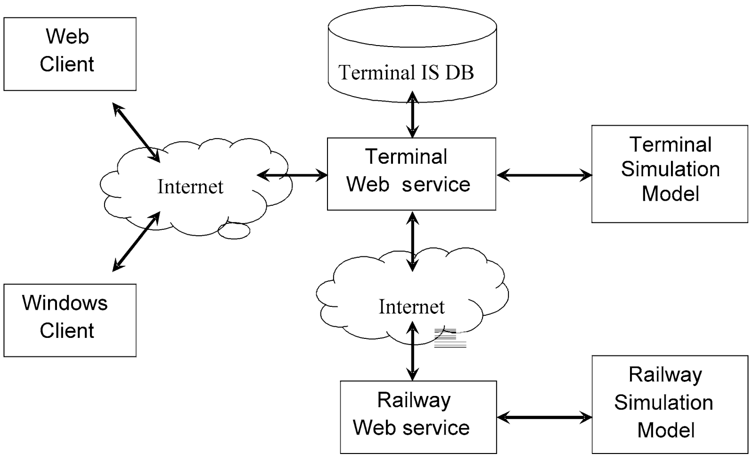


Figure 7. Architecture of TLOCS system.

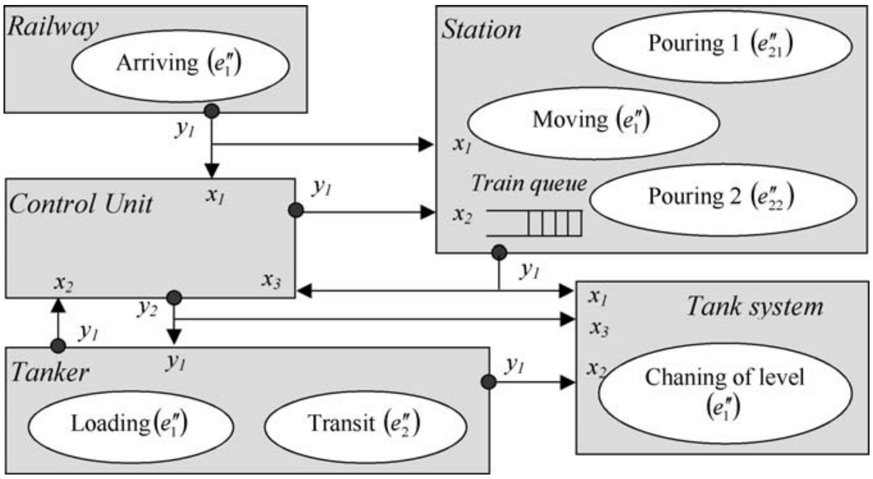


Figure 8. Aggregate model of oil terminal.

Terminal Simulation Model of Loading Processes in Terminal (Figure 8). Oil is delivered to terminal by train. Single kind of oil is transhipped in terminal. Duration of transhipment from train tanks to reservoir depends on oil temperature and this dependency is known. Two platforms are used for transhipment of oil in terminal. Arrived trains are placed to queue when both platforms are occupied. Trains are served according FIFO servicing strategy. Terminal has one embankment.

Mathematical description of two aggregates (Railway and Station) is presented below.

Aggregate RAILWAY

1. The set of input signals: $X = \emptyset$.
2. The of output signals: $Y = \{y_1\}$, $y_1 \in \{arrive\_train\}$, when arrive_train – the train has arrived.
3. The set of external events: $E' = \emptyset$.
4. The set of internal events: $E'' = \{e_1''\}$, e_1'' – event which occurs, when train arrives.
5. Controlling sequence: $e_1'' \to \{\eta\}_{i=0}^{\infty}$, η – time duration, after which arrives a next train.
6. The set of discrete co-ordinates of state: $\nu(t) = \emptyset$.
7. The set of continuous co-ordinates of state: $z_\nu(t) = \{w(e_1'', t)\}$, where $w(e_1'', t)$ – time instance when arrives a new train.
8. Initial state: $w(e_1'', t_0) = t_0 + \eta_0$.
9. Transition operators:
$H(e_1'')$: $w(e_1'', t_{m+1}) = t_m + \eta_m$.
$G(e_1'')$: $Y = \{y_1\}$, where $y_1 = \{arrive\_train\}$.

Aggregate STATION

1. The set of input signals: $X = \{x_1, x_2\}$, $x_1 \in \{arrive\_train\}$, $x_2 \in \{shunt\}$
2. The of output signals: $Y = \{y_1\}$, $y_1 \in \{start\_pump, end\_pump\}$.
3. The set of external events: $E' = \{e_1', e_2'\}$.
4. The set of internal events: $E'' = \{e_1'', e_{21}'', e_{22}''\}$.
5. Controlling sequence: $e_1'' \to \{\eta_i\}_{i=1}^{\infty}$, $e_{21}'' \to \{\xi_{1i}\}_{i=1}^{\infty}$, $e_{22}'' \to \{\xi_{2i}\}_{i=1}^{\infty}$, η_i, ξ_{ij} – durations of operations.
6. The set of discrete co-ordinates of state: $\nu(t) = \{Q(t), rate_{21}(t), rate_{22}(t)\}$, $Q(t)$ – queue in which is stored arrival times of trains, $rate_{2i}(t)$ – intensity of pouring oil from the i-th platform.
7. The set of continuous co-ordinates of state:
$z_\nu(t) = \{w(e_1'', t), w(e_{21}'', t), w(e_{22}'', t)\}$
8. Initial state: $\#Q(t_0) = 0$, $rate_{21}(t_0) = 0$, $rate_{22}(t_0) = 0$.
Transition operators:
$H(e_1')$:
$Q(t_{m+1}) = Enq(Enq(Q(t_m), t_m), t_m)$, where Enq – operator placing of new element to $Q(t)$.
$H(e_2')$: $Q(t_{m+1}) = Deq(Q(t))$, $w(e_1'', t_{m+1}) = t_m + \eta_m$.
$H(e_1'')$: $w(e_{2i}'', t_{m+1}) = t_m + \xi_{im}$, $rate_{2i}(t_{m+1}) = \xi_{im}$, $i = \min\{j \mid rate_{2j}(t_m) = 0\}$.
$G(e_1'')$: $Y = \{y_1\}$, where $y_1 = \{start\_pump\}$.
$H(e_{2i}'')$: $rate_{2i}(t_{m+1}) = \xi_{im}$,
$G(e_{2i}'')$: $Y = \{y_1\}$, where $y_1 = \{end\_pump\}$.

Railway Simulation Model of Transportation Oil Through Railway (Figure 9). Model simulates transportation of oil products from oil plants to Klaipeda oil terminal.

This imitation model consists of three kinds of aggregates:

PLANT, simulates the loading process of oil products and the departing of the trains from the oil plant.

LINE, simulates the moving of the train over the certain railway line. There is a ability to connect a few aggregates of the same type and to get needed configuration of the railway.

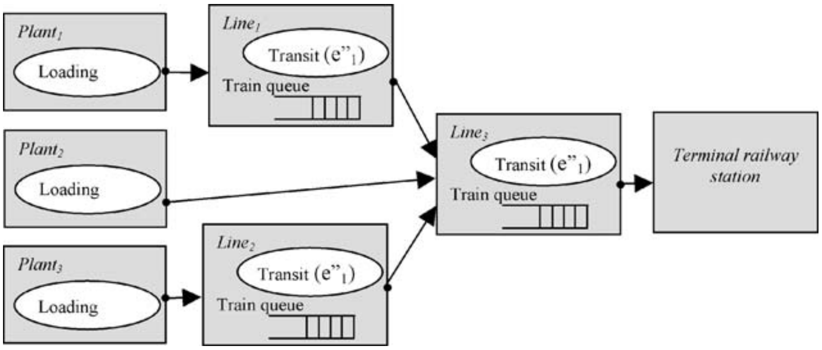


Figure 9. Railway aggregate model.

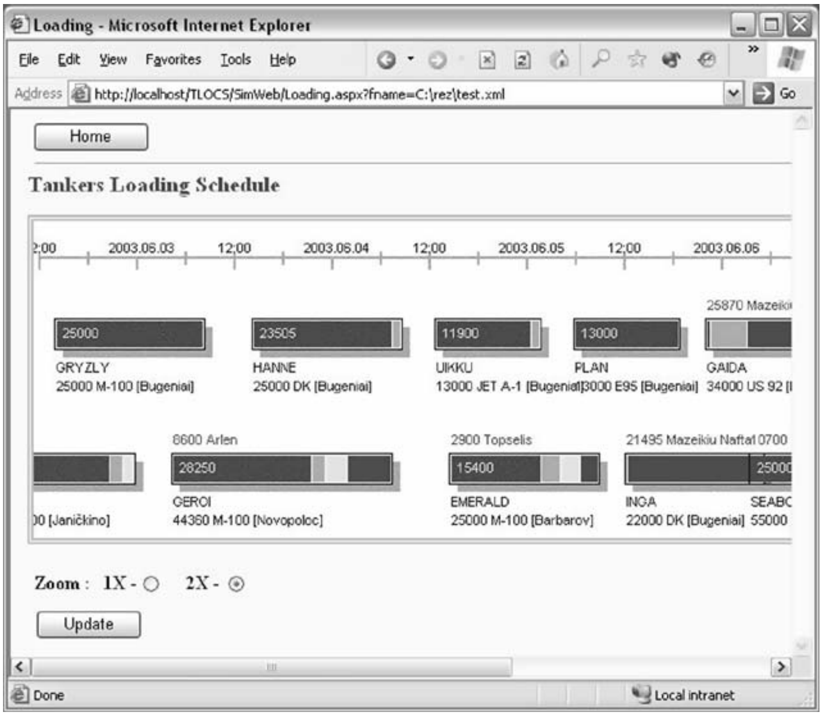


Figure 10. Tanker loading schedule.

TERMINAL RAILWAY STATION, simulates the departing of the train to the terminal railway station.

Client program for visualization tankers loading processes. Web client program of the described task solution has been created. Structure of window for graphical chart of tankers loading schedule is presented in Figure 10.

6. CONCLUSIONS

The created formal specification of oil terminal activity processes, the simulation model and the performed researches showed that terminal operative management may be improved by using simulation models of oil products transportation through the railway and simulation models of technological processes of oil loading in the terminal. These models must be integrated into corresponding terminal and railway information systems in which information about oil transportation process in the railway and loading works in the terminal is fixed.

Such model system allows forecasting technological processes run in oil terminal.

The PLA formalism and the object-oriented library PRANAS allowed creating imitational models programme realizations using the MS Framework.NET technologies. These technologies allowed integrating imitational models with corresponding information systems databases and realizing models interoperability.

REFERENCES

Brown, A., Johnston, S., and Kelly, K., 2003, Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications. A Rational software whitepaper from IBM (September 15, 2003); http://www.rational.com/media/whitepapers/TP032.pdf.

Matthew, M., 2003, *Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting*, Microsoft Press, Washington, p. 692.

Pranevicius, H., 1992, Aggregate approach for specification, validation, simulation and implementation of computer network protocols, in: *Lecture Notes in Computer Science No 502*, Springer-Verlag, pp. 433–477.

Pranevicius, H., Pilkauskas, V., and Makackas, D., 1999, Interaction of various kinds of transportation at Klaipeda Harbour, in: *Proceedings of The International Workshop "Harbour, Maritime & Industrial Logistics Modelling and Simulation"*, Genoa, pp. 124–129.

Pranevicius, H., and Makackas, D., 2002, Simulation of stewodoring works in the Klaipeda oil terminal, *Journal of Vilnius Gediminas Technical University and Lithuanian Academy of Sciences, TRANSPORT*, Vol. XVII, N 5, pp. 188–193.

Pranevicius, H., and Pilkauskas, V., 2003, The use of Simulation Models in Maritime Information Systems, in: *The International Workshop on Harbour, Maritime and Multimodal Logistics Modelling & Simulation*, Riga, pp. 338–344.

Short, S., 2002, *Building XML Web Services for the Microsoft .NET Platform*, Microsoft Press, Washington, p. 426.

ASPECT-ORIENTED ANALYSIS AND DESIGN USING ACTIVITY BASED COSTING

Arjan Visser and Kees van Slooten*

1. INTRODUCTION

In business, reducing and controlling the costs of business activities has management's attention. The real costs of business activities as part of a business process to support the customer are often higher than the budget, and there are often insufficient possibilities for control. To get a better understanding of the costs of business activities, an organization may apply Activity-Based Costing (ABC) to determine prices of products and services. There exists a lot of literature about Activity-Based Costing, e.g. Glad and Becker (1997). Activity-Based Management (ABM) uses ABC to collect data about costs. Hixon (1995) defines ABM as follows: "Activity-based management is the management and control of enterprise performance using activity-based information as the primary means of decision support". Hammer et al. (1993) define BPR as follows: "Re-engineering is the fundamental rethinking of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service and speed". Achieving lower costs by rethinking business processes and business activities is a main goal of such approaches. ABC and ABM may play an important role in this rethinking process, i.e. analyzing and designing business processes and activities. Different aspects of analyzing and designing an object system like a business process will be dealt with in this paper. Van Slooten and Brinkkemper (1993) define an object system as follows: "…the part of reality, or universe of discourse, considered as problem area…", and distinguish the following important aspects of the analysis and design process: problem, organization, process, information, and behavior. The problem-oriented aspect uses methods, techniques, and tools to articulate and solve the problems of the object system; e.g. ISAC analysis of change (Lundeberg, 1982) is a good example supporting this aspect. The organization-oriented aspect uses methods, techniques, and tools to specify the organizational structure and culture to support the problem-solving aspect; e.g. organization charts and responsibility matrices may be applied to model the structural aspects. Process-oriented methods

* University of Twente (BBT), Department of Business Information Systems, P.O. Box 217, 7500 AE Enschede, The Netherlands, a.a.visser@alumnus.utwente.nl, c.vanslooten@utwente.nl. Fax: +31.53.4893509.

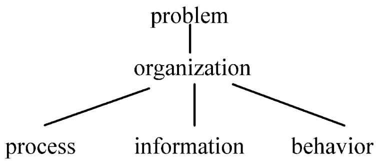


Figure 1. Hierarchy of aspects.

structure the processes and functions of the object system; e.g. decomposition of the object system as explained by (Van Slooten and Brinkkemper, 1993) supported by Lano matrices (Lano, 1979). The focus of information-oriented methods is on modeling information; e.g. Lano matrices and/or entity/relationship models. The behavior-oriented aspect encompasses temporal and dynamic aspects of the object system; e.g. dynamic, Petri-net-based modeling of the procedures of the object system. The process, information, and behavior aspects support the organization aspect, which supports the problem aspect implying a hierarchical relationship between aspects (Figure 1, based on Van Slooten, 1995). Situated Method Engineering (Van Slooten, 1996) may be applied to decide upon fitting methods or method components.

In Section 2 some principles of ABC and ABM will be explained, followed by the construction of a new model for reducing and controlling costs in Section 3. A case study to test the proposed model is given in Section 4, followed by our conclusions in Section 5.

2. ABC AND ABM

Activity-Based Costing is a costing system that takes into account the fact that some costs are not related to the volume, but are dependent on the sort and number of activities that have to be executed for the generation of products and services. The ABC method is developed in the 1980's in the United States, primarily to meet manager's needs for accurate information with regard to the costs of resources, necessary for products, services, clients and channels (Cooper and Kaplan, 1988).

Activity-Based Costing divides costs into direct costs and indirect costs. Direct costs are costs specifically incurred for a certain cost object, and indirect costs are costs not specifically incurred for a certain cost object, but for the entire production and sale.

The ABC concept is based on the assumption that the generation of products and services generates an internal request for activities (Staubus, 1988). By applying the ABC approach, indirect costs are assigned to a product by linking costs to activities. The allocation of costs in Activity-Based Costing is represented in Figure 2 (Glad and Becker, 1997).

The direct costs are assigned directly to the diverse *cost objects* (product types). For this assigning process relatively simple methods can be used, in which costs are proportionally spread over the cost objects. The indirect costs are assigned indirectly to the cost objects in several successive steps. In the first phase, all actions executed in the departments, which are directly or indirectly connected to the production or service are grouped. In this way groups of activities are formed. Furthermore, an *activity driver* determines the

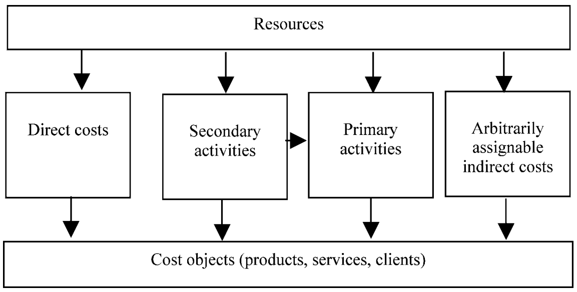

Resources
Direct costs
Secondary activities
Primary activities
Arbitrarily assignable indirect costs
Cost objects (products, services, clients)

Figure 2. The ABC system.

costs of an activity. An example of an activity driver is the cost per resource unit. Activities can be distinguished into primary and secondary activities. Primary activities are activities, which directly contribute to the department's objective. Secondary activities are activities which support a primary activity. The costs of secondary activities are assigned to the primary activities. The second phase will assign these costs to the cost object by a so-called *cost driver*. An example of a cost driver is the cost per hour for executing the activity. The last phase relates to the assignment of the arbitrarily assignable indirect costs. These are costs, which just have almost no relationship with any activity or cost object. Examples of these costs are general management expenses and small expenses such as mail which cannot be related to a specific activity directly. Glad and Becker (1997) state that these expenses are normally of a small amount in relation to the total expenses (less than 5 % of total expenses) and can be spread over the cost objects proportionally.

Besides the static analysis of the ABC system, one can try to influence the system in order to lower costs. With respect to cost reduction two aspects are relevant: The size of reduction and the method of reduction. The size of reduction is about the cost level to which costs have to be reduced. The method of reduction is about the way costs are reduced. According to the ABC system, costs can be reduced on two points. On the one hand, directly by reducing the costs of resources and, on the other hand, indirectly by taking care that fewer resources are needed to perform the activities. The latter point can be achieved for example by performing the activities more efficiently, eliminating them or outsourcing them. In the one case, a cost reduction plan is delivered and, in the other case, a route for the restructuring of activities is delivered.

This is the point where Activity-Based Management (ABM) comes in. ABM tries to influence the ABC system in order to lower costs. The purpose of ABM is to reach a continuous improvement of performance on important business processes through cost control. Hixon (1995) defines ABM as follows: "Activity-based management is the management and control of enterprise performance using activity-based information as the primary means of decision support". So, ABM uses the ABC system to collect data on costs. These are initially used to expose the origin of the costs of the supporting activities and next to carry out a so-called *value analysis* to determine which activities must be improved to enhance the value created by the performed activities.

In conclusion, it can be stated that cost savings can be realized by analyzing costs and by controlling them through improved structuring of the organization's activities.

3. A NEW MODEL FOR ASPECT ORIENTED ANALYSIS AND DESIGN USING ABC AND ABM

Activity-Based Costing is a costing system for fixing cost prices. To reduce and control costs, it is necessary to have insight into costs. Insight into the construction of costs can be gained on various levels. It can vary from insight into the total costs to insight into the costs per unit. However, to gain insight on the level of a cost price, the total costs have to be known first. Then the total costs can be broken down by means of methods for cost allocation, such as the ABC method, in order to gain insight on the level of a cost price. Insight into the total costs also enables insight into the development of costs, but on a very global level. However, to be able to analyze, explain and forecast cost patterns, it is of importance to know the behavior of costs. The behavior of costs indicates whether costs are direct or indirect and fixed or variable. Other influences on cost behavior are the economic lifetime and the depreciation method of the relative product. To trace the behavior of costs, there has to be insight on the level of the cost price. This implies that costs of all units have to be known, so that it can be stated whether costs are direct or indirect and fixed or variable. The development of costs also shows how costs develop during the year, and which costs contribute to the budget possibly being exceeded.

To reduce costs, it is necessary to analyze where reductions in costs can be made, based on the ABC system. On basis of the information about the construction of costs and the related activities, an ABM program can be delivered. Using this ABM program, an assessment can be made about costs and activities in order to take measures for reducing these costs. Besides, ABM delivers the information for the improvement or redesign of activities, which are involved in the controlling of processes. So, ABM is used to determine the origin of certain costs, on the basis of which ways to reduce costs can be decided on. Then, on the one hand, the direct costs are to be reduced, and so, on the other hand, are the indirect costs by improving the activities. To define which activities are to be considered for improvement, a value analysis (of activities) can be applied. The value analysis determines the performance of the various activities. The activities for each department have to be defined first. The usual procedure begins with an interview with the manager concerned (Cooper, 1990; Cooper et al., 1992; Innes and Mitchell, 1990) in order to draw up a list of activities needed to manufacture the products or perform the services. The definition of activities is followed by these indications:

- Significant activities ($5\% <$ consuming time/department's time $\times$ 100);
- Estimated time percentage spent on each activity;
- Activity input(s), output(s) and output measure(s);
- Classify activities as primary or secondary and value- or nonvalue-added;
- Link activity to a business process.

Thereupon, the departmental resources required to perform the activities are traced for each activity. According to Pryor and Sahm (1989), at least 90 percent of the departmental resources should be spent on primary activities and not more than 10 percent on secondary activities. A value-added activity is an activity that tries to meet the customer's or organization's demand and expectations. A nonvalue-added activity is an activity that does not try to meet this. According to the ABM method, companies aim to reduce the percentage of resources spent on nonvalue-added activities to zero. The value-analysis activity provides both information for the enhancement of activities' performance and information for a better structuring and design of activities. This forms the basis for the reduction in indirect costs and creates better possibilities for the control of the indirect costs of the ABC system. The problem with ABM is that it is programmed to deny and annihilate anything which is not on its list of routine activities, whether it is of genuine value or not (Armstrong, 2002). Besides, Armstrong (2002) says of ABM: "ABM is really nothing more than an updated and partially automated form of cost reduction and control". So ABM just delivers an assessment about activities, but does not provide a method for activity improvement. Despite attempts to represent ABM as an instrument of quality management, the most it can actually achieve is rescaling of existing activities, not a modification of them (Armstrong, 2002).

In order to enhance the performance of the activities, the scaling of the activities or the activities in their own right have to be influenced. Based on the current process structure and the information from the ABC system, Aspect-Oriented Analysis and Design may indicate how an improved process structure can lead to a more efficient ABC system. By applying Aspect-Oriented Analysis and Design, quality improvement and cost reduction are achieved by redesigning processes from the bottom up or at least by efficiently cutting in processes, and integrating and efficiently linking them together (workflow management). Aspect-Oriented Analysis and Design provides a complete method for the reengineering of processes. During the ABM-phase, the activities considered for improvement are determined. Aspect-Oriented Analysis and Design reflects the important elements for the redesign of activities in order to obtain significant improvements on performance and efficiency. The collection of activities to be changed or influenced is the object system. The aspects of the object system to analyze are: problem, organization, process, information and behavior. Based on this analysis, the needs for change can be identified in order to make recommendations for a renewed object system with a higher performance on activities to reduce indirect costs.

Figure 3 shows the relationships between Activity-Based Costing, Activity-Based Management and Aspect-Oriented Analysis and Design. This model shows the sequential steps to be taken to reduce costs and gain better control on these costs. The model is based on the assumption that Activity-Based Costing is applied as the costing system.

The first step of the model consists in determining the construction of costs according to the ABC system, the development of costs and at which points budget exceeding can be expected. Having gained insight into costs, the next step is to try to manage the costs of the ABC system. ABM traces the origin of costs, on the basis of which recommendations can be made towards reducing costs or adjusting the budget. The indirect costs are analyzed on the relative performances of the causal activities by means of a value analysis in order to define which activities can be improved. The third step then consists in improving the (structuring of) activities with the aim of achieving better control of the costs caused by these activities. To this end, the problems as experienced with the object system are defined. Subsequently, the organization, process and information function, and the behavior of the object system are analyzed in order to make recommendations for improving the (structuring of) activities. All aspects are involved into the design of the new object system.

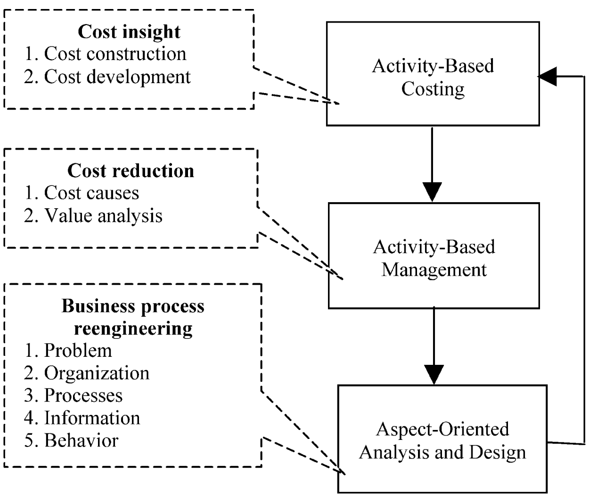


Figure 3. Model for reducing and controlling costs.

Possibly, the new structuring of activities may show a need for adjustment of the ABC system, such as a redefining of primary and secondary activities, resources, activity drivers or cost drivers.

This model concretely shows the sequential steps to be taken in order to reduce costs and get a better control on costs. The model consists of elements that are generic in situations where ABC plays an important role. Due to the general character of the model, it can be seen as a generic instrument for cost reduction and cost control improvement.

4. CASE STUDY

In 2003, a large international bank in Europe showed a need for the reduction of telephone call costs and for a better control on these costs. The management had indicated a huge overspend of the telephony budget with a lack of hands-on instruments to get a grip on these costs. The bank used the ABC system to define the cost prices of its telephony cost objects. In order to make recommendations for reducing these costs and to get a better control, the model as described above was applied.

Activity-Based Costing

The first step of the model is to gain insight into costs applying the ABC system. Cost insight comprises two elements: What is the construction of costs and how do costs develop? The ABC system classifies activities as primary and secondary (see Table 1).

The primary activities of the ABC system were the ordering and delivery of telephone calls related products and services, and the passing on of those costs. The secondary, or supporting, activities were the processing of the requests, the execution of an administration system, the control of invoices, and the management of the exploitation account. The

Table 1. ABC system of telephony

Resource	Secondary activity	Primary activity	Cost object
Cost of data network	-	-	Telephone call standard
Cost of products	-	-	Telephone call standard
Call charges	-	-	Telephone usage
Personnel cost	Processing request Execution Administration	Ordering and delivery	Telephone call standard
	Control Invoices Management exploitation account	Cost charching	

costs of the activities were determined by assigning the costs of the resources needed to these activities. The costs of the resources needed for the execution of these activities were the personnel costs, the call charges, the costs of the data network, and the costs of telephone call products. The final stage was to allocate the costs of the activities to the ultimate cost objects. The direct costs were allocated directly to the cost objects. The indirect costs were allocated to the cost objects by assigning them to activities. The following cost objects were identified: fixed telephone call standard, mobile telephone call standard, fixed telephone usage and mobile telephone usage. The allocation of the costs of resources to the cost objects occurred as follows. The call charges were traced to the telephone-usage cost objects directly. The costs of the data network and the costs of telephone call products were traced to the telephone call standard cost objects. The personnel costs were indirect costs and related to the activities of the ABC system, and therefore had to be allocated in a different way. The indirect costs were linked to the activities by means of the activity driver. The activity driver for the personnel costs was the cost price per hour for the performance of the activities. Using the cost driver, the costs of the activities were allocated to the cost objects. The cost driver of the activities was the cost price per user for using the service.

Having gained insight into the ABC system on telephone call costs, the development of costs was determined. The development of costs depends on the cost properties and the expected quantitative development of costs over a given time span. The cost's properties indicate whether costs are direct or indirect, and fixed or variable. By quantifying costs, the costs expected to be realized are fixed in relation to the budget. So, the development of the costs indicates the cost properties and the costs realized in relation to the budget. From the estimation of the development of costs it became clear that the call charges and the costs of telephone call products were expected to exceed by far the budgetary restrictions on these costs.

Activity-Based Management

To make recommendations for the reduction in the telephone call costs, the causes of the budget overrun had to be traced. To this end, interviews were conducted with relevant managers and workers in the telephone call organization. The costs of calling appeared to be higher than budgeted for both fixed and mobile telephone calls, but the cellular phone costs were a factor 1.5 higher than the fixed calling costs, although the number of fixed telephone connections was three times higher than that of mobile ones. Closer investigation revealed a disproportionately large number of private calls by cellular phone users outside office hours. This indirectly implied that the possessors of cellular phones took advantage of the uncontrolled and unlimited use of the business telephones. On the other hand, the costs of telephone call products were expected to greatly exceed the budget. From the interview results it appeared that it had something to do with the fact that there were product orders still passing through the request process that were not taken into account when drawing up the budget. Besides, there were a lot of cellular phones in circulation, which were no longer used but still being depreciated.

The next step of Activity-Based Management is to look for possibilities for cost reduction. To reduce the calling costs, a so-called calling limit per traffic category was set up. Before, there was just a total budget on call charges which could be easily exceeded, as there was no control on these costs. By setting up a calling limit per traffic category, the call charges per worker could be limited and better controlled. A traffic category defines the rights according to telephone calls: is the person authorized to make international calls, cellular phone calls et cetera? Each worker was assigned to one of the traffic categories. This could be any one of the categories classified according to function or department. For each traffic category a calling limit was fixed. On the basis of this calling limit per traffic category a more exact total budget for call charges could be fixed. Thus, budgets could be set up for each department as derived from this total budget. In doing so, managers are held responsible for the call charges in their department, while affording decentralized control to reduce the call charges.

In addition to reducing telephone call costs, the budget had to be prevented from being exceeded by telephone call costs by making adjustments to the framing of the budget. This required a more accurate determination of call charges, which could be done by using the calling limits per traffic category. Also, running costs had to be taken into account to a greater extent when designing the budget, and unique costs had to be determined more accurately. By applying these points more exactly during the framing of the budget, one could better protect it from an unrealistic estimation of real costs.

The last stage of ABM consists in executing the value analysis. For this, all activities of the ABC system are listed and judged in terms of performance. It is determined for each activity whether it is a significant activity, what the estimated time percentage is that is spent on each activity, what the activity input(s), output(s) and output measure(s) are, whether the activity is primary or secondary and value or nonvalue-added, and to which business process the activity is linked. Based on this analysis, the following could be concluded: 85 % of the total time spent on activities was consumed by the secondary activities; most of the time was spent on the processing of the requests, the executing of the administration and the control of invoices; and there were too many activities with

Table 2. Problem groups

Problem group	**Interest group**
Administration of data is not transparent	Administrative organization
Running time of administration and invoicing is too long	Management Administrative organization Finance Client
Customers' requests are not standardized*	Administrative organization Client

* By customer(s) and client(s) the internal worker(s) of the bank is (are) meant.

no added value for the customer or the organization. This demonstrated that there was a sufficient basis for the rescaling of activities in order to get a better control on costs.

Aspect-Oriented Analysis and Design

The method of Aspect-Oriented Analysis and Design is used for rescaling the activities as determined by the value analysis. This method analyzes the object system on aspects of problem, organization, processes, information and behavior. The object system analyzed was the administrative organization of the telephone costs.

For the analysis of the problem aspects of the object system, the change analysis from the ISAC approach of Lundeberg (1982) was used. For this, the problems as experienced by the diverse interest groups of the object system were summarized and grouped. From the interviews with the diverse interest groups the following groups of problems emerged (see Table 2): the administration of data was not clear, the running time of administration and invoicing was too long and the customers' requests were not transparent.

For the analysis of the organization aspects of the object system, the telephony organization was schemed using an organization chart and a responsibility matrix. Figure 4 shows the responsibility matrix for the telephony organization.

From the analysis of the organization it became clear that the activities of the telephone call system were too much fragmented over various departments. Many people were responsible for partial aspects of the telephony, but no one could be held ultimately responsible for all telephony activities.

While a lot of activities having a close connection with each other were executed separately by various departments, cooperation was being hindered and so was the running time of the processes.

To analyze the process and information aspects, all processes with the corresponding information were schemed in so-called Lano matrices (Lano, 1979). Lano matrices can be drawn up to conveniently scheme the process and information flows of the object system. Lano matrices were drawn for all main and some subprocesses. Figure 5 shows the Lano scheme for the primary process of ordering and delivering. From the analysis of the Lano matrices it appeared that a lot of information systems were stand-alone, although on the basis of the findings from the Lano matrices an exchange of data between those systems was expected. This slowed down the running time of the processes.

Primary activity	**Ordering and delivery**		**Cost charging**	
Secondary activity / *Department*	**Processing requests**	**Execution administration**	**Control invoices**	**Management expl. acc.**
Order Intake	R	I		
Administration & Configuration		R		
Financial Control			R	
Financial Accounting				R
ICT Helpdesk	I	I		
Client	I			
Supplier	I			

R = Responsible
I = Involved

Figure 4. Responsibility matrix for the telephony organization.

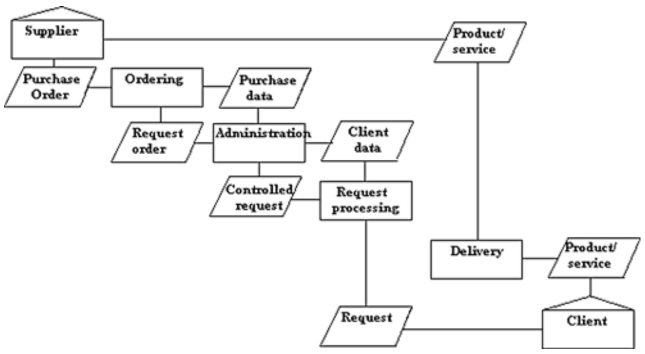


Figure 5. Lano matrix for the primary process of ordering and delivery.

To represent the behavior of the object system, methods from MERISE were used (Tardieu, 1986). MERISE is a Petri-net-based dynamic modeling technique that makes it possible to scheme the relationships between events and processes chronologically. From the findings of the MERISE schemes it appeared that many people were occupied full-time with the processing and entering of request and mutation data. In addition, there were several ways of recording a request, which resulted in a lack of clarity for customers about the request process and in difficulties in managing the request process.

According to the change analysis (Lundeberg, 1982), on the basis of the analyses and the goals of the interest groups, the needs for change per interest group were listed. The following needs for change were identified: make the recording of requests and cost charging data clearer; shorten the running time of the processes for ordering, delivery and cost charging; and make the request process more transparent for customers. To meet the needs for change several change alternatives were formulated and evaluated. On the basis of the evaluation one change alternative was selected, namely, that of embedding the solution in an infrastructure already available for the administration of IT products.

The new object system was designed in accordance with the aspects from the method of Aspect-Oriented Analysis and Design. In the new organizational structure, two new departments would be initiated, i.e. a service department and a finance department. The service department is charged with the processing of clients' requests. The finance department is charged with all money circulation concerning the telephone calls, from invoice control to cost charging. The increased clustering of activities benefits cooperation.

According to the new organizational structure, two primary processes are centralized: the service process and the accounting process. The service process deals with the requests for services (delivery of a product or a service), the processing of the service request, the delivery of the service and the administration. The accounting process ensures that incoming invoices are compared with the internal administration, and that these costs are passed on to the client.

The behavior of the new object system is designed as follows. All service requests are centralized by a web portal, accessible by clients via a personalized login. The client can enter a request for a product or a service. This can be a request for an order, a request for the solution of a problem, a request for change or a request for information. The request data will be stored in the back-portal information systems and processed by the service department. Data for an order are stored in an order system, which can automatically generate a purchase order to the supplier and receive invoices from the supplier. First, the service department checks whether the product is in stock or whether the service can be delivered internally. When the product or service can be delivered, an email is sent to the client with specific information. Data about the clients, the order and the costs is stored in another system. This system also keeps the data about the traffic categories and the corresponding calling ceilings up to date. Another system stores all telephony-related troubles as entered by the clients. The service department attends to those troubles via a helpdesk which is also accessible by telephone. All those systems are linked together for optimized data exchange. On the other hand, clients could view their data and call charges via de web portal. The web portal also shows a product catalogue with all information about telephone call related products and services.

As a result of the renewed scaling of processes and activities of the new object system, some new main activities and corresponding supporting activities could be pointed to. For comprehensive cost control, it is important that the cost price of the cost objects is built up on the basis of these activities. The main processes (service, accounting) were dedicated as the primary activities of the ABC system. The support and, therefore, the new secondary activities according to the primary-activity service were the requests for services, the processing of the service request, the delivery of the service and the administration. The secondary activities according to the primary-activity accounting were invoice control, cost charging and cost administration.

This case illustrates the way the telephone call costs at a large bank were reduced and could be better controlled by applying the model for reducing and controlling costs as described above. By sequentially following the steps of the model, insight could be gained into the costs according to the ABC system, costs could be reduced, and the activities of the ABC system could be rescaled to improve performance of these activities for cost savings to be realized. Simultaneously, the rescaling of activities brought up a need for changing the ABC system through which cost prices could be determined more exactly and costs could be managed more efficiently.

5. CONCLUSIONS

Organizations use ABC for an accurate determination of cost prices. In this context the business activities are considered as being responsible for the origin of costs. However, organizations are continuously looking for opportunities to reduce and control costs. ABM can be used to find out the causes of costs and provide a judgment on the activities. ABM is a static instrument for making assessments, however, and does not provide assistance for the improvement of activities. To this end, Aspect-Oriented Analysis and Design can be used to analyze the structuring of processes and improve this structuring in order to reduce costs and gain better control on these costs. The model is based on the assumption that ABC is applied for cost price fixing in the organization. First, the construction and development of the costs are examined to gain optimal insight into the costs. Thereupon, an activity-based management program is initiated geared to reducing costs. To this end, the costs are quantified, the causes of the costs are located and a value analysis is executed on the activities. Finally, the object system of the relevant processes and activities is analyzed on the basis of Aspect-Oriented Analysis and Design. According to this method, the following aspects regarding the object system have to be determined: problem, organization, processes, information and behavior. From a rescaling of activities it may appear that the ABC system needs some adjustment in order to get sharper cost price fixing.

The model as described above has been put into practice at the telephony organization of a large bank in Europe. From this research it appeared that the model is a practical instrument for cost reduction and cost control improvement in accordance with the activities of the ABC system. Application of the model disclosed many malfunctions of the organization of the underlying activities. Due to the improved structuring of processes, one could make considerable savings on costs of the ABC system, and maintain far better control on these costs.

REFERENCES

Armstrong, P., 2002, The costs of activity-based management, *Journal of Accounting, Organizations and Society* (27):99–120.

Cooper, R., 1990, Implementing an activity-based costing system, *Journal of Cost Management*, Spring, pp. 33–42.

Cooper, R., and Kaplan, R. S., 1988, How cost accounting distorts product costs, *Management Accounting*, April, pp. 20–27.

Cooper, R., and Kaplan, R. S., 1991, *The Design of Cost Management Systems, Text, Cases and Readings*, Prentice-Hall, Inc., New Jersey.

Cooper, R., et al., 1992, *Implementing Activity-Based Cost Management: Moving from Analysis to Action: Experiences at Eight Companies*, Institute of Management Accountants (USA) / Peat Marwick.

Glad, E., and Becker, H., 1997, *Activity-Based Costing and Management*, Juta & Company Limited, Cape Town.

Hammer, M., and Champy, J., 1993, *Reengineering the Corporation*, Harper Business.

Hixon, M., 1995, Activity-based management: its purpose and benefits, *Management Accounting (CIMA)* **73**(6):30–31.

Innes, J., and Mitchell, F., 1990, Activity based costing research, *Management Accounting (CIMA)* **68**(5):28–29.

Lano, R. J., 1979, *A Technique for Software and Systems Design Methodologies*, North-Holland, Amsterdam.

Lundeberg, M., 1982, The ISAC Approach to Specification of Information Systems and its Application to the Organisation of an IFIP Working Conference, in: *Information Systems Design Methodologies: A Comparative Review*, T. W. Olle, H. G. Sol, and A. A. Verrijn-Stewart, eds., North-Holland, pp. 173–234.

Pryor, T., and Sahm, J., 1989, *Using Activity Based Management for Continuous Improvement, A Step-by-Step Approach*, ICMS, Inc., Arlington, Texas.

van Slooten, K., and Brinkkemper, S., 1993, A method engineering approach to information systems development, in: *Information System Development Process*, N. Prakash, C. Rolland, and B. Pernici, eds., Elsevier Science Publishers, North-Holland.

van Slooten, K., 1995, Integrating method fragments for information systems analysis and design, in: *Proceedings of ECIS'95*, June, Athens.

van Slooten, K., 1996, Situated method engineering, in: *Information Resources Management Journal*, Summer, Idea Group Publishing, pp. 24–31.

Staubus, G. J., 1988, *Activity Costing for Decisions*, Garland Publishing, New York & London.

Tardieu, H., 1986, *La Méthode Merise*, Les Editions d'Organization, Paris.

ON CODING AND CODEBOOKS IN MULTIMEDIA INFORMATION SYSTEMS

Moshe Porat*

Abstract Multimedia tools are becoming an essential means in human interaction, electronic commerce and markets such as the worldwide web system. This increasing amount of information transmitted via communication channels has to be handled in an efficient manner to avoid congestion and delays in the network service. In this work, a new approach to multimedia data transmission and storage is proposed. One of its basic assumptions is that in many situations the multimedia information could be decomposed into basic components or codewords that represent a significant part of the data. To illustrate the capabilities of the proposed approach, a compression system for video streams is developed and tested. The system is based on the observations that in most multimedia systems, building blocks of recent history of the stream can describe present details of the multimedia information. A straightforward example is taking the background image of a video sequence as a codeword. However, there are more complex codewords such as typical repeated movements in the transmitted image e.g., lip and eye movements of humans in a scene. Experimental results indicate a high compression ratio of more than 100:1, obtaining almost lossless visual reproduction with signal-to-noise ratio higher than 1000:1 (30 db). Unlike MPEG or the like, the proposed approach can be similarly used also for other components in the multimedia stream. Our conclusion is that codewords and codebooks could be very efficient in developing multimedia information systems and transmission methods.

1. INTRODUCTION

Multimedia tools play an important role in visual information systems. With today's technology, a significant part of human interaction is based on transmitted voice, images and video streams over communication channels – land-lines and wireless. Some of these communication links and storage means are also becoming essential tools in electronic commerce and markets such as the worldwide web system. This increasing amount of

* Department of Electronics, Computers and Communication, Faculty of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel, mp@ee.technion.ac.il.

information transmitted via band-limited channels has to be handled efficiently to avoid congestion in the network service. Information compression is thus becoming an inherent part of most multimedia networks, with the goal of allowing nearly lossless reproduction of the original information despite a significant reduction in the transmitted data rate.

The main part of the multimedia stream is images and video, which contains more than 90% of the transmitted data. In many situations, however, the inherent redundancy in images and image sequences is relatively much higher than in speech. In fact, for many sequences, the differences between consecutive frames are often very minor. For example, in the case of video communication (videophone), where commonly the source image comprises head-and-shoulder information, the inherent redundancy is very significant. Another situation is in e-commerce where a product is shown from different angles. Under such conditions, where the background remains almost unchanged and the major relevant information relates to localized limited changes, improved techniques may be used for multimedia transmission.

Generally, there are four basic approaches to image coding: (1) entropy coding in which no loss of information is expected; (2) predictive strategies where mainly the changes are dealt with; (3) transform-based coding where a transformed version of an image is coded; and (4) cluster coding which basically relates to vector quantization (VQ).

In other cases of data coding for transmission, such as in speech coding and facsimile, it is known that better compression might be available if additional information related specifically to the source of the transmitted data is considered. Usually this was done by developing a model for the information source and coding the parameters of the model instead of the raw information. The source of data in speech, the human voice, has been described by many models. The digital facsimile can be also considered as based on modeling the scan of a typical printed page, where short sequences of black and some longer sequences of white are more common than other combinations. In the area of image coding, however, it is generally agreed that images are not created by a well defined source and thus lack common characteristics.[1] Such models as exist in the image processing area rely on general assumptions like spatial high correlation, as used in the Markov process model.[1] There are other models related to the influence of blurring and defocusing, for example, but not many attempts have been made to model image sources, except perhaps a few works related to the human face as consists of several moving objects.[2] These models are of limited use. The basic assumptions, even though stated as general as possible, do not fit into many images, and by their very nature do not rely on information related directly to specific applications for which the visual communications system is designed.

A review of more sophisticated approaches to image coding is "Second Generation Image-Coding Techniques" by Kunt et al.,[3] however, with regard to still images. One of the emphases in this review is on coding edges as an important feature of the visual information. A similar approach to coding of oriented edges is reported also by Giunta et al.[4] for the case of low bit rate coding. Those methods are based on decomposition of the sequence into several bands,[5, 6] with various techniques applied to different bands. This approach of subband decomposition is in accordance with many findings related to the basic structure of the human visual system, where cells and groups of cells are sensitive to limited spatial-frequency bands, and are likely to be parts of different processing mechanisms of the human pattern recognition systems.[7]

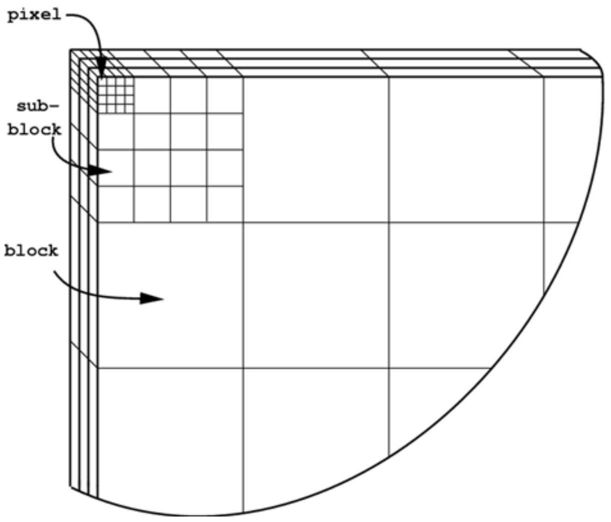


Figure 1. Blocks and sub-blocks. Each block and sub-block contains 4 consecutive frames.

Accordingly, a need exists for a multimedia coding system which encodes and communicates at a lower bit rate and uses information about the specific characteristics of the source, the viewer, and the visual system, to reproduce the information with minimal distortion. In this paper a new approach is introduced based on the above principles and motivation.

2. THE MODEL AND THE SYSTEM

The model used in this study assumes specific properties related to the nature of multimedia streams. In particular, the main assumption is that changes or events of recent history are likely to repeatedly appear in the same area of the image where they have previously appeared. If not similarly repeated (up to allowed distortion), they are likely to be encoded as transformed versions of previous details.[8] Furthermore, if a specific event cannot be matched accurately according to coarse partitioning of the frame, a more refined description can be used.[9] According to this model a frame or a frame is divided into blocks and sub-blocks as shown in Figure 1.

In this figure an image is divided into sub-blocks or vectors of $4 \times 4 \times 4$ pixels. In this figure, sixteen vectors comprise a block. Typically, an image is divided into at least 25 or 36 blocks, each of them plays a localized role in the compression of the information.

Based on the above assumptions and structure, the proposed system is presented in Figures 2 and 3.

In the first stage of the process (Figure 2) the blocks of the image after *Image Partition* are used for training localized codebooks (*CB*).[10] These codebooks are based on the recent localized history of the sequence, thus contain significant information which can be readily

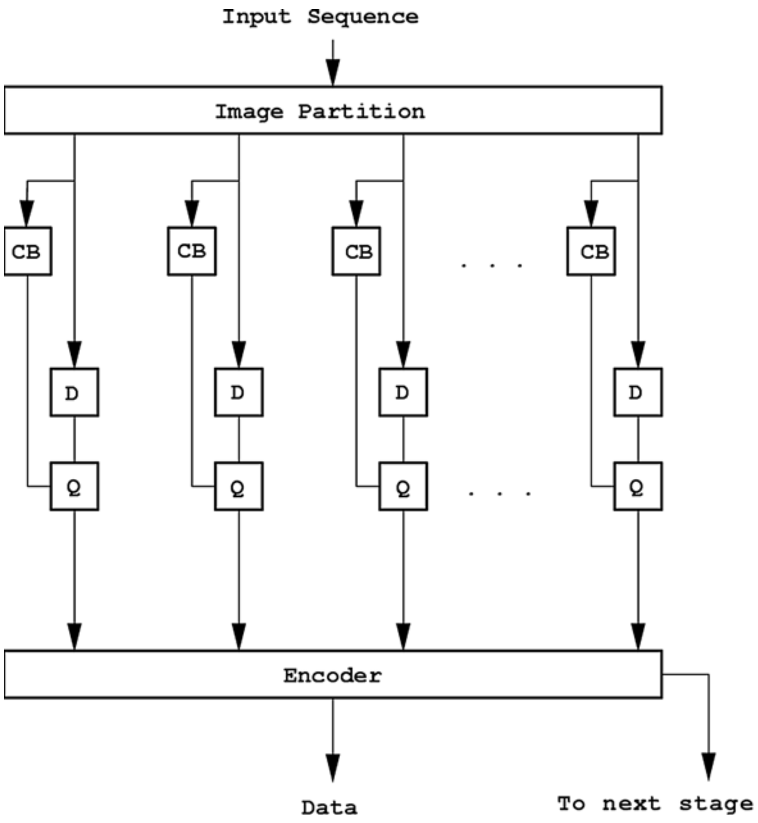


Figure 2. Block diagram of the first stage of the system. Localized codebooks (CB) are trained by the recent history of the sequence, and then used to encode the next frames. D indicates Delay, and Q represents Quantizer.

used for quantizing the next frames of the sequence. The Delay (*D*) is needed to allow updating of the codebooks before quantization (*Q*).

An illustration which indicates the size of the localized codebooks is shown in Figure 4. This example relates to a sequence of head-and-shoulder (Miss America) which represents a typical videophone interaction. In this illustration, codebooks of the background are small in size (indicated by dark grey and black) while codebooks of active areas like the head, and in particular the eyes and the mouth, are larger (light grey and white).

Most of the sub-blocks (vectors) of each block are adequately encoded in this stage by relatively sparse codebooks, with up to 512 code-words in the illustrated example. Some of the vectors, however, as shown in Figure 5, require additional attention due to distortion above a pre-determined threshold. These vectors are shown in the example of Figure 5 by black areas. Usually most of these blocks can be encoded using adjacent codebooks as shown schematically in Figure 3. It is assumed that these vectors refer to motions whichhave crossed the borders of their block (codebook) thus can be found in one of the adjacent codebooks, possibly after Transformation (*T*) of rotation or scaling.

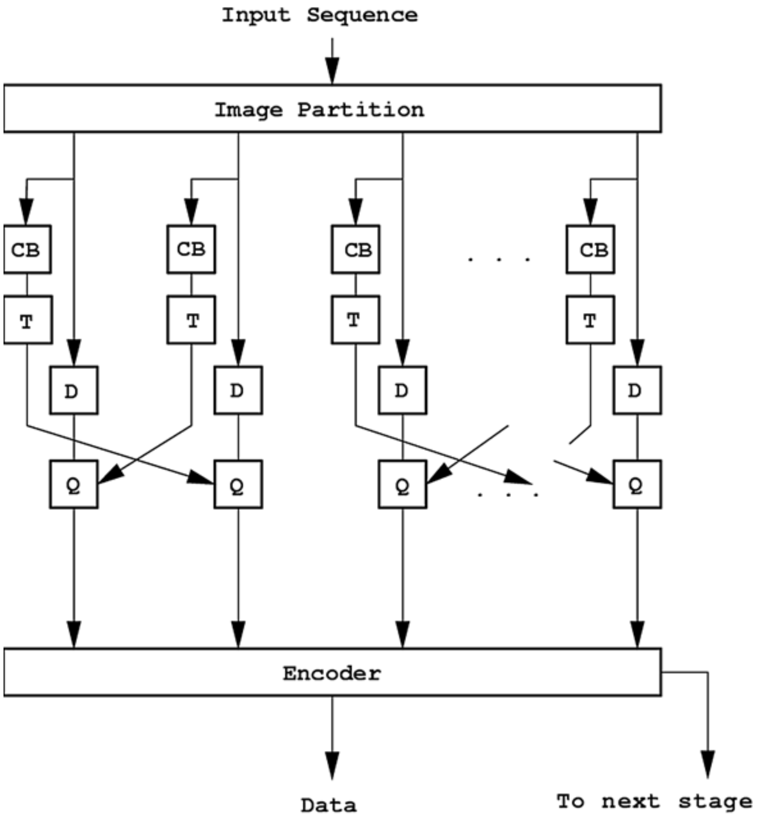


Figure 3. In the second stage of the system, adjacent localized codebooks are searched for best match, using – if required – Transformed (T) versions of codewords. CB, D and Q are as indicated in the first stage.

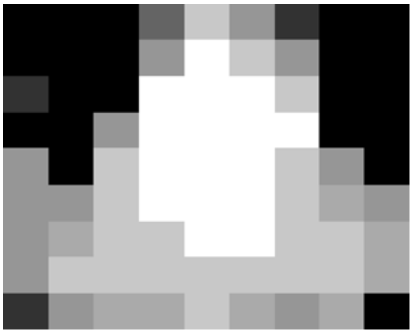


Figure 4. Example of the size of the codebooks used in the basic level of the encoder. There are 9x9 blocks in this example (head-and-shoulder scene of Miss America, see next figures), where the grey level of each block represents the number of code-words used. The dark areas use small size codebooks, brighter areas use larger codebooks. The maximum number of codewords in a codebook here is 512 (white).

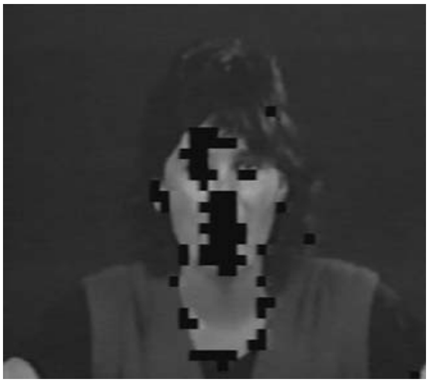


Figure 5. Reconstructed frame based on vectors found in localized codebooks, as shown in the block diagram of Figure 2. Black areas were not encoded properly in this stage and are searched for best match in adjacent codebooks in the second stage (Figure 3).

The model has several parameters. In terms of history, two parameters relate to the duration of the history used, and to the typical duration of each movement, respectively. The latter also determines the minimum expected delay of the system. A localization parameter relates to the size of areas considered as likely to contain repeated motions during the timeperiod defined as history. In the next stages of the system, a refined process is carried out with regard to those areas which were not encoded adequately in the first two stages. Usually only very small number of vectors require this additional process. A pyramidal approach that can be adopted here may introduce additional parameters, mainly the factor by which the size of the blocks is changed between adjacent levels of the pyramidal representation.

3. IMPLEMENTATION DETAILS AND RESULTS

The system was implemented according to the structure illustrated in Figures 2 and 3 with three pyramidal levels. The length of the history used for training the codebooks was of one second, with delay of 4 frames (about 1/10 of a second). To avoid the need to transmit the codebooks to the receiving end, the recent history of the *transmitted* sequence was used for training the codebooks so that the same codebooks could be created at the remote receiver, thus only an index representing each codeword was transmitted via the transmission line.

Typical results are shown in Figures 6–9. To afford comparison to previously proposed schemes, a three-dimensional vector quantization with one *global* codebook is used in the example of Figure 6. In this example small vectors of $2\times2\times4$ pixels were used obtaining high quality results at a compression ratio of 14:1. Increasing the size of the vectors to $4\times4\times4$, still with a global codebook, improves the compression without significant loss of quality: in Figure 7, the same quality is obtained at a compression ratio of 62:1. In comparison, the new model-based approach is presented in Figure 8.

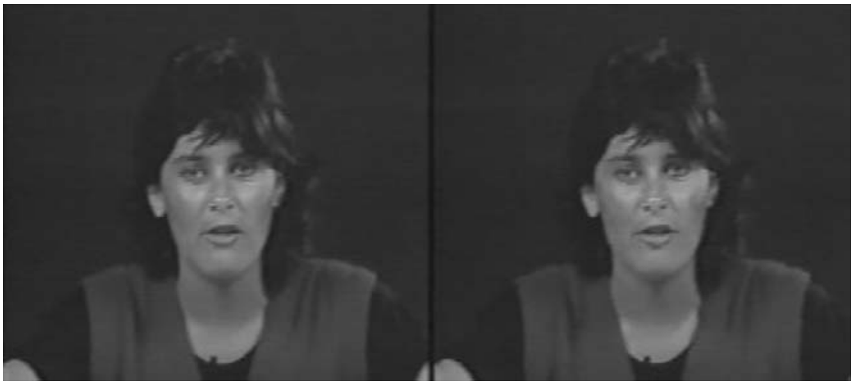


Figure 6. Original (left) and reconstructed (right) frame from the sequence Miss America, using a global codebook with $2 \times 2 \times 4$ blocks. The compression here is 14:1.

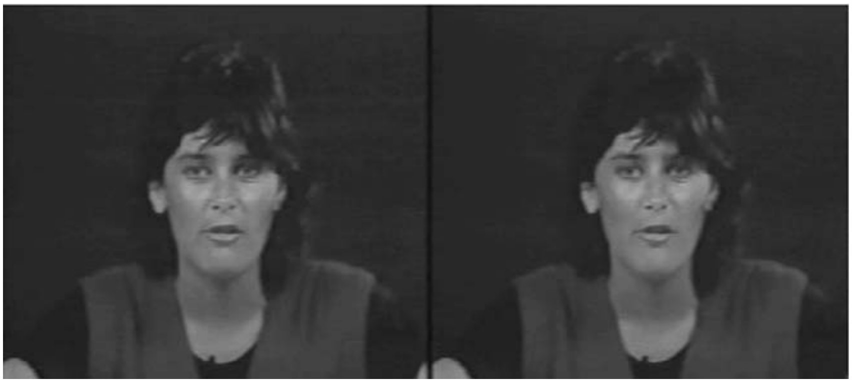


Figure 7. Original (left) and reconstructed (right) frame from the sequence Miss America, using a global codebook with $4 \times 4 \times 4$ blocks. The compression here is 62:1.

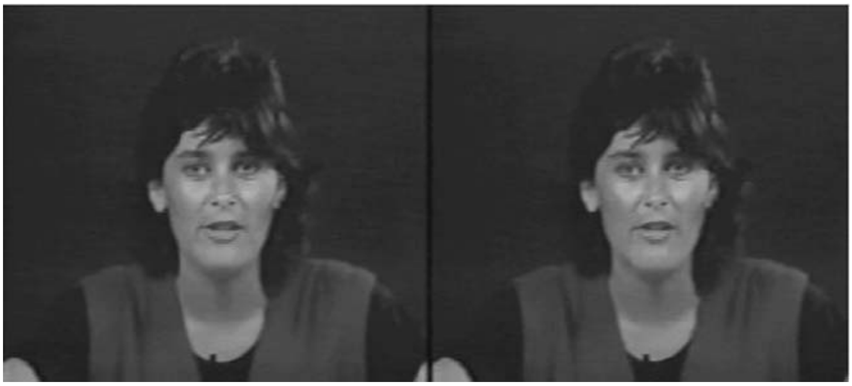


Figure 8. Original (left) and reconstructed (right) frame from the sequence Miss America, using localized codebooks with $4 \times 4 \times 4$ blocks. The compression is 100:1.

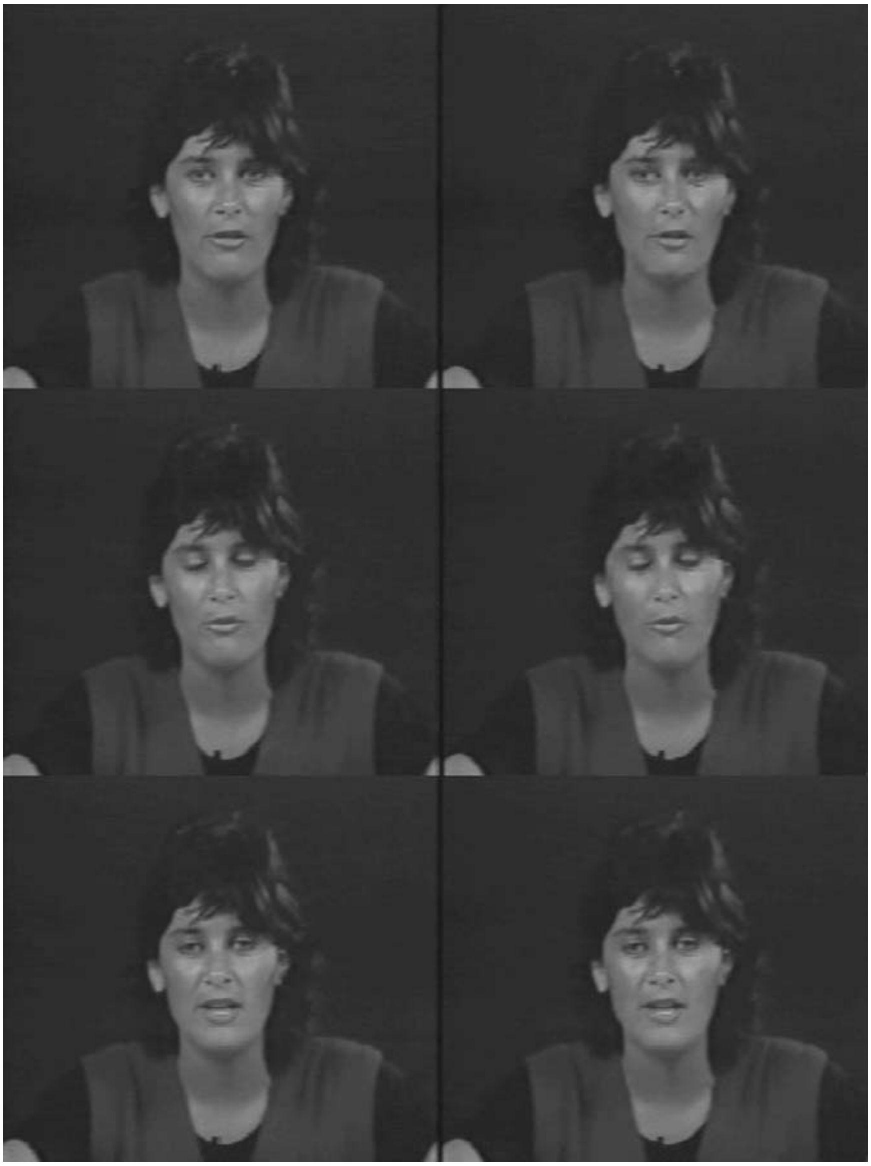


Figure 9. Comparison of the Global approach (left, 62:1) to the localized version (right, 100:1).

This localized approach provides a much higher compression ratio of more than 100:1, with the same quality of the perceived information. For additional comparison of the localized vs. the global approach, three examples of the global (left) and localized (right) versions are shown in Figures 9.

4. SUMMARY AND CONCLUSIONS

This paper has presented a new history-based approach to multimedia coding using localized history of the sequence as a training set for vector quantization. In addition to the quality of the resultant sequences, the implementation of this approach can be systematically organized in a parallel manner by its very nature since localized codebooks are created for each block and searched for independently. It should be noted that even for communication by serial machines, reduced complexity is achieved by processing of many small codebooks instead of a combined one.This approach resembles the technique described in [12, 13], however, here the codebooks and the VQ process are localized, thus providing both higher quality and more efficient results. There is also a major difference when compared to the popular MPEG system in both the use of history and in its applicability to additional components of the multimedia stream. Based on its performance, it is suggested that the new localized approach to multimedia coding be further analyzed and integrated in presently available methods.[14]

ACKNOWLEDGMENTS

This research was supported in part by the HASSIP Research Program HPRN-CT-2002-00285 of the European Commission, and by the Ollendorff Minerva Center. Minerva is funded through the BMBF.

REFERENCES

1. Special Issue on Sequence Coding, *IEEE Trans. Image Processing* (September, 1994).
2. C. S. Choi, H. Harashima, and T. Takebe, Analysis and Synthesis of Facial Expressions in Knowledge-based Coding of Facial Expressions in Knowledge-based coding of facial Image Sequences, *ICASSP*, 1991.
3. M. Kunt and M. Kocher, Second-generation image-coding techniques, *Proc. of the IEEE* **73**(4), 549–573 (1985).
4. C. I. Podilchuk, N. S. Jayant, and P. Noll, Sparse Codebooks for the Quantization of Non-dominant Subbands in Image Coding, *International Conference on Acoustic Speech and Signal Processing*, 1990, pp. 2101–2104,
5. G. T. R. Giunta, Image sequence coding using oriented edges, *Image Communication* **2**(4), 429–440 (1990).
6. C. I. Podilchuk, N. S. Jayant, and N. Farvardin, 3-D subband coding of video, *IEEE Trans. on Image Processing* **4**(2), 125–139 (1995).
7. J. W. Woods and S. D. O'Neil, Subband coding of images, *IEEE Trans. on Signal Processing* **ASSP-34**(5), 1278–1288 (1986).
8. M. Porat and Y. Y. Zeevi, The generalized gabor scheme in biological and machine vision, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-10**(4), 452–468 (1988).
9. N. Katzir, M. Lindenbaum, and M. Porat, Curve segmentation under partial occlusion, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-16**(5), 513–519 (1994).
10. M. Porat and Y. Y. Zeevi, Localized texture processing in vision: Analysis and synthesis in the Gaborian space, *IEEE Trans. on Biomedical Engineering* **BME-36**(1), 115–129 (1989).
11. Y. L. Linde, A. Buzo, and R. M. Gray, An algorithm for vector quantizer design, **IEEE-COM 28**, 84–95 (1980).
12. S. Panchanathan and M. Goldberg, Adaptive algorithms for image coding using vector quantization, *Signal Processing: Image Communication* **4**, 81–92 (1991).

13. G. Goldberg and H.-F. Sun, Image sequence coding by three-dimensional block vector quantization, *IEE Proceedings, Pt. F* **133**(5), 482–486 (1986).

14. D. Furman and M. Porat, On content-based very low bitrate video coding, *The Very-Low Bitrate Video (VLBV) Conference* (Madrid, 2003).

LINKING AND PROPAGATING BUSINESS RULE CHANGES TO IS DESIGN

Wan M. N. Wan Kadir and Pericles Loucopoulos*

1. INTRODUCTION

Among the main challenges faced by today's information system (IS) are the frequent changes of its business environment.[1] Many approaches in IS development attempt to provide techniques, and tools for producing a more resilient software system.

Experience has shown that a practical solution for the above problem is very hard to achieve unless we address the root cause for business change which in turn impacts on the need to change the support IS. One such cause is traced to *business rules*.[2] We subscribe to the view that business rule changes bring the highest impact on both software and business processes compared to other changes such as altering code for elegance or speeding execution.[3, 4] This viewpoint raises the need to explicitly consider business rules in software modeling for assisting future evolution.[5]

To date many business rule approaches have emerged that attempt to address the evolution problem. These approaches can be divided into two broadly defined areas: (a) *business rule specification* and (b) *business rule externalization*. The former aims to provide complete guidelines to capture and specify rules, whilst the latter attempts to separate business rules from other parts of software system. Whilst the field is well served in each area the issue of linking business rules specified during an analysis phase to a software architecture defined during a design phase remains open.

The aim of our research work is to bridge the gap between the two. It is based on the premise that rapid changes in a business environment need to be rapidly implemented on the support software system. By having a specification of business rules in the first place the process of the evolving these rules during analysis is greatly facilitated. If however, this business rules specification requires a laborious interpretation and analysis of the necessary changes to the software architecture, the advantages are minimized.

* Wan M. N. Wan Kadir, Software Engineering Dept., Faculty of Comp. Sci. and Info. Systems, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia (currently on research leave at UMIST). Pericles Loucopoulos, Department of Computation, University of Manchester Institute of Science & Technology (UMIST), P.O. Box 88, Manchester, M60 1QD, U.K.

The objective of this paper is to report on the theory, techniques and tools developed by our research work in addressing the business rules tracing problem. The paper demonstrates the way in which changes to business rules can be automatically propagated to object-oriented software design.

The paper is organized as follows. Section 2 reviews other works related to our research. Section 3 provides a synoptic discussion to our meta-model. Section 4 gives a brief description of the MediNET system, an industrial size system that is used as a case study in applying and explaining the techniques. Section 5 gives examples of business rules specification and software design for the case study. Section 6 discusses the way in which business rules propagation is automated. Finally, Section 7 concludes the paper with some observations and future issues related to the work presented here.

2. RELATED WORKS

Most of the business rule approaches focus on specification issues such as the typology and structure of business rules. For example, the *Business Rules Group* (BRG) classified business rules into three main types i.e. structural assertions, action assertions, and derivations.[6] Structural assertion is a statement about concept or relationship of something of importance to the business. Action assertion is concerned with the dynamic aspect of the business. It includes a conditional action, integrity constraints, and optional actions. Finally, a derivation is a derived fact or value that is created by an inference or a mathematical calculation from terms, facts, other derivations, or action assertions.

Business Rule-Oriented Conceptual Modeling (BROCOM) introduced a metamodel that formalizes business rules in conceptual modeling.[7, 8] In BROCOM, a business rule is composed of three components namely event that triggers business rules, condition that should be satisfied before an action, and action that describes the task to be done. Morgan suggested a formalization in terms of the pattern of business rule statements which is capable to be translated into formal logic.[9] There are five rule statement patterns namely basic constraint, list constraint, classification, computation, and enumeration. Morgan suggested to link business rules to a fact model which contain business objects, their relationships, and their attributes, and store the constraints of the fact model as business rules. Ross proposed the functional categories of business rules i.e. rejectors, projectors, and producers.[10] He also provided a set of rule sentence templates for specifying and capturing business rules.

There are some efforts which are related to externalization of business rules in object-oriented software development. Among the leading approaches is *Adaptive Object Model* (AOM), which is defined as "a system that represents classes, attributes, and relationships as metadata".[11, 12] It is a meta-architecture that allows users to manipulate the concrete architectural components of the model such as business objects and business rules. These components are stored in a database instead of code. Thus, a user only needs to change the metadata instead of changing the code to reflect domain changes. Simple rules such as defined types of entities, legal subtypes, relationships, and cardinality, are normally controlled by object-oriented modeling semantics. *Strategies*[13] and *RuleObjects*[14] are used to model complex rules.

Coordination Contract aims to separate core business entities which are relatively stable and volatile business products which keep changing for the business to remain competitive.[15] Volatile business products are implemented as contracts. Contract aims to externalize the interactions between objects (core entities) by explicitly defining them in the conceptual model.

Business Rule Beans (BRBeans), formerly known as Accessible Business Rules,[16, 17] is a framework that provides guidelines and infrastructures for the externalization of business rules in a distributed business application. Business rules are externally developed, implemented and managed to minimize the impact of their changes on other components such as core business, application, and user interface objects. They are implemented as server objects, which are fired by embedded trigger points in application objects. The rule management facility is provided to help users to understand the existing rules and to locate the rules when changes are required.

Diaz et al. provide method to explicitly identify, design and implement business policies in object-oriented software system.[18]

In summary, whilst approaches in both areas provide many advantages in each individual area the important question of "how do business rules changes affect the software design?" remains unanswered. This question is addressed by the `Link Model` which is part of an ongoing research project on business rules known as the MBRM (Manchester Business Rules Management) approach.[19, 20]

3. THE LINK MODEL APPROACH

One aspect of MBRM deals with the *linking of conceptual specifications of business rules to software designs*. The metamodel that has been developed to support the link of business rules specification to software designs is based on the requirements that (i) it should naturally define business rules from user's perspectives and, (ii) at the same time, be well structured enough to be implemented or linked to software design.

The three essential components that are required in order to propagate chages of business rules from conceptual domain to software domain are discussed in this section. They are: the *business rule metamodel*, the *software design metamodel* and the *linking elements* between the two.

3.1. Business Rule Metamodel

The simplified version of the business rule metamodel is shown in Figure 1. As shown in Figure 1, the metamodel classifies business rules into three main types i.e. *Constraint*, *Action Assertion*, and *Derivation*.

Constraint rules specify the characteristics of a business entity. They are used to check for the result of the execution of business events. *Constraint* rules are further divided into Attribute and Relationship constraints. The former specifies the uniqueness, optionality (null), and value check of an entity's attribute, whilst the later asserts the types, cardinality, and role of a relationship between two entities.

Action Assertion, which is also called Active or ECA rule, is a statement that concerns a dynamic aspect of the business. It specifies the action that should be activated on the occurrence of a certain event and/or on satisfaction of a certain condition. The number of

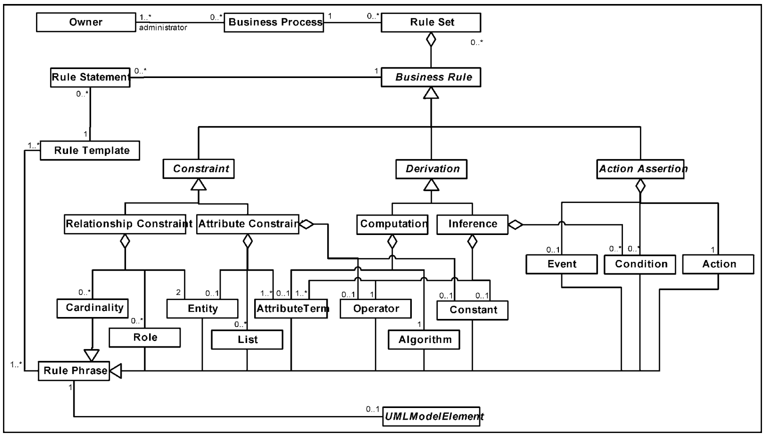


Figure 1. The metamodel.

Table 1. Business rule templates

Types	Templates
Constraint	<entity> must have <attConstraint><attributeTerm> <attributeTerm1> must be <comparison> <attributeTerm2> <attributeTerm1> must be <comparison> <constant> <attributeTerm> must be in <list> <cardinality> <entity1> is a/an <role> of [<cardinality>]<entity2> <entity1> must have <cardinality> <entity2> <entity1> is a/an <entity2>
Action Assertion	[when <event>] if <condition> then <action>
Derivation	<attributeTerm>|<value> is computed as <algorithm> if <condition> then <attributeTerm><operator><attributeTerm>|<constant>

tested events or conditions can be more than one, and they can be connected using logical connectives AND, OR, or NOT.

Derivation is a rule that derives a new fact based on the existing terms and facts. It can be divided into two types i.e. *Computation*, which uses a mathematical calculation to derive a new arithmetic value, and *Inference*, which uses logical deduction or induction to derive a new fact. *Inference* rule is also used to represent permission such as user policy for data security.

As a guideline to capture and implement rules, each type of business rule is associated with one or more *templates*. Table 1 lists the identified templates for each business rule type.

3.2. Software Design Metamodel

The Unified Modeling Language (UML)[21] metamodel is used to represent software design since it is widely accepted in research and industry communities. In general, the UML metamodel consists of three packages i.e. Foundation, Behavioral Elements and

Model Management. These packages define various models developed using UML. Although most of these models were used for MediNET design, for the purposes of the work reported in this paper, we only describe two models namely class diagram and statechart diagram.

Class diagram shows the kind of things that exist (such as classes), their internal structure, and their relationships to other things. It is often used to represent or to generate the program structure of software applications. Class diagram is specified by the Core package, which is a sub-package of Foundation package. The main model element in Core package, i.e. *Class*, has zero or more features. Feature is an abstract super-class of *Attribute* and *Operation*. An attribute is a named property of a class that describes a range of values that the instances of the property may hold. An operation is the implementation of a service that can be requested from any object of a class to affect behaviour. The semantic relationship between classes is defined by an Association element. The instances of an association are a set of tuples relating objects of the classes. Each association has at least two *AssociationEnds*. An *AssociationEnd* is an endpoint of an association, which connects the association to a class. The aggregation attribute of *AssociationEnd* defines the kind of relationship none, aggregate, or composite. Apart from *Association*, there is also *Generalization* relationship that connects a class to its super-class so that the class inherits all of the features from its super-class.

Statechart diagram models the possible states of an object or system as well as the transitions from one state to another. It specifies the sequence of states, the event that cause the transition, the guard (condition) that should be satisfied, and the action triggered by the transition. It is useful to show the lifecycle of the class with a complex behaviour.

The syntax and semantics of statechart diagram is described in the State Machines package, which is a sub-package of Behavioral Elements package. In State Machines package, *State* is a static situation such as waiting for event or dynamic condition such as performing a process. It has one or more outgoing and incoming *Transitions*. *Transition* shows the change from the first state to the second state when the specified *Event* occurred and the *Condition* satisfied. It also specifies the *Action* triggered by the state change. In the UML metamodel, each model element is a subclass of the *ModelElement* abstract class.

3.3. Linking Elements – the Rule Phrase

Rule phrases are considered as the building blocks for the rule statements. They can be maintained independently, in other words, they are not deleted when the business rule is deleted. However, the modification and deleting of a rule phrase is not recommended since a careful effort is needed in reviewing its aggregated business rules. Each rule phrase is associated with zero or more UML model elements.

Most of the rule phrases are directly connected to the class diagram model elements. However, *Event*, *Condition*, and *Action* phrases are connected to statechart diagram and consequently map to class diagram. In addition to operation specification, A*lgorithm* can also be linked to activity diagram. The last three rule phrase types i.e. *List*, *Operator*, and *Constant*, are not connected to any UML model elements. They contain either a single value or a set of enumerated values. The associations are listed in Table 2.

Table 2. The associations between rule phrases and design elements

Rule Phrase Type	Software Design Elements
Entity	Class
Attribute Term	Class.attribute
AttConstraint	Attribute.isUnique, Attribute.notNull
Cardinality	AssociationEnd.multiplicity
Role	AssociationEnd.role
Event	Transition.event → Class.operation
Condition	Transition.guard, Operation.specification
Action	Transition.action → Class.operation
Algorithm	Operation.specification
List	- (enumeration)
Operator	- (enumeration)
Constant	- (literal value)

4. THE MEDINET CASE STUDY

MediNET is an internet-based application that allows various components of the healthcare industry to exchange business data instantaneously and automates their routine administrative tasks. In general, MediNET users can be divided into three categories: *paymasters*, *healthcare providers*, and a *supplier*. Paymasters are those who pay for medical services. They use MediNET to maintain the basic parts of the patient records. Healthcare providers (HCPs) are the professionals who dispense medical treatment. HCPs perform patient records management, patient billing and paymaster invoicing. The supplier is the company that owns and maintains the MediNET applications. The supplier lets users to access MediNET applications and charges them based on the number of performed transactions.

In MediNET, there are three main business processes: patient registration, billing, and invoicing. Patient registration can be done by the HCP or the paymaster. For each visit to HCP, each patient must be registered for consultation. The consultation registration is used to validate the patient eligibility and to prepare necessary information prior to the consultation. Next, the verified patient is put in a queue. After consultation is completed, a bill is issued to the patient based on the doctor's prescriptions. Cash patients must pay their bills whilst the bills for panel patients are sorted and verified before they are inserted into an invoice as invoice items. Finally, the invoice is sent to the paymaster.

From the analysis of MediNET requirements, a number of business rules were identified. Business rule typology and templates are used as a guideline to identify and specify the business rules. Some examples of the captured business rules are shown in Table 3.

5. DESIGNING MEDINET USING THE MBRM APPROACH

During analysis, the initial business rule statements were identified as given by the examples in the previous section. Each rule statement will be further refined to a more formal or structured specification during the design phase. In this section, we present the deliverables from the MediNET design phase i.e. class and statechart diagrams, and business rules specification.

Table 3. Examples of MediNET business rules

B Rule Category	Business Rule
Attribute Constraint	The amount of a panel patient's bill must be less than the maximum bill amount set by the panel company.
Relationship Constraint	Clinic item is an item type of bill item.
Action Assertion	When the invoice is created, if the invoice interval is monthly, then set invoice end date to the end day of the month.
Computation	The amount of HCP usage invoice is computed as the sum of monthly subscription fee plus transaction fee.
Inference	A paymaster (panel company) is under probation if the paymaster has an invoice with category 1 past due and the current balance is more than RM 5,000.00.

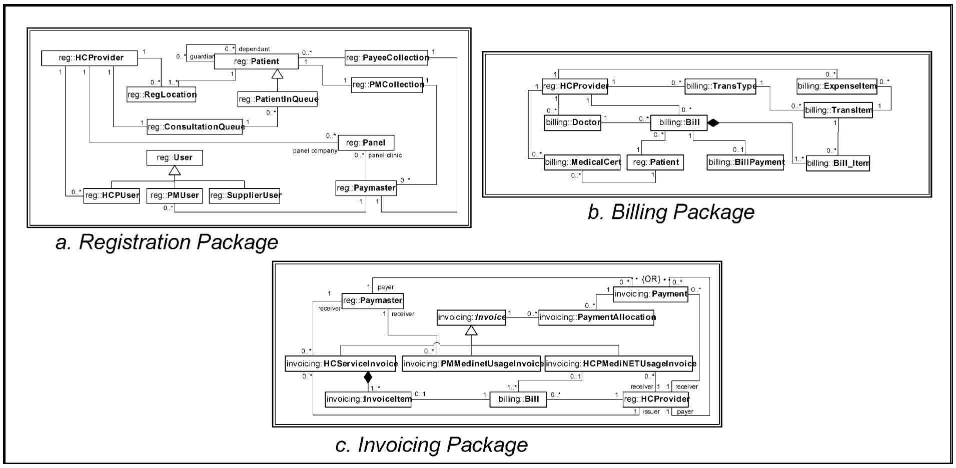


Figure 2. MediNET class diagrams for the core package.

5.1. Class and Statechart Diagrams

In Figure 2, we present the class diagrams of the *core* package i.e. a business objects layer that is commonly used by all MediNET subsystems. The core package is divided into three sub-packages: registration, billing, and invoicing. The attributes and operations of the class diagrams are suppressed to reduce the presentation complexity.

The example of statechart diagram for an invoice object is shown in Figure 3. It shows the states and transitions from the creation of an invoice object until the invoice is archived. The diagram also detailed the events and conditions for dealing with overdue payments.

5.2. Business Rules Specification

The initially identified business rules are refined to derive the rule phrases and consequently link to software design element. If necessary, the rules can be rewrite to match the

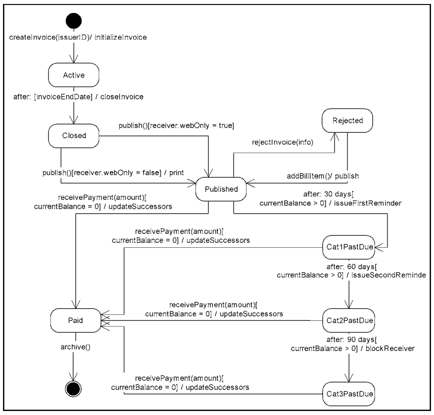

createInvoice(issuerID)/ initializeInvoice
Active
after: [invoiceEndDate] / closeInvoice
Closed
publish()[receiver.webOnly = true]
Rejected
rejectInvoice(info)
publish()[receiver.webOnly = false] / print
Published
addBillItem()/ publish
receivePayment(amount)[
currentBalance = 0] / updateSuccessors
after: 30 days[
currentBalance > 0] / issueFirstReminder
receivePayment(amount)[
currentBalance = 0] / updateSuccessors
Cat1PastDue
after: 60 days[
currentBalance > 0] / issueSecondReminde
receivePayment(amount)[
currentBalance = 0] / updateSuccessors
Paid
Cat2PastDue
archive()
after: 90 days[
currentBalance > 0] / blockReceiver
receivePayment(amount)[
currentBalance = 0] / updateSuccessors
Cat3PastDue

Figure 3. Statechart for invoice object.

Table 4. The examples of the derived rule phrases

B Rule Category	Business Rule Phrases	Software Design Elements
Attribute Constraint	<attributeTerm> = 'the amount of a panel patient's bill'	Bill.amount
	<comparison> = 'less than'	- (logical operator '<')
	<attributeTerm> = 'the maximum bill amount of the panel company'	Paymaster.maxBillAmount
Relationship Constraint	<cardinality> = 'one and only one'	- (cardinality '1')
	<entity> = 'clinic item'	TransItem (class)
	<role> = 'item type'	AssociationEnd.name
	<entity> = 'bill item'	Bill_Item (class)
Action Assertion	<event> = 'the invoice is created'	Invoice.createInvoice() (trans.event)
	<condition> = 'the invoice interval is monthly'	Invoice.invoicePlan == 'monthly' (trans.guard)
	<action> = 'set invoice end date to the end day of the month.'	Invoice.initialiseInvoice() (via trans.action)
Computation	<attributeTerm> = 'the amount of HCP Usage invoice'	HCPUsageInvoice.amount
	<algorithm> = 'the sum of monthly subscription fee plus transaction fee'	HCPUsageInvoive.calculateAmoun()
Inference	<attributeTerm> = 'a paymaster is under probabation'	Paymaster.status == 'under probation'
	<condition> = 'the paymaster has an invoice with category 1 past due' AND 'the current balance is more than RM 5,000.00'	Paymaster.setStatus().specification

available templates, to derive more suitable rule phrases, or to achieve completeness and correctness of the business rules. However, this task must be carefully done to preserve the original meaning of the rules. The examples of rule phrases derived from business rule examples from Table 3 are shown in Table 4.

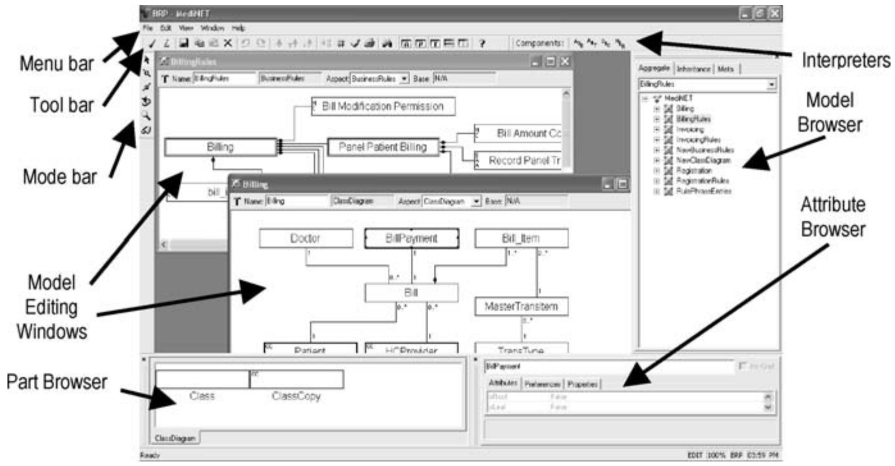


Figure 4. BRP tools main window.

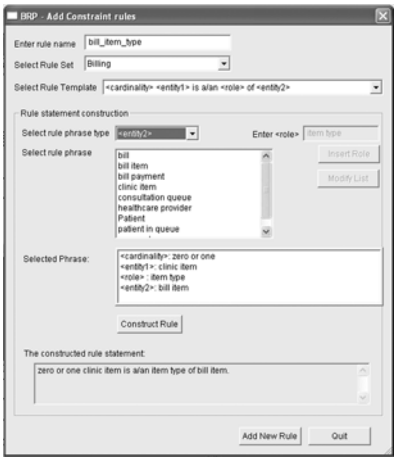
6. AUTOMATING RULE CHANGES

A key criterion for the success of propagating changes defined at the business rule specification level to software design level is the availability of appropriate tool support to automate these changes. Such a tool, known as the BRP (Business Rules Propagation) has been developed specifically to support the `Link Model`. The BRP tool was developed using the Generic Modeling Environment (GME). GME is a configurable toolset that supports the easy creation of domain specific modeling and program synthesis environments.[22] The interface of the BRP modeling environment generated by GME based on our meta-model is shown in Figure 4.

BRP allows users to create business rules or software design models by right-clicking at a particular node in the Model Browser window. The models can be edited in Model Editing windows. To add any element into the model, users may simply drag the element from Part Browser window and drop the element on the Model Editing window. The attributes of a model element can be edited using a form-based interface in Attribute Browser window. Several interpreters that manipulate the model information and provide form-based interfaces have been developed to simplify certain modeling tasks such as managing rule phrase entries, browsing rules, adding new rules and modifying existing rules.

6.1. Linking Business Rules to Software Design

The first step in linking business rules to software design involves users populating the rule phrase entries. Users must enter the phrase written in natural language as a rule phrase entry and select its type. Based on the selected type and the mapping listed in Table 2, BRP automatically lists the possible software design elements to be linked to the rule phrase. For examples, '`bill item`' is linked to `Bill_Item` class, and '`clinic item`' is linked to the amount attribute of `TransItem` class. This process is repeated until the number of entries is adequate for documenting business rules.



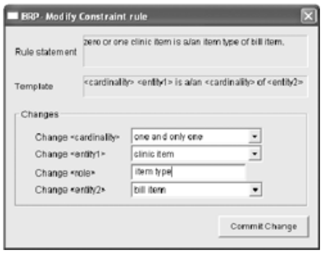
Enter rule name: bill_item_type
Select Rule Set: Billing
Select Rule Template: <cardinality> <entity1> is a/an <role> of <entity2>

Rule statement construction
Select rule phrase type: <entity2>
Enter <role>: item type
Select rule phrase:
bill
bill item
bill payment
clinic item
consultation queue
healthcare provider
Patient
patient in queue

Selected Phrase:
<cardinality>: zero or one
<entity1>: clinic item
<role> : item type
<entity2>: bill item

The constructed rule statement:
zero or one clinic item is a/an item type of bill item.

Figure 5. Adding new relationship constraint rule.



Rule statement: zero or one clinic item is a/an item type of bill item.
Template: <cardinality> <entity1> is a/an <cardinality> of <entity2>

Changes
Change <cardinality>: one and only one
Change <entity1>: clinic item
Change <role>: item type
Change <entity2>: bill item

Figure 6. Modifying Relationship Constraint Rule.

After the initial set of rule phrase entries has been populated, users may start documenting a more formal or structured business rules specification. First, a user must select the type of the rule to be added. BRP will invoke an interface based on the selected type. The interface to add Constraint rules is shown in Figure 5. Second, the name of the rule must be entered, and the rule set and template are selected. Third, a user must select rule phrases from the rule phrase types, which are listed based on the selected template. Finally, the respective buttons can be clicked to generate and store the rule. The above steps can be repeated to add a new rule and the rule phrases can be added as necessary. In the example shown in Figure 6, a business rule named `bill_item_type` with the rule statement '`zero or one clinic item is a/an item type of bill item`' is added to the `Billing` rule set.

6.2. Changing Business Rules

The most frequent task in business rule management is to modify the existing rules, which is often made by business users. Consider the previous `bill_item_type` rule as an example. This rule implies that it is not mandatory for each bill item to be associated with a clinic item as the item type. In other words, users are allowed to manually enter bill item details during a billing process. Assume that HCP management wish to change this policy in order to impose stricter control over what can be inserted in the bill item. Thus, the new rule statement would be '`one and only one clinic item is a/an item type of bill item`'. Users may add, modify, and delete rules using BRP. However, these tasks are limited to the existing rule phrases. If the users want to introduce a new phrase, they need to have some software design knowledge.

6.3. Propagating Business Rule Changes

BRP automates the propagation of most rule changes to software design. In the previous example, the cardinality of the association relationship between `Bill_Item` and TransItem classes will be automatically changed. The changes of different types of rules trigger different types of software design changes.

Constraint rule changes are propagated to object relationships and attributes. BRP will determine whether to change the attribute uniqueness, attribute optionality, relationship type (association, aggregation, or inheritance), the role name, or the cardinality based on the rule template and the changed rule phrase(s).

Action assertion rule changes are not directly propagated to a class diagram. Rather, they are propagated to statechart diagram, which is consecutively propagated to class diagram. The changes of event, condition, or action in an action assertion rule are respectively propagated to event, guard, and action of a state transition in the statechart diagram. Based on our case study, condition is the most common and frequently changed component of action assertion rules.

Derivation rules describe the way to compute a certain value or to derive a new fact. Thus, their changes should be propagated to a specific operation of a class. Currently, BRP simply links the derivation rules to operation specification, which can be written in any chosen specification language.

7. CONCLUSION AND FUTURE WORKS

We found that our approach is practical in facing the challenges of volatile business environment since it deals with the source of changes, i.e. business rules, and links to the widely accepted UML models. Moreover, the application of the approach in the MediNET case study, and its implementation using the BRP toolset, indicate that this approach is useful and practical in dealing with business rule changes. Using our approach, the changes can be managed prior to the implementation of IS.

With regard to future works, we highlight two efforts that are currently under way with the view to improving the current status of the theory and practice. First, the metamodel and templates could be enhanced via the experiences from more case studies. Even though we have extensively tested the proposed templates using the MediNET case study, we strongly believed that more case studies may prove to be useful in improving the current templates and rule phrases. Second, the supported tools could be made even more practicable by including additional features such as code generation and reverse propagation. Although it is hard to achieve the absolute code generation, it may improve the time to market, consistencies, and provide code traceability. Reverse propagation is concerned with the effect of changing software design to business rules and rule phrases.

REFERENCES

1. K. H. Bennett, M. Munro, N. Gold, et al., An Architectural Model for Service-Based Software with Ultra-Rapid Evolution, in: *IEEE International Conference on Software Maintenance* (Florence, Italy, 2001), pp. 292–300.
2. P. J. Layzell and P. Loucopoulos, A Rule Based Approach to the Construction and Evolution of Business Information Systems, in: *IEEE Computer Society Conference on Software Maintenance* (Phoenix, Arizona, USA, 1988), pp. 258–264.
3. N. Chapin, J. E. Hale, K. M. Khan, et al., Types of software evolution and software maintenance, *Journal of Software Maintenance and Evolution: Research and Practice* **13**(1), 3–30 (2001).
4. L. Andrade and J. Fiadeiro, Coordination Technologies for Managing Information System Evolution, in: *13th Conference on Advanced Information Systems Engineering* (Interlaken, Switzerland, 2001), pp. 374–387.
5. P. Loucopoulos, B. Theodoulidis, and D. Pantazis, Business Rules Modelling : Conceptual Modelling and Object-Oriented Specifications, in: *IFIP WG8.1 Working Conference on the Object-Oriented Approach in Information Systems* (Quebec City, Canada, 1991).
6. D. Hay and K. A. Healy, Defining Business Rules ~ What Are They Really?, the Business Rules Group Technical Report Rev 1.3, 2000 (unpublished).
7. H. Herbst, Business Rules in Systems Analysis: A Meta-model and Repository System, *Information Systems* **21**(2), 147–166 (1996).
8. H. Herbst, *Business Rule-Oriented Conceptual Modeling* (Physica-Verlag, Germany, 1997).
9. T. Morgan, *Business Rules and Information Systems: Aligning IT with Business Goals* (Addison-Wesley, Boston, MA, 2002).
10. R. G. Ross, *Principles of the Business Rule Approach* (Addison-Wesley, Boston, USA, 2003).
11. D. Riehle, M. Tilman, and R. Johnson, Dynamic Object Model, Dept. of Computer Science, Washington University Technical Report WUCS-00-29, 2000 (unpublished).
12. J. W. Yoder, F. Balaguer, and R. Johnson, Adaptive Object Models for Implementing Business Rules, in: *Third Workshop on Best-Practices for Business Rules Design and Implementation, Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2001)* (Tampa Bay, Florida, USA, 2001).
13. E. Gamma, R. Helm, R. Johnson, et al., *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley, 1995).
14. A. Arsanjani, Rule Object: A Pattern Language for Adaptable and Scalable Business Rule Construction, in: *7th. Pattern Languages of Programs Conference* (Monticello, Illinois, USA, 2000).
15. L. Andrade and J. Fiadeiro, Evolution by Contract, in: *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications 2000, Workshop on Best-practice in Business Rules Design and Implementation* (Minneapolis, Minnesota USA, 2000).
16. I. Rouvellou, L. Degenaro, K. Rasmus, et al., Extending Business Objects with Business Rules, in: *33rd International Conference on Technology of Object-Oriented Languages and Systems (TOOLS Europe 2000)* (Mont Saint-Michel/St-Malo, France, 2000), pp. 238–249.
17. I. Rouvellou, I. Degenaro, K. Rasmus, et al., Externalizing Business Rules from Enterprise Applications: An Experience Report, in: *Conference on Object-Oriented Programming, Systems, Languages, and Applications* (Denver, Colorado, 1999).
18. O. Diaz, J. Iturrioz, and M. G. Piattini, Promoting business policies in object-oriented methods, *Journal of Systems and Software* **41**(2), 105–115 (1998).
19. P. Kardasis and P. Loucopoulos, Managing Business Rules during the Requirements Engineering Process in Rule-Intensive IT Projects, in: *6th International Conference on Business Information Systems (BIS 2003)* (Colorado Springs, Colorado, U.S.A., 2003), pp. 239–247.
20. W. M. N. Wan Kadir and P. Loucopoulos, Relating Evolving Business Rules to Software Design, in: *International Conference on Software Engineering Research and Practice (SERP)* (Las Vegas, Nevada, USA, 2003), pp. 129–134.
21. OMG, *UML Specifications ver 1.5* (Object Management Group, 2003).
22. A. Ledeczi, M. Maroti, A. Bakay, et al., The Generic Modeling Environment, in: *Workshop on Intelligent Signal Processing*, (Budapest, Hungary, 2001).

USING ENTERPRISE MODELING FOR IDENTIFICATION AND RESOLUTION OF HOMONYM CONFLICTS IN VIEW INTEGRATION

Peter Bellström*

1. INTRODUCTION

Schema integration has been a research area since the late 1970s and many methods have been proposed (e.g. Batini et al., 1984; Lee and Ling, 2003; Navathe and Gadgil, 1982; Shoval, 1990). A comprehensive survey of methods and related work is found in Batini et al. (1986). During the years two main paths of schema integration has been developed: *view integration* and *database integration*. Kohler et al. (2000) divide database integration into two additional paths: the data warehouse approach and the database federation approach. Batini et al. (1986) defines schema integration as "[. . .] the activity of integrating the schemas of existing or proposed databases into a global, unified schema." (p. 323). A more precise and explaining definition is given by Boman et al. (1997): "In the context of information systems development, the integration of a number of local schemas into a global one is called *view integration*. A similar activity, called *database integration*, occurs in distributed database design, where a set of schemas for existing information systems is merged into a single global schema." (p. 150, italic in original).

The differences between the two paths are that view integration is a task in conceptual database design and database integration is a task in distributed database design. View integration focus on integrating all the local schemas into one global conceptual schema and database integration focus on providing a global view of all the existing databases. One common task in all integration methods, independent of path and approach, is to identify and resolve conflicts, like for instance homonyms, synonyms, type and dependency conflicts (e.g. Bhargava and Beyer, 1992; Embury et al., 1999; Lawrence and Barker, 2001; Larson et al., 1989; Lee and Ling, 2003; Li and Clifton, 1993; Navathe and Gadgil, 1982; Parent and Spaccapietra, 1998; Sheth and Kashyap, 1992). These conflicts arise because end-users have different references and different vocabulary. Johannesson (1993) explains the problem in the following way: "Fundamental to most problems with schema integration

* Division for Information Technology, Karlstad University, SE-651 88, Karlstad, Sweden, Peter.Bellstrom@kau.se.

is that a Universe of Discourse can be modelled in many different ways. The same phenomenon may be seen from different levels of abstraction, or represented using different properties. Different terms can denote the same concept, and different modelling structures can represent the same reality." (p. 10).

The aim and organization of this paper is:

- to investigate how to identify and resolve homonym conflicts in view integration using enterprise modeling
- to discuss a subset of enterprise modeling and investigate how it can be used as a tool for conceptual database design and view integration
- to discuss conflicts that appear in view integration with focus on homonym conflicts
- to discuss resolution techniques for homonym conflicts identified during view integration

2. USING ENTERPRISE MODELING FOR CONCEPTUAL DATABASE DESIGN AND VIEW INTEGRATION

Today there exist many methods, including drawing techniques, for conceptual database design (e.g. Chen, 1976; Engels et al., 1992; Teorey et al., 1986). One of the first, which also focused on semantic modeling, was the entity-relationship model with its entity-relationship diagrams proposed by Chen (Chen, 1976) in the mid 1970s. The entity-relationship model is one of the most popular conceptual database design tool existing today (Fahrner and Vossen, 1995) and it has been extended and used in several database design methods (e.g. Engels et al., 1992; Teorey et al., 1986). Two reasons for its popularity is its use for communicating different definitions of data and relationships with end-users (Teorey et al., 1986) and its use of concepts, naturally occurring in database design and information systems (Lee and Ling, 2003). Since view integration is part of conceptual database design, and also the first path in the definition of schema integration, the success and progress of it is close connected to the method used in this design step. Although the entity-relationship model has been widely used and is easy to understand it has been criticized like for instance for its generality of the relationship concept (Engels et al, 1992). A more generic problem with many of the existing methods is that they tend to focus only on the implementation level and the technical system part of the future information system and database (Gustas and Gustienė, 2002). One method that takes a broader approach is enterprise modeling. It deals with modeling and integration of business processes of the organization, with its technical parts, that is in focus (Vernadat, 1996). Enterprise modeling use three levels (a pragmatic, a semantic and a syntactic) to model the future information system and database. It can be viewed as a generalization and an extension of system analysis and design. The main idea is to end up with a complete specification of the desired information system and database. The specification should also be coherent and consistent (Gustas and Gustienė, 2003). A more comprehensive discussion about the three levels and how they can be used is found in Gustas and Gustienė (2003). Enterprise modeling has also been used for extending other methodologies. One example of that is found in Gustas and Gustienė (2002) and an example of research carried out today is found in Jacobsson and

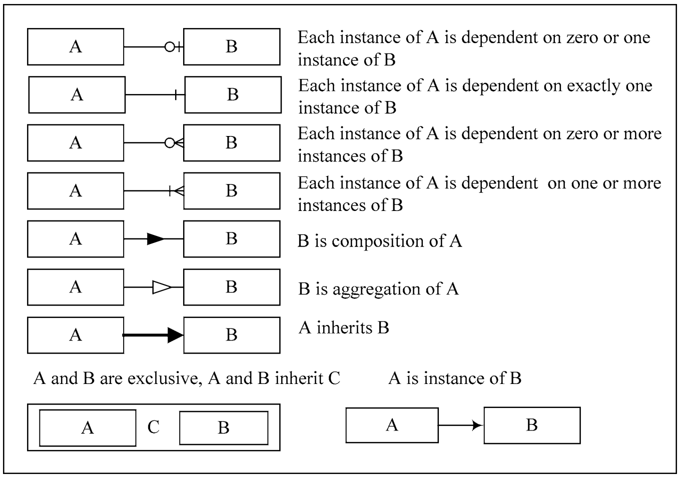

A
B
Each instance of A is dependent on zero or one instance of B
Each instance of A is dependent on exactly one instance of B
Each instance of A is dependent on zero or more instances of B
Each instance of A is dependent on one or more instances of B
B is composition of A
B is aggregation of A
A inherits B
A and B are exclusive, A and B inherit C
A is instance of B
C

Figure 1. Representation of static dependencies in enterprise modeling (Gustas and Gustienė, 2003, p. 82).

Gustas (2004). The second path of schema integration is database integration. In the rest of this paper the definition given by Boman et al. (1997) is applied when talking about it. Database integration takes place during the distributed database design and has the same problems as view integration – conflicts.

How to use enterprise modeling for conceptual database design and view integration is discussed and the representation of static dependencies used in enterprise modeling is therefore illustrated in Figure 1. A reason to use enterprise modeling for this task is its comprehensive way of representing the static dependencies. One dependency of special interest is *instance of* which is often missing in the traditional conceptual database design methods like for instance the traditional entity-relationship model.

In conceptual database design the organization in focus is defined and a conceptual schema is the final result. Chen (1976) uses the synonym term diagram. One way to do this is to involve the customer with its end-users and define smaller schemata, so called views, for each end-user or user-group. These views are then integrated into one global conceptual schema using some kind of method (e.g. Batini et al., 1984; Lee and Ling, 2003; Navathe and Gadgil, 1982; Shoval, 1990). According to Batini et al. (1986) any integration methodology is a mixture of the following four activities: *preintegration*, *comparison of the schemas*, *conforming the schemas* and finally *merging and restructuring*.

Let us now assume that two end-users (E1 and E2) have defined their view of the concepts: *Item*, *Product* and *Type*. The resulting view is illustrated in Figure 2. The interpretation of Figure 1 is as follows. Each *Product* is dependent on one or more *Items* and each *Item* is dependent on exactly one *Product*. Each *Product* is dependent on exactly one *Type* and each *Type* is dependent on one or more *Products*.

Since E1 and E2 define the concepts exactly the same, at least on the surface, the view integration is not a problem. There exists no conflict between them therefore it is just to use it as the integrated and global schema. Let us now assume that E1 and E2 have

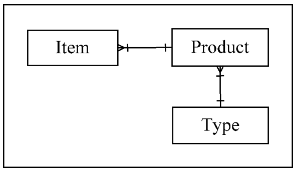


Figure 2. View one for end-user one (E1) and end-user two (E2).

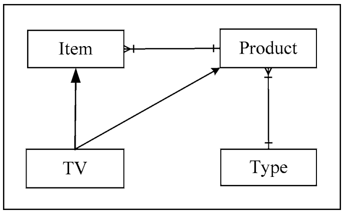


Figure 3. View two for end-user one (E1).

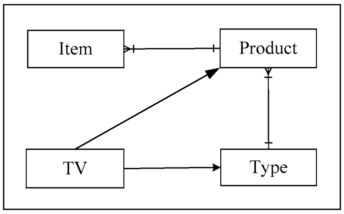


Figure 4. View two for end-user two (E2).

extended their schema with a forth concept called *TV*. The resulting views are illustrated in Figure 3 and Figure 4. Studying and comparing the two views it is clear that there exist several homonym conflicts between them. One example is the usage of the concept *TV*. By introducing it E1 and E2 define *TV* differently and two additional dependencies are also used in both views: *inherits* and *instance of*.

The interpretation of the concepts *Item*, *Product* and *Type* in Figure 3 is the same as in Figure 2. The differences appear when interpreting the concept *TV* and the dependencies connected to it. Each *TV* inherits *Item*, *TV* is as subtype to *Item*, and at the same time *TV* is an instance of *Product*.

The interpretation of the concepts *Item*, *Product* and *Type* in Figure 4 is also the same as in Figure 2. The differences appear when interpreting the concept *TV* and the dependencies connected to it. Each *TV* inherits *Product*, *TV* is as subtype to *Product*, and at the same time *TV* is an instance of *Type*.

By studying and comparing Figure 3 and Figure 4 several conflicts is immediately identified through the use of different dependencies. One example of this is the dependencies used in Figure 3 and Figure 4 for the *TV* concept. In Figure 3 it is defined as a subtype to *Item* and in Figure 4 it is defined as a subtype to *Product*. Conflicts must be resolved

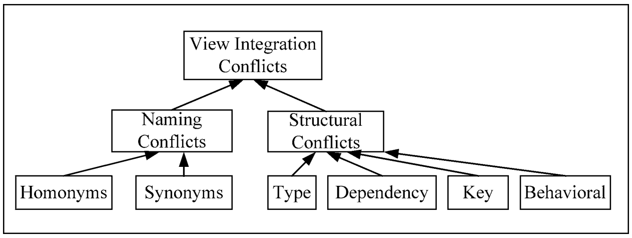


Figure 5. View Integration Conflicts.

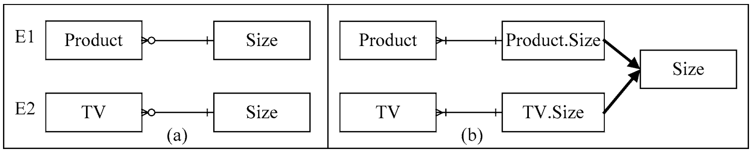


Figure 6. Homonym conflict before and after resolution.

before a merge of the local schemata can be done. The next step is therefore to study the local schemata and identify conflicts between them.

3. CONFLICTS IN VIEW INTEGRATION

As already mentioned one of the tasks in schema integration is *comparison of the schemas* (Batini et al., 1986). According to Johannesson (1993) this task is divided into three activities: *name comparison*, *structural comparison* and *identification of interschema properties*. The two first of these three activities match the conflict classification given by Batini et al. (1986) and the third activity, identification of interschema properties, concerns concepts that are not exactly the same but are related to each other by some constraints. Many conflict classifications have been proposed by researchers (e.g. Batini et al., 1984; Dupont, 1994; Fang et al., 1991; Kim and Seo, 1991; Sheth and Kashyap, 1992; Shoval, 1990; Spaccapietra and Parent, 1991). The largest difference between them is which perspectives the conflicts are classified according to. In this paper the classification given by Batini et al. (1986) is used. The reason for this is that it is well known and gives an illustrating picture over the conflicts that can appear during view integration. Batini et al. (1986) divide the conflicts into naming and structural conflicts and each of these are further refined into smaller conflicts. Figure 5 illustrates the conflict classification as an inheritance hierarchy.

Homonyms are conflicts that appear if one name is used for two or more concepts. One example of a homonym conflict is when end-user one (E1) use *Size* to define the *Product* volume *size* and end-user two (E2) use *Size* to define the *TV Size* in inches (Figure 6 (a)). Figure 6 (b) illustrates a schema where this conflict has been resolved by using prefixing. In Subsection 3.1 a more comprehensive discussion about this conflict takes place.

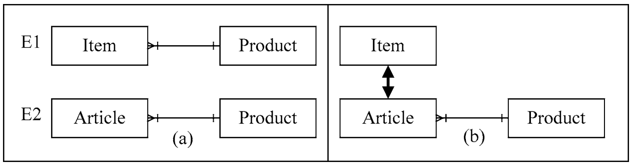


Figure 7. Synonym conflict before and after resolution.

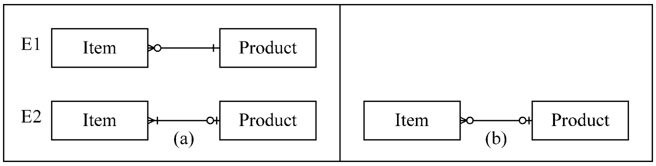


Figure 8. Dependency conflict before and after resolution.

The resolution technique used in the homonym conflict is called prefixing. In enterprise modeling prefixing is used for compound concepts to unambiguously identify a concept in a schema. A definition for prefixing is: "The operation of prefixing B in the context of A can be denoted by A.B if and only if concept B is dependent by any kind of the simple static dependency from A. The compound concept A.B defines the subset of concept B instances which are associated with one instance of A." (Gustas, 1997, p. 163).

Synonyms are conflicts that appear if two or more names are used for the same concept. One example of a synonym conflict is when E1 use *Item* and E2 use *Article* to define the same concept (Figure 7 (a)). Figure 7 (b) illustrates a schema where the conflict has been resolved by using mutual inheritance.

The resolution technique used in the synonym conflict is mutual inheritance and a definition is: A◀━━▶B if and only if A ━━▶B and B ━━▶A (Gustas, 1997).

Type conflicts appear when E1 and E2 use different modeling constructs for the same concept. This type of conflict does not appear using enterprise modeling. The reason for this is that every new concept is modeled as a box and nothing else. This is another reason to use enterprise modeling instead of a traditional conceptual database design model like for instance the entity-relationship model.

Dependency conflicts appear when E1 and E2 use different dependencies defining the same concepts. An example of dependency conflict is when E1 define the dependency between *Item* and *Product* as each *Item* is dependent on exactly one *Product* and each *Product* is dependent on zero or more *Items*. E2 define the same dependency as each *Item* is dependent on zero or one *Product* and each *Product* is dependent on one or more *Items* (Figure 8 (a)). Figure 8 (b) illustrates a schema where the conflict has been resolved by introducing semantic weaker dependencies between the concepts.

The resolution technique for the dependency conflict is to introduce semantic weaker dependencies for both E1 and E2 (Parent and Spaccapietra, 1998). In this case this is the only way to resolve the conflict since E1 uses a weaker dependence for *Item* and E2 a weaker dependency for *Product*.

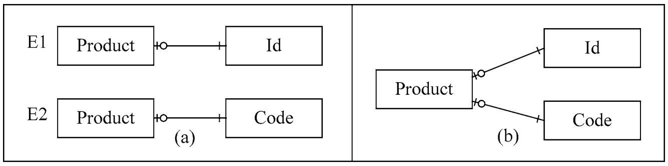


Figure 9. Key conflict before and after resolution.

Key conflicts appear when E1 and E2 use different keys for the same concept. An example of key conflicts is when E1 use a company standardized *Product* key, *Id*, and E2 use an invented *Product* key, *Code* (Figure 9 (a)). Figure 9 (b) illustrates a schema where the conflict has been resolved by including both *Id* and *Code* into the schema.

The resolution technique for the key conflict is to include both keys in the integrated schema. This is the case because to choose which key that is to be used for implementation is a task for logical database design which is out of the scope of this paper.

Behavioral conflicts appear when E1 and E2 use different policies for insertion and deletion. This type of conflict only appears when the used model supports this kind of behavioral properties. Behavioral conflicts are out of the scope of this paper since it does not take dynamics and behavioral aspects into account.

3.1. Homonym Conflicts in View Integration

A homonym conflict is classified as a name conflict, which is also illustrated in Figure 5. This type of conflict appears when the same name is used for two or more concepts. Several homonym conflicts are identified through the usage of different dependencies comparing Figure 3 and Figure 4. First, in Figure 3 the concept *TV* is defined as a subtype to *Item* and as an instance of *Product*. In Figure 4 the concept *TV* is defined as a subtype to *Product* and as an instance of *Type*. Second, in Figure 3 the concept *Type* is defined as a *Type* in general containing all the *Types* that are to be stored. In Figure 4 the concept *Type* is defined as a *TV Type* and nothing else. Third, in Figure 3 the concept *Product* is defined to contain only *TV*. In Figure 4 *Product* is defined as a *Product* in general containing all the *Products* that are to be stored. Finally, in Figure 3 the concept *Item* is defined as *Item* in general containing all the *Items* that are to be stored. In Figure 4 the concept *Item* is defined as Product *Items*.

The homonym conflict is well known and often mentioned by researchers and in the literature (e.g. Batini et al., 1986; Boman et al., 1997; Johannesson, 1993). The type of homonym conflicts that can appear depends on the chosen drawing technique (e.g. entity-relationship model and enterprise model) and the level of abstraction that the view integration is going to be performed at (e.g. conceptual or logical level). For instance using entity-relationship diagrams, on the conceptual level, homonym conflicts can arise for entities, relationships and attributes. Using the enterprise model homonym conflicts can only arise for concepts. One large difference between the entity-relationship model and the enterprise model is that the entity-relationship model uses a general relationship concept and the enterprise model uses a more specific one which helps to minimize the ambiguity. In enterprise modeling the relationship concept is classified into association, composition, aggregation, specialization, generalization or instance of (Figure 1). Another large difference between the entity-relationship model and the enterprise model is that in enterprise modeling type conflicts can not appear since every concept is modeled as a box. If it is the logical level, with the relationship model in focus, homonym conflicts can arise for table and attribute names (Kim and Seo, 1991).

Since the homonym conflict is well know, many methods to identify it has been proposed. A general way to identify homonym conflicts is for instance found in Batini and Lenzerini (1984) where "concept unlikeness" is used. A concept unlikeness is "[. . .] every situation in which two concepts (entities, attributes, relationships) have the same name and different related modeling features [. . .] in two schemata." (ibid, p. 656). Similar ways to identify homonym conflicts is suggested by Navathe and Gadgil (1982) where they compare key and non-key attributes of two objects looking for incomplete match and in Batini et al. (1992) where they focus on finding different properties and constraints.

Larson et al. (1989) suggests a more detailed way to identify homonym conflicts where they search for attribute equivalence by comparing attribute values and domains. A similar way is found in Li and Clifton (1993) where they compare field specifications to determine attribute equivalence. Sheth and Kashyap (1992) use a dual perspective that takes both semantic similarities and schematic differences into account searching for conflicts and finally Bhargava and Beyer (1992) use semantic information about the concepts to identify name conflicts.

One justification to the study in this paper is that the enterprise model has a comprehensive set of graphical symbols to model the static dependencies; the *instance of* dependency is of special interest. Another is that every concept in the enterprise model is modeled as a box which eliminates at least one conflict, the type conflict, compared with the entity-relationship model and in enterprise modeling both static and dynamic dependencies, behavioral aspects, can be modeled in one schema. Although it is important to point out that dynamic dependencies and behavioral aspects is outside the scope of this paper. Finally in enterprise modeling a concept can be interpreted differently depending on dependencies in focus (Gustienė and Gustas, 2002). An example of that is found in Figure 10 where the concept *TV* can be interpreted both as an object and as a class.

As discussed above there exist many methods for identification of naming conflicts in view integration. Although the differences between them every method compares, one way or another, two names from two different views trying to identify conflicts.

4. RESOLUTION OF HOMONYM CONFLICTS IN VIEW INTEGRATION

Homonym conflicts arise because the same name is used to define two or more concepts. The risk for this conflict is higher when the vocabulary of terms is small (Batini et al., 1992). One reason for homonym conflicts is the use of incomplete names for the concepts (Kim & Seo, 1991) and therefore traditional resolution techniques focus on renaming. Although this type of conflict is often mentioned in the literature the methods suggested to resolution is not always clear. The resolution technique is often renaming and nothing further. Table 1 summarizes three traditional resolution techniques with reference examples to each of them.

Table 1. Traditional resolution techniques for homonym conflicts

Conflict resolution technique	**Reference examples**
Renaming one or more names	Batini et al., 1984; Dupont, 1994; Johanesson, 1993; Larson et al., 1989; Shoval, 1990
Prefix the names	Engels et al., 1992; Parent and Spaccapietra, 1998
Standardization of names	Lawrence and Barker, 2001

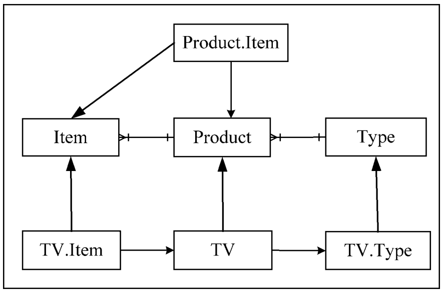


Figure 10. An integrated schema merged from the views in Figure 3 and Figure 4.

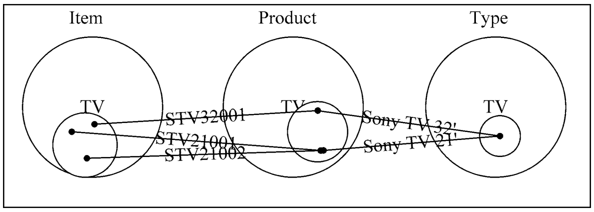


Figure 11. A clarifying example of Figure 10.

The resolution techniques found in Table 1 works for the homonym conflict discussed in Figure 6. But how about the conflicts identified comparing Figure 3 and Figure 4 where several homonym conflicts exist? Traditional resolution techniques change the names by the usage of prefixing which is also illustrated in Figure 10. By integrating the two views the system boundary move and dependencies change.

Since *TV*, in Figure 3, is defined as a specialization of *Item* it becomes *TV.Item* and *TV*, in Figure 4, remains unchanged. The concept *Type* defined in Figure 3 remains unchanged and *Type* defined in Figure 4 becomes *TV.Type*. At the same time it becomes a specialization of *Type*. The reason for this is that *TV.Type* only contains *TV* and *Type* contains all the *Types* of interest. The instance of dependency in Figure 3, between *TV* and *product*, is moved to the specialization *TV* since *Product* in Figure 3 only define *TV* and *Product* in Figure 4 is more general and defines all *Products* of interest. The concept *Item* defined in Figure 3 remains unchanged and the concept *Item* defined in Figure 4 becomes *Product.Item* and at the same time becomes a specialization of *Item*. The dependency, between *Product.Item* and *Product*, is also changed to instance of. Figure 11 clarifies the integrated schema in Figure 10 by using the TV example.

Figure 11 should be interpreted as follows. *TV.Type* is the *TV Type* in general. *TV.Product* (*TV* in Figure 10) is an instance of *TV.Type* and includes all *TV Products* (e.g. Sony TV 32' and Sony TV 21'). *TV.Item* is an instance of *TV.Product* and includes all the *TV Items* produced for sale e.g. Sony TV 32' identified with STV32001 or two Sony TV 21' identified with STV21001 and STV21002.

After integration of Figure 3 and Figure 4 all concepts have unique names and the homonym conflicts have been resolved. One problem that still exists is the ambiguity regarding a few of the concept names used in the Figure 10. The usage of the concept *TV* (*TV.Item*, *TV*, *TV.Type*) is an example of that. Since Figure 10 has concepts with names that still are ambiguous an integration of Figure 3 and Figure 4 is not to be purposed without consideration of the differences between them.

5. CONCLUSIONS

In conceptual database design and view integration some method including a drawing technique has to be used. By using a traditional modeling technique without the instance of and inherits dependencies the customer, with its end-users, and the developer would probably agree on the definitions found for the concepts *Item*, *Product* and *Type* in Figure 2. This indicates that the primitives used in these methods are not enough. One way to resolve, at least improve, the problem is to introduce a method that use these like for instance enterprise modeling. The end-users can then better discuss, explain and define the meaning of each used concept using both natural language and a view or a schema.

Several reasons to use enterprise modeling for conceptual database design and view integration has been identified and discussed. First of al, enterprise modeling has a comprehensive set of graphical primitives to use in the design process. This is illustrated in Figure 3 and Figure 4 where the dependencies instance of and inherit is of great importance. Without these two dependencies the final and integrated schema would probably contain a lot of ambiguities. Secondly, in enterprise modeling every concept is modeled as a box and nothing else. This reduces the types of conflicts that can appear in view integration. An example of that is the type conflict that can appear using a traditional database modeling technique. Third, in enterprise modeling both static and dynamic dependencies, behavioral aspects, can be modeled in one and same schema. This is important if it in the future is to be used for more than conceptual database design. Finally, in enterprise modeling a concept can be interpreted differently depending on the dependencies in focus. An example of this is found in Figure 10 where the concept *TV* is interpreted as an object when focusing on the dependency between *TV.Type* and *TV* and as a class when focusing on the dependency between *TV* and *TV.Item.*

The homonym conflict that is studied a bit deeper in this paper is identified through the usage of different dependencies between two views using the same name for different concepts. The traditional conflict resolution technique, renaming, suggested by most methods is not always clear and needs to be more investigated and clarified. This is important because the developer, together with the end-users, have to identify situations where integration is not even suitable because the views are too different. One example of that schema situation is found in Figure 10 where the concept *TV* is used in several concept names and is therefore ambiguous.

The study in this paper indicates that it is important to first develop local views for each end-user (user group) and then integrate them into one global conceptual schema. By neglecting this, the global conceptual schema could contain errors and ambiguity. Similar conclusions is found in a case study performed by Parsons (2002) where he writes "[...] local schemas preserve and highlight differences among views, whereas a global schema can mask them." (p. 162) and "[...] database designers should not build a global conceptual schema without first building and verifying local schemas that directly reflect user views." (p. 173). These quotations illustrates even better that developers should first involve end-users and define views for each of them and thereafter integrate them into one global conceptual schemata instead of just develop one global conceptual schema.

REFERENCES

Batini, C., Ceri, S., and Navathe, S., 1992, *Conceptual Database Design An Entity-Relationship Approach*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, California.

Batini, C., and Lenzerini, M., 1984, A methodology for data schema integration in the entity relationship model, *IEEE Transactions on Software Engineering* **10**:650–664.

Batini, C., Lenzerini, M., and Navathe, B. L., 1986, A comparative analysis of methodologies for database schema integration, *ACM Computing Surveys* **18**:323–363.

Bhargava, H. K., and Beyer, R. M., 1992, Automatic detection of naming conflicts in schema integration: experiments with quiddities, in: *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, B. D. Shriver, ed., pp. 300–310, Vol. 2.

Boman, B., Bubenko Jr, J. A., Johannesson, P., and Wangler, B., 1997, *Conceptual Modelling*, Prentice Hall, Great Britain.

Chen, P., 1976, The entity-relationship model – toward a unified view of data, *ACM Transactions on Database Systems* **1**:9–36.

Dupont, Y., 1994, Resolving fragmentation conflicts in schema integration, in: *Proceedings of the 13th International Conference on the Entity-Relationship Approach Business Modelling and Re-Engineering*, P. Loucopoulos, ed., Springer-Verlag, pp. 513–532.

Embury, S. M., Jones, C. J., Sutherland, I., Gray, W. A., White, R. J., Robinson, J. S., Bisby, F. A., and Brandt, S. M., 1999, Conflict detection for integration of taxonomic data sources, in: *11th International Conference on Scientific and Statistical Database Mangement*, D. C. Martin, ed., pp. 204–213.

Engles, G., Gogolla, M., Hohenstein, U., Hulsmann, K., Lohr-Richter, P., Saake, G., and Ehrich, H. D., 1992, Concepual modelling of database applications using an extended ER model, *Data & Knowledge Engineering* **9**:157–204.

Fahrner, C., and Vossen, G., 1995, A survey of database design transformations based on the entity-relationship model, *Data & Knowledge Engineering* **15**:213–250.

Fang, D., Hammer, J., and McLeod, D., 1991, The identification and resolution of semantic heterogeneity in multidatabase systems, in *First International Workshop on Interoperability in Multidatabase Systems*, Y. Kambayashi, M. Rusinkiewicz, and A. Sheth, eds., pp. 136–143.

Gustas, R., 1997, *Semantic and Pragmatic Dependencies of Information Systems*, Monograph, Technologija, Kaunas.

Gustas, R., and Gustienė, P., 2002, Extending Lyee methodology using the enterprise modelling approach, in: *Frontiers in Artificial Inteligence and Applications New Trends In Software Methodologies, Tools and Techniques*, Hamido Fujita and Paul Johannesson, eds., IOS Press, Amsterdam, pp. 273–288.

Gustas, R., and Gustienė, P., 2003, Towards the enterprise engineering approach for information system modelling across organisational and technical boundaries, in: *Proceedings of the Fifth International Conference on Enterprise Information Systems*, O. Camp, J. Filipe, S. Hammoudi, and M. Piattini, eds., pp. 77–88, Vol. 3.

Gustienė, P., and Gustas, R., 2002, On a problem of ambiguity and semantic role relativity in conceptual modelling, *International conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*, L'Aquila, Italy.

Jacobsson, L., and Gustas, R., 2004, Towards a systematic modeling of component based software architectures, *International SSCCII-2004*, Amalfi, Italy.

Johannesson, P., 1993, *Schema Integration, Schema Translation and Interoperability in Federated Information Systems*, PhD thesis, Department of Computer & System Science, Stockholm University, Royal Institute of Technology, No 93-010-DSV, Edsbruk.

Kim, W., and Seo, J., 1991, Classifying schematic and data heterogeneity in multidatabase systems, *IEEE Computer* **24**:12–18.

Kohler, J., Lange, M., Hofestadt, R., and Schulze-Kremer, S., 2000, Logical and semantic database integration, in: *IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, D. C. Young, ed., pp. 77–80.

Larson, J. A., Navathe, S. B., and Elmasri, R., 1989, A Theory of attribute equivalence in databases with application to schema integration, *IEEE Transactions On Software Engineering*, **15**:449–463.

Lawrence, R., and Barker, K., 2001, Integrating relational database schemas using standardized dictionaries, in: *16th ACM Symposium on Applied Computing*, ACM Press, pp. 225–230.

Lee, M. L., and Ling, T. W., 2003, A methodology for structural conflict resolution in the integration of entity-relationship schemas, *Knowledge and Information System* **5**:225–247.

Li, W-S., and Clifton, C., 1993, Using field specifications to determine attribute equivalence in heterogeneous databases, in: *Third International Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, H.-J. Schek, A. P. Sheth, and B. D. Czejdo, eds., pp. 174–177.

Navathe, S. B., and Gadgil, S. U., 1982, A methodology for view integration in logical database design, *Proceedings of the Eighth International Conference on Very Large Data Bases*, Morgan Kaufmann, pp. 142–164.

Parent, C., and Spaccapietra, S., 1998, Issues and approaches of database integration, *Communications of the ACM*, **41(5es)**:166–178.

Parsons, J., 2002, Effects on local versus global schema diagrams on verification and communication in conceptual data modeling, *Journal of Management Information Systems* **19**:155–183.

Sheth, A., and Kashyap, V., 1992, So far (schematically) yet so near (semantically), in: *Proceedings of the IFIP WG2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)*, D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, eds., pp. 283–312.

Shoval, P., 1990, A methodology for integration of binary-relationship conceptual schemas, in: *International Conference on Databases, Parallel Architectures and Their Applications*, N. Rishe, S. Navathe, and D. Tal, eds., pp. 435–437.

Spaccapietra, S., and Parent, C., 1991, Conflicts and correspondence assertions in interoperable databases, *ACM SIGMOD* **20**:49–54.

Teorey, T. J., Yang, D., and Fry, J. P., 1986, A Logical design methodology for relational databases using the extended entity-relationship model, *Computing Surveys* **18**:197–222.

Vernadat, F. B., 1996, *Enterprise Modeling and Integration Principles and Applications*, Chapman & Hall, London.

A CRITICAL REVIEW OF CHALLENGES IN HYPERMEDIA SYSTEMS DEVELOPMENT

Michael Lang*

1. INTRODUCTION

Over the past few years there has been considerable academic interest in the development of "Web-based" systems, much of it surrounding the contention that it is somehow different from "traditional" or "conventional" systems development. However, the debate is clouded because of confusion over the meaning of the phrase "Web-based system".[1] Indeed, it may be argued that "Web-based" is just an adjective which does not essentially alter the meaning of the term "information system" when prefixed to it.[2] This view is upheld by an examination of definitions from the literature, which clearly make no fundamental distinction:

> *"Web-based information systems (WIS) are information systems (IS) that are based on Web technology and they are likely to be tightly integrated with conventional IS such as databases and transaction processing systems"*[3]
>
> *"Web Information Systems are Systems, not Pages. WISs are information systems first, and Web systems second"*[4]

As Lockwood and Constantine put it, "current development tools make it easy to 'browserize' almost anything",[5] so little if any redesign may be required to effect basic migration of a system to the Web. For example, some intranet projects have been as straightforward as Web-enabling existing back-end applications such as Lotus Notes databases. It is therefore obvious that although a system may be said to be "Web-based", this doesn't necessarily imply it is any different from a non-Web-based system as regards software design considerations.

Some clarity can be introduced by thinking of the Web as a "global hypermedia system".[6] Hypermedia technologies support much richer user interfaces, more complex navigation mechanisms and more varied forms of information than conventional systems. As Web-based systems assume hypermedia functionality, they become distinct from con-

* Department of Accountancy & Finance NUI Galway, University Road, Galway, Ireland. Telephone: +353 91 750301 Fax: +353 91 750565, Michael.Lang@nuigalway.ie.

ventional systems. Although the Web is not an ideal hypermedia environment, it is nevertheless the most common platform for hypermedia systems at the present time. In recognition of this point, the ACM Special Interest Group on Hypertext and Hypermedia (formerly SIGLINK) now goes by the acronym SIGWEB. This paper therefore considers interactive Web-based systems within the broader classification of hypermedia systems. This interpretation is also guided by Lee's point that IS researchers should aspire to produce timeless contributions.[7] "Hypermedia systems" is a better term than "Web-based systems" not just because it is less ambiguous, but also because it is a more enduring concept and embraces technologies that pre-date the Web (*e.g. on-line help, encyclopaedia CD-ROMs*) as well as those that follow the Web (*e.g. WAP and interactive TV applications*). Indeed, the term "Web-based information system" may soon become a redundant anachronism. The scope of the Web is expanding with the range of delivery devices, – witness "e-commerce" giving birth to offshoot terms such as "m-commerce" and "t-commerce", – what Botterweck and Swatman call "Web-like applications".[8] It is likely that over time most systems will be ported to the Web and that Web-based interfaces will become the norm, as testified by Microsoft's zeal to win the "browsers war". If most systems were to become "Web-based", the adjective would probably be dropped, as has happened with "*multimedia* PC", a popular term from the early 1990s that is now defunct.

Before proceeding, a number of other terms warrant clarification. Where used in this paper, *traditional systems development* refers to popular practices in the period from the 1970s to the late 1980s, until displaced by new approaches upon the arrival of visual programming, RAD tools, GUIs, object-orientation, and open systems architectures. The subsequent period from the early 1990s to present shall be referred to as the *modern age*. *Conventional systems* refer to systems and applications of the standard types encountered within organisations, such as transaction processing systems and management information systems. Lastly, *traditional hypermedia development* refers to applications from the pre-Web era, such as interactive CD-ROMs, online help systems, and Apple Hypercard applications.

The objectives of this paper are to critically review a number of issues encountered within hypermedia systems development that are often argued with little if any justification to be radically different, and to assess if these issues are indeed *new* (that is, not experienced in traditional systems development or in other disciplines) or *unique* (that is, not experienced in the development of conventional systems in the modern age, or in other disciplines). In brief, these issues are: cognitive challenges of designing non-linear navigation mechanisms, pressures of accelerated development in a "Web-time" environment, problems arising out of the external "virtual" nature of the end-user population, the appropriateness of traditional design methods and techniques, and difficulties attributable to the multidisciplinary composition of hypermedia design teams.

2. ISSUES AND CHALLENGES IN HYPERMEDIA DEVELOPMENT

There has been much speculation that the development of "Web-based" information systems poses new or unique challenges.[9–11] However, the assumption of "newness" is a common weakness in both systems development research and practice.[12–14] Of late, there is a growing posse of dissenters who argue that there is nothing substantially different about

Web-based systems development.[1, 2, 15, 16] Given the problems earlier alluded to concerning the definition of "Web-based systems", this ought not be surprising. In the following sub-sections, each of the purported new/unique challenges shall be considered within the broader umbrella of hypermedia systems development. Just as it is appropriate to situate a discussion of Web-based systems within the context of hypermedia, we should also consider hypermedia within the traditions of its contributory root disciplines, which include graphic design, information science, technical writing, literary theory, media production, and database systems.

2.1. Complexity of Navigation Mechanisms

Some definitions of hypermedia emphasise non-linearity as its essential differentiating characteristic. Because of this non-linearity, navigation mechanisms within hypermedia systems can become quite complex. Arbitrary hyperlinking results in chaotic structures, leading to problems such as "getting lost in cyberspace", locating information, visualising system organisation, and managing content. This is where diagrammatic modelling techniques become useful for system developers, as they help to overcome the cognitive difficulties of understanding complex, abstract structures. However, it has been argued that diagramming techniques from traditional systems development are inadequate for modelling hypermedia systems.[9, 17]

Structural complexity arising from interconnectedness is not a new problem in software design. In the early days, there was the 1960s practice of 'go-to' programming, which Dijkstra criticised as being "*an invitation to make a mess*".[18] In the 1970s, complex navigation and relationship structures were a feature of network databases. Out of those experiences arose a number of principles and techniques, such as modularity and normalisation, which are readily applicable to the design of information nodes within hypermedia systems.[19] More recently, the flow of control in visual event-driven and object-oriented programming languages is such that traditional techniques such as structured flowcharts are of limited use. As a consequence, modern age software diagramming techniques such as UML are being applied, as well as techniques drawn from traditional dynamic media (e.g. storyboarding). Storyboarding is borrowed from the film industry, where non-linear narrative has long been used for dramatic effect. Both storyboarding and UML can also be applied to hypermedia systems modelling, – indeed, a number of UML extensions have been proposed specifically for hypermedia design.[20, 21]

Looking to traditional literature, many authors have experimented with non-sequential interactive fiction (e.g. Borges[22]), and there are numerous examples stretching back to antiquity of branching stories and interlocking commentaries. The discipline of library information science also has long experience of non-linear systems that require the investigation of side links, and librarians' skills are relevant to the design and evaluation of Web sites.[23, 24] Likewise, techniques from technical writing and electronic documentation can be applied to hypermedia design.[25, 26] Even within traditional printed media, there are certain types of material that are specifically designed to be used in a random-access non-linear manner, such as encyclopaediae, thesauruses and reference works. Physically these are linear sequences of content units, but logically they are more complex because the units may be indexed and cross-referenced. According to Whitley, hypermedia systems are different from other types of software applications because their design involves "*a process* *of structuring ideas, describing the order of presentation, and conceptual exploration... the developers have to set up a number of alternatives for readers to explore rather than a single stream of text*".[27] However, technical writers have long had to set up navigable paths in designing online help systems, and likewise have authors of non-linear printed materials.

2.2. Accelerated Development Cycles

A much-cited "new" challenge is the pressure of accelerated development cycles, often referred to as "Web time" or "Internet speed".[28, 29] Certainly, looking at trends in IS development over the past 20 years, delivery times have dramatically shortened. In the early 1980s, Jenkins et al reported that the average project then lasted 10.5 months.[30] By the mid-1990s, the duration of typical projects had fallen to less than 6 months,[31] and average delivery times for Web-based systems are now less than 3 months.[9, 32–34] Such compressed timeframes are unprecedented in traditional systems development or traditional hypermedia development, and are made possible by the combined effect of two factors. Firstly, the Web is an immediate delivery medium which, unlike traditional IS and off-the-shelf software applications, is not impeded by production, distribution and installation delays. The second enabling factor is modern age rapid application devlopment tools. Arguably, this second factor is the more important, and the jargon term "Web time" is misleading because it tends to suggest that the coming of the Web alone brought about these accelerated timescales.

Ever-shortening product cycles has always been an observable fact of life within the IT industry, even before the advent of the Web.[35] Back in the 1960s "space age", NASA were rushing to produce software in the race to the moon. Short deadlines and limited resources have long been the bane of IS project managers.[36, 37] As such, "Web time" may be said to be just a continuation of this general tendency. Moreover, it is a phenomenon that is not specific to Web or hypermedia development, but applies also to conventional systems.[38] This is reflected by the growing interest in high-speed approaches such as agile methods, RAD, timeboxing, and COTS configuration amongst the general community of software developers.

Indeed, one could say that this trend is reflective of a greater urgency in business in general, brought about not just by the Web but also by other advances in telecommunications, transportation, and computing technologies, as well as practices such as JIT and BPR. Business in the modern age is characterised by a faster metabolism, time-based competition, shorter windows of opportunity, rapidly changing and uncertain environments, and a need for greater flexibility and adaptability.[39, 40] Considered thus, the phenomenon of "Web time" is not unique to Web, hypermedia, or conventional systems development, although it must be acknowledged that product life cycles are much shorter in the IT industry than in many other industries.[35] However, for many "new media" companies, "Web time" is not at all new, because they have always faced strict, pressing deadlines and have accordingly adapted their processes to the pressures of Web delivery.[41, 42]

2.3. Virtual End-User Population

Russo and Graham make the point that "*Web applications differ from traditional information systems [because] the users of Web applications are likely to be outside of the* *organization, and typically cannot be identified or included in the development process*".[9] This, like the previous issue, is not hypermedia-specific and arises out of the nature of the Web. However, because most modern hypermedia applications are delivered via the Web or "Web-like" platforms (such as PDAs and interactive TV), it shall be considered under the banner of hypermedia development.

It is plainly true that for most Web-based information systems, with the obvious exception of intranets, end-users are external to the organisation. This is indeed new territory for IS development, because information systems traditionally served internal functions.[43] Collecting requirements from a virtual population is difficult, and traditional ISD techniques cannot be easily applied if at all.[44] As a consequence, requirements are often vague. Baskerville and Pries-Heye state that "*an inability to pre-define system requirements is the central, defining constraint of Internet time development . . . often a project starts without a requirement specification*".[29]

Let us examine both these aspects more closely. Firstly, the notion of a virtual population, although new to IS developers, is quite typical for mass-market off-the-shelf software production and new product development.[43] In such situations, the marketing department fulfils a vital role as the voice of the customer.[44] Established marketing research techniques can be used in conjunction with traditional requirements elicitation techniques and Web usage analysis techniques to define requirements for a virtual population.[45] For example, Tognazzini describes how a team of designers, engineers, and human factors specialists used scenarios to define requirements based on an understanding of the profiles of target users as communicated by marketing staff.[46]

Secondly, the phenomenon of vague requirements is by no means new. Glass remarks that "*back in the earliest days of software development . . . [specifications] often didn't exist, or when they did, they were written on the back of an envelope*",[47] Walz et al observe that users often "don't know what they want" and fuzzy requirements are common,[48] and of course there is the classic problem of "I'll know it when I see it" requirements that are not pre-specifiable.[49]

Another aspect of a virtual user population, which is different from traditional IS development, is that end users often cannot be personally trained how to use the system. Constantine and Lockwood assert that "*much more so than standard software, Web applications must focus on the user experience*".[50] Certainly, issues such as usability and interaction design are paramount for Web/hypermedia systems, but they are arguably no less a consideration for off-the-shelf software applications. Indeed, the importance of ease-of-use for mass-produced goods has long been emphasised by industrial designers. There is however a further element of Web-based user experience design which traditionally has not been considered by software designers, – branding and corporate image. Unlike off-the-shelf applications and conventional systems development, Web systems have a "public relations" aspect.[51] This new aspect of IS development is best handled by those with specialist skills, – marketing personnel, graphic designers and communications consultants.

2.4. Need for Specialised Development Methods and Approaches

It is often argued that approaches and methods from traditional systems development are inappropriate for Web/hypermedia development.[3, 9, 17, 42] Murugesan et al speak of "*a pressing need for disciplined approaches and new methods and tools*",[52] taking into account "*the unique features of the new medium*". As demonstrated in this paper, it is arguable if many of the features of hypermedia are indeed unique, for many parallels may be drawn with traditional/conventional software design and other root disciplines. Merely because an application is based on new technologies, its design should not necessarily require an altogether new or different approach.[4, 13] It may well be true that traditional systems development methodologies are ill-suited to hypermedia development. However, for the same reasons, those methodologies can be argued to be inappropriate for conventional systems development in the modern age.[53] Modern approaches, methods and techniques, – such as rapid prototyping, incremental development, agile methods, use case modelling, class diagrams, GUI schematics, model-view-controller framework, and heuristic evaluations, – may be said to be as applicable to hypermedia development as to conventional systems development.

However, traditional methodologies ought not be entirely discarded for hypermedia development. Some authors maintain that an adapted form of the classical SDLC remains the most appropriate process model,[54, 55] an assertion that is further supported by empirical evidence that the SDLC and variants thereof remain in popular use.[56–58] Furthermore, Barry and Lang reveal that traditional approaches from other root disciplines, such as graphic design and media production, are also being used in hypermedia systems development.[34] It would therefore seem that new, specialised hypermedia-specific approaches are not required. Although many such approaches have been proposed in the literature, – such as RMM, OOHDM, VHDM, EORM, WSDM, WebML, and OntoWebber, – a recent study of hypermedia development practice reveals that these are not being used, but that methods from traditional and conventional systems development are being used.[58]

2.5. Multidisciplinary Design Teams

A notable aspect of hypermedia systems development is that design teams typically involve members from a diversity of professional backgrounds.[32, 34, 59] Of course, skills diversity is not unique to hypermedia systems development, – many conventional projects, particularly large ones, necessitate the integration of various knowledge domains.[48] However, in conventional systems development, designers tend to be primarily "computer professionals", which is typically not the case with hypermedia systems development. This is especially true of Web-based hypermedia systems, where many developers do not have a background in traditional software design or programming.[9, 33, 51] The challenge of managing communication and collaboration within multidisciplinary design teams is by no means trivial, and if mismanaged is potentially disastrous. Experiences from Web and interactive multimedia development projects reveal that discrepancies in the backgrounds of team members can give rise to significant communication and collaboration problems.[51, 60] Kim makes the point that:

> *"disciplines are like cultures: for disciplines to work well together they must learn to appreciate one another's language, traditions, and values . . . Different disciplines have different priorities, different thinking styles, different values. When people from different disciplines get together, their values collide."*[61]

This is analogous to Kuhn's notion of "paradigms".[62] It has been noted that separate paradigms can co-exist within the same field yet be mutually ignorant of each other, either deliberately or inadvertently.[63] Although "paradigms" and "disciplines" are not strictly equivalent, they are closely related. Within hypermedia systems development, two such disciplines are software engineering and graphic design. Despite being the two foremost disciplines in hypermedia development,[58] it has been observed that they appear to operate in distinctly different worlds and have quite different value systems.[64, 65] For example, software engineering and graphic design have markedly different interpretations of concepts such as "quality", "elegance" and "structure".

On the face of it, this appears to be a new challenge, although one could argue that communication problems are not new for they have traditionally plagued systems development projects.[66] The need to address the balance between functionality and aesthetics has previously arisen in other disciplines, such as civil engineering versus decorative architecture, automobile design, and computer game design. Within the realm of conventional systems development, user interface design has also long been a multidisciplinary, collaborative activity.[67, 68] Kautz and Nørbjerg argue that the move towards multidisciplinary teams is a continuation of an observable trend across systems development in general.[16] This position is supported by the contention of Fafchamps and Garg that flexible teams, characterised by composite membership and roles and diverse disciplines and skills, are "*a new type of organizational entity that will become more prevalent in the future*".[69] Whereas hypermedia systems development seems to call for a greater level of multi-/cross-skilling than traditional systems development,[42, 56, 70] this may likewise be argued to be an increasingly common phenomenon in the digital networked economy (i.e. "reprogrammable labour").

3. SUMMARY AND CONCLUSIONS

Jakob Nielsen has commented that "*software design is a complex craft and we sometimes arrogantly think that all its problems are new and unique*".[71] As presented in Table 1, many of the challenges of hypermedia systems development are neither unique nor entirely new. IS researchers have often been guilty of jumping onto the bandwagon of the lastest fad, leading Keen to remark that many "*issues seen as 'new' turn out to have long roots*".[12] With the advent of so-called "new media", much of the discourse amongst communications scholars has likewise been at fault of lacking a "historical consciousness".[72]

It is clear that hypermedia development is a multidisciplinary domain which can potentially draw from lessons and experiences across a variety of disciplines – traditional IS development, software engineering, HCI, graphic design, visual communications, marketing, technical writing, library information science, media production, architecture, and industrial design. For now, this is what most distinguishes hypermedia development from traditional/conventional systems development, although this distinction is likely to disappear in the future as software development in general is headed in this direction.

The multidisciplinary nature of design teams must be acknowledged in devising mechanisms to resolve challenges of hypermedia development. Design approaches, integrated working procedures, diagramming techniques, toolset selection, and methods for specifying and managing requirements must all take this central aspect into consideration.

Table 1. Summary of key points

Challenge	Related Trends, Technologies, and Experiences
Complexity of Navigation Mechanisms *(designing and visualising non-linear structures)*	Traditional systems development • go-to spaghetti code (structured programming techniques). • network and relational databases (data modelling techniques). Conventional systems development in the modern age • visual, event-driven, object-oriented programming (UML). Other disciplines • technical writing techniques for production of electronic documentation. • cataloguing mechanisms in library information science. • planning of scenes and shooting order for film pre-production (storyboarding).
Accelerated Development Cycles *(pressures of "Web time")*	Traditional systems development • short deadlines and tight resources have long been a reality. • ever-shortening product cycles has been a notable trend. Conventional systems development in the modern age • high speed approaches such as rapid prototyping, RAD, agile methods, COTS configuration are growing in popularity. Modern business environment • faster metabolism, greater urgency, need for agility are norms. Other disciplines • "old media" firms moving to "new media" have long experience of pressing deadlines e.g. news production.
Virtual End-User Population *(requirements definition, user experience design)*	Traditional/conventional systems development • traditional IS development was in-house, but traditional off-the-shelf software development has always been for an external population. • initially vague requirements and fuzzy specifications are an age-old issue. • "I'll know it when I see it" requirements are common in the modern age. • usability is recognised as an important consideration for all modern systems. Other disciplines • marketing researchers have always had an external focus. • visual communications discipline has experience of developing brands in diversified marketplaces. • in manufacturing, industrial designers have long emphasised intuitive, easy-to-use features in the development of mass-produced goods.
Need for Specialised Methods and Approaches	Traditional/conventional systems development • traditional IS development methods may be inappropriate for hypermedia development, but likewise are inappropriate for conventional systems development in modern age. • modern methods, approaches and techniques for conventional systems development can be used for hypermedia development. Hypermedia development • variants of classical SDLC remain in popular use in hypermedia development. • specialised approaches e.g. OOHDM, WSDM are not being used in practice Other disciplines • traditional approaches from graphic design and media production can also be adapted to hypermedia development.
Multidisciplinary Design Teams	Traditional/conventional systems development • traditional IS project teams have long been cross-functional (e.g. accounting, marketing, etc.) but developers were primarily software engineers. • design teams for interactive systems with visual GUIs are often multidisciplinary. Modern business environment • multi-skilling and cross-skilling are facets of digital networked economy. • flexible teams are becoming an increasingly common organisational entity. Other disciplines • trade-offs between functionality and aesthetics have been experienced elsewhere e.g. building design, automobile design., computer game development

The software engineering and ISD literature is filled with a multitude of methods and techniques aimed towards Web/hypermedia design, very few of which are used in practice. This might be explained in a variety of ways: lack of awareness, lack of tool-based support, lack of guidance on how to use them, or inertia being obvious possibilities. However, there is a much more fundamental matter which has received very little attention, – that of the narrow world-view of the method developers. Morgan makes the point that researchers in all branches of science "*often approach their subject from a frame of reference based upon assumptions that are taken-for-granted*".[73] For methods and techniques to be of value, the assumptions upon which they are founded should be in congruence with the context within which they are intended to be used. Methods and techniques for hypermedia systems development should therefore recognise its multidisciplinary nature and take a cosmopolitan world-view, integrating and adapting approaches from the various root disciplines. To paraphrase a well-known quotation, those who choose not to draw from the well of cumulative knowledge are bound to foolishly repeat past mistakes. This paper therefore concludes with a call to hypermedia design researchers to reach out and explore the historical experiences of other related disciplines.

4. FURTHER RESEARCH

This literature review paper is the product of an ongoing doctoral research project and draws upon insights gained from exploratory empirical work.[58, 74] The next stage of this project involves the definition of a conceptual framework to analytically compare the various disciplines that contribute to hypermedia systems development. This comparison shall analyse data captured from literature trawls and interviews with developers. Researchers with similar or related interests are welcome to contact the author.

REFERENCES

1. J. Holck, 4 perspectives on Web information systems, in: *Proceedings of 36th Annual Hawaii International Conference on System Sciences (HICSS)* (IEEE Computer Society Press, Hawaii, USA, January 2003).
2. C. Barry, Web-based information systems – time for the revisionists, in: *Proceedings of Collaborative Electronic Commerce Technology and Research (CollECTeR) Conference*, edited by T. Acton and P. Swatman (CISC, National University of Ireland, Galway, June 24, 2003), pp. 37–50.
3. S. Wang, Toward a general model for web-based information systems, *International Journal of Information Management* **21**(5), 385–396 (2001).
4. A. R. Dennis, Lessons from three years of Web development, *Communications of the ACM* **41**(7), 112–113 (1998).
5. L. Lockwood and L. Constantine, Taming Web development, *Software Development Magazine* (April 1999); http://www.sdmagazine.com/documents/sdm9904h/.
6. M. Andreessen and E. Bina, NCSA Mosaic: a global hypermedia system, *Internet Research* **4**(1), 7–17 (1994).
7. A. S. Lee, *The Timeliness of Publications in MIS Quarterly* (ISWorld Mailing List, June 5, 1999).
8. G. Botterweck and P. A. Swatman, Towards a contingency based approach to Web engineering, in: *Proceedings of 7th Australian Workshop on Requirements Engineering (AWRE'2002)*, edited by J. L. Cybulski et al. (Melbourne, Australia, December 2–3, 2002), pp. 47–64.
9. N. L. Russo and B. R. Graham, A first step in developing a Web application design methodology: understanding the environment, in: *Methodologies for Developing and Managing Emerging Technology Based Information Systems: Proceedings of 6th International BCS Information Systems Methodologies Conference*, edited by A. T. Wood-Harper et al. (Springer, London, 1999), pp. 24–33.

10. T. Isakowitz, M. Bieber and F. Vitali, Web information systems, *Communications of the ACM* **41**(7), 78–80 (1998).
11. D. B. Lowe and J. Eklund, Client needs and the design process in Web projects, *Journal of Web Engineering* **1**(1), 23–36 (2002).
12. P. G. W. Keen, Relevance and rigor in information systems research: Improving Quality, Confidence, Cohesion and Impact, in: *Information Systems Research: Contemporary Approaches and Emergent Traditions*, edited by H.-E. Nisson et al. (Elsevier Science Publishers, 1991), pp. 27–49.
13. E. Yourdon, Developing Applications for the Internet: advice for the Java generation, *American Programmer*, 36–41 (December 1996).
14. R. S. Pressman, What a tangled web we weave, *IEEE Software* **17**(1), 18–21 (2000).
15. T. Butler, An institutional perspective on developing and implementing intranet- and internet-based information systems, *Information Systems Journal* **13**(3), 209–231 (2003).
16. K. Kautz and J. Nørbjerg, Persistent problems in information systems development: the case of the World Wide Web, in: *Proceedings of 11th European Conference on Information Systems (ECIS)*, edited by C. Ciborra et al. (Naples, Italy, June 16–21, 2003).
17. K. Siau and M. Rossi, Information modeling in the Internet age – challenges, issues and research directions, in: *Information Modeling in the New Millennium*, edited by M. Rossi and K. Siau (Idea Group Publishing, Hershey, PA, 2001), pp. 1–8.
18. E. W. Dijkstra, Go to statement considered harmful, *Communications of the ACM* **11**(3), 147–148 (1968).
19. M. J. Taylor, S. Wade and D. England, Informing IT system Web site design through normalisation, *Internet Research: Electronic Networking Applications and Policy* **13**(5), 342–355 (2003).
20. H. Baumeister, N. Koch and L. Mandel, Towards a UML extension for hypermedia design, in: *UML'99: The Unified Modeling Language – Beyond the Standard, Second International Conference, Fort Collins, CO, USA, October 28–30, 1999, Proceedings. Lecture Notes in Computer Science 1723*, edited by R. B. France and B. Rumpe (Springer, 1999), pp. 614–629.
21. J. Conallen, *Building Web Applications with UML* (Addison Wesley, Reading, MA, 2000).
22. J. L. Borges, The garden of the forking paths, in: *Labyrinths: Selected Stories and other Writings*, edited by D. A. Yates and J. E. Irby (Penguin Books, Harmondsworth, 1981), pp. 44–54.
23. J. Conklin, Hypertext: an introduction and survey, *IEEE Computer* **20**(9), 17–20;32–41 (1987).
24. S. Shropshire, Beyond the design and evaluation of library web sites: an analysis and four case studies, *The Journal of Academic Librarianship* **29**(2), 95–101 (2003).
25. B. B. Zimmerman, Applying Tufte's principles of information design to creating effective Web sites, in: *Proceedings of 15th ACM Conference on Systems Documentation* (ACM Press, Snowbird, Utah, USA, October 19–22, 1997), pp. 309–317.
26. L. B. Eriksen, Limitations and opportunities for system development methods in Web information system design, in: *Organizational and Social Perspectives on Information Technology, IFIP TC8 WG8.2 International Working Conference on the Social and Organizational Perspective on Research and Practice in Information Technology, June 9–11, 2000, Aalborg, Denmark*, edited by R. Baskerville et al. (Kluwer, Boston, MA, 2000), pp. 473–486.
27. E. A. Whitley, Method-ism in practice: investigating the relationship between method and understanding in Web page design, in: *Proceedings of 19th International Conference on Information Systems (ICIS)* (Helsinki, Finland, December 13–16, 1998), pp. 68–75.
28. D. Thomas, Web time software development, *Software Development Magazine*, 78–80 (October 1998).
29. R. Baskerville and J. Pries-Heye, Racing the e-bomb: how the Internet is redefining information systems development methodology, in: *Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective. Proceedings of International Federation for Information Processing (IFIP) Working Group 8.2 Conference, Boise, Idaho, USA, 27–29 July 2001*, edited by N. L. Russo et al. (Kluwer Academic Publishers, Boston, MA, 2001), pp. 49–68.
30. M. A. Jenkins, J. D. Naumann and J. C. Wetherbe, Empirical investigation of systems development practices and results, *Information & Management* **7**(2), 73–82 (1984).
31. B. Fitzgerald, The use of systems development methodologies in practice: a field study, *Information Systems Journal* **7**(3), 201–212 (1997).
32. P. R. Vora, Designing for the Web: a survey, *ACM interactions* **5**(3), 13–30 (1998).
33. S. McClure, *Web Application Development Developer Perspectives: An IDC White Paper* (International Data Corporation, Framingham, MA, 1998).
34. C. Barry and M. Lang, A comparison of "traditional" and multimedia information systems development practices, *Information and Software Technology* **45**(4), 217–227 (2003).

35. H. Mendelson and R. R. Pillai, Clockspeed and informational response: evidence from the information technology industry, *Information Systems Research* **9**(4), 415–433 (1998).
36. T. Gilb, Deadline pressure: how to cope with short deadlines, low budgets and insufficient staffing levels, in: *Information Processing*, edited by H. J. Kugler (Elsevier Science Publishers B. V., 1986), pp. 293–299.
37. E. Yourdon, *Death March: The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects* (Prentice Hall, Upper Saddle River, NJ, 1997).
38. D. Kurata, Do OO in "Web time", *Visual Basic Programmer's Journal* **11**(1), 70 (2001).
39. J. C. Wetherbe and M. N. Frolick, Cycle time reduction: concepts and case studies, *Communications of the AIS* **3**(13), 1–42 (2000).
40. G. Scott, Internet/Web systems development: what can be learned from hi-tech new product strategic planning, in: *Proceedings of 36th Annual Hawaii International Conference on System Sciences (HICSS)* (IEEE Computer Society Press, Hawaii, USA, January 2003).
41. V. Bellotti and Y. Rogers, From Web press to Web pressure: multimedia representations and multimedia publishing, in: *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems* (ACM Press, Atlanta, Georgia, USA, March 22–27 1997), pp. 279–286.
42. J. Greenbaum and D. Stuedahl, Deadlines and work practices in new media development: its about time, in: *PDC 2000 Proceedings of Participatory Design Conference*, edited by T. Cherkasky et al. (New York, USA, November 28 – December 1, 2000), pp. 70–77.
43. J. Grudin, Interactive systems: bridging the gaps between developers and users, *IEEE Computer* **24**(4), 59–69 (1991).
44. J. Lazar, E. Hanst, J. Buchwalter and J. Preece, Collecting user requirements in a virtual population: a case study, *WebNet Journal* **2**(4), 20–27 (2000).
45. M. S. Lane and A. Koronois, A balanced approach to capturing user requirements in business-to-consumer Web information systems, *Australian Journal of Information Systems* **9**(1), 61–69 (2001).
46. B. Tognazzini, *Tog on Software Design* (Addison Wesley, Reading, MA, 1995).
47. R. L. Glass, Who's right in the Web development debate?, *Cutter IT Journal* **14**(7), 6–10 (2001).
48. D. B. Walz, J. J. Elam and B. Curtis, Inside a software design team: knowledge acquisition, sharing, and integration, *Communications of the ACM* **36**(10), 63–77 (1993).
49. B. Boehm, Requirements that handle IKIWISI, COTS, and rapid change, *IEEE Computer* **33**(7), 99–102 (2000).
50. L. L. Constantine and L. A. D. Lockwood, Usage-centered engineering for Web applications, *IEEE Software* **19**(2), 42–50 (2002).
51. P. H. Carstensen and L. Vogelsang, Design of Web-based information Systems – new challenges for systems development?, in: *Proceedings of 9th European Conference on Information Systems (ECIS)* (Bled, Slovenia, June 27–29 2001), pp. 536–547.
52. S. Murugesan, Y. Deshpande, S. Hansen and A. Ginige, Web engineering: a new discipline for development of Web-based systems, in: *Proceedings of 1st ICSE Workshop on Web Engineering* (ACM Press, Los Angeles, California, USA, May 16–17 1999), pp. 1–9.
53. B. Fitzgerald, Systems development methodologies: the problem of tenses, *Information Technology & People* **13**(3), 174–185 (2000).
54. T. A. Powell, D. L. Jones and D. C. Cutts, *Web Site Engineering: Beyond Web Page Design* (Prentice Hall, Upper Saddle River, NJ, 1998).
55. C. Urquhart, Exploring methodologies for Web based design: a case study of a design business, in: *Proceedings of 1st AIS SIGeBiz Workshop on e-Business (WeB 2002)* (Barcelona, Spain, December 14–15 2002), pp. 1–9.
56. C. Britton, S. Jones, M. Myers and M. Sharif, A survey of current practice in the development of multimedia systems, *Information and Software Technology* **39**(10), 695–705 (1997).
57. J. Paynter and M. Pearson, *A Case Study Of The Web-Based Information Systems Development* (Department of Management Science and Information Systems, University of Auckland, New Zealand, 1998).
58. M. Lang, Hypermedia systems development: a comparative study of software engineers and graphic designers, *Communications of the AIS* **12**(16), 242–257 (2003).
59. L. L. Scarlatos, R. P. Darken, K. Harada, C. Heeter, R. Muller and B. Shneiderman, Designing interactive multimedia (Panel), in: *Proceedings of 5th ACM International Conference on Multimedia* (ACM Press, Seattle, Washington, USA, November 9–13 1997), pp. 215–218.
60. O. Koechlin, *Methods and Tools to Improve Software Quality for Multimedia Productions. Final Report ESSI Project No. 21545* (European System and Software Initiative, December 1997).

61. S. Kim, Interdisciplinary cooperation, in: *The Art of Human-Computer Interface Design*, edited by B. Laurel (Addison Wesley, Reading, MA, 1990), pp. 31–44.
62. T. S. Kuhn, *The Structure of Scientific Revolutions* (University of Chicago Press, Chicago, IL, 1970).
63. M. Dogan, Paradigms in the social sciences, in: *International Encyclopedia of the Social & Behavioral Sciences*, edited by N. J. Smelser and P. B. Baltes (Elsevier Science, Oxford, 2001), pp. 11023–11027.
64. L. Vertelney, M. Arent and H. Lieberman, Two disciplines in search of an interface: reflections on a design problem, in: *The Art of Human-Computer Interface Design*, edited by B. Laurel (Addison Wesley, Reading, MA, 1990), pp. 45–55.
65. S. Gallagher and B. Webb, Competing paradigms in multimedia systems development: who shall be the aristocracy?, in: *Proceedings of 5th European Conference on Information Systems (ECIS)*, edited by R. D. Galliers et al. (Cork Publishing Ltd., Cork, Ireland, June 19–21, 1997), pp. 1113–1119.
66. B. Curtis, H. Krasner and N. Iscoe, A field study of the software design process for large systems, *Communications of the ACM* **31**(11), 1268–1287 (1988).
67. J. Grudin and S. E. Poltrock, User interface design in large corporations: coordination and communication across disciplines, in: *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems* (ACM Press, Austin, Texas, USA, April 30 – May 4, 1989), pp. 197–203.
68. J. Preece, *Interaction Design* (Wiley, New York, 2002).
69. D. Fafchamps and P. Garg, Computing environments for flexible teams, in: *Software Engineering and Human-Computer Interaction. ICSE '94 Workshop on SE-HCI: Joint Research Issues, Sorrento, Italy, May 16–17, 1994, Proceedings (LNCS 896)*, edited by R. N. Taylor and J. Coutaz (Springer-Verlag, Berlin, 1995), pp. 174–184.
70. N. P. Kotamraju, Keeping up: Web design skill and the reinvented worker, *Information, Communication & Society* **5**(1), 1–26 (2002).
71. J. Nielsen, Learning from the real world, *IEEE Software* **14**(4), 98–99 (1997).
72. E. Huhtamo, From cybernation to interaction: a contribution to an archaeology of interactivity, in: *The Digital Dialectic: New Essays on New Media*, edited by P. Lunenfeld (MIT Press, Cambridge, MA, 1999), pp. 96–110.
73. G. Morgan, Paradigms, metaphors, and puzzle solving in organization theory, *Administrative Science Quarterly* **25**(4), 605–622 (1980).
74. M. Lang, Reconsidering the "software crisis": a study of hypermedia systems development, in: *Proceedings of IADIS International WWW/Internet 2003 Conference*, edited by P. Isaías and N. Karmakar (IADIS Press, Algarve, Portugal, November 5–8 2003), pp. 307–313.

TRANSFORMATIONS OF UML DIAGRAMS FOR RECONCILIATION OF REQUIREMENTS

Lina Ceponiene and Lina Nemuraite*

1. INTRODUCTION

In OMG Model Driven Architecture (MDA) (Frankel, 2003; Siegel, 2001) models are primary artifacts in software development process, which differs significantly from earlier processes where the purpose of models was an aid to understanding and communication between participants of development project. In MDA models constitute the definition of the system under development. This definition must be precise and comprehensive in order to generate meaningful code. Development is prosecuted as a stepwise process where four types of UML models are created:

Business Model → Platform Independent Model (PIM) → Platform Specific Model (PSM) → Program Code.

MDA activities are concentrated on going from PIM to PSM and from PSM to code. The very important role there is played by the quality of PIM, i.e. its capability to adequately represent system under development. By our view, for ensuring the quality of PIM it is purposeful to ensure the quality of definition of requirements. Elaborated requirements model (based on (Reggio et al., 2001; Astesiano and Reggio, 2002)) named as Design Independent Model (DIM), was described in (Ceponiene et al., 2003; Ceponiene and Nemuraite, 2004). For automation of development process, the design model may be derived from requirements definition, integrating it with chosen software architecture model (transformation DIM → PIM). During development of DIM, while analysis of interactions in requirements phase remains under responsibility of analyst, computer supported generation of state charts from interactions may help to harmonize multiple sequence diagrams to comprehensive model of desired behaviour of the system.

UML class, sequence and state diagrams, representing system under development, really are partially overlapping views (e.g. Breu et al., 1998) over the common model of that system. The deficiency of the current UML CASE tools is that different diagrams are only partially related and developer often creates UML metamodel instances with non-overlapping element sets.

* Kaunas University of Technology, Studentu 50–308, LT-51368 Kaunas, Lithuania, kavalina@soften.ktu.lt, nemur@soften.ktu.lt.

In this work, derivation of state diagrams from sequence diagrams and vice versa is considered. DIM is applicable to systems oriented to services, that is, systems under consideration are capable to perform services responding to received requests, possibly collaborating with other systems. The shape of state machines representing service-oriented behavior is different from state machines used in Action Semantics Language (Mellor and Balcer, 2002) or UML 2.0 protocol state machines (Unified Modeling Language Superstructure…, 2003). Some principles are similar to (UML Profile…, 2002), where states of entities and behavioral classifiers are integrated; also information about senders of events, sent to other systems, is captured (that is missing in UML 2.0 protocol state machines and in state machines in general).

The paper is composed of 7 sections. In Section 2, the proposed principles of Information System modeling at elaborated requirement level are presented. In Section 3 DIM-corresponding canonical forms of sequence diagrams and state chart diagrams are introduced. Sections 4 and 5 are devoted for definition of transformations from sequence diagrams to state charts and vice versa. Section 6 discusses related approaches. Finally, Section 7 concludes the paper and discusses possible future work.

2. INTEGRATION OF SEQUENCE DIAGRAMS AND STATE CHARTS

Design Independent Model for service oriented Information System presents subset of UML 2.0 metamodel with additional constraints. More detail, DIM enables specification of information systems constituted of entities and interfaces to behavior. Functional requirements are defined as conceptual operations of interfaces (Cheesman and Daniels, 2000; D'Souza and Wills, 1999; Sendall and Strohmeir, 2000). Reconciliation of requirements is achieved by binding model elements. In Figure 1, DIM elements used in transformations between interactions and state charts are represented.

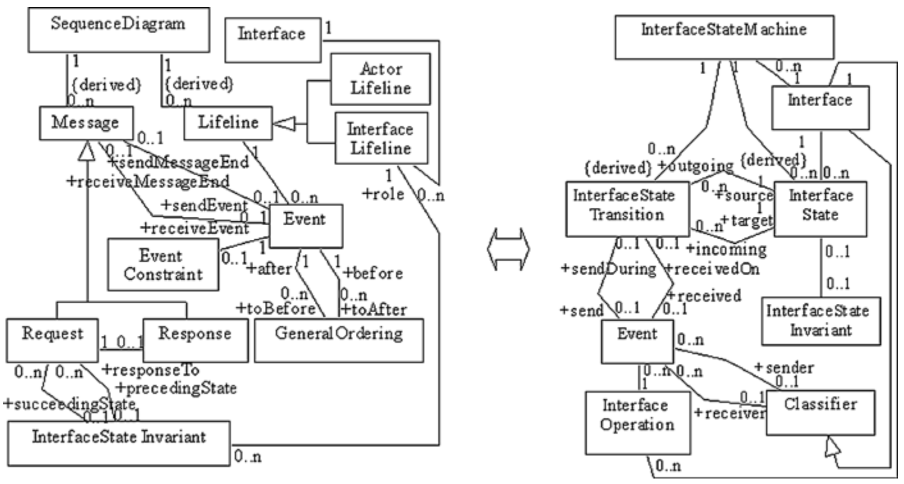


Figure 1. Transformation between elements of Sequence Diagrams and Interface State Machines metamodels.

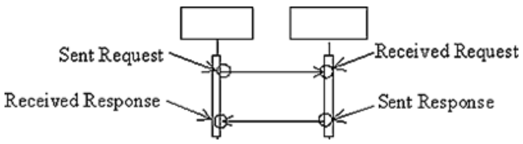


Figure 2. Event types.

In DIM, persistent state of the system is defined by states of entities; transient states correspond to execution of operations, specified by interfaces. Signatures of operations are expressed in terms of entities; pre and post conditions are defined using states of entities that are changed by operations. DIM contains no design decisions, that is, there are no control classes or components; these elements arise in design phase.

The main elements of sequence diagram are *Lifelines* (used in UML 2.0 instead of classifier roles from previous versions of UML) and *Messages*. For clear modeling of behavior, two types of *Messages* were distinguished: *Requests* and *Responses*. *Events* were differentiated to *Send Events* corresponding to one message end and *Receive Events* corresponding to the other one (Figure 2).

As events are present in sequence diagrams and in state charts, the main problem is to map messages, sent between different actors and interfaces from various sequence diagrams, to states and state transitions happening during life cycles of interfaces and represented in state machines of these interfaces.

3. TRANSFORMATION OF DIAGRAMS INTO CANONICAL FORM

Analyzing correspondences between interactions and states, the following rules were deduced: *Request* event, received on lifeline, is always associated with transition to new *Interface State* in state machine of lifeline owner; *Sent Response* always is associated with transition from *Interface State* to *Wait State* (for service oriented systems it is supposed that interface to service always has default state named *Wait State*); *Sent Request* is associated with event sent during transition; *Received Response* is associated with activation of transition to state dependent from succeeding event (to *Wait State*, if the succeeding event is *Sent Response*, and to the current state, if it is *Sent Request*).

3.1. Canonical Sequence Diagrams

For mapping, the sequence diagram initially is transformed to canonical form (Figure 3), where all *Request* messages are supplemented with *Response* messages if latter do not persist. *Response* message is inserted before the first *Received Request* succeeding considered *Request* having no *Response* (or at the end of sequence of events). If there are asynchronous messages in input sequence diagram, the apparent response messages are added that denote successful sending of asynchronous requests.

This transformation may be defined by DIM metamodel interface operation transformSD2CSD(), using OCL 2.0 (Unified Modeling Language: OCL..., 2003):

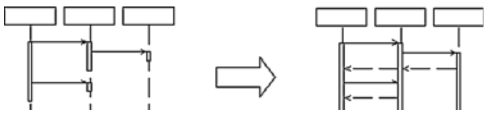


Figure 3. Transformation to canonical sequence diagram.

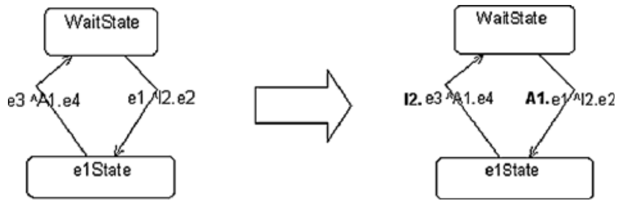


Figure 4. UML state chart and canonical state chart with information about senders of events.

```
Context DIM :: transformSD2CSD(sd:SequenceDiagram):SequenceDiagram
                post : result = sd.addResponses2SD(sd.orderSD(sd))
```

Operations addResponses2SD() and orderSD() (described in Appendix) represent intermediate transformations: addResponses2SD() adds missing responses; orderSD() orders events in sequence diagram. Using orderSD() operation events are ordered and converted to sequence of events, using information from metamodel class *General Ordering*. It is necessary because during creation of sequence diagrams messages are created and deleted and they may appear in *General Ordering* instance set in any order, not corresponding to their sequence during interaction.

3.2. Canonical State Chart Diagrams

For bi-directional transformation between sequence diagrams and state charts, information about senders and receivers of events must exist in state chart diagram. Relationships for providing this information are present in UML metamodel, but senders are not captured in usual state diagrams. Notation used for describing the send event and its receiver in UML is ^*Receiver.SendEvent*. The notation for describing sender of received event in canonical state chart may be similar: *Sender.Event* (Figure 4). If designer has not provided this kind of information, default sender may be added to *Received Requests* (e.g., '*Xrequestor*', where *X* denotes the name of interface); senders of *Received Responses* correspond to receivers of *Sent Requests*.

As long as events on transitions in state machine correspond to some *Events* of *Request* or *Response* messages from sequence diagram, transitions can be dependent on preceding transitions. *Event e1* depends on event *e2* if it is *Received Response* to *Sent Request e2*: *e1.receiveMessageEnd.responseTo = e2.sendMessageEnd*. These constraints are clearly tangible in sequence diagrams, but in state chart diagram they are badly expressed. State charts would be more informative if the order of transitions would be explicitly represented (compare Figure 5(a) and (b)).

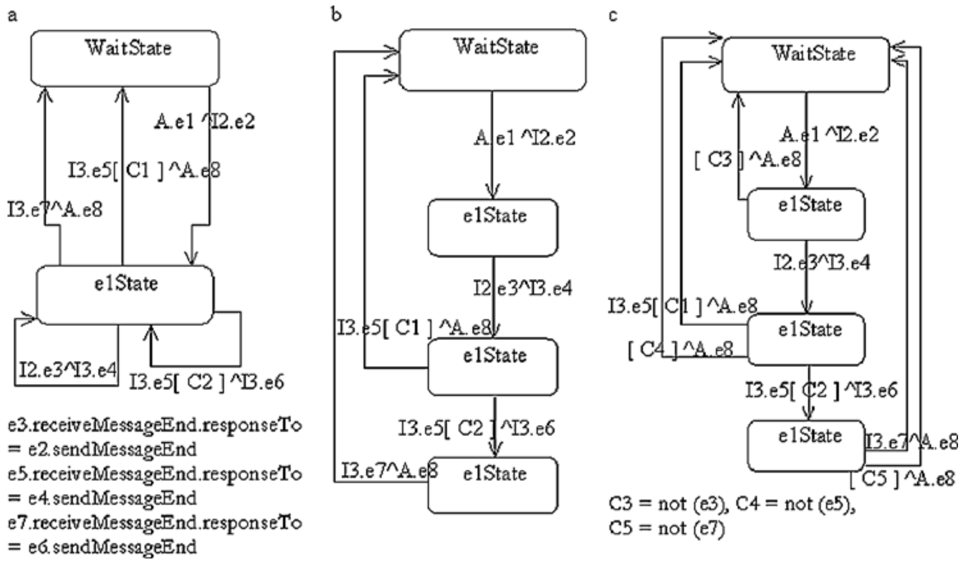


Figure 5. Usual (a) and rich (b) state chart representation; improved state chart (c).

Reasoning about correctness of state machine gives the possibility to improve behavior model described by sequence diagrams. To ensure completeness some additional constraints may be added to representation of behavior. E.g. for every synchronous *Sent Request* message at least two responsive transitions must be provided: one for the case, when response is received, and the other one, when response is not received, i.e., *e1* and *not(e1)* (such condition may express timeout for waiting of event occurrence) etc. In Figure 5 (c) the improved state chart is represented.

4. TRANSFORMATION FROM SEQUENCE DIAGRAMS TO STATE CHARTS

Canonical sequence diagrams may be transformed to state chart diagrams, mapping events of every lifeline to sequences of pairs (tuples). In each pair the first element represents event corresponding to state and the second element represents at most two events corresponding to state transition (received event that activates the transition and event sent during this transition). Pairs may be represented as instances of an auxiliary class *Pair* (Figure 6).

Route corresponds to sequence of pairs obtained from one sequence diagram. *Interface State Machine* may represent several *Routes*. The order of pairs in the sequence is significant as source state of the transition corresponds to the first element of one pair and target state corresponds to the first element of the succeeding pair. Class *Pair* actually is a view representing complex relation between events; the instances of this relationship may be described as *Tuple* in OCL 2.0:

```
Pair: Tuple(first:Event,second:Tuple(received:Event,sent:Event))
```

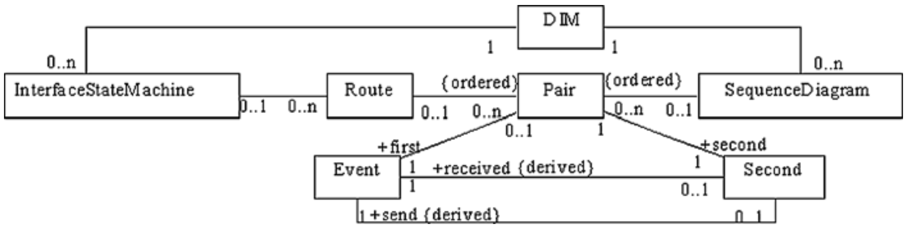
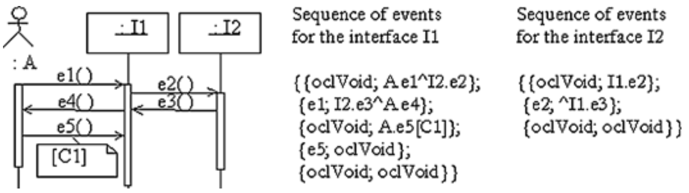


Figure 6. Auxiliary classes, representing views under original classes of DIM metamodel.



Sequence of events for the interface I1

```
{{oclVoid; A.e1^I2.e2};
 {e1; I2.e3^A.e4};
 {oclVoid; A.e5[C1]};
 {e5; oclVoid};
 {oclVoid; oclVoid}}
```

Sequence of events for the interface I2

```
{{oclVoid; I1.e2};
 {e2; ^I1.e3};
 {oclVoid; oclVoid}}
```

Figure 7. Example of sequences of events obtained from sequence diagram.

Operation for transformation of events of one lifeline to state machine may be represented by sequence of intermediate transformations: obtaining the sequence of pairs for the *Lifeline* and transforming this sequence to *Interface State Machine*:

```
Context DIM:: transformCSD2CSM(sd:SequenceDiagram,l:Lifeline):InterfaceStateMachine
let r:Route = sd.transformCSD2P(sd, l) in
 result = DIM.transformP2CSM(r,l)
```

Operation transformCSD2P() is described in Section 4.1 and operation transformP2CSM() is described in Section 4.2.

4.1. Obtaining Sequence of Pairs

Operation transformCSDL2P() (OCL definition presented in Appendix) is a query obtaining a sequence of pairs of the certain lifeline from sequence diagram. The sequences of pairs obtained from the sequence diagram are presented on Figure 7.

4.2. Transforming Sequence of Pairs to State Machine

Operation transformP2CSM() (OCL definition can be found in Appendix) adds one route (sequence of pairs) of one lifeline to canonical state machine of the interface. The results of applying the operation to lifelines of sequence diagram (Figure 7) are illustrated on Figure 8.

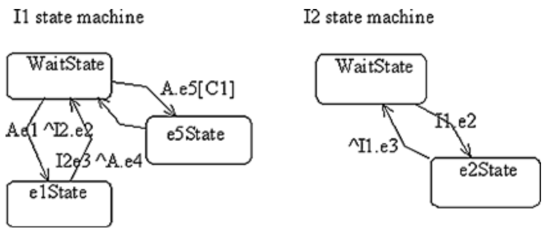


Figure 8. Resulting state machines for I1, I2.

5. TRANSFORMING CANONICAL STATE MACHINES TO SEQUENCE DIAGRAMS

The algorithm for generation of sequence diagrams from state charts should produce a set of sequence diagrams where every sequence diagram corresponds to one sequence of events from state chart.

Operation for transforming the sequences of events from the state machine to sequence diagrams may be represented by two intermediate transformations: obtaining sequences of pairs and transforming these sequences to the set of *Sequence Diagrams*:

```
Context DIM :: transformCSM2CSD(ism:InterfaceStateMachine):Set(SequenceDiagram)
result = ism.transformCSM2P(ism) → iterate(r; acc: Set(SequenceDiagram)={}|
acc → append(ism.transformP2CSD(r)))
```

Operation transformCSM2P() (Section 5.1) is used for obtaining the sequences of pairs from statechart and operation transformP2CSD() (Section 5.2.) transforms these sequences into sequence diagrams. OCL definition of both operations can be found in Appendix.

5.1. Obtaining Sequence of Pairs

Sequence of pairs from state machine is created by walking through the state chart from *Wait State* to *Wait State* by all possible ways and remembering the path. In one sequence each transition can be taken not more than once (this helps to avoid endless traces). The sequences obtained from a state chart are described in the same manner as for generation of state charts from sequence diagrams. Operation transformCSM2P() transforms canonical *Interface State Machine* to sequences of pairs:

```
Context InterfaceStateMachine :: transformCSM2P(ism:InterfaceStateMachine):Sequence(Route)
let ws:InterfaceState = ism.InterfaceState → select(is | is.name = 'WaitState') in
  result = ism.iroute(ws) → iterate(tu;acc:Sequence(Route)={}|
    acc → union(ism.AllRoutes(tu.second.received.receivedOn.target, tu)))
```

Here recursive operation allRoutes() (described in Appendix) is used for obtaining all routes starting from the given state. Two traces obtained from the state chart diagram from Figure 5 (b) are represented on Figure 9.

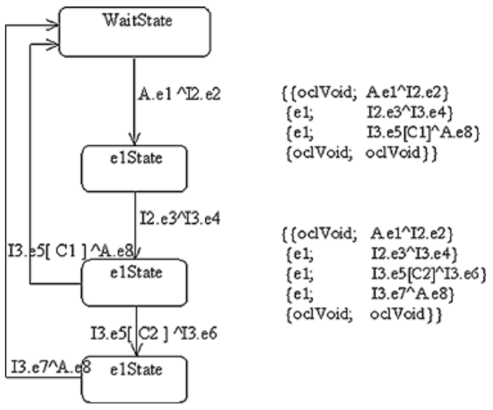


Figure 9. Example state chart and sequences obtained from this state chart.

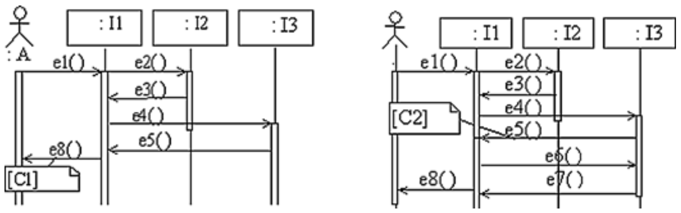


Figure 10. Sequence diagrams derived from state machine on Figure 9.

5.2. Transforming Sequence of Pairs to Canonical Sequence Diagram

Operation transformP2CSD() transforms sequence of pairs to sequence diagram (Figure 10) (operation is similar to transformLP2CSM() and it was not presented here because of limits of the paper). Again pairs are analyzed according to their sequence:

- If *Lifeline* of the state machine owner, sender or receiver does not exist in the sequence diagram, it is added;
- If the first member of a pair is *OclVoid*, the second member is mapped to *Request* message, received by state machine owner; *Sent Event* of this second member is mapped to *Request* message, sent by the state machine owner;
- If the first member of a pair is not *OclVoid*, but the first member of subsequent pair is *OclVoid*, the second member is mapped to *Response Message* and *Sent Event*, if any, is mapped to *Response Message*, sent by state machine owner;
- If the first member of a pair and the first member of subsequent pair is not *OclVoid*, the second member is mapped to *Response Message*, sent to state machine owner from *Received Event* sender; *Sent Event*, if any, is mapped to *Request Message*.

The main transformations of described above were experimentally implemented using Argo UML tool (Nemuraite et al., 2002). For solid implementation of DIM component full collection of operations should be created extending current framework (Figure 11) with

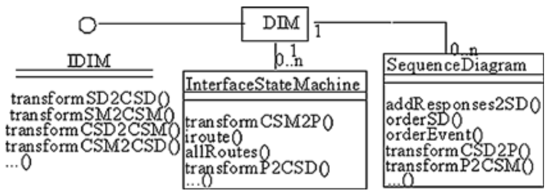


Figure 11. Current version of DIM component.

operations enabling developer to transform, match, insert diagrams and their elements, keeping model of IS consistent or querying about inconsistencies.

6. RELATED WORK

Currently, transformations between UML models are intensively investigated. Several proposals (Gardner et al.; 2003; MOF, 2004) are made in response to OMG request for proposal to MOF Query/View/Transformation (MOF, 2002). Principles of simple language for transformations are presented in (Kleppe, 2003). In (Braun, 2003), a highly formal language is proposed for transformation of object-oriented models. In (Varro et al., 2002) transformations are based on graph transformation techniques. All of these proposals use a declarative approach to transformations rather than an imperative one. In our work, transformations are described by the set of "forward" and "backward" rules that is deficiency from declarative bijective mappings perspective. But currently capabilities of existing transformation languages are demonstrated just for relatively simple transformations (e.g. object to relational mapping) that are remarkably different from mappings between interactions and state charts. Transformations in this work are described in OCL 2.0 that provides significantly more possibilities to manipulate with UML models than the earlier versions (Warmer, 2000; Balsters, 2003).

In current work the major attention is devoted to integration of behavior pursuing (Bock, 1999; Bock, 2000). There were works on theme of generation of state charts from sequence diagrams (Bordeleau, 2000; Makinen and Systa, 2000; Whittle and Schumann, 2000) but bi-directional transformations (without loss of information) were not possible.

7. CONCLUSION

Perfection of modeling approaches remains a challenge in the information systems field. The idea of this work is in consequent derivation of model fully representing design independent state and behavior of intended information system. Use cases are formalized as interfaces; elaborated requirements may be presented in visual form and in OCL. Diagrams supported with OCL constraints are integrated and derivable from the other ones. Direct and reverse mappings between sequence diagrams and state charts may be fulfilled, and analysis of state charts may reveal incomplete and inconsistent pieces. Transformations between diagrams proposed in this work are described as operations of DIM metamodel component interface.

In the future work, two targets are endeavored:

- To develop the complete collection of operations enabling developer to manipulate sequence, state and class diagrams and their elements;
- To map requirements model (DIM) to design model, integrating DIM with software architecture model using rules of chosen design method. The development of methodologies of conceptual modeling allows substantial progress to be expected in this area. Some experimental mappings have been made, but currently they are at instance level.

REFERENCES

Astesiano, E., and Reggio, G., 2002, *Knowledge Structuring and Representation in Requirement Specification*, in: *Proceedings of SEKE 2002*, G. Tortora and S. K. Chang, eds., ACM Press, New York, pp. 143–150.

Balsters, H., 2003, Modelling Database Views with Derived Classes in the UML/OCL-framework, in: *"UML" 2003 – The Unified Modeling Language: Modeling Languages and Applications*, P. Stevens, J. Whittle, and G. Booch, eds., Springer-Verlag, Heidelberg, LNSC **2863**, pp. 295–309.

Bock, C., 2000, Goal-Driven Modeling, *Journal of Object-Oriented Programming* **13**(5):48.

Bock, C., 1999, Three kinds of Behavior Model, *Journal of Object-Oriented Programming* **12**(4).

Bordeleau, F., and Corriveau, J. P., 2000, From Scenarios to Hierarchical State Machines: A Pattern-Based Approach, in: *Proc. of OOPSLA 2000 Workshop: Scenario-based round-trip engineering*, T. Systa, ed., Tampere University of Technology, Software Systems Laboratory Report No. 20, pp. 13–18.

Braun, P., and Marschall, F., 2003, *BOTL. The Bidirectional Object Oriented Transformation Language*, Institut für Informatik Technische Universität München Report (unpublished), p. 170.

Breu, R., Grosu, R., Huber, F., Rumpe, B., and Schwerin, W., 1998, Systems, Views and Models of UML, in: *The Unified Modeling Language, Technical Aspects and Applications*, M. Schader and A. Korthaus, eds., Physica Verlag, Heidelberg, pp. 93–108.

Ceponiene, L., and Nemuraite, L., 2004, Design independent modeling of information systems using UML and OCL, in: *Databases and Information Systems*, *Sixth International Baltic Conference BalticDB&IS 2004*, J. Barzdins et al., eds., Riga, Latvia, June 6–9, **672**, pp. 357–372.

Ceponiene, L., Nemuraite, L., and Paradauskas, B., 2003, Design of schemas of state and behaviour for emerging information systems, in: *Computer Science Reports*, Branderburg University of Technology at Cottbus, B. Thalheim and G. Fiedler, eds., **14**, pp. 27–31.

Cheesman, J., and Daniels, J., 2000, *UML Components*, Adison Wesley, Boston, p. 208.

D'Souza, D. F., and Wills, A. C., 1999, *Objects, Components, and Frameworks with UML. The Catalysis Approach*, Addison Wesley, Boston, p. 816.

Frankel, D., 2003, *Model Driven Architecture: Applying MDA to Enterprise Computing*, Wiley Publishing, Inc., Indianapolis, Indiana, p. 352.

Gardner, T., and Griffin, C., 2003, *A Review of OMG MOF 2.0 Query/Views/Transformations Submissions and Recommendations Towards the Final Standard*, OMG document ad/03-08-02 (May 3, 2004); http://www.omg.org.

Kleppe, A., Warmer, J., and Bast, W., 2003, *MDA Explained. The Model Driven Architecture: Practice and Promise*, Addison-Wesley, Boston, p. 192.

Makinen, E., and Systa, T., 2000, An interactive approach for synthesizing UML statechart diagrams from sequence diagrams, in: *Proc. of OOPSLA 2000 Workshop: Scenario-Based Round-Trip Engineering*, T. Systa, ed., Tampere University of Technology, Software Systems Laboratory Report No. 20, October, 7–12, Tampere, pp. 7–12.

Mellor, S. J., and Balcer, M. J., 2002, *Executable UML. A Foundation for Model-Driven Architecture,* Addison-Wesley, Boston, p. 368.

MOF 2.0 Query/Views/Transformations RFP, 2002, OMG document ad/2002-04-10 (May 3, 2004); http://www.omg.org.

MOF 2.0 Query/Views/Transformations Submission. Second Revised submission, 2004, submited by DSTC, IBM, CBOP, OMG document ad/2004-01-06 (May 3, 2004); http://www.omg.org.

Nemuraite, L., Kavaliauskaite L., and Ambrazevicius, E., 2002, Towards ensuring consistency to UML models, *Informacijos Mokslai* **24**:85–97.

Reggio, G., Cerioli, M., and Astesiano, E., 2001, Towards rigorous semantics of UML supporting its multiview approach, in: *Fundamental Approaches to Software Engineering FASE'01*, H. Hussmann, ed., Springer-Verlag, Berlin, LNSC **2029**, pp. 171–186.

Sendall, S., and Strohmeir, A., 2000, From use cases to system operation specifications, in: *The Third International Conference on the Unified Modeling Language-UML'2000*, A. Evans, S. Kent, and B. Selic, eds., Springer-Verlag, Heidelberg, LNCS **1939**, pp. 1–15.

Siegel, J., 2001, *Developing in OMG's Model-Driven Architecture*, OMG document omg/01-12-01 (May 3, 2004); http://www.omg.org.

UML Profile for Enterprise Distributed Object Computing Specification, 2002, OMG document ptc/o2-02-05 (May 3, 2004); http://www.omg.org.

Unified Modeling Language Superstructure Specification Version 2.0, 2003, OMG document ptc/03-08-02 (May 3, 2004); http://www.omg.org.

Unified Modeling Language: OCL Version 2.0, 2003, OMG document ptc/03-08-08 (May 3, 2004); http://www.omg.org

Varro, D., Varro, G., and Pataricza, A., 2002, Designing the automatic transformation of visual languages, *Science of Computer Programming* **44**:205–227.

Warmer, J. B., and Kleppe, A. G., 2000, *The Object Constraint Language: Precise Modeling with UML*, Addison Wesley, Boston, p. 112.

Whittle, J., and Schumann, J., 2000, Generating statechart designs from scenarios, in: *International Conference on Software Engineering*, C. Ghezzi, M. Jazayeri, and A. L. Wolf, eds., ACM Press, New York, pp. 314–323.

APPENDIX

Operation addResponses2SD():

```
Context SequenceDiagram:: addResponses2SD(sd:SequenceDiagram):SequenceDiagram
post:result.Lifeline = sd.Lifeline and result.Message = sd.Message →
 iterate(m:Message;acc:Set(Message)=sd.Message |
if m.oclIsKindOf(Request) and m.response→ isEmpty() then
 let m1:Message = m
 let prev:Event = if (m.MessageSort = asynchCall or m.MessageSort = asynchSignal) then
    m1.receiveEvent else m.receiveEvent.Lifeline.Event→ select(e |
 m.receiveEvent.toAfter→ includes(e) and e.receiveMessageEnd.oclIsKindOf(Request))→
   asSequence()→ first().toBefore.Before endif in
 if prev→ isEmpty() then prev= m.receiveEvent.Lifeline.Event→
        select(e | m.receiveEvent.toAfter→ includes(e))→ asSequence()→ last() endif
 let ma=Response{name='resp'.concat(m1.name)} in
 ma.responseTo=m1 and ma.sendEvent.Lifeline = m1.receiveEvent.Lifeline and
 ma.receivedEvent.Lifeline = m1.sendEvent.Lifeline and
 ma.sendEvent.toBefore.before=prev and ma.sendEvent.toAfter.after=prev.toAfter.after
 and ma.receivedEvent.toBefore.before = ml.sendEvent and
 ma.receivedEvent.toAfter.after= ml.sendEvent.toAfter.after and acc→ including(ma) endIf)
```

Operation orderSD():

```
Context SequenceDiagram::orderSD(sd:SequenceDiagram):SequenceDiagram
post: result.Lifeline = sd.Lifeline and
result.Lifeline→ forAll(l | l.Event = sd.orderEvent(sd.Lifeline→ select(l1=l).Event) )
```

The utility operation orderEvent() used in operation orderSD():

```
Context SequenceDiagram::orderEvent(s:Set(Event)):Sequence(Event)
post:
result = s→ iterate(e;acc:Sequence(Event)={s→ select(e | e.toBefore.before→ isEmpty()}) |
 acc.append(e1 | e1=acc.last().toAfter.after))
```

Operation transformCSD2P():

```
Context SequenceDiagram::transformCSD2P(sd:SequenceDiagram,l:Lifeline):Route
Let rseq:Sequence(Event)=sd.l.Event→
   select(e |    e.receiveMessageEnd.oclIsKindOf(Request))→ asSequence()
result = rseq → iterate(er; acc: Route = { } |
 if er.toAfter.after.sendMessageEnd.oclIsKindOf(Request) then
acc→ append(Tuple{first=oclVoid, second=Tuple{received=er,sent=er.toAfter.after}})
else acc→ append(Tuple{first=oclVoid, second=Tuple{received= er, sent=oclVoid}}) endif
let es:Sequence(Event)=sd.l.Event→ select(e |   e.receiveMessageEnd.oclIsKindOf(Response)
       and er.toAfter→ includes(e) and if er<>rseq→ last() then
        rseq→ at(rseq.indexOf(er)+1).toAfter→ excludes(e) endif)→ asSequence() in
if es→ notEmpty() then es→ iterate(e | if e.toAfter.after.sendMessageEnd→ notEmpty()
      then acc→ append(Tuple{first=er,second=Tuple{received=e,sent=e.toAfter.after}})
      else acc→ append(Tuple{first=er,second=Tuple{received=e,sent=oclVoid}}) endif)
else if er.toAfter.after.sendMessageEnd.oclIsKindOf(Response) then
  acc→ append(Tuple{first=er,second=Tuple{received=oclVoid,sent=er.toAfter.after}})
endif endif
if er= rseq→ last() then acc→ append(Tuple{first =oclVoid, second = Tuple{received= oclVoid,
   sent=oclVoid}}) endif)
```

Operation transformP2CSM():

```
Context SequenceDiagram:: transformP2CSM(r:Route,l:Lifeline):InterfaceStateMachine
post:
result.InterfaceState=r.Pair→
 iterate(p;acc:Set(InterfaceState)=l.Interface.InterfaceState) |
   let st:InterfaceState = {st.name = if p.first = OclVoid then
   'WaitState' else p.first.name.concat('State')} in
      if acc.InterfaceState→ excludes(st1 = st) then acc→ including(st) endIf)
        and result.InterfaceStateTransition = r.Pair→
          iterate(p;acc:Set(InterfaceStateTransition)=
            InterfaceStateMachine.InterfaceStateTransition →
         select(st | st.source.Interface = l.Interface and st.target.Interface = l.Interface) |
         let t:InterfaceStateTransition = {t.name= if p.second.received→ notEmpty()
      then p.second.received.Lifeline.name.concat('.').concat(p.second.received.name) else '' endif
      if p.second.sent→ notEmpty() then .concat('´)
        .concat(p.second.sent.sendMessageEnd.receiveEvent.Lifeline.name)
          .concat('.').concat(p.second.sent.name) endif} in
           t.source.name = if p.first = OclVoid then 'WaitState'
       else p.first.name.concat('State') endIf and
        t.target.name = if r.Pair → at(r.Pair → indexOf(p)+1).first()= OclVoid then
       ‘WaitState’ else r.Pair → at(r.Pair→ indexOf(p)+1).first().name.concat(‘State’) endIf
       and t.received = p.second.received and t.send = p.second.sent and
        t.received.EventConstraint = p.second.received.EventConstraint and
         t.sent.EventConstraint = p.second.received.EventConstraint and
          if acc → excludes(tr = t) then acc → including(t) else
           let t1 = acc → select(tr | tr.received=t.received and tr.send=t.send and
            (tr.received.EventConstraint<> t.received.EventConstraint
             or tr.send.EventConstraint<> t.send.EventConstraint))
               if t1→ notEmpty() then
         t1.received.EventConstraint.body.concat(t.received.EventConstraint) and
         t1.send.EventConstraint.body.concat(t.send.EventConstraint) endif endIf)
```

Recursive operation allRoutes() (here sr is a variable used for saving the sequence of obtained routes during recursion):

```
Context InterfaceStateMachine def: sr : Sequence(Route)={}
Context InterfaceStateMachine::allRoutes(st:InterfaceState,r:Route):Sequence(Route)
if st.name<>‘WaitState’ then ism.iroute(st)→ iterate(tu;acc:Route=r | acc→ append(tu)
    ism.allRoutes(tu.second.received.receivedOn.target,acc))
else r→ append(Tuple{first=oclVoid, second=Tuple{received=oclVoid,sent=oclVoid}})
     if sr→ excludes(r) then sr→ append(r) endif endif result=sr
```

The utility operation iroute() used for obtaining all possible pairs from one state and its outgoing transitions:

```
Context InterfaceStateMachine :: iroute(is:InterfaceState): Sequence(pair)
Post:result = is.outgoing → iterate(t; acc:(Sequence(pair))= {} |
   acc → append(Tuple{first.name=
if is.name=‘WaitState’ then OclVoid else is.name.substring(1,is.name.size()-5),
   second=Tuple{received =t.received,sent=t.sent}})
```

FROM USE CASES TO WELL STRUCTURED CONCEPTUAL SCHEMAS

Lina Nemuraite and Bronius Paradauskas*

1. INTRODUCTION

Today, the growing interest on design of schemas of information systems is noticed. Traditional schema design is focused on data storage, i.e. enterprise database development. In modern systems, virtual schemas and explicit behavioural schemas of business processes or services also are needed. Apart relational, object relational or object databases, these schemas may be implemented in XML (Ullman and Widom, 2002; Elmasri et al., 2002) or in languages based on XML (e.g. WSDL, WSCI, BPEL4WS (Arkin et al., 2002; Andrews et al., 2003)).

It is difficult to overestimate the importance of schemas and their intended evolution. Manipulation with schemas enable automation of development of software in CASE tools, supported by Model Driven Architecture (MDA) (Frankel, 2003; Kleppe et al., 2003; Mellor and Balcer, 2002); management of e-business processes; they are cardinal in Semantic Web and Data Grids. Resources of data and services on the Web should be supplied with instrumentality for understanding not only structural, but also behavioural features, e.g. business processes exposed as Web services. Specifications of DAML-S (DAML-S, 2003), the language for computer interpretable description of behavioural semantics of Web services, also should be generated from behavioural schemas.

Traditionally, there are 3 schema levels: conceptual, logical, and physical (Ullman and Widom, 2002). The most of novel technologies still are based on low-level (i.e. implementation) schema design issues. It contrasts with ideas of raising level of abstraction in software development, "programming" with models, fostered by OMG (Object Management Group, 2004) that supports set of standards (UML, OCL, MOF, CWM, XMI etc.) enabling the paradigm shift from the focus on programs toward models. The newest version UML 2.0 is intended to give a very precise definition of the semantics of its models enabling direct translation of UML models into programs. In conjunction with other standards, it gives promise to serve as universal language enabling high level of abstraction and interoperability between peoples, systems, tools, etc.

* Kaunas University of Technology, Studentu 50, LT-51368 Kaunas, Lithuania, nemur@soften.ktu.lt, parad@soften.ktu.lt.

The goal of this paper is to propose principles for development of well-structured conceptual schemas describing state and behaviour of today service oriented information systems, firstly at the requirements level. Design Independent Model introduced in (Ceponiene et al., 2003) means elaborated model of requirements where entities of problem domain and conceptual operations following from user goals are precisely specified at relevant level of abstraction. Independence from design means that in these models no decisions are made with regards to software architecture – these decisions are postponed to design stage (Platform Independent Model (PIM) (Siegel, 2001). Such strict separation of concerns serves for clarification of development phases and potential formalization of design process.

Well-structured schema is meant normalized analogously to the relational schemas on the base of project/join dependencies (Ullman and Widom, 2002; Paradauskas and Nemuraite, 2002). Such conceptual schema must be successfully mapped to relational, object-relational, object or XML logical model and implemented with corresponding implementation level schema. Step-wise development procedure is based on ordering dependencies between entities of problem domain. This proposal is tied to fact, that the definitive corollaries in UML, OCL, and related studies are nearing to valuable features founded in Extended Entity–Relationship EER and Relational models. Historically, many authors (e.g. Thalheim, 1996; Paradauskas, 1994; Paradauskas and Nemuraite, 2002; Balsters, 2003) have studied integration of EER and relational modeling with object–oriented aspects.

The rest of the paper is organised as follows. In Section 2, the framework for interface-based representation of use cases having semantics of business transactions is proposed. In Section 3, principles of development of conceptual schema of information system from analysis of use cases are presented. Section 4 is devoted to definition of conceptual join operations and re-using of the main database development criterion to development of well-formed schema of service-oriented information system. Finally, Section 5 concludes the paper and discusses possible future work.

2. INTERFACE-BASED REPRESENTATION OF USE CASES HAVING SEMANTICS OF BUSINESS TRANSACTIONS

For definition of software requirements, use cases are widely acknowledged. In (Ceponiene and Nemuraite, 2004) it was proposed to represent use cases as interfaces of system under development. This conception fits well with UML 2.0 (Unified Modeling Language Superstructure Specification, 2003) definition:

"A Use Case is a kind of behavioural classifier that represents a declaration of an offered behaviour". It may be associated with subject – classifier, representing system under development. "Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors". Alone use cases have a little meaning; they must be provided with specifications, describing scenarios of interactions between actors and the system. Specification of use cases in natural language is well suited for statement of initial requirements and communicating with stakeholders. But there are no means for detail specification of elaborated use cases representing consistent requirements and their association with the rest of system. The definition of interface is nicely suitable for this purpose:

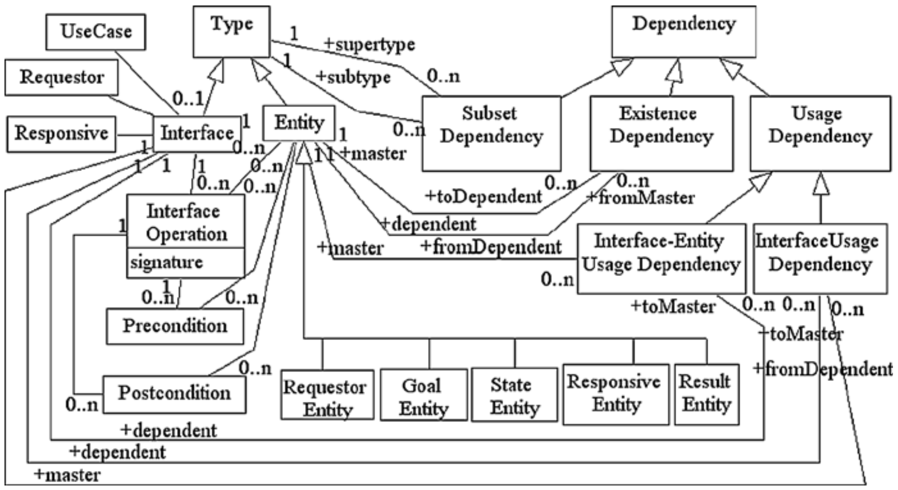


Figure 1. Metamodel of use case specification conceptualised as extended communicative action loop.

"An interface declares a set of public features and obligations that constitute a coherent service offered by a classifier. Interfaces provide a way to partition and characterize groups of properties that realizing classifier instances must possess". Shortly, interface is "a named set of operations that characterize the behaviour of an element" (Unified Modeling Language Infrastructure Specification, 2003).

Use cases also may represent business transactions, from atomic to complex, including B2B (Business-to-Business) transactions executed by multiple participants (UN/CEFACT, 2002). At some abstraction level, they may be thought as specifications of joint actions or operations performed by several agents. Business use cases also may be specified as interfaces to behaviour (business process). So use cases are powerful instrument for abstraction and may be applied at many levels.

In this paper, use cases are used for definition of conceptual schema of a system under development at requirement level. Resulting conceptual schema, visualised as UML class diagram, contains entities of problem domain and interfaces; it defines desirable state and behaviour along with integrity constraints and derivation rules, as primarily stated in (ISO/TR 9007, 1987).

In (Nemuraite et al., 2002) the extended communicative action loop was proposed as canonical unit for modeling of use cases, representing business transactions. Communicative action loop is model introduced by Winograd, elaborated in Language Action Perspective (later Communicative Action Perspective). It is widely reused and sophisticated in various forms, in large variety of methods and tools (e.g. (Gustas, 1997; Jayaweera, 2002; UN/CEFACT, 2002). In our case the communicative action loop is supplemented with appropriate information model and serves as canonical pattern leading the development of a conceptual schema. Metamodel of contents of use case specification is presented on Figure 1, where taxonomy of schema elements (subset of generic UML *Types*) and their roles in business transaction are given. *Entities* (often used as stereotypes in UML profiles) represent object types describing state of the system.

The metamodel includes service *Requestor* (initiating actor role, client), *Responsive* (reacting actor role, delivering service or transferring request to collaborating actor roles), and set of *Interfaces* to services. Every *Interface* has a set of *Interface Operations*, defined by their *Signature*, *Precondition* and *Post Condition*. *Existence Dependencies* relate subtypes (roles) of entities (every entity may play different roles with regards of different *Use Cases* or even in *Precondition/Post Condition* of the same operation).

Entity in role of *Requestor/Responsive* represents information about request sender/ receiver (identity and location that are important sending request through the net, user login, password etc). Object of this type is created every once when new *Requestor* appears in the system or response is created by *Responsive* (the system, permanent or transient, is bounded by its context). Entity in role of *Goal* represents information about requested service. Object of this type is analysed every once then new request is got. Entity in role of *State* represents system state relevant for request and response. Instances of these entities are created/modified every once then system changes its state (only entities, associated with *Interface Operation Preconditions/Post Conditions*, are treated). Entity in role of *Result* represents response (indicating successful achievement of the *Requestor* goal, unsuccessful execution or fault). Instance of this entity is created every once when response is sent to *Requestor* from the system.

Interface Usage Dependency represents dependency of current interface from other interfaces, i.e. possibility to transfer *Request* to other collaborating actor roles offering services that are required for delivering service by *Responsive* for fulfilling request. *Interface-Entity Usage Dependencies* represent dependencies between interface and entities, corresponding to arguments of interface operations.

So this metamodel determines what kinds of information objects (entities) *Analyst* must define in requirement specification, and what dependencies relate these entities with other entities and interfaces. Three kinds of dependencies may be revealed between elements of conceptual schema: *Subset* dependency $P \prec Q$; *Existence* dependency $P \overset{exist}{\longleftarrow} Q$; *Usage* dependency $P \overset{use}{\longleftarrow} Q$; here P, Q are object types.

Object type P is in *Subset* dependency of object type Q, if for every object $p \in P$ also $p \in Q$. The above dependency implies generalization hierarchy on classes.

Object type P is in *Existence* dependency on object type Q, if the life of each occurrence of object p of type P is embedded in the life of single and always the same occurrence q of type Q (Snoeck et al., 1999; Paradauskas and Nemuraite, 2002).

Object type P is in *Usage* dependency on object type Q, if every object of type $p \in P$ requires object $q \in Q$ for its full implementation or operation (Unified Modeling Language Superstructure Specification, 2003). In UML metamodel, *Usage* is a dependency in which the client requires the presence of the supplier. All these dependencies are reflective, antisymmetric, transitive and acyclic; they define partial order on schema elements.

It is possible to establish generic partial-ordering relationship $P \leq Q$ between schema elements as generalization of above dependencies, following the rules: *Existence* dependency is proper dependency between entities; *Usage* dependency is inherent dependency between interfaces and between interfaces and entities; *Subset* dependency is common for both types of schema elements; if schema elements are not specialized as entities and interfaces (i.e. all elements are treated as classes), *Existence* dependencies and *Usage* dependencies may overlap.

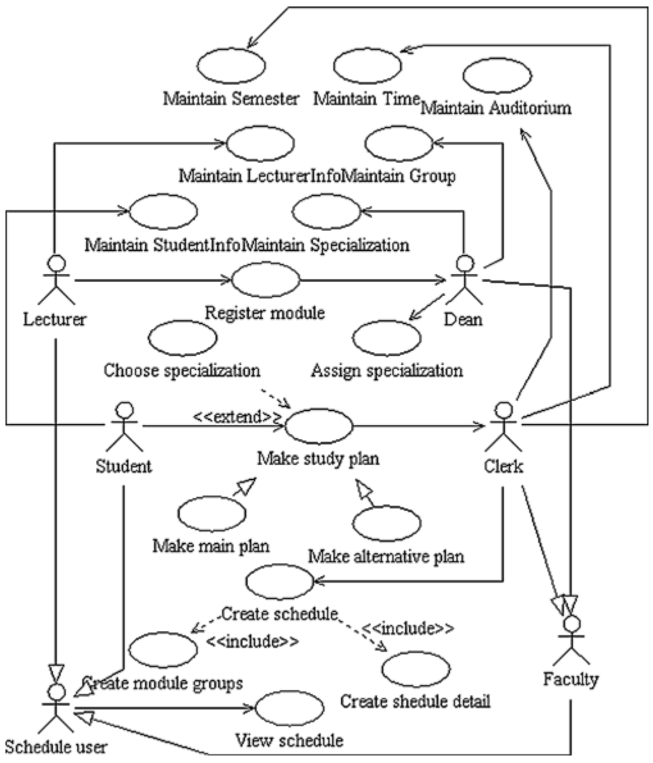


Figure 2. Use cases of schedule information system.

3. DEVELOPMENT OF SCHEMA OF INFORMATION SYSTEM

The principles of schema development are demonstrated by example. Use cases of Schedule Information System are presented on Figure 2.

The Schedule Information System may be ordinary information system on a local net as well as global e-Learning system. Despite use cases do not exhibit order of execution of business process, they are arranged according to sequence of business process steps. Thinking by scheme on Figure 1, collaborating entities and interfaces corresponding to each use case are analysed.

In Figure 3, interfaces *IMakestudyPlan, ICreateModuleGroups* and *ICreate* S*chedule-Detail* are created for use cases *Make study plan*, *Create module groups* and *Create schedule details*. Every interface may be defined by set of operations. At this abstraction level, one use case is represented as conceptual operation (one operation is devoted for every interface) and entities are grouped according to their participation in *Preconditions* (*Pre*) and *Post conditions* (*Post*). Operations, representing business transactions, may be decomposed till atomic operations, corresponding to use case steps.

In Figure 3, *Existence* dependencies are incident between entities, associated with *Preconditions*, and entities, associated with *Post conditions*; operation arguments reference to

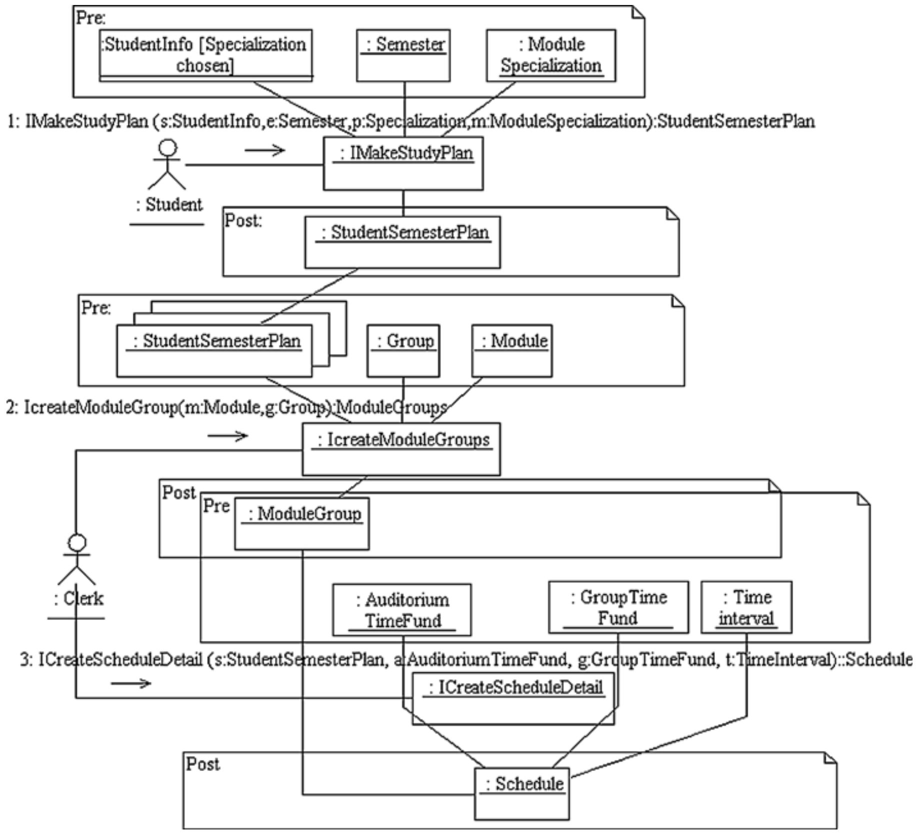

Pre:
:StudentInfo [Specialization chosen]
: Semester
: Module Specialization
1: IMakeStudyPlan (s:StudentInfo,e:Semester,p:Specialization,m:ModuleSpecialization):StudentSemesterPlan
: IMakeStudyPlan
: Student
Post:
: StudentSemesterPlan
Pre:
: StudentSemesterPlan
: Group
: Module
2: IcreateModuleGroup(m:Module,g:Group):ModuleGroups
: IcreateModuleGroups
Post
Pre
: ModuleGroup
: Clerk
: Auditorium TimeFund
: GroupTime Fund
: Time interval
3: ICreateScheduleDetail (s:StudentSemesterPlan, a:AuditoriumTimeFund, g:GroupTimeFund, t:TimeInterval)::Schedule
: ICreateScheduleDetail
Post
: Schedule

Figure 3. Objects collaborating in use cases.

entities, incorporated in *Preconditions*. Starting from the first step of process, new entities and interfaces are gradually introduced; some of them are independent (for example, *Auditorium* is invented analysing *IcreateScheduleDetail* (Figure 4)). Every time, when new entity is introduced and relationship between two entities must be established, proportion between life cycles of these entities is considered.

If entities are existence-independent, but must be related, the new entity is added to represent this relationship (for example, *ModuleSpecialization* is introduced to relate *Module* and *Specialization*). In class diagram (Figure 4), association ends having multiplicity with lower bound "1" are expressing existence dependencies, and dashed arrows (dependencies in UML) are expressing usage dependencies (stereotype «use» is omitted because of overloaded visibility).

Functional features at requirement level are represented as interfaces corresponding to initial use cases. «include» relationship between use cases is mapped to generalization relationship between interface, corresponding included use case, and interface, corresponding base use case. It follows from definition, that behavior of included use case is obligatory in performance of base use case.

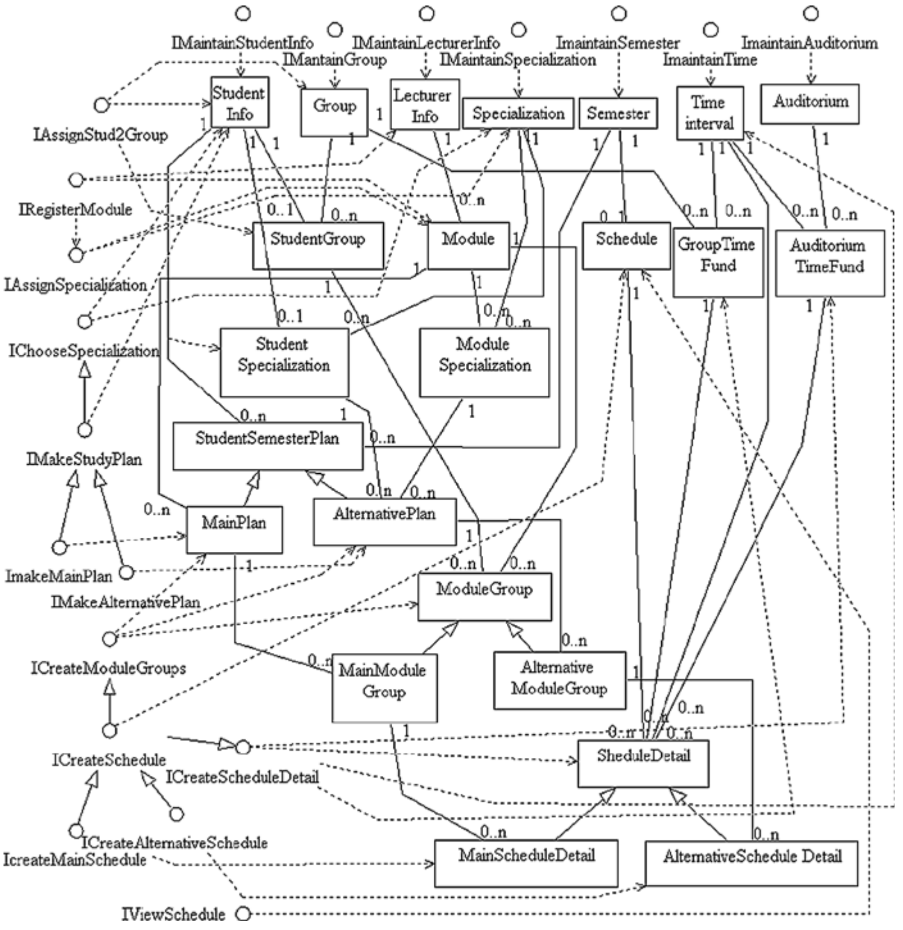


Figure 4. Class diagram representing entities of problem domain and interfaces.

Generalization relationship between use cases is mapped to generalization, «extend» relationship is mapped to generalization between interface, corresponding to extension use case, and interface, corresponding to base use case. It follows from definition that extended use case renders additional behavior that may be incorporated if some condition is satisfied.

For implementation, generalization between interfaces may be changed to usage relationship; it does not influence the generic ordering in conceptual model. Usage dependency is established between interface and entities that are used by operations of interface; also this kind of dependency is incident between interfaces belonging to different actors or systems.

Besides analytic and visual clarity, resulted conceptual schema has advantage in that fact that no cyclic or reflexive relationships occur. Ordering of schema at conceptual level has great value for derivation of schemas of logical and implementation levels, evenly for schemas, representing system state or behavior. Acyclic models are desirable for database schemas, XML schemas (Cagle et al., 2001; Elmasri et al., 2002), software components (D'Souza and Wills, 1999; Cheesman and Daniels, 2000; Martin, 2003), and schemas of business processes (Andrews et al., 2003; Neiman et al., 2002).

4. DEVELOPMENT CRITERION FOR WELL FORMED CONCEPTUAL SCHEMAS

Well-formed object-oriented conceptual schema may be defined similarly to Relational Data model that is based on powerful theory of dependencies and integrity constraints (Thalheim, 1996). The benefits of the relational model are to provide simple and easily used data structures. Object-oriented models have more rich type system where part of constraints is implied in model structure. The essential object-oriented concepts are abstraction, generalization, aggregation, composition, and polymorphism. It has disadvantages in that some concepts are ambiguously understood by different subjects and have different implementations in different programming languages and databases, or are not supported at all.

For precise modeling, ambiguous concepts may be replaced by set of well-defined simple concepts. For example, authors of precise modeling (Starr, 2002; Mellor and Balcer, 2002) propose to use only associations and complete, disjoint generalizations; multiple generalizations if properly stated fit as well. In (Gogolla and Richters, 2002) it is shown, how all kinds of relationships may be converted to associations and constraints. OCL 2.0 (Cook et al., 2002; Unified Modeling Language: OCL, 2003; Warmer and Kleppe, 2003), MOF (Meta Object Facility (MOF) Specification, 2004) object models include only core concepts (object types, attributes, associations, roles, multiplicities, and generalizations). In general, conceptual schema completely may be defined by its elements and constraints. In our case, the main constraints of conceptual schema are *Subset*, *Existence*, and *Usage* dependencies; *Coexistence* dependencies relate schema elements that both have the same existence or usage dependent type. Objects related by *Coexistence* dependency must exist together during some time interval coincided with objects that are existence-dependent from them. *Coexistence* dependency may be derived from *Existence* dependencies.

Interfaces are related by *Usage* dependencies with entities, representing states, and other interfaces. Interfaces and entities together represent the subschema (view) of business transaction that may be performed using interface of service offered by information system, supporting business system. Views are usual in database design, but it is purposeful to define concept of views at conceptual model level. As in database design, view is relationship (set of tuples) between object types obtained by join of these object type instance sets.

Creating the view of conceptual model means creation of auxiliary class (root), having associations with classes that comprise the view. Such approach is presented in (Balsters, 2003), where join operation is appreciated at attribute level. In current work, the conceptual join operation relates objects. It has sense considering views as views play the main role in integration of databases or interactions between different systems, queries requesting Web services or manipulation with UML models for generation of implementation code under MDA. View under conceptual model may be defined by join operation analogous to join operation of relational algebra.

As in relational model, conceptual normalization should take place. Normalization concept is rarely analysed in object-oriented literature. In (Ambler, 1998) three normal forms are presented. Shortly, object schema is in third normal form when all of its classes encapsulate only one set of cohesive behaviour that is specific to class itself. Using above process of development of conceptual model, the fourth and fifth normal form, may be obtained. Then criterion of well-formed schema may be expressed analogously to main development criterion of relational database systems:

If relationship between schema types conforms to projection/join dependency Schema(Set(Class)), it is possible to project it to class schemas and reproduce original schema by performing natural join operation on these class schemas. The result of join may be obtained from Cartesian product of class instance sets satisfying object-linking constraints from UML model. The development criterion of well-formed conceptual schema may be expressed as invariant in OCL:

```
Context Schema
def:sch:TupleType = self.allConnections → ordered() → make()
inv: {1,..self.sch.allAttributes → size()} → iterate (i:Integer; acc:Set(sch.AsType()) = {}|
 acc = acc → union (self → collect (c:Class | at(i).name = c.name))) → join() = self
```

Here ordered() denotes operation that aligns schema elements according to partial ordering relationship that generalizes subset, existence and usage dependencies. Join operation join() can be derived from operation of Cartesian product; objects of conceptual schema may be linked on the base of values of their structural features (association ends), that may be accessed by operation from UML 2.0 metamodel allConnections,[*] which results in set of all association ends of class including association ends of all its parents.

Natural join operation:[†]

```
Context Schema::join(s:Set(Class)):Schema
pre:
  s → forAll (t | self.allConnections → exists (ae | ae = t)
post:
result = s → ordered() → nproduct() → let T:sch.asType = s.ordered() → make() in
iterate(tu:T; acc:Set(T) = {} | acc = acc → union (tu.allAttributes →
  iterate (a:Class; acc1:T = tu.allAttributes → first() | let k = tu.allAttributes → size() - 1 in
    if tu.indexOf(a) < k then acc1.at (indexOf (a) + 1) =
      if a.allConnections → includes((tu.allAttributes - acc1.allAttributes) →
        at (indexOf (a) + 1))
        then tu.allAttributes → at (indexOf (a) + 1) else k = 0 and acc1 = {}
      endIf endIf )))
```

Here nproduct(set(Class)) is OCL operation for n-ary Cartesian product generalized from operation of binary product(). Operation nullJoin() (allowing undefined values) may be described:

[*] There may be some discrepancies between names of operations in different versions of OCL and UML metamodels, e.a, allConnections in UML 2.0 specification correspond to allAssociationends in OCL 2.0 specification.

[†] The definition of join will be slightly different, taking the existing class as a root.

```
Context Schema::join(s:Set(Class)):JoinSchema
pre:
  s → forAll (t | self.allConnections → exists (ae | ae = t)
post:
result = s → ordered() → nproduct() →
 let T:sch.asType = s → ordered() → make() in
iterate (tu:T; acc:Set(T) = {} | acc = acc → union (tu.allAttributes →
  iterate (a:Class;acc1:T = tu.allAttributes → first() |
    if tu.indexOf(a) < tu.allAttributes → size () - 1
      then acc1.at (indexOf (a) + 1) =
        if a.allConnections → includes ((tu.allAttributes - acc1.allAttributes) →
         at (indexOf(a) + 1))
          then tu.allAttributes → at (indexOf (a) + 1)
            else OclVoid endIf endIf )))
```

Other kinds of conceptual joins corresponding to LEFT, RIGHT, and others, defined in SQL standard (Melton, 2003) and used in Database Management Systems, may be described in similar manner.

Definition of θ-join in OCL:

```
Context Schema:: θ-join (s:Tuple (classes (Set(Class)),θ :Set (OclExpression)) :JoinSchema
pre:
  s → forAll (t | self.allConnections → exists (ae | ae = t)
post:
result = s → ordered() → nproduct() →
 let T:TupleType = s → ordered() → make() in
iterate(tu:T; acc:Set(T) ={} | acc = acc → union (let r:T =
 tu.allAttributes → iterate(a:Class; acc1:T = tu.allAttributes → first() |
    let k = tu.allAttributes → size() - 1 in
  if tu.indexOf (a) < k then acc1.at (indexOf (a) + 1) =
    if a.allConnections → includes ((tu.allAttributes - acc1.allAttributes) →
     at (indexOf (a) + 1))
    then tu.allAttributes at (indexOf (a) + 1)
     else k = 0 and acc1 = {}
    endIf endIf ))) → iterate (r:T; acc2:Set(T)={} | acc2 → including
     if θ → forAll (θi | θi.evalθ(r) = true) then r endIf )
```

Here by evalθ(r) is denoted operation that evaluates constraints, described in θ-join(), against tuple r that is element of set of results of θ-join.

5. CONCLUSION

In this paper, the attempt is made to present principles of development of schemas of Information Systems at conceptual level based on concepts similar to Relational Model. In contrast to relational schemas, UML models schemas of service oriented Information Systems should include specifications of operations and constraints, described in declarative manner. As elements of such schemas interfaces and entities are proposed. Constraints that are not implied by model structure may be specified in OCL and related to schema elements.

The way to develop well-formed schemas is proposed on the base of subset, existence and usage dependencies among schema elements. These dependencies are plain expressed by associations and multiplicities, but model must be structured in specific way, based on analysis of life cycles of entities used in business transactions.

For manipulation with elements of schema, conceptual join operation is useful. Join operation is not included in OCL 2.0, but it may be derived using other OCL operations corresponding operations of relational algebra. Namely, Cartesian product was used to define join operation manipulating with objects.

Linking objects in tuples comprising the result of conceptual join means possibility to integrate features of these objects. If entities are linked, their attributes may be associated in result of query. If interface is linked with entity, it means, that this entity is used as argument in operation of this interface. Sub-schema of business transaction is comprised of interface; entities linked with this interface by usage dependency, and possible other interfaces that are required by interface under consideration.

Conceptual join operation may be extended to feature level that has no principal difference as attributes may be complex types and conceptual operations may be composed of other operations. Linking of operations means that preceding operation op1() may call succeeding operation op2(), i.e., sending message expression ^op2 is presented in post condition of op1(). It means that precisely specified schema can carry all information about structure and behaviour of system under development, and interactions as well as state transitions may be generated from conceptual schema for verification of consistency.

These ideas will be employed in further work. Schema conception may be functionally used in practical applications for generation of software code and multiple standard specifications required in development of today service-oriented systems (database and XML schemas, WSDL, compositions of web services etc.), various problems of integration and querying on the web.

Considering conceptual schema consisting of entities and interfaces limits models to requirements specification level. For definition of design level schema, requirements model must be integrated with software architecture, and control or component classes representing design decisions should appear. It is believed, that this process may be automated in future.

REFERENCES

Ambler, S. C., 1998, *Building Object Applications That Work*, Cambridge University Press, Cambridge, p. 506.

Andrews, T., et al., 2003, *Business Process Execution Language for Web Services*, BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems (May 3, 2004); http://www-128.ibm.com/developerworks/library/ws-bpel.

Arkin, A., et al., 2002, *Web Service Choreography Interface*, BEA Systems, Intalio, SAP, Sun Systems (May 3, 2004); http://www.w3.org/TR/wsci.

Balsters, H., 2003, Modelling Database Views with Derived Classes in the UML/OCL-framework, in: *"UML" 2003 – The Unified Modeling Language: Modeling Languages and Applications*, P. Stevens, J. Whittle, and G. Booch, eds., Springer-Verlag, Heidelberg, LNSC 2863, pp. 295–309.

Cagle, K., et al., 2001, *Professional XML schemas*, Wrox Press, Birmingham, p. 691.

Ceponiene, L., Nemuraite, L., and Paradauskas, B., 2003, Design of schemas of state and behaviour for emerging information systems, in: *Computer Science Reports*, Branderburg University of Technology, Cottbus, B. Thalheim and G. Fiedler, eds., **14**, pp. 27–31.

Ceponiene, L., and Nemuraite, L., 2004, Design independent modeling of information systems using UML and OCL, in: *Databases and Information Systems, Sixth International Baltic Conference BalticDB&IS 2004*, Riga, Latvia, June 6–9, J. Barzdins et al., eds., Riga, Latvia, June 6–9, **672**, pp. 357–372.

Cheesman, J., and Daniels, J., 2000, *UML Components*, Addison-Wesley, Boston, p. 208.

Cook, S., et al., 2002, The Amsterdam Manifesto on OCL, in: *Object Modeling with the OCL, The Rationale behind the Object Constraint Language,* A. Clark and J. Warmer, eds., Springer-Verlag, London, LNSC 2263, pp. 115–149.

D'Souza, D. F., and Wills, A. C., 1999, *Objects, Components, and Frameworks with UML: The Catalysis Approach,* Addison-Wesley, Boston, p. 816.

DAML-S, 2003, *DAML-S: Semantic Markup for Web Services* (May 3, 2004); http://www.daml.org/services.

Elmasri, R., et al., 2002, Conceptual modeling for customized XML schemas, in: *Conceptual Modeling – ER 2002: 21st International Conference on Conceptual Modeling Tampere, Finland, October 7–11, 2002*, S. Spaccapietra, S. T. March, and Y. Kambayashi, eds., Springer-Verlag, Heidelberg, LNCS 2503, pp. 429–443.

Frankel, D., 2003, *Model Driven Architecture: Applying MDA to Enterprise Computing*, Wiley Publishing, Inc., Indianapolis, Indiana, p. 352.

Gogolla, M., and Richters, M., 2002, Expressing UML class diagrams properties with OCL, in: *Object Modeling with the OCL, The Rationale behind the Object Constraint Language,* A. Clark and J. Warmer, eds., Springer-Verlag, London, LNSC 2263, pp. 85–114.

Gustas, R., 1997, *Semantic and pragmatic dependencies of information systems*, Monograph, Technologija, Kaunas, p. 274.

ISO/TR 9007, 1987, *Concepts and Terminology for the Conceptual Schema and Information Base*, ANSI, New York, p. 120.

Jayaweera, P. M., 2002, *A Methodology to Generate e-Commerce Systems: A Process Pattern Perspective (P^3)*, Philosophy Thesis, Stoholm University and Royal Insititute of Technology, Stockholm, p. 97.

Kleppe, A., Warmer, J., and Bast, W., 2003, *MDA Explained. The Model Driven Architecture: Practice and Promise*, Addison-Wesley, Boston, p. 192.

Martin, R. C., 2003, *Agile Software Development, Principles, Patterns, and Practices*, Prentice Hall, Upper Saddle River, p. 529.

Mellor, S. J., and Balcer, M. J., 2002, *Executable UML. A foundation for model-driven architecture,* Addison-Wesley, Boston, p. 368.

Melton, J., 2003, *(ISO-ANSI Working Draft) Foundation (SQL/Foundation)*, WG3:HBA-003 H2-2003-305, August, p. 1121.

Meta Object Facility (MOF) Specification, 2004, Version 1.4, ISO/IEC 19502::2004(E), p. 122.

Neiman, S., Kraunelis, L., and Schmelzer, R., 2002, *Ebxml*, Addison-Wesley, Boston, p. 400.

Nemuraite, L., Paradauskas, B., and Salelionis, L., 2002, Extended communicative action loop for integration of new functional requirements, *Information Technology and Control* **2**(23):18–26.

Object Management Group, 2004 (June 1, 2004); http://www.omg.org.

Paradauskas, B., 1994, Denotational semantics of objects relationship, in: *Proc. of the Baltic Workshop on National Infrastructure Databases: Problems, Methods, Experiences*, Trakai, Lithuania, **1**, pp. 230–241.

Paradauskas, B., and Nemuraite, L., 2002, *Data Bases and Semantic Models*, Monograph, Technologija, Kaunas (in Lithuanian), p. 303.

Snoeck, M., et al., 1999, *Object-Oriented Enterprise Modeling with MERODE*, University Press, Leuven, p. 227.

Starr, L., 2002, *Executable UML. How to build class models,* Prentice Hall, Upper Saddle River, p. 418.

Thalheim, B., 1996, An overview on semantical constraints for database models, in: *6th International Conference "Intellectual systems and Computer science"*, Moscow, December 1–10.

Ullman, J., and Widom, J., 2002, *A First Course in Database Systems*, 2nd ed., Prentice-Hall, Upper Saddle River, p. 511.

UN/CEFACT Modeling Methodology, 2002, UNCEFACT/TMG (May 3, 2004); http://www.unece.org/cefact.

Unified Modeling Language Infrastructure Specification Version 2.0, 2003, OMG document ptc/03-09-15 (May 3, 2004); http://www.omg.org.

Unified Modeling Language Superstructure Specification Version 2.0, 2003, OMG document ptc/03-08-02 May 3, 2004); http://www.omg.org.

Unified Modeling Language: OCL Version 2.0, 2003, OMG document ptc/03-08-08 (May 3, 2004); http://www.omg.org.

Warmer, J. B., and Kleppe, A. G., 2003, *The Object Constraint Language: Precise Modeling with UML*, Addison-Wesley, Boston, p. 112.

A MODEL OF INFORMATION SYSTEMS DEVELOPMENT FOR LEARNING VIRTUAL ORGANIZATIONS

Mart Roost, Rein Kuusik, Karin Rava, and Tarmo Veskioja*

1. INTRODUCTION

The information society generates virtual subjects and their organizations. These organizations form, function, and develop as a result of system work that cooperative subjects perform on the level of information systems. The success of such organizations depends on their ability to adapt to the environment and learn.

The Virtual Organization (VO) forms on the basis of uniting (and/or separating) resources of independent work units called subjects or actors, where such a subject dynamically creates needed roles into multiple environments of different VO-s over the world, continuing at the same time its independent existence. Development of such organizations is related to the problem of decentralized or subject-centred development of information systems that is based on synchronization of the business model of the subject with business models of the environment (that means the related subjects).

The Information System (IS) is the main organizational interface of a contemporary and/or future organization (as a subject of information society) with embracing world, one's immediate functioning and learning environment. Development of this environment (in other words, analysis, design and building of IS) has to be a central role of the Learning Virtual Organization (LVO). To perform this role, the organization needs a methodology that enables the members – subjects/actors of the organization to develop (analyze, design, build etc.) the IS immediately in their natural work environment, which is the IS under development. This concept, the self-advancing information system, contains the subject as well as the environment of development (Roost, 1996). We call the development process of such an information system *IS self-development*. How can it be accomplished? The key problem here is building adequate space for development (Roost et al., 1998), based on decentralized models of system work and development (Lyytinen et al., 1998), service-

* Tallinn University of Technology, Raja tn. 15, 12618 Tallinn, Estonia, mart@cc.ttu.ee, kuusik@cc.ttu.ee, rava@cc.ttu.ee, tarmov@staff.ttu.ee.

oriented architectures (Apshankar, 2002), and virtual communities of Information Systems Development (ISD) practice (Powazek, 2002; Preece, 2000; Wenger et al., 2002).

In the next section of the paper, the problem space and the key concepts are defined and analyzed. In the third section, the core solutions are developed.

2. PROBLEM ANALYSIS

In this section, at first the general problem space is defined, then the key concepts in this space are defined and analyzed, and the motivation and outline for the (rest of the) paper is given.

2.1. Problem Statement

The business environment is quickly changing. A learning (intelligent) organization is an organization that is able to adapt to its environment by changing its business model (Marshall, 1999). The key for such organizations success is a continuous process of their business model innovation. The organization and the IS must evolve dynamically (and partially automatically) with business according to changes in the business model and in the business environment (Russo, 2000). A new development situation has emerged, where traditional ISD models and paradigms do not work (da Cunha et al., 2001).

How do we develop an IS for LVO-s that evolve dynamically and automatically with business?

- What are the main requirements to IS and ISD in the context of LVO?
- What is the ISD approach/model applicable in this context?
- What is IS self-development?
- What is the meta-model behind this concept?
- How can it work in the practice?
- What is the role of the ISD community in this approach/model?

To answer these questions, some of the core concepts of this problem space are defined and analyzed next.

2.2. Virtual Organization

VO is defined as "a temporary network of independent institutions, enterprises or specialized individuals that through the use of Information and Communication Technology spontaneously unite to utilize an apparent competitive advantage. They integrate vertically, bring their core competencies and act as a single organizational unit". Another definition states that "a VO is an identifiable group of people or organizations that make substantially more use of Information and Communication Technologies than physical presence to interact, conduct business and operate together, in order to achieve their objectives." (Strausak, 1998)

To be exact, we can mention that a VO must not be always a temporal organization, but VO-s can be developed also in order to generate some strategic, sustainable advantages for enterprises. This is so in the case of an extended enterprise (Wenger et al., 2002) that is a VO, too. In the information society, each organization serves also as a VO.

VO forms on the basis of uniting (and/or separating) core competencies and resources of independent work units called subjects or actors, where such a subject creates dynamically needed roles into multiple environments of different VO-s over the world, continuing at the same time its independent existence (Bultje et al., 1998; Reithofer et al., 1997; Strausak, 1998). Each subject, participating in a VO, should be handled also as a VO, if it integrates and organizes, in the same way, other subjects, their resources and services on the level of its IS to achieve some common goals.

VO is an organization of virtual subjects in an IS, where a virtual subject is a representation of some "real" subject in the IS. The IS "belongs" to one of these subjects who represents this VO as a whole. This IS serves as a common work and development space for the related subjects. The architecture of this space should be based on the role concept in VO.

2.3. Learning Organization

A VO, as each organization in the information society, is normally a learning organization. This is an organization able to adapt to its environment (Marshall, 1999). As the basis of success for an organization in fast changing environment, more and more the process of the business model innovation becomes (da Cunha et al., 2001). IS and VO must evolve dynamically and partially automatically with business, according to changes in the business environment and in the business model.

The development of a LVO can be handled in the framework of decentralized or subject-centred development of information systems (Roost, 1996; Roost et al., 1998). For each subject in LVO, this is based on synchronization the business model of the subject with the business model of the environment, where the environment here is the subject, who represents the LVO as a whole. The two different approaches applicable here are Business Reengineering (Kirikova, 1998; Scheer, 2000) and Bottom-Up Planning of Virtual Enterprises (Reithofer et al., 1997). Both of them can be considered as System Work (Mikli et al., 1998) in Work Systems (Alter, 2002).

2.4. Work System

A Work System or Soft System (Checkland et al., 1990) is a view of work as occurring through a purposeful system (Alter, 2002). This system has an active, "social" component or view called Subject (work unit, actor). The Subject, usually a human/organization, has requirement and capability to develop (analyze, design, etc.) its environment, that is an organization of multiple work systems coordinated to accomplish goals that these work systems cannot accomplish individually. See the Figure 1 below for clarifying some concepts.

Similarly with VO-s and Subjects (see in the subsection 2.2 before), each organization is also a work system and a work system can be handled as an organization that consists of several work systems that participate in it. The participation is possible only through particular roles in the organization.

In the behavioural or Role view (see also in 3.2 below), a Work System (as a Subject) performs System Work. This is usually a collective work of related Subjects playing roles in the Work System's organization. This work is directed to ensuring normal/desired func-

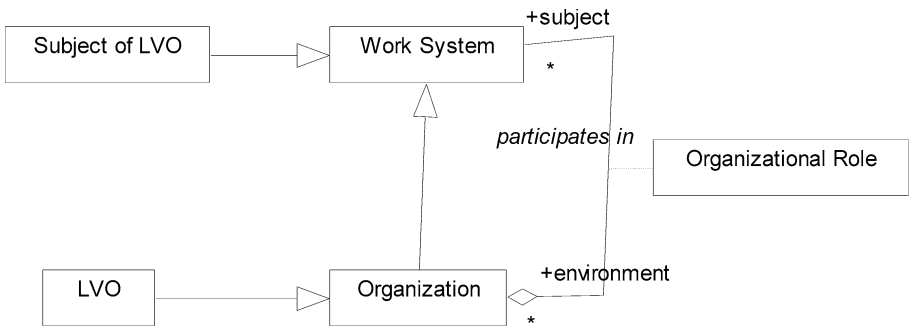


Figure 1. Relationships between Work Systems and Organizations (UML Class diagram).

tioning and progress of the system/organization. Usually with the term "System Work" development processes (like System Analysis and Design) are automatically included. Modelling, management, quality control and training are the main activities of System Work. LVO is a Work System that forms, functions and develops as a result of system work that cooperative Subjects perform on the level of IS.

2.5. Information System

According to Davis (Davis, 2000), an IS is "a part of an organization that provides information and communication services required by the organization". Another definition states that IS is a view of organization's information work as occurring through a purposeful system (Mikli, 1997), so an IS can be handled as a specialized Work System.

In the context of LVO, the IS provides and mediates not only the information and communication services but also the development services of the organization and its IS. This makes it possible to handle ISD processes in the context of the organizational roles in the business organization. Specialized development organizations can then focus on services for ISD.

According to our interpretation, the IS is an active information level of a subject in an information society, defined by and generated from models of the subject. An IS is a virtual model-system that generates the actual/virtual environment (development space) of the subject. This concept, the self-advancing IS, contains the subject as well as the environment of development. We call the development process of such an information system IS self-development (Roost et al., 1999 and 2001).

2.6. Self-Development

In the new development situation, related to LVO, responsibility for decisions on IS and applications is divided between all business functions of the organization (da Cunha et al., 2001) and internal or external parties performing these functions (IS function [Davis, 2000] is one of them). In such context, each party or subject is responsible for developing its own capability areas and the respective view of IS as a whole (see 3.1, Figure 2). Such approach we call IS self-development that is subject to the decentralized model of ISD.

Here the whole system is not directly developed, but autonomous parts/subjects of the system are designing itself (auto-design) in mutual cooperation and in a common space of development. In the context of LVO, this space is formed in the IS under development. Consequently, each IS must contain a development subsystem as its core, that provides high-level services for supporting IS self-development by the subjects of the LVO. In the role of the service provider potentially the whole global community of ISD practice (Wenger et al., 2000) is seen. The development subsystem should operate as two-directional interface between the subjects of the LVO (local ISD communities) and the global ISD community. To accomplish this vision, the first step is to design a role-based meta-model to build an adequate space of ISD (see in 3.2).

2.7. Motivation for the Paper

The research questions, stated and analyzed before, are concentrated on modelling of the framework for the IS self-development, made to tailor a LVO.

This framework can't be based on the classical ISD approaches and paradigms because these were created for entirely different development situations (da Cunha et al., 2001) not proper to LVO. A classical development situation is subject to the centralized model of ISD. Here ISD is treated predominantly as a "one-time effort" with fixed final results. The whole process is managed from one centre, which is, depending on the development strategy, the IS/IT department of the customer organization, or a specialized IT organization. In both case, the whole ISD tends to become developer-centric, because the business functions (excluding the IS function) in the customer organization do not have enough autonomy in the framework of the hierarchical development/project organization and the total top-down development approach.

An examination of well established systems development methods and schools reveals that they are targeted to support the centralized development models (Structured Analysis and Design methods, like the Oracle CASE Method in [Barker, 1991]), or the decentralized form of development is not outlined (in Object-Oriented Analysis and Design with Unified Process [Jacobson et al., 1999], Soft Methods [Checkland et al., 1990], and Agile Methods [Larman, 2001]).

The aim of the rest of the paper is to introduce a role-based meta-model as a basis for a decentralized ISD framework needed in the context of LVO-s. This model can be accomplished on the technical basis of the service-oriented architectures (Apshankar, 2002), and the social basis of virtual communities of ISD practice (Powazek, 2002; Preece, 2000; Wenger et al., 2002).

3. DESIGNING A SOLUTION

From the previous section we know, what is IS self-development and how it works, in general, in the context of LVO-s. In this section we explain, how the self-development can be accomplished. A short answer is: by building an adequate space of development.

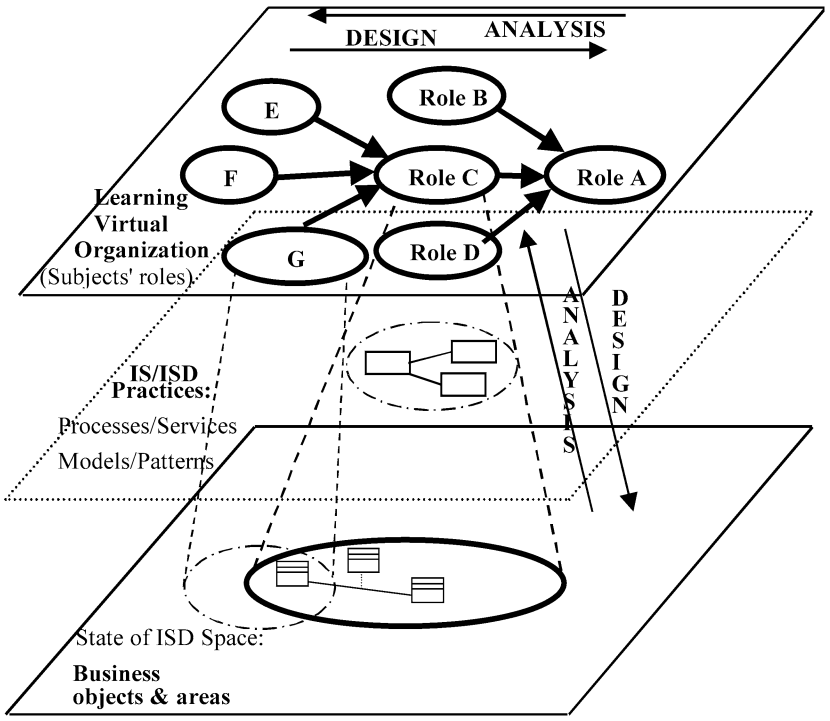


Figure 2. The general role-based architecture for the ISD Space.

3.1. ISD Space

System Work and ISD take place in some space, determined by strategic dependencies (Kirikova, 1998) between a customer organization, that is the subject of development, and the developer organization(s), that represent the environment for development. In the role of the developer organization, potentially the whole global community of ISD practice (service providers) may be seen. An ISD Space can be defined as a multidimensional logical space for organizing ISD-related artefacts and patterns. Analyzing and designing the ISD space in the context of a particular organization is the main task of the IS strategic development (strategic analysis, strategic design).

In the self-development approach, analysis and design of the ISD space is based on the Role-Model (Roost, 1996; Roost et al., 1999). The central dimension here is the one of roles as main modelling patterns for LVO and its IS. ISD is handled in the framework of organizational roles of the customer organization, which form the basic (self-advancing) structure (architecture) for the ISD space (see Figure 2).

On the upper level of the figure, we see a LVO including its subjects and their roles in this organization (subjects of IS/ISD). Developer organizations and roles (ISD service providers) also will be handled as subjects of this virtual organization (extended enterprise). In the context of IS self-development, the LVO operates as a distributed community of ISD practice (Wenger et al., 2002).

The lower level represents the state of the ISD Space as the object of system work and ISD. This state is structured by business objects, which form business areas (the circles). An area belongs to a role responsible for the managing and development of this business area. In the organizational context, we refer to a business area as an area of competence. In the ISD context, the business objects are also objects/artefacts of system work and ISD. Around these objects local communities of ISD practice will form.

The intermediate level represents IS/ISD practices: processes, services, models and patterns that make it possible to manage and develop the objects and to perform IS self-development by changing the state of the ISD Space.

3.2. The Role-Model

In this section, we present a general structure view of the meta-model behind our methodology. The model is called a Role-Model.

The central architectural unit of the development space is called subject's role – a work system modelled by sentence "Subject develops his environment", or "Environment develops the subject", in the dual view. This is a profile of development activities oriented to identification and development of some static requirement on the level of the whole organization. The global space of development divides into open local development spaces according to named profiles/roles.

On the basis of the Role-Model, a Work System has three main views (see Figure 3):

- *Subject* as the "closed" view of the system, embracing its environment. Subject is an active intelligent component or a view of the system having requirements and capabilities to develop the environment. For our ISD framework (see 3.1, Figure 2), this is a view of a Subject who represents the LVO and its IS as a whole, including its Customer and Developer Communities, both modelled here as "black boxes". As a member of the Developer communities, the Subject provides/mediates high-level IS/ISD services/capabilities for the Customer commu-

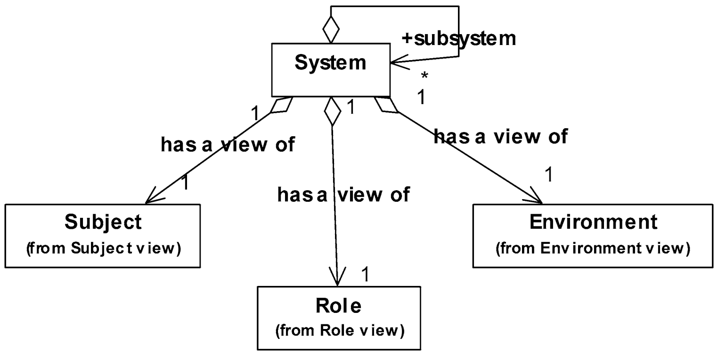


Figure 3. The general structure of the Role-Model (UML class diagram).

nities self-development. As a member of the Customer communities, the Subject mediates/publishes their common IS/ISD requirements for the Developer communities. The Subject manages the IS self-development on the basis of its business model, the requirements and the services.

- *Environment* as the "open" view of the system for the members of both the Customer and Developer Communities. In our framework, this view is modelled as a federated ISD portal that is a basis for forming the both communities as the internal and external environments, respectively, in the context of the IS self-development. Each member of any community is a Subject with its own space of development that recursively follows the Role Model.
- *Role* as the central, integrating, development view of the system, a dynamic interface between the Subject and the Environment synchronizing their (business) models. In our framework, this view is modelled as the Development Subsystem that finally implements the high-level ISD services and processes needed to accomplish the IS self-development. The main keywords are change management, system work, analysis and design.

A more formalized description of the meta-model and each of its main views are presented in (Roost, 2004). Here we give only a summary of this using the format of patterns (Larman, 2001).

Summary (pattern) for the Environment view:
Pattern (Candidate) name: Environment
Problem: How do we organize the System (LVO IS)?
Solution: Organize the System as an open network of Subjects' Information Systems (SIS). Each SIS can be (recursively) handled by the same model as the System.

Summary (pattern) for the Subject view:
Pattern (Candidate) name: Subject
Problem: How do we manage wholeness of the System (LVO IS)?
Solution: Specialize one of the Subjects to model, manage and develop the Environment. This is the Subject that represents the System as a whole.

Summary (pattern) for the Role view:
Pattern (Candidate) name: Role
Problem: How do we manage and support the self-development of the System (LVO IS) by its Subjects?
Solution: Assign to each Subject the (generic) Role to manage IS Self-Development in its Environment. Adjust/specialize the generic Role for the context of each particular Subject/Environment relationship (the Subject's Role). The generic Role is a Subject that represents the development view of the System, manages the model of IS self-development, and synchronizes the (business) model of the Subject with (business) models of the Environment. Supporting software, that we call the Development Subsystem, is needed here.

Summary (pattern) for the Role-Model
Pattern (Candidate) name: Role-Model

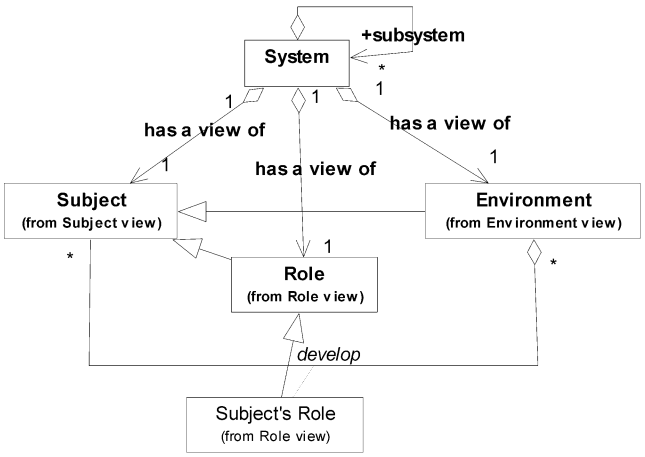


Figure 4. The Role-Model as a composite pattern (UML class diagram).

Problem: How do we architect IS (System) in the context of LVO? What are the main characteristics of this System?
Solution: Define the Subject, Environment, and Role views of the System. The System (LVO IS) must be closed (in the Subject view), open (in the Environment view) and self-advancing (in the Role view).
This concluding pattern is illustrated by the Figure 4 that integrates ideas previously illustrated separately in Figures 1 and 3.

This meta-model can be useful as a possible basis for building a common architecture for the community of ISD practice that serves as the social framework for achieving IS self-development in the context of LVO-s. If this idea is acceptable and will be supported on the level of the "global" community, the next development phase (elaboration in terms of the RUP [Jacobson et al., 1999]) of the proposed model should be good to plan and perform as "community based". Currently we are elaborating and prototyping the model in the context of some particular (extended) enterprises in our country.

4. CONCLUSIONS

An introduction to and an overview of a model of ISD for LVO were presented. The key concepts of the problem space were analyzed, an overview of a role-based framework for IS self-development in LVO-s, and the general structure of the meta-model for the framework were described.

The behavioural part of the meta-model was initially developed, but could not be presented in the framework of this paper. Also a case study, that explains and elaborates the meta-model on a more concrete level, was planned, but could not be included in this paper. We hope to include both of the missing aspects into our next paper.

We are elaborating the vision using UML and patterns, and developing prototypes for the core of the solution. One of our work directions is to develop a UML Profile for the development subsystem architecture on the basis of the proposed model and following the MDA concept (Frankel, 2003).

REFERENCES

Alter, S., 2002, Work System Method for Understanding Information Systems and Information Systems Research, *Communications of the Assotiation for Information Systems* **9**(6).

Apshankar, K., Sadhwani, D., Samtani, G., Siddiqui, B., Clark, M., Fletcher, P., Hanson, J. J., Irani, R. M., Waterhouse, M., and Zhang, L. J., 2002, Web Services Business Strategies and Architectures, *Expert Press*.

Barker, R., 1991, *Oracle CASE Method: Tasks and Deliverables*.

Bultje, R., van Wijk, J., 1998, Taxonomy of Virtual Organisations, based on definitions, characteristics and typology, in: *VoNet: The Newsletter* **2**(3):9–12; (March 1, 2001), http://www.virtual-organization.net.

Checkland, P., and Scholes, J., 1990, *Soft Systems Methodology in Action*, Wiley, London.

da Cunha, P. R., and de Figueiredo, A. D., 2001, Information Systems Development as Flowing Wholeness, in: *Proceedings of the IFIP TC8/WG8.2 Working Conference on Realining Research and Practice in Information Systems Development*, Boise, USA, pp. 29–48.

Davis, G. B., 2000, Information systems conceptual foundations: looking backward and forward, in: *Organizational and social perspectives on information technology: Proceedings of the IFIP TC8 WG8.2 International Working Conference*, Aalborg, Denmark, pp. 61–82.

Frankel, D. S., 2003, *Model Driven Architecture: Applying MDA to Enterprise Computing*, Wiley Publishing, Indianapolis, USA.

Jacobson, I., Booch, G., and Rumbaugh, J., 1999, *The Unified Software Development Process*.

Kirikova, M., 1998, Completeness and Levels of Business Models, *Proceedings of the Third International Baltic Workshop of Databases and Information Systems*, Riga, Latvija, pp. 39–51.

Larman, C., 2001, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, Prentice Hall PTR, New Jersey.

Lyytinen, K., Rose, G., and Welke, R., 1998, The Brave New World of development in the internetwork computing architecture (InterNCA): or how distributed computing platforms will change systems development, *Information Systems Journal* 8(4):241–253.

Marshall, C., 1999, *Enterprise Modelling with UML – Designing Successful Software through Business Analysis*.

Mikli, T., 1997, *Introduction to Information Systems*, Tallinn, Estonia.

Mikli, T., and Rava, K., 1998, System Work in System Development, in: *Organisation Structures, Management, Simulation of Business Sectors and Systems: Proceedings of the IFORS Special Conference* (SPC8), Kaunas, Lithuania, pp. 213–217.

Powazek D. M., 2002, *Design for Community: The Art of Connecting Real People in Virtual Places*, New Riders Publishing, Indianapolis, USA.

Preece, J., 2000, *Online Communities: Designing Usability, Supporting Sociability*, John Wiley & Sons, Chichester, England.

Reithofer, W., and Naeger, G., 1997, Bottom-up planning approaches in enterprise modelling – the need and state of the art, *Computers in Industry* (33):223–235.

Roost, M., 1996, Information Systems Development on the basis of the Role-Model, in: *Databases and Information Systems: Proceedings of the Second International Baltic Workshop*, Tallinn, pp. 37–47.

Roost M., Kuusik, R., and Elmik, L., 1998, A Subject-Centred Framework for Information System and Organisation Development: Analysis and (Re)Design, in: *Organisation Structures, Management, Simulation of Business Sectors and Systems: Proceedings of the IFORS Special Conference* (SPC8), Kaunas, Lithuania, pp. 222–228.

Roost M., Kuusik, R., Elmik, L., Veskioja, T., and Rava, K., 1999, The Role-Model: An OO Framework for Information Systems Self-Development, *Poster Abstracts of Conf. UML'99*, Fort Collins, USA, pp. 7–8.

Roost, M., Kuusik, R., Veskioja, T., 2001, A Role-Based Framework for Information System Self-Development, in: *Proceedings of the IFIP TC8/WG8.2 Working Conference on Realining Research and Practice in Information Systems Development*, Boise, USA, pp. 95–105.

Roost, M., 2004, A Model of Self-Development of Information Systems, in: *The Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2004), Volume I, Information Systems, Technologies and Applications*, Orlando, USA, July 18–21, pp. 126–131.

Russo, N. L., 2000, Expanding the horizons of information systems development, in: *Organizational and social perspectives on information technology: Proceedings of the IFIP TC8 WG8.2 International Working Conference*, Aalborg, Denmark, pp. 103–112.

Scheer, A. W., 2000, *ARIS – Business Process Frameworks*, Springer Verlag.

Strausak, N., 1998, Resumee of VoTalk, Organizational Virtualness, P. Sieber and J. Griese, *Proceedings of the VoNet–Workshop*, April 27–28. Simowa Verlag Bern, 1998; (11.05.2001), http://virtual-organization.net/news/proc-98.pdf.

Wenger, E., McDermott, R., and Snyder, W. M., 2002, *A Guide to Manage Knowledge: Cultivating Communities of Practice*, Harvard Business School Press, Boston, USA.

SURVEY OF REQUIREMENTS ENGINEERING PRACTICE IN LITHUANIAN SOFTWARE DEVELOPMENT COMPANIES

Raimundas Matulevičius*

1. INTRODUCTION

The requirements engineering (RE) process is described as a sequence of actions, during which the list of requirements for a new software system is elicited, analyzed, validated and documented into a formal, complete and agreed requirements specification.[1] There are various RE approaches and methods, but the RE process still remains immature.[2] Introduction of new RE methods leads to the RE process improvements. However, the gap between academic and industrial practice exists, and academic suggestions seldom find applicability in the software development organizations. An empirical research is the first step to find out how the industry actually works, and to gather the knowledge about the improvement possibilities for the RE process. This paper explores the state of the RE practice in Lithuanian software development companies. The research addresses the following questions:

1) What is the perceived importance of different activities, executed during the RE process?
2) How much of the total project development time does the RE take?
3) What software development methods, techniques and tools are used for the RE?

The paper is structured as follows. Section 2 analyzes the related work. Section 3 describes the research method. Section 4 analyzes the findings and provides the discussion over the results. The survey results indicate the important trends to describe the RE process in Lithuanian software development organizations. Section 5 discusses the potential validity threats. Finally, Section 6 provides conclusions and future work.

* Dept. of Computer and Information Science, Norwegian Univ. of Science and Technology, Sem Sælands vei 7-9, NO-7491 Trondheim, Norway, raimunda@idi.ntnu.no.

2. RELATED WORK

Empirical surveys of the RE practice could be divided into several groups: analysis, which concern only RE issues,[3–6] works which analyze RE for the particular application domain,[7] and studies, which include geographical coverage.[8–10]

The field study of the software design process for large system development[4] concludes that knowledge integration, change facilitation, and broad communication are the key issues impacting the project success. Elsewhere,[6] the best practice of RE is identified as involving customers and users throughout RE, assigning skilled project managers and team members, providing specification template, developing complementary models together with prototypes, maintaining traceability, and using peer reviews, and scenarios to validate requirements. Further, the analysis of RE processes[5] suggests seven factors for RE process improvement. They are package consideration, managing the level of detail of functional process models, examining the current system, user participation, managing uncertainty, benefits of CASE tools, and project management capabilities. The survey results[3] show that the requirements capture and analysis are iterative. Allocation of the resources (time, efforts, cost, and people) depends on the project type, the team members' and users' attitude, and the management.

An RE for commercial off-the-shelf software packages is analysed in Swedish organizations.[7] The authors suggest to improve the RE process by solving the communication gaps between marketing and development, and the problems of balancing the influence between marketing and requirements development on requirements decisions.

Ten organizations in the United Stated are interviewed in order to find out how they define, interpret, analyze, and use requirements.[9] The results conclude that organizational solutions are preferred over technological ones and general-purpose tools are more common than RE tools. A survey[10] of small and medium-size enterprises in Finland indicates that the key software development needs are the development of RE process adaptation, RE process improvement and automation of RE practices.

The recent study[8] in 22 Norwegian software development companies, reports that RE is usually performed using office and CASE tools. The main reasons of not using targeted RE tools basically include the economical factors, like high costs, low returns on investment and non-awareness of the existing RE tools.

This work is an empirical investigation of the RE processes in Lithuanian software development companies. Lithuania is a country rapidly increasing the use of the information technology,[11] which in turn generates interesting RE trends. This analysis concludes with the challenging issues to improve the RE process in the software developing companies.

3. RESEARCH METHOD

The main steps of the research method are shown in Figure 1. The exploratory research formulates research questions, constructs, and validates the questionnaire. The research questions are formulated after the analytical study of the related work. Results of a focus group discussion help to construct the questionnaire, which is validated during in depth interviews.

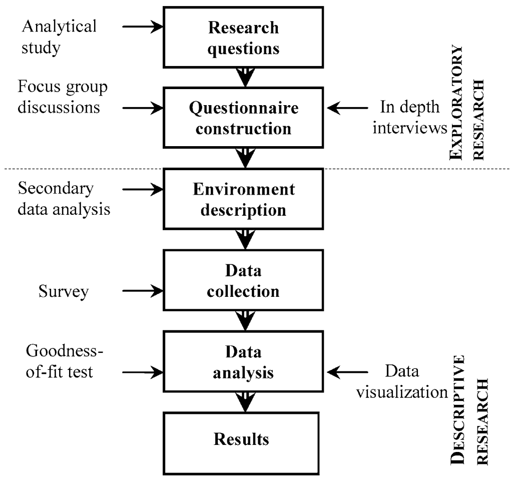


Figure 1. Research method.

The descriptive research collects the data about the RE from different sources, analyzes the collected information and provides the results. Environment and participant description are gathered from the secondary data sources (e.g. previous survey[11]). The web-based questionnaire helps to perform the survey of the RE process in the software development companies and the results are considered using data tabulation and statistical tests.

3.1. Research Questions

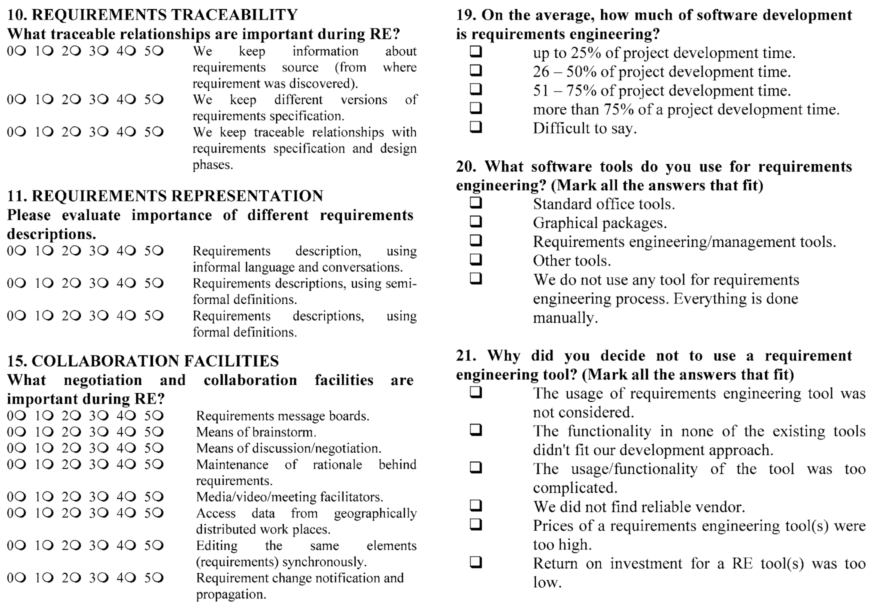
The RE process is characterized as a network of activities, like elicitation, documentation, analysis, validation and negotiation.[12–15] The activities are customized by choosing the appropriate techniques to specific applications. This leads to the following research question:

Question 1. *What is the perceived importance of different activities, executed during RE process?*

In order to answer the question the structuring evaluation framework[16] is adapted. The framework describes the RE process along the three orthogonal dimensions[1] – representation, agreement and specification. Framework features correspond to the requirements categories, which are followed with the lists of activities.[17–18] While this framework was originally developed to evaluate RE tools, it mainly decomposes RE into various activities, and can thus be used also to evaluate the RE process even if it is not supported by the RE tools. The framework is adapted to the questionnaire (Figure 2).

Depending on the software development approach,[15] the RE phase takes a certain amount of time. The following research question is:

Question 2. *How much of the total project development time does the RE take?*



10. REQUIREMENTS TRACEABILITY
What traceable relationships are important during RE?

0O 1O 2O 3O 4O 5O	We keep information about requirements source (from where requirement was discovered).
0O 1O 2O 3O 4O 5O	We keep different versions of requirements specification.
0O 1O 2O 3O 4O 5O	We keep traceable relationships with requirements specification and design phases.

11. REQUIREMENTS REPRESENTATION
Please evaluate importance of different requirements descriptions.

0O 1O 2O 3O 4O 5O	Requirements description, using informal language and conversations.
0O 1O 2O 3O 4O 5O	Requirements descriptions, using semi-formal definitions.
0O 1O 2O 3O 4O 5O	Requirements descriptions, using formal definitions.

15. COLLABORATION FACILITIES
What negotiation and collaboration facilities are important during RE?

0O 1O 2O 3O 4O 5O	Requirements message boards.
0O 1O 2O 3O 4O 5O	Means of brainstorm.
0O 1O 2O 3O 4O 5O	Means of discussion/negotiation.
0O 1O 2O 3O 4O 5O	Maintenance of rationale behind requirements.
0O 1O 2O 3O 4O 5O	Media/video/meeting facilitators.
0O 1O 2O 3O 4O 5O	Access data from geographically distributed work places.
0O 1O 2O 3O 4O 5O	Editing the same elements (requirements) synchronously.
0O 1O 2O 3O 4O 5O	Requirement change notification and propagation.

19. On the average, how much of software development is requirements engineering?

☐ up to 25% of project development time.
☐ 26 – 50% of project development time.
☐ 51 – 75% of project development time.
☐ more than 75% of a project development time.
☐ Difficult to say.

20. What software tools do you use for requirements engineering? (Mark all the answers that fit)

☐ Standard office tools.
☐ Graphical packages.
☐ Requirements engineering/management tools.
☐ Other tools.
☐ We do not use any tool for requirements engineering process. Everything is done manually.

21. Why did you decide not to use a requirement engineering tool? (Mark all the answers that fit)

☐ The usage of requirements engineering tool was not considered.
☐ The functionality in none of the existing tools didn't fit our development approach.
☐ The usage/functionality of the tool was too complicated.
☐ We did not find reliable vendor.
☐ Prices of a requirements engineering tool(s) were too high.
☐ Return on investment for a RE tool(s) was too low.

Figure 2. A fragment of the questionnaire.

The automated support for the software development processes is recognized in the literature.[2, 19–20] RE tools support elicitation (brainstorming and interviewing tools, tools for highlighting and extracting requirements), analysis and documentation (tools for classifying requirements and keeping requirements traceability), and validation (tools for simulating requirements models). The third research question is:

Question 3. *Which software tools are used to support the RE process?*

3.2. Questionnaire Construction

An initial questionnaire was design after the analytical study of the related work and focus group discussion. The questionnaire was validated during the in-depth interviewing (face-to-face meetings and e-mail correspondences) of the experts, who have experience in software development. Finally, the questionnaire was sent out to three test respondents to software companies. The questionnaire validation discovered the undefined and not clear questions, and it helped to remove choice-answer ambiguities.

The information about the respondent and his organization is asked, first. The information includes the organizational profile, the product role, employee number in organization, and respondent's position. The most difficult problem during questioning is to motive the respondents to answer an unsolicited survey.[21–23] People are motivated if they can see that the study results are likely to be useful to them. The questionnaire is accompanied with several key pieces of information: what the study purpose is, why each individual's participation is important, how confidentiality will be preserved.

The RE definition[24] keeps the unified understanding. The evaluation scheme from '0' to '5' is introduced ('0' – not important, '5' – very important RE activities). The respondents have to evaluate the importance of stakeholders, requirements discovery, traceability, representation languages, grouping and attributing, negotiation and collaboration facilities, reuse of requirements, and documentation. Next, the survey questions consider the average time of RE phase, and the automated RE support. Due to the space limits Figure 2 displays only the sample of the questions.

3.3. Environment Description

In 2000 Lithuanian information technology (IT) market was identified as 67% hardware, 16% software and 17% IT services.[11] But lately an increasing demand for the more sophisticated software has speeded up the growth rates of software sales (20% growth during 2000–2001) and IT service market are forecast to become the most rapidly growing segment in the next few years.

According to Lithuanian Statistics Department and Infobalt Association the number of IT companies is assessed to be around 250 in 2001. The majority of them are small, employing up to 10–20 people. But the market is dominated by a number of medium and large companies. The most frequent business activities include customer specific tailored-made software, local area networks, and information systems. The service lines consist of data warehousing, accounting, financial, and business process management.

3.4. Data Collection

A self-administered data collection method[22] was used. First, the respondents were contacted by phone. The purpose of the study was explained to an individual participant. To motive participants, we have promised to send the study results. If the participant agreed to take part in the study, the questionnaire URL was sent to him or her by e-mail.

3.5. Participants

28 (out of 75, response rate 37%) companies agreed to participate in the survey. This included 12 (42,9%) small (up to 20 employees), 8 (28,6%) average (21–100 employees), 5 (17,9%) large (more than 100 employees) companies, and 3 (10,7%) non-software development companies, which have their own software development departments.

The responses came from companies representing a variety of domains: data warehousing (15,6%), accounting (33,3%), content (13,3%), and business process management (15,6%), specific domain applications (22,2%). At the organization profile level 50% of respondents indicated themselves as consulting companies, 75% – software development companies, 32,14% – software users (employing software). At the project development level 67,86% of companies are producing off-the-shelf software, 82,14% develop products for external customers, and 35,71% are developing software for internal organizational use.

Finally, there are responds from 9 project leaders, 5 marketing executives, 6 system developers, 4 managing directors, 2 communication experts and 2 user support stuff.

3.6. Data Analysis

Three techniques to analyze the collected data are used: data tabulation,[21] data visualization[25] and chi-square goodness-of-fit test.[21] Data tabulation entails counting the number of cases that fall into separate categories. Data visualization helps to identify the leading results of the empirical data. Chi-square goodness-of-fit test determines whether a set of observed frequencies departs significantly from a set of expected frequencies. Concretely, the chi-square test helps to examine the importance of the RE methods and techniques. The calculations are provided in the appendix.

4. RESULTS AND DISCUSSION

The section discusses the survey results, which are provided in the appendix. The importance of the RE process activities is analyzed along three dimensions.[1] Next, the RE share in the software development and the automated RE support are considered.

4.1. Representation Activities

The requirements representation deals with representation forms and relationships between them. The results (Table 1) show a strong importance of informal and semi-formal specification languages. The most important representation language is natural language (Table 2). There is an interesting difference between semi-formal and formal languages: semi-formal languages are used for requirements representation, but there is no agreed commonly used language. Formal languages are used quite rarely.

Table 1. Importance of requirements representation

Requirements representation	Observed evaluations						M	V	χ^2	Importance
	5	4	3	2	1	0				
Requirements description, using informal language and conversations.	9	10	6	2	0	1	3,96	0,88	3,25	**High**
Requirements descriptions, using semi-formal definitions.	9	12	4	3	0	0	3,96	0,92	5,25	**High**
Requirements descriptions, using formal definitions.	6	7	9	2	3	1	3,41	1,56	8,40	**Low**

Table 2. Importance of requirements representation languages and techniques

Requirements representation languages	Observed evaluations						M	V	χ^2	Importance
	5	4	3	2	1	0				
Natural language.	16	6	3	1	0	2	4,42	0,73	4,58	**High**
State charts.	5	9	3	2	2	7	3,62	1,55	40,30	Not imp.
DFD.	4	10	3	2	2	7	3,75	1,25	41,51	Not imp.
Use Case diagrams.	8	5	5	2	1	7	3,81	1,46	38,80	Not imp.
Use Case templates.	2	14	2	1	2	7	3,62	1,15	46,71	Not imp.
UML.	6	5	5	2	2	8	3,55	1,73	53,38	Not imp.
ER diagrams.	3	10	3	1	2	9	3,58	1,37	70,50	Not imp.
OMT.	1	4	4	4	3	12	2,75	1,53	138,08	Not imp.
Action semantic.	3	2	2	5	3	13	2,80	2,17	164,64	Not imp.
Algebraic specifications.	0	2	5	3	5	13	2,19	1,23	186,61	Not imp.
Z-schemas.	0	2	2	3	8	13	1,87	1,27	210,74	Not imp.
Aggregate approach.	0	1	5	3	6	13	2,07	1,07	186,61	Not imp.

Table 3. Importance of requirements traceability techniques

ACTIVITIES, METHODS, AND TECHNIQUES OF REQUIREMENTS TRACEABILITY	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
We keep different versions of requirements specification.	14	9	4	1	0	0	3,96	1,07	2,96	**High**
We keep information about requirements source.	14	9	3	2	0	0	4,25	0,86	2,71	**High**
We keep traceable relationships with requirements specification and design.	8	10	3	6	1	0	3,64	1,50	10,3	**Low**

Table 4. Importance of requirements discovery methods

ACTIVITIES, METHODS, AND TECHNIQUES OF REQUIREMENTS DISCOVERY.	OBSERVED EVALUATIONS						M	V	χ^2	IMPORTANCE
	5	4	3	2	1	0				
Analyzing existing (similar) software.	10	11	6	1	0	0	4,07	0,74	4,96	**High**
Performance of surveys (market, stakeholders).	6	11	5	4	2	0	3,54	1,44	8,38	**High**
Meetings and conversations with (potential) software stakeholders.	26	2	0	0	0	0	4,93	0,07	10,2	**High**
Reuse of domain specific requirements from predefined repositories.	6	6	8	5	2	1	3,33	1,54	9,90	**Low**
Analyzing competitive documentations, which were provided by software vendors.	6	8	6	5	3	0	3,32	3,33	11,4	Not imp.

The use of natural language is supported in[7–8, 10, 19]. Semi-formal techniques are ER diagrams,[6, 8–9, 26] state charts,[6–7] data flow diagrams,[8–9, 26] UML,[7] and Use Case templates.[8] Furthermore,[6,9] the requirements are described using knowledge and quality function deployment matrices. Formal languages are not used for RE,[9] but Lithuanian practice shows a weak support for formal languages, but none of them were identified.

The results (Table 3) suggest strong importance of relationships between requirements definitions, the requirements source, and through continues requirements development. Several companies[6, 9] maintain a traceability matrix to track a requirement from its origin through its specification to its implementation.

4.2. Agreement Activities

The requirements agreement deals with the agreement about requirements model. The common practice to reach agreement about requirements model is face-to-face negotiation and discussions (Table 10). The communication makes sure that requirements are interpreted properly.[5, 6] However, the communication gaps among stakeholders are indicated.[4, 7] The knowledge sharing and change facilitation is argued.[3, 4] The rationale in customer-specific projects is not worth the effort,[9] but in market-driven projects the rationale for decisions and assumptions should be recorded. In Lithuanian companies the rationale is kept and it helps to solve the emerging conflicts about the requirements.

The most important elicitation activities are meetings, surveys, and analysis of similar systems (Table 4). The same activities are considered in[3, 5, 6, 9]. Elsewhere,[6, 7] the requirements databases and document analysis as the requirements source are argued.

Sorting and viewpoint definitions (Table 6) involve different stakeholders: project managers, software developers, domain experts, marketing personnel and end users (Table 5). Experienced project manager should be able to use the RE tools, and have knowledge of the system development process.[3, 5] Software developers play an important role[7]

Table 5. Importance of importance of stakeholders

Stakeholders	Observed evaluations						M	V	χ^2	Impor-tance
	5	4	3	2	1	0				
Project managers.	10	7	6	2	2	1	3,78	1,56	2,46	**High**
Software developers.	10	8	5	2	2	1	3,86	0,87	1,58	**High**
Marketing personnel.	19	6	2	1	0	0	4,54	0,63	3,06	**High**
Domain experts.	18	6	2	2	0	0	4,43	0,85	4,00	**High**
End-users.	8	14	5	1	0	0	4,04	0,63	8,58	**High**
Training and user support staff.	7	6	7	6	1	1	3,44	1,49	10,70	**Low**
Communication experts.	5	8	3	5	5	2	3,12	2,11	24,91	Not imp.
Indirect stakeholders.	2	8	12	4	2	0	3,14	1,02	20,31	Not imp.
Technical authors.	4	6	9	7	2	0	3,11	1,36	22,30	Not imp.
User interface designer/ usability expert.	2	8	7	4	5	2	2,92	1,59	26,31	Not imp.
Project/product sponsors.	7	4	4	6	3	4	3,25	2,11	24,88	Not imp.

Table 6. Importance of requirements attributes

Activities, methods, and techniques related to requirements attributes	Observed evaluations						M	V	χ^2	Impor-tance
	5	4	3	2	1	0				
Sorting according to attributes.	8	9	5	1	1	4	3,92	3,82	9,38	**Low**
Definition of views according to attributes.	7	10	4	2	1	4	3,83	1,19	10,21	**Low**
Definition of requirements attributes.	9	8	6	0	1	4	4,00	1,04	10,40	**Low**
Filtering according to attributes.	7	7	6	1	1	6	1,12	1,20	26,34	Not imp.

Table 7. Importance of requirements groups

Requirements groups	Observed evaluations						M	V	χ^2	Impor-tance
	5	4	3	2	1	0				
Functional requirements.	16	6	3	1	1	1	4,30	1,14	2,58	**High**
End-users requirements.	11	10	3	1	0	3	4,24	0,69	6,33	**High**
Architectural requirements.	6	14	4	0	1	3	3,96	0,79	8,27	**Low**
Adaptability/ reliability/ security/ usability requirements.	9	10	3	0	1	5	4,13	0,94	18,50	Not. imp.
Domain/ user/ organizational requirements.	10	8	3	0	2	5	4.04	1,41	19,58	Not imp.
Developers' requirements.	5	10	6	2	1	4	3,67	1,10	11,78	Not imp.
Non-functional requirements.	6	6	6	3	3	4	3,38	1,81	18,00	Not imp.
Financial requirements.	5	8	5	5	0	5	3,57	1,17	24,11	Not imp.

during requirements analysis and validation. Users should participate in the project from the very beginning.[3, 5, 6, 9] Domain experts are the most regarded participants.[3, 4–6, 9] They understand a domain as well as the users, and domain experts can often correct the users.

Most of Lithuanian software development companies are departments of some international organizations. The central offices consider software marketing and support requirements. In the survey respondents indicated (Table 7), that they deal with functional and architectural requirements and do not analyze non-functional ones.

4.3. Specification Activities

The requirements specification is written according to standards and guidelines used. The results (Table 9) indicate that companies prefer organizational standards to standards agreed by international communities.[8] To support the final specification, a big amount of documentations is needed (Table 11). However, the creation of requirements documents is not self-evident.[9, 10] Projects that are developing products for a potential market tend not to document requirements in as much detail, if at all, and seldom follow any standards other than internal corporate or division-wide guidelines.

Table 8. Importance of requirements reuse techniques

Activities, methods, and techniques for requirements reuse	Observed evaluations						M	V	χ^2	Impor-tance
	5	4	3	2	1	0				
Selection and extraction of domain specific requirements from other projects.	8	6	8	1	3	2	3,64	1,91	7,38	**Low**
Maintenance of data repository.	10	3	8	1	3	3	3,58	1,69	13,68	Not imp.
Reverse engineering of requirements from code/ design.	4	5	8	4	5	2	2,96	1,80	26,18	Not imp.

Table 9. Importance of requirements specification standards

Activities, methods, and techniques for final requirements specification	Observed evaluations						M	V	χ^2	Impor-tance
	5	4	3	2	1	0				
Standard for requirements specification, defined by your organization.	16	8	1	1	1	1	3,45	1,97	0,89	**High**
Standards requirements specification (f.e: IEEE std 830-1998).	7	4	6	2	3	6	4,37	1.01	34,04	Not imp.
Requirements specification, agreed between all stakeholders, defined in formal language.	13	4	3	4	0	4	4,08	1,38	14,33	Not imp.

Table 10. Importance of negotiation and collaboration facilities

Activities, methods, and techniques for negotiation and collaboration	Observed evaluations						M	V	χ^2	Impor-tance
	5	4	3	2	1	0				
Means of brainstorm.	8	11	4	1	2	2	3,83	1,34	4,96	**High**
Means of discussion/negotiation.	16	7	3	1	0	1	4,41	0,71	3,21	**High**
Maintenance of rationale behind requirements.	7	12	5	0	1	3	3,96	0,87	10,33	**High**
Requirement change notification and propagation.	9	9	4	0	2	4	3,96	1,35	11,77	Not imp.
Requirements message boards.	6	10	3	3	2	4	3,63	1,55	13,01	Not imp.
Media/video/meeting facilitators.	3	10	7	5	1	2	3,35	1,12	11,31	Not imp.
Access data from geographically distributed work places.	7	7	3	5	2	4	3,50	1,70	16,64	Not imp.
Editing the same elements (requirements) synchronously.	5	7	6	2	3	5	3,96	1,35	23,14	Not imp.

Table 11. Importance of requirements documentation

Activities, methods, and techniques for requirements documentation	Observed evaluations						M	V	χ^2	Impor-tance
	5	4	3	2	1	0				
Reports and documentation, which help to understand requirements.	11	9	5	1	1	1	4,04	1,11	0,96	**High**
Reports and documentation of representations.	13	10	4	0	0	1	4,33	0,54	3,58	**High**
Reports and documentation about requirements agreement.	14	9	3	1	1	0	4,21	1,06	2,21	**High**

Only selection and extraction of domain specific requirements from other project is indicated with weak importance (Table 8). The reuse activities are performed manually,[9] in the best case using "copy-paste" typing operations.

4.4. RE Share

The analysis shows that the time spent for RE is 25–50% of the whole development (Figure 3). The RE is a new and not mature activity.[2, 13] The survey respondents could consider and include not only the process of new software development, but also the software support, which takes part after software release. The result corresponds to other findings, where the average amount of RE time equals to 38.6 % of total project duration,[6] and which argues, that companies invest heavily in terms of time.[3]

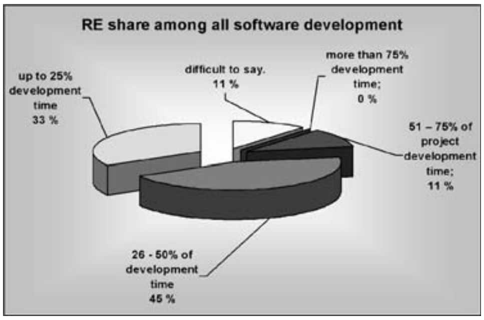


Figure 3. RE share among all software development.

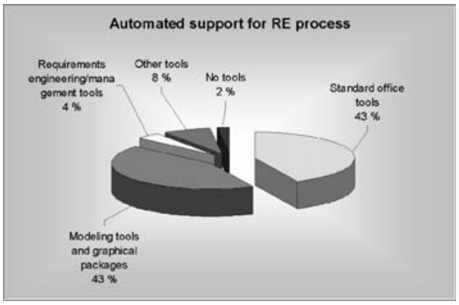


Figure 4. Automated support for RE process in Lithuanian companies.

4.5. Automated Support for RE

Automated support for RE process is not adequate. The mainstream of RE practice relies on word processors (43%), modelling tools (43%) (Figure 4) rather than the RE tools (4%). Other surveys[6–10] also reports the lack of the automated support. The most commonly used tools are text editors and spreadsheets[3, 8–10] (e.g. Norwegian practice in Figure 5), web sites, accessible to all stakeholders,[6] and Visio.[26] Some projects combine text editors and database management systems for traceability, but such integrations can only be used if configuration management policies are enforced.[9]

Reasons for not using the RE tools include financial causes, like high RE tool price (31%), low return on investment (21%), companies do not consider the possibilities to adapt RE tools to their organizational needs (38%). The Norwegian practice[8] reports non-awareness of the existing RE tools and the large tool costs as the main reasons for not using them. A lack of well defined RE process and a lack of team training in the selected tools caused the non-sufficient support for the RE activities.[6] In order to adopt the tool, an infrastructure must be set to support a tool and a company must be willing to invest in putting such an infrastructure in the environment.[5] This includes personnel training,[7] tool support groups,[26] funding for the tool implementation.[5] However, the management of such companies usually expects the unrealistic expectations, as for example immediate pay-off.

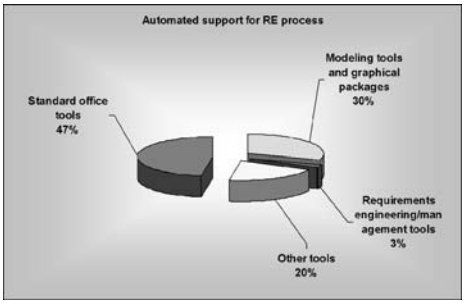


Figure 5. Automated support for RE process in Norwegian companies.

5. THREATS TO VALIDITY

The following possible threats to the validity of this study have been identified:

- The research target is the geographical area, but not the domain. The study provides the general understanding of RE in software development companies.
- Impact of the company size and the software development life cycle model applied in the companies is not taken into account for the study results.
- The sample size of respondents is relatively low. But the threat is compensated by a relatively high response rate.
- Difficulty to find the right person. As RE could not be executed by one person, it is important to find the respondent who is able to fill in the questionnaire.
- Respondents had different educational experience, which influences the organizational knowledge and work style, and affects the survey results, because each individual interprets the RE activities according to his own experience.
- The usual threat with the self-administered questionnaire is to state the questions in a comprehensive way. RE is dealing with a number of concepts, which could be interpreted differently. To achieve a common understandability the definitions were provided.

Unsolicited questionnaire does not help to reason why respondents prefer one or another RE activity. The in-depth investigation could contribute with a close insight look into the RE practice. But, a web-based questionnaire was appropriate due to the geographical distance, as well as enabling an easy and relatively non-expensive data collection.

6. CONCLUSIONS AND FUTURE WORK

This paper analyzes the RE process in Lithuanian software development companies. Although several companies use the predefined RE process, but the survey confirms that the RE process is not mature.[2] The findings emphasize for the RE challenges:

- The analysis of non-functional requirements would stimulate the creativity of the software engineers. The common non-functional requirements could be reused in several projects and reduce the time and resource needs.

- Reuse of requirements specification would be effective, if requirements repositories are introduced. Repository tends to support storage of requirements metadata and traceability when relevant requirements are modified.
- Adaptation of RE tools for organizational needs could contribute towards RE process improvement. The market suggests RE tools, which allow integration with other modelling environments and support RE activities such as scope management, version and configuration control, quality assurance and quality control. The requirements specifications of different types, contents, and level of details could be automatically generated from the requirements repositories using appropriate selection criteria and templates.

The most effective software development cycles[19] are iterative, incremental, parallel, and timeboxed. Elsewhere,[27] the RE process is discovered not as a smooth and incremental evolution, but characterized by occasional "crisis" points where the requirements are reconceptualised, restructured, and simplified. The work findings suggest the investigation not only the importance of the individual RE activities, but also the interaction and relationships between them. This leads to the follow up questionnaire with more in-depth investigation in particular software development companies.

ACKNOWLEDGMENT

The author would like to thank Guttorm Sindre for the feedback concerning drafts of this paper and Rita Butkienė for the help during the survey process in Lithuania.

REFERENCES

1. K. Pohl, The three dimensions of requirements engineering: a framework and its applications, *Information systems* **19**(3), 243–258 (1994).
2. H. Kaindl, S. Brinkkemper, J. Bubenko, Jr., B. Farbey, S. J. Greenspan, C. L. Heitmeyer, J. C. S. do Prado Leite, N. R. Mead, J. Mylopoulos, and Siddiqi, Requirements engineering and technology transfer: obstacles, incentives, and improvement agenda, *Requirements Engineering* **7**, 113–123 (2002).
3. P. Chatzoglou, Factors affecting completion of the requirements capture stage of projects with different characteristics, *Information and Software Technology* **39**, 627–640 (1997).
4. B. Curtis, H. Krasner, and N. Iscoe, A field study of the software design process for large systems, *Communication of the ACM* **31**(11), 1268–1287 (1998).
5. K. El Emam and N. H. Madhavji, A Field Study of Requirements Engineering Practice in Information Systems Development, in: *RE'95. Second IEEE Int. Symposium on Requirements Engineering* (1995), pp. 68–80,
6. H. F. Hofmann and F. Lehner, Requirements Engineering as a Success Factor in Software Projects, *IEEE Software*, pp. 58–66 (2001).
7. L. Karlsson, A. G. Dahlstedt, J. Natt och Dag, B. Regnell, and A. Persson, Challenges in Market-Driven Requirements Engineering – an Industrial Interview Study, in: *REFSQ'02. Proceedings of the 8th International Workshop on Requirements Engineering – Foundation for Software Quality* (German).
8. S. Ekremsvik and E. M. Tiset, Kravspek-verktøy, marketstudiet, TDT4730 Information System Specification Report, IDI-NTNU, Trondheim, Norway, December, 2003.
9. M. Lubars, C. Potts, and C. Richter, A Review of the State of the Practice in Requirements Modelling, in: *RE'93. First IEEE Int Symposium in Requirements Engineering* (RE'93, 1993), pp. 2–14.
10. U. Nikula, J. Sajaniemi, and H. Kälviäinen, A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises, TBRC Research Report 1 (Telecom Business Research Center Lappeenranta, Lappeenranta University of Technology, 2000).

11. M. Nissinen, The Baltics as a Business Location for Information Technology and Electonics Industries, *VTT Research Notes 2169* (2002); http://www.infobalt.lt.
12. P. L. Ferdinandi, *A Requirements Pattern. Succeeding in the Internet Economy* (Addison-Wesley, 2002).
13. G. Kotonya and I. Sommerville, *Requirements Engineering: Process and Techniques* (Wiley, 1998).
14. D. Leffingwell and D. Widrig, *Managing Software Requirements. A Unified Approach* (Addison-Wesley, 2000).
15. P. Loucopoulos and V. Karakostas, *System Requirements Engineering* (McGraw-Hill, 1995).
16. R. Matulevičius, Validating an Evaluation Framework for Requirement Engineering Tools, in: *Information Modeling Methods and Methodologies (Adv. Topics of Database Research)*, edited by J. Krogstie, T. Halpin, and K. Siau (Idea Group Publishing, 2004), pp. 148–174.
17. J. Krogstie and H. Jørgensen, Quality of Interactive Models, in: *IWCMQ'02, Proceedings of the 1st International Workshop on Conceptual Modeling Quality* (Finland, 2002), pp. 115–126.
18. M. Lang and J. Duggan, A tool to support collaborative software requirements management, *Requirement Engineering* **6**, 161–172 (2001).
19. D. Firesmith, Modern requirement specification, *Journal of Object Technology* **2**(1), 53–64.
20. W. Harrison, H. Ossher, and P. Tarr, Software Engineering Tools and Environments: a Roadmap, in: *The Future of Software Engineering*, edited by A. Finkelstein (ACM Press, 2000).
21. G. A. Churchill, Jr., *Basic Marketing Research* (The Dryden Press, 2001).
22. D. A. Dillman, *Mail and Internet Surveys* (John Wiley & Sons Inc., 2000).
23. B. Kitchenham and S. L. Pfleeger, Principles of survey research. Part 4: questionnaire evaluation, *Software Engineering Notes* **27**(3), 20–23 (2002).
24. P. Zave, Classification of research efforts in requirements engineering, *ACM Computing Surveys* **29**(4), 315–321 (1997).
25. B. Regnell, M. Host, J. Natt och Dag, and T. Hjelm, Visualization of Agreement and Satisfaction in Distributed Prioritization of Market Requirements, in: *REFSQ2000. Proceedings of the 6th International Workshop on Requirements Engineering – Foundation for Software Quality* (Sweden, 2000).
26. I. Davies, P. Green, and M. Rosemann, Modelling in the Australian Practice – Preliminary Insights, Centre for Information Technology Innovation, Queensland University of Technology.
27. L. Nguyen and P. A. Swatman, Managing the requirements engineering process, *Requirements Engineering* **8**, 55–68 (2003).

APPENDIX

The appendix presents the mean (M), variance (V), and chi-square (χ^2) calculations for the RE activities. χ^2 is calculated like

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i},$$

where O is the observed pattern, E is the expected pattern. The table value of χ^2 for five degree of freedom and $\alpha = 0.05$ is 11.07. Two importance levels – high and low - are selected according to the E mean threshold, which is equal to 3.79. High importance is if E equals to {18, 6, 1, 1, 1, 1} and {12, 8, 4, 2, 1, 1}. Low importance is if the E equals to {9, 10, 6, 1, 1, 1} and {10, 9, 5, 2, 1, 1}. If the calculation is higher than χ^2 value (with all E), then the activity is considered not important.

FUNCTIONALITY OF INFORMATION SYSTEMS SPECIFICATION LANGUAGE: CONCEPT, EVALUATION METHODOLOGY, AND EVALUATION PROBLEMS

Albertas Caplinskas and Jelena Gasperovic*

1. INTRODUCTION

The ISO/IEC 9126 standard (ISO/IEC 9126, 1991) distinguishes internal quality and quality in use. It defines internal quality as "the totality of attributes of a product that determines its ability to satisfy stated and implied needs when used under specified conditions" and quality in use as "the extent to which a product used by specified users meets their needs to achieve specified goals with effectiveness, productivity and satisfaction in specified context of use." In other words, internal quality is defined without any particular context; quality in use depends on the context. In (Caplinskas et al., 2002) and (Caplinskas and Gasperovic, 2004) we have investigated the problem of evaluation of quality of IS specification languages, proposed the evaluation procedure for quality in use and defined the taxonomy of quality characteristics for internal quality. Following the philosophy of ISO/IEC 9126, we argued that internal quality of a specification language could be described by four groups of quality characteristics: functionality, reliability, usability, and efficiency. However, these characteristics, when applicable to specification languages, should be understood in a quietly different way. The main purpose of this paper is to analyse internal structure of functionality of IS specification language and discuss issues of operationalisation of sub-characteristics of this characteristic. The paper discusses internal structure of this characteristic and its semantic and ontological aspects.

The rest of the paper is organised as follows. Section 2 surveys related works. Section 3 analyses the concept of functionality. Section 4 shortly discusses the problems of operationalisation. Finally, Section 5 concludes the paper.

* Institute of Mathematics and Informatics, Akademijos str. 4, LT-08663, Vilnius, Lithuania, alcapl@ktl.mii.lt, j.gasperovic@algoritmusistemos.lt.

2. RELATED WORKS

Functionality (from Latin *functio* meaning "to perform") means the degree to which the designed product will perform to meet its intended purpose. In the case of specification languages, functionality is understood as the set of features necessary to describe requirements of a future system. Despite the fact that functionality is one of the most important characteristics of internal quality of a specification language, only separate aspects of functionality, mostly expressiveness and ontological completeness, have been discussed in scientific literature.

A language is said to be more expressive than another one if it can express more concepts than the other. The concept of expressiveness has been studied formally already in logic (Kleene, 1952; Troelstra, 1973). The main result is that the additional symbols of core logic can be eliminated (i.e. expressed) if there exists a translation from extended logic to its core that satisfies some given conditions.

The concept of expressiveness has been extensively studied also in the realm of programming languages. The earliest works (e.g. (Chandra and Manna, 1975)) in this field have been based on comparative schematology. In this context, some authors consider such concept as computable functions and define maximally expressive class of languages as those that are equivalent to Turing machine. However, as it is pointed out in (Felleisen, 1990), "Comparing the set of computable functions that a language can represent is useless, because the languages in question are usually universal; other measures do not exist." Other authors (Reynolds, 1981; Steele et al., 1976) studied expressiveness of imperative programming languages modelling common programming constructs in terms of a simple applicative language based on a lambda calculus. They demonstrated that a number of programming constructs can be modelled locally, without restructuring the whole program, and analysed to which extent a language can support the organisation of a problem. Steele and Sussman (1976) pointed out "The emphasis not should be on eliminating "bad" language constructs, but on discovering or inventing helpful ones." Further, Felleisen (1990) suggested the following analogy between formal systems and programming languages: the set of phrases of a programming language corresponds to the expressions of a formal system, the set of programs corresponds to the set of well-defined formulae, and the set of terminating programs corresponds to the set of theorems. On the basis of this analogy and the works (Kleene, 1952; Troelstra, 1973) he made an attempt to develop a formal theory of expressiveness for programming language. In this approach a common language universe U that comprises all constructs of interest is constructed. A language L is less expressive than language L_1 if L_1 can express all the constructs from U, which L can express. Felleisen concluded that programs in less expressive languages exhibit repeated occurrences of programming patterns and suggested that such programming style is harmful.

Ontological completeness is the ability of a specification language and associated reasoning system to represent all phenomena of interest in the domain of discourse (Wand and Weber, 1993) or, in other words, the ability to describe social reality at a certain level of granularity (Colomb and Weber, 1998). Ontological completeness of a number of specification languages (mostly diagrammatic languages) has been analysed in (Colomb and Weber, 1998; Rosemann and Green, 2002; Wand and Weber, 1993; Wand and Weber, 1990). As a theoretical basis to evaluate ontological completeness in this approach so called Bunge-

Wand-Weber (BWW) models have been used. These models are based on the ontology proposed by the philosopher Mario Bunge. In other words, ontological completeness of the language in question is evaluated with regards to Bunge's ontology. So, BWW play the role of universe of ontological primitives. In this sense BWW approach is similar to Felleisen's approach. A language L is less expressive than language L_1 if L_1 can express all the ontological BWW primitives, which L can express. The shortcoming of both approaches is that they can be accepted only in the case when one shares objectivistic point of view that the proposed universes are universal indeed, that is represent everything that may be important for the user of programming or specification language, and those representations are language neutral and independent of any user's interest.

Another similar approach to evaluate ontological completeness has been proposed in (Milton et al., 1998). In this work, it is proposed to use Chisholm's ontology instead of Bunge's ontology and to evaluate ontological completeness not from the viewpoint of language constructs but from the point of its ability to specify a variety of situations.

Sølvberg and co-authors (Krogstie and Sølvberg, 1999; Lindland et al., 1994), see this problem in a slightly different way. Discussing wider issues of appropriateness (Krogstie, 2001; Krogstie, 2002; Krogstie, 2003), they analyse also the concept of expressive power that differs from the discussed concept of ontological completeness in the measure that could be used not only to evaluate is any statement in the domain expressible in the language, but also to evaluate whether it is impossible to express in the language any statement, which is not in the domain. In (Krogstie and Sølvberg, 1999; Lindland et al., 1994; Krogstie, 2001, Krogstie, 2002; Krogstie, 2003) the concept of expressiveness is defined using set-theoretical approach. This definition, in principle, can be considered as an abstract measure, however it is unclear how one can operationalise this measure.

3. ANALYSIS OF THE CONCEPT OF FUNCTIONALITY

3.1. Main Characteristics of Internal Quality

We consider the concept of functionality to be significantly wider than expressiveness or ontological completeness. In our attempt to define internal quality of specification language we follow the approach proposed by ISO/IEC 9126 standard (ISO/IEC 9126, 1991).

Although this standard addresses quality of software, its conceptual basis is significantly wider and can be applied to evaluate quality of languages, too (King and Maegaard, 1998; QStudio® for Java, 2003). Thus we define internal quality through four lower-level sub-characteristics: functionality, reliability, usability, and efficiency (Figure 1). We do not include maintainability and portability, because these sub-characteristics are relevant rather to software tools supporting production of specifications than to specification language itself.

It should be noted that sub-characteristics of internal quality can be considered from conceptual as well as from representational points of view (Chandra and Manna, 1975; Kleene, 1952) because any language has two most important aspects: its semantic and its syntax. Although we share Felleisen's viewpoint (Felleisen, 1990) that analogy between formal system and language could be highly useful, we argue, however, that for specification languages more useful is analogy with first-order languages. In (Caplinskas et al.,

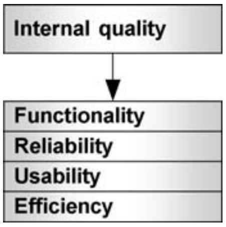


Figure 1. Sub-characteristics of internal quality.

2002) we introduced the notion of the linguistic system defining a formal structure beyond the language. Linguistic system has been defined as four-tuple:

$$\boldsymbol{\Phi} = \langle \boldsymbol{\alpha}, \boldsymbol{\Sigma}, \boldsymbol{\Xi}, \boldsymbol{\Omega} \rangle,$$

where
$\boldsymbol{\alpha}$ is a nonempty set of basic concepts (primitive concepts),
$\boldsymbol{\Sigma}$ is a set of constructors used to construct composite concepts,
$\boldsymbol{\Xi}$ is a nonempty set of constructors used to construct statements,
$\boldsymbol{\Omega}$ is a reasoning apparatus.

In this definition a concept is considered as some abstract meaning that can be represented in many different ways using syntactically different notations. It is also supposed that the notion of constructor comprises also the rules of correct application of constructors. So the set of composite concepts corresponds to the set of the terms of first-order language and the set of statements corresponds to the set of well-defined formulae.

It is supposed further that the set $\boldsymbol{\alpha}$ is defined by the chosen ontology (more exactly, by the chosen conceptualisation) and provides a set of ontological primitives. So it answers the question: In what terms does a language describe the systems? The set $\boldsymbol{\Sigma}$ defines a kind of “algebra of concepts” (Nilsson, 2000) on $\boldsymbol{\alpha}$. It provides the apparatus to define domain-oriented conceptual primitives and operators, including so-called epistemological (abstraction/structural) primitives, that allows constructing complex concepts from primitive ones. In some cases (e.g. for domain-oriented specification languages) domain-oriented primitives can be built-in into linguistic system itself. It is supposed that in such cases domain-oriented primitives have the status of ontological primitives and are considered as elements of the set $\boldsymbol{\alpha}$. The set $\boldsymbol{\Xi}$ defines apparatus to express assertions (more exactly, axioms) about the properties of the system in question.

It should be noted that specification languages can be seen as modelling languages that are used to build an abstract model (a theory) of system in question or, in other words, to define a set of assertions about the required properties of an existing or a future system. Linguistic system describes modelling facilities and, like description logics, distinguishes two modelling levels, conceptual and assertional, although language itself cannot distinguish those levels explicitly. Employing conceptual level apparatus one can define conceptual primitives and create complex concepts. Assertional level apparatus allows formulating statements about the properties of instances of concepts.

It should also be noted that specification languages are used to describe systems as “black boxes” and to define their external properties; design and programming languages deal with the "transparent boxes" and are used to describe implementation details. The purpose of design language is to refine operational specification and the purpose of programming language is to refine design specification. Therefore it is desirable that the transition from specification to design and further to implementation would be seamless. On the other hand, the development of an information system starts usually with business modelling and is completed by the implementation of a number of applications. Architecture of IS should be aligned with business goals and mission, and architectures of applications should be aligned with goals and mission of the IS. Consequently, it is highly desirable that the transition from business level to information processing level and further to application requirements specifications would be seamless, too. Thus, in general case, it is desirable that IS specification language would be suitable also to specify business systems as well as applications and would be applicable at all stages of development. From this we conclude that functionality should characterise the degree of potential appropriateness of a language to specify properties of any kind of system (i.e. business system, information system, or application) and at any stage of development. The higher is the value of functionality as an aggregate quality characteristic of a specification language the less is the probability that this language cannot be applied to specify some system in question.

3.2. Main Structure of Functionality

Usually a language provides core facilities allowing specifying in sufficient details any aspect of "typical" systems in the realm. The functionality of core facilities can be restricted in two different ways: by restricting the application area or by restricting the meaning of used concepts. A flexible language should provide also some apparatus allowing extending or adapting core facilities in order to specify "untypical" systems.. Thus in addition to suitability (i.e. core facilities) the concept of functionality comprises also flexibility of a language. Consequently, we decompose functionality into two sub-characteristics: suitability and flexibility (Figure 2).

Suitability, like functionality, is an aggregate characteristic. It is decomposed further into lower level characteristics (Figure 2) for two reasons: to understand better the nature of functionality and to facilitate the operationalisation of measures of functionality.

Suitability characterises how sophisticated statements about potential systems in a particular realm a specification language is able to express, and at what level of granularity it can be done. The measure of suitability is the probability that any statement about any system of a particular kind can be formulated in terms of a specification language. The probability becomes equal to 1, if a language allows formulating statements about any property of a system in a given realm and expressing it with the needed degree of precision. Therefore the higher is suitability of a language the more sophisticated systems can be described and in the more details it can be done. Because of this dualistic nature of suitability, we split it further into two sub-characteristics: completeness and expressive adequacy (Figure 2).

3.3. Completeness

Completeness is the measure of the ability of specification language to describe any system in question sufficiently and exhaustively. The value of this measure is the proba-

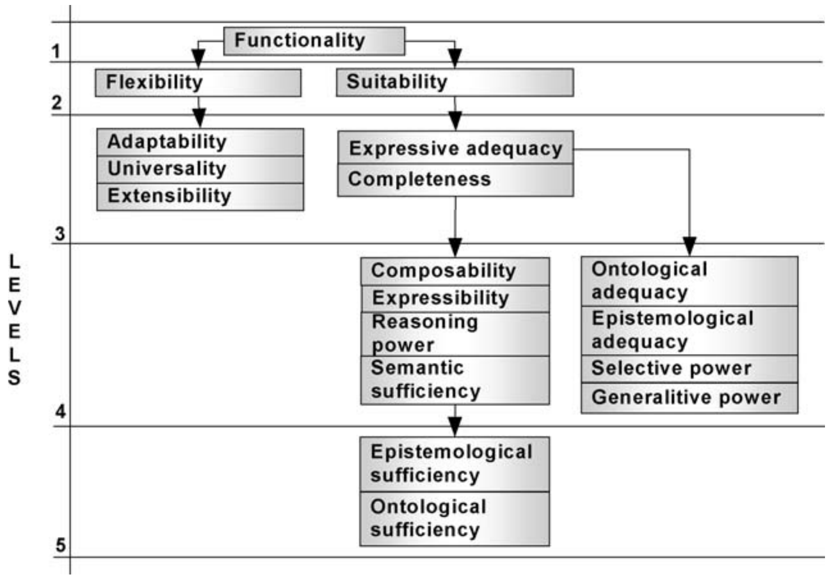


Figure 2. Sub-characteristics of functionality.

bility that any required property of a given system can be described in terms of evaluated specification language. However, completeness does not ensure that statements about these properties will be expressed in adequate terms or even that they will be expressed with sufficient degree of precision. It has several aspects (Figure 2). One of the most important from them is semantic sufficiency.

Semantic sufficiency characterises the conceptual level of the linguistic system. It is the measure of the ability of specification language to specify all "things" that might be necessary for analysis and design of any system in question. Semantic sufficiency is measured by the probability that any required concept can be expressed in terms of specification language in question. In other words, semantic sufficiency answers the question: In which degree the sets α and Σ of the linguistic system beyond the language in question are sufficient for the needs of IS engineer? Consequently we split further semantic adequacy into ontological sufficiency and epistemological sufficiency (Figure 2).

We use the term "ontological sufficiency" instead of the traditional term "ontological completeness" because ontological completeness characterises a specification language with regards to a chosen ontology. A language L is ontologically complete with regards to ontology O if there exists a total mapping f from a set of ontological constructs of α_O of this ontology to the set α_L of ontological primitives of the linguistic system beyond the language L. In other words, language L is ontologically complete if there exists a construct of L that can be used to represent each ontological construct of O. We aim, however, to define an ontology-independent measure of completeness. Ontological sufficiency is the probability that any system in question will be conceptualised successfully through categories provided by α_L. It answers the question: In which degree the set α of the linguistic system beyond the language in question is sufficient for the needs of IS engineer?

Epistemological sufficiency characterises the ability of the linguistic system to express epistemological primitives or, in other words, the constructive power of "algebra of concepts" provided by $\mathbf{\Sigma}$. Here we use the term "epistemology" in a very narrow sense, namely, to refer to concept structuring/abstraction machinery beyond a language L or, more exactly, to the facilities allowing to define conceptual primitives on the basis of ontological ones and to combine defined concepts further in order to form more complex concepts. Epistemological sufficiency can be measured by the probability that all required conceptual structures will be modelled using language constructs. It ensures the ability to structure any given system in an adequate way and answers the question: In which degree the set $\mathbf{\Sigma}$ of the linguistic system beyond the language in question is sufficient for the needs of IS engineer?

Semantic sufficiency does not characterise completeness of a specification language exhaustively because it describes only conceptual level of the linguistic system beyond language. Assertional level can be described by two additional characteristics: expressibility and reasoning power.

Expressibility of a language is a measure of what it can be used to say. However, we use this term in more narrow sense. In our case, it describes the ability of a language to express statements about properties of instances of concepts. In other words, expressibility characterises the class of formulas expressible in language L and answers the question: In which degree the set Ξ of the linguistic system beyond a language L is sufficient for the needs of IS engineer? The measure of the expressibility is the probability that any statement about the system in question can be expressed using language constructs.

Reasoning power of the linguistic system beyond a language L is closely related to its expressibility. Reasoning power characterises the ability of reasoning apparatus $\mathbf{\Omega}$ to derive new statements about properties of conceptual primitives and their compositions. Mainly, reasoning power is important in cases when a specification is used to reason about a system in question, for example, to prove some system properties. The measure of reasoning power is the probability that any property of the system that in principle can be derived from the stated basic assumptions can be proved using reasoning apparatus $\mathbf{\Omega}$.

One more characteristic of completeness is composability. It characterises the degree of possibility to compose language constructs and features together. Limited composability limits functionality because not all meaningful compositions can be expressed using a language in question. In such cases composition cannot be realised because the composition scheme adopted in the language does not support it, although composition is possible from logical perspective. So the measure of composability is the probability that any intuitively and logically correct composition is realisable in the language. In terms of the linguistic system, it means that constructors of $\mathbf{\Sigma}$ and Ξ are free from any artificial restrictions on their applicability.

3.4. Expressive Adequacy

Completeness characterises how exhaustively can be described any system in question using a specification language L. However, it ensures neither that statements about the properties of this system will be expressed with sufficient degree of precision nor that they are described in adequate terms. Expressive adequacy (Figure 2) is a characteristic of internal quality of a language L that describes the ability of this language to specify the properties of a system in question in an adequate way. The measure of this characteristic is the probability that all statements about any system in question can be formulated in adequate terms and can be expressed with required degree of precision. Expressive adequacy has several aspects: ontological adequacy, epistemological adequacy, selective power, and generalitive power. Thus we split expressive power into four sub-characteristics (Figure 2).

Ontological commitments can be regarded as decisions about interpretation of statements in a given language. They can be defined not only by mapping to ontological primitives directly. Even a language of symbolic level can be used to define ontological commitments. For example, they can be defined on the basis of set-theoretical formalism as in specification language Z. Ontological adequacy is the ability of the linguistic system to express its ontological commitments within this system itself. It focuses on the system-model link and ensures that primitives from $\boldsymbol{\alpha}$ are linked directly to categories describing IS. The concern about ontological adequacy is that we can adequately capture peculiarities of IS through ontology O beyond the language L. The measure of ontological adequacy is the probability that any system in question can be conceptualised adequately using ontological primitives provided by $\boldsymbol{\alpha}$. Thus ontological adequacy answers the question: How well do specifications written in the language represent real-world phenomena?

Epistemological adequacy is closely related to ontological adequacy. The term epistemological adequacy is used in many different senses, for example, as a degree to which the language is able to reflect all distinctions that are important. We define epistemological adequacy as the characteristic that describes the degree to which the linguistic system beyond the language is able to express epistemological primitives directly. In other words, epistemological adequacy ensures that constructors provided by $\boldsymbol{\Sigma}$ are linked directly to such epistemological schemes as generalisation, aggregation, and etc. The measure of epistemological adequacy is the probability that any useful epistemological primitive has his counterpart in $\boldsymbol{\Sigma}$.

The notion of selective power has been developed in the realm of relational databases. The measure of selective power of a query language is relational completeness. The language L is relationally complete if any relation derivable by means of relational calculus can be retrieved by means of this language. We argue the notion of expressive adequacy is applicable also to other languages, including specification languages. For example, we can say that the language of first order logic has limited selective power because it cannot distinguish two elementary-equivalent structures (Rudys, 2004). The measure of selective power of the specification language L is the probability that any two different concepts, any two different instances of a concept and any two properties of an instance of a concept can be described in the language L in a distinguishable way.

Finally, the fourth sub-characteristic of expressive adequacy is generalitive power. It characterises the ability of the language to describe system at different levels of granularity. Generalitive power allows suppressing irrelevant details while preserving essential properties of the system. The influence of generalitive power on internal quality of the language is ambiguous. The language that attempts to cover too many levels of granularity is likely to be overly complex. On the other hand, the language that supports only one level of granularity is likely to be overly restricted. However we suppose that measuring functionality other aspects of quality can be ignored (they are described by other characteristics

(Caplinskas and Gasperovic, 2004)) and define the measure of generalitive power as the probability to describe any system in question at any required level of granularity.

3.5. Flexibility

Suitability characterises the power of built-in facilities of the language. Flexibility supplements suitability because it in some sense characterises scope of applicability of the language. The more flexible is the language the easier is to adjust built-in facilities to situations that have been not provided in advance by language designers. Thus flexibility describes the extent to which the language can be adjusted to specify preliminary not intended properties. The measure of flexibility is the probability that the language can be applied to the whole spectrum of IS-related systems, namely, business systems (from the IS design perspective), information systems, and applications.

Flexibility can be achieved in different ways: by universalisation of the language constructs, by adaptability or by extensibility. So we split flexibility into three sub-characteristics: universality, adaptability, and extensibility (Figure 2).

Universality characterises degree of generality of ontological primitives beyond the language.* If the language is based on domain-oriented primitives, it has comparatively low degree of universality because constructs of this language cannot be applied to systems in other realms. On the other hand, such ontological primitives as class or relation are universal and can be applied almost in any realm. Therefore the measure of universality is the probability that ontological primitives, provided by α, are suitable to model concepts from the whole spectrum of systems in question. In other words, universality answers the question: To which extent the language can be considered as a general-purpose language?

Adaptability is in some degree is the characteristic that is opposite to universality. It describes the ability of the language to configure syntax and semantics to adapt it for arbitrary domain. Of course, adaptability is important for general-purpose languages only. Mechanisms of adaptability still are studied insufficiently. An example of such mechanism is tailoring that allows selecting some predefined constructs of general-purpose language and specialising these constructs in order to generate the language dedicated to some specific uses. In general, mechanisms of adaptability allow offering domain-specific notation, constructs and abstractions on the basis of syntax and semantics of a general-purpose specification language. Therefore adaptability characterises both linguistic system and representational system beyond the language. The measure of adaptability is the probability that any required domain-specificity can be introduced using adaptability mechanisms provided by the language.

Extensibility is closely related to adaptability. It also allows introducing domain specificity by extending a general-purpose language with new features. However, we argue that adaptability and extensibility should be considered as separate characteristics, because they characterise presence of different mechanisms in the language and their power. Adaptability answers the question: To which extent is the language reconfigurable? Extensibility answers the question: To which extent can the language be extended by new features? A well-known mechanism of extensibility is so-called problem domain-specific profiles in

* It should be noted that composability also can be considered as some aspect of universality.

UML that allow introducing some domain specificity into this language. The measure of extensibility is the probability that any required domain-specific features can be introduced using extensibility mechanisms provided by the language.

4. OPERATIONALISATION OF THE MEASURES OF FUNCTIONALITY

The proposed approach provides that the values of all characteristics of internal quality are treated as probabilities. Indeed, probabilities in our approach are used as a kind of rating levels. Such approach allows calculating values of aggregate characteristics from the values of lower level characteristics easily. However, characteristics of the lowest level such as ontological and epistemological sufficiency, ontological and epistemological adequacy, selective and generalitive power, universality, adaptability, and extensibility should be measured directly. Ontological completeness is measured usually (Rosemann and Green, 2002; Wand and Weber, 1993; Wand and Weber, 1990) by comparing language constructs to categories of some chosen ontology. However, this approach cannot be used to measure neither ontological nor epistemological adequacy, because we aim to develop ontology-independent approach to evaluate internal quality of the specification language. We argue to use for this aim assessment case suites, as it is common for certification of compilers. We are working currently on the development of such suites.

The measurement of expressibility and reasoning power is a complex problem. We attempt to develop for this aim the measurement techniques that are based on classification of formulae and reasoning methods and on analysis of peculiarities of Ξ and $\mathbf{\Omega}$. The limited space of this paper does not allow us to discuss the issues of operationalisation in more details. We intend to consider this problem in a separate work.

5. CONCLUSIONS

This paper continues research on the evaluation of the quality of specification languages, which has begun in (Caplinskas et al., 2002; Caplinskas and Gasperovic, 2004). In (Caplinskas and Gasperovic, 2004) we have proposed the taxonomy of characteristics of internal quality and methodology how to evaluate quality in use (external quality) on the basis of measurements of internal quality. Despite the fact that the quality of specification languages has strong impact on the quality of specifications the research in this field is still in beginning. No commonly accepted agreement exists about the required set of quality characteristics and even about their names. We hope that the analysis of the concept of functionality that has been done in this paper will contribute both to the research on evaluation of the quality of existing specification languages and to the development of new ones.

REFERENCES

Chandra, A. K., and Manna, Z., 1975, The power of programming features, *Journal of Programming Languages* **1**: 219–232.

Caplinskas, A., and Gasperovic, J., 2004, A taxonomy of characteristics to evaluate specification languages, in: *BalticDB&IS 2004. Proc. of the Sixth International Baltic Conference on Databases and Information Systems, Riga, Latvia, June 6–9, 2004*, Vol. 672, J. Barzdins, ed., University of Latvia, Riga, pp. 321–336.

Caplinskas, A., Lupeikiene, A., and Vasilecas, O., 2002, A framework to analyse and evaluate information systems specification languages, in: *ADBIS 2002. Proc. of the 6th East European Conference, Bratislava, Slovakia, September 2002, LNCS 2435*, Y. Manolopoulos and P. Navrat, eds., Springer, pp. 248–262.

Colomb, R. M., and Weber, R., 1998, Completeness and quality of ontology for an Information System, in: *FOIS'98. Proc. of the International Conference of Formal Ontology in Information System, Trento, Italy, June 6–8, 1998*, N. Guarino, ed., IOS Press, Amsterdam, pp. 207–217.

Felleisen, M., 1990, On the expressive power of programming languages, in: *ESOP '90. Proc. of the 3^{rd} European Symposium on Programming, Copenhagen, Denmark, May 1990*, Vol. 432, N. Jones, ed., Springer-Verlag, pp. 134–151; http://citeseer.ist.psu.edu/cache/papers/cs/633/ftp:zSzzSzftp.cs.indiana.eduzSzpubzSzscheme-repositoryzSzdoczSzpubszSzexpress.pdf/felleisen90expressive.pdf/.

ISO/IEC 9126, 1991, *Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their use*. International standard, first edition, 1991-12-15, reference number ISO/IEC 9126: 1991(E).

King, M., and Maegaard, B., 1998, Issues in natural language systems evaluation, in: *LREC. Proc. of the First Conference on Language Resources and Evaluation, Granada, Spain, May 28–30, 1998*, Vol. 1, pp. 225–230; http://citeseer.nj.nec.com/514907.html.

Kleene, S. C., 1952, *Introduction to Metamathematics*, D. Van Nostrand Co., Inc., New York, N. Y.

Krogstie, J., 2003, Evaluating UML using a generic quality framework, in: *UML and the Unified Process*, L. Favre, ed., Idea Group Publishing, Hershey, PA, USA, pp. 1–22.

Krogstie, J., 2002, A semiotic approach to quality in requirements specifications, in: *Proc. of the IFIP TC8/WG8.1 Working Conference on Organizational Semiotics: Evolving a Science of Information Systems, Montreal, Quebec, Canada, July 23–25, 2001*, K. Liu, R. Clarke, P. B. Andersen, and R. Stamper, with El-Sayed Abou-Zeid, eds., Kluwer Academic Publishers, Boston, pp. 231–249.

Krogstie, J., 2001, Using a semiotic framework to evaluate UML for the development of models of high quality, in: *Unified Modelling Language: Systems Analysis, Design and Development Issues*, K. Siau and T. A. Halpin, eds., Idea Group Publishing, Hershey, pp. 89–106.

Krogstie, J., and Sølvberg, A., 1999, *Information Systems Engineering: Conceptual Modeling in a Quality Perspective* (the draft of the book), Information Systems Groups, NTNU, Trondheim, Norway.

Lindland, O. I., Sindre, G., and Sølvberg, A., 1994, Understanding quality in conceptual modelling, *IEEE Software* **11**(2), pp. 42–49.

Milton, S., Kazmierczak, E., and Keen, C., 1998, Comparing data modelling frameworks using Chisholm's ontology, in: *ECIS'98. Proc. of the 4th European Conference on Information Systems, Aix-en-Provence, France, June 1998*, vol. 1, J-A. Bartoli, ed., Euro-Arab Management School, Aix-en-Provence, pp. 260–272.

Nilsson, J. F., 2000, A conceptual space logic, in: *Information Modelling and Knowledge Bases XI*, E. Kawaguchi et al., eds., IOS Press/Ohmsha, Amsterdam, pp. 26–40;
http://www.imm.dtu.dk/~jfn/publications/conceptspaces.ps.

QStudio® for Java, 2003, The Software Health Tool for Java. QA Systems BV, The Netherlands;
http://www.qa-systems.com/welcome.html.

Reynolds, J. C., 1981, The essence of Algol, in: *Algorithmic Languages*, Jaco W. de Bakker and J. C. van Vliet, eds., North-Holland, Amsterdam, pp. 345–372.

Rosemann, M., Green, P., 2002, Developing a meta-model for the Bunge-Wand-Weber ontological constructs, *Journal of Information Systems* **27**(2):75–91.

Rudys, A., 2004, Elementary-equivalence, Lecture 21, in: *Logic in Computer Sciences* (course material), M. Y. Vardi, ed., Department of Computer Science, Rice University;
http://www.cs.rice.edu/~vardi/comp409/2001/lec21.ps.

Steele, G. L. Jr., and Sussman, G. J. Lambda, 1976, The ultimate imperative, Memo 353, MIT AI Lab.;
ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-353.pdf.

Troelstra, A. S., 1973, *Metamathematical Investigation of the Intuitionistic Arithmetic and Analysis*, LNM, 344, Springer-Verlag.

Wand, Y., and Weber, R., 1993, On the Ontological Expressiveness of Information Systems Analysis and Design Grammars, *Journal of Information Systems* **3**(4):217–237.

Wand, Y., and Weber, R., 1990, Mario Bunge's ontology as a formal foundation for Information Systems concepts, in: *Studies on Mario Bunge's Treatise*, P. Weingartner, and G. J. W. Dorn, eds., Rodopi, pp. 123–149.

TEMPORAL SEMANTIC ABSTRACTION BASED ON LAYERED MULTIMEDIA DATA MODEL

Yan Jian-feng, Li Zhan-huai and Guo Chen-juan*

1. INTRODUCTION

With the development of the technique of data gathering and the capability of computer storage, more and more multimedia data has been introduced into the computer system; therefore, researches on multimedia database have become more important than ever before. Temporal feature is one of the most important features in multimedia data, such as video and audio. The multimedia database, which is the kernel part of multimedia information retrieval system, should be able to provide detailed descriptions and queries to such information.[1]

Compared to traditional alphanumeric data, multimedia data have a number of content-based information and semantic information. So, the multimedia data model should not only provide how to describe the alphanumeric data, but also describe some specific data, such as temporal features in video or audio data and constraints among them. Semantic information of temporal and spatial features are the most typical different between multimedia data and other data, therefore, the conceptual model for describing multimedia data should be able to distinguish different objects and the temporal or spatial relationships among them. Then, based on these, it can answer the queries including temporal and spatial features. For video data, temporal features are the base of spatial features. Therefore, we have to introduce temporal dimension into the model when building such data models.

Many works have been done in order to introduce temporal features into the multimedia data model. Time line is a directed descriptive method for media objects. It uses absolute time to annotate temporal features of media objects in time axis. Breiteneder proposed a model based on time line,[2] which combined the video data into different groups through entities such as Movie, Tracks, Media and Layer, and relationships such as *is-derived* and *t-comp*. This model can distinct differentiation in expressive scene and transmitted time. However, it does not have related description for temporal relationship, either introduces temporal abstraction, therefore what can be described for semantic is limited.

* Northwestern Polytechnical University, Xi'an, Shaanxi, 710072, P. R. China, jfyan@mail.nwpu.edu.cn, lizhh@nwpu.edu.cn, cjguo@mail.nwpu.edu.cn.

Based on the Petri-Net, Little has proposed a descriptive method called OCPN,[3] which is able to describe comparatively simple temporal relationship like synchronism and continuity, by using *place*, which describes multimedia object, and *transition*, which describes temporal synchronism. It can also describe absolute temporal features precisely by mapping *place* information into real numbers. However, this model can't provide abstract description for temporal features, and cannot support description for user's interactive operation. Therefore when it is desired to enrich media information and recombine media information that required specific domain knowledge, this model seems helpless.

The temporal complicacy of media objects is represented in comparative temporal relationship. For example, there exist many temporal relationships in video object, such as *before*, *after*, *overlaps*, *equals* and so on. Allen has concluded thirty correlations in two temporal intervals.[4] For the point of view of application, Schloss[5] has discussed a method to build layered model multimedia data, and proposed a method to operate temporal relation through calculations of multimedia event. Little[6] has proposed TIB (Time-Interval-Based model), which not only describes absolute temporal intervals in object, but also describes relative temporal relationships. Continuous research[7] discusses the queries in temporal intervals when there exists disconnection in temporal intervals. Zhang[8] introduced this idea into data stream management, which extends the application of temporal interval model.

The description and management of information must satisfy different abstract layers that users required. According to the rich semantic queries of multimedia temporal/spatial features, Megalou[9] proposed a method for multimedia temporal/spatial semantic abstraction, which divides multimedia objects into conceptual objects and presentational objects, and the formal describe the content information of multimedia information, while the latter describe temporal/spatial feature of the conceptual objects. Although this work provided an operable semantic model for temporal/spatial semantic abstractions, temporal/spatial semantic abstractions of multimedia object would change according to different application and background. And so do the related generalization layers and aggregation granularities. Therefore it is difficult to use this model to do everything.

Further more, multimedia data management should provide description for the complicated semantic information when describing temporal information. M. Hacid[10] proposed a model named $CoPaV^2$, which focuses on describing high semantic information by semantic objects and describing temporal information by temporal aggregation objects. These two kinds of objects build temporal relationships by object reference. The author also discusses the queries based on the model and designs a query language to query the temporal relationships between semantic objects. But this model can't provide precise description and query for temporal interval, which is its limitation.

Our multimedia data model,[11] which can provide a higher abstraction level, is designed. And this model cans provide a comprehensive description for multimedia content information, such as physics features, structure features and semantic features. Based on this, in this paper, we will discuses description and query for absolute temporal intervals and relative temporal relationships. Through the analysis of generalization and aggregation, we also propose a structural algorithm and query method of relative temporal relationship constrains in multimedia data content information management, and these can be used in different generalization layers and satisfy different aggregation granularities.

2. THE MDU DATA MODEL

2.1. Multimedia Object

A multimedia object, which describes a meaningful section of multimedia data, can be identified by an identifier, which is one of the elements in identification set *ID*. *ID* is a countable and infinite set. Different multimedia objects have different object identifiers. In the life circle of an object, this element of *ID* is stable, and system can access an object by its identifier.

The attributes of an object are used to describe the features of the object, and the value of one attribute corresponds to one feature value of multimedia data. Multimedia objects, which own temporal and spatial attributes, should be able to describe the temporal and spatial features of this part of multimedia data. Those features are described by the attributes of temporal interval and spatial interval, which describe the place where this object is in the temporal interval domain and spatial interval domain, respectively.

Temporal interval domain T is defined as a set of tuples, one of which looks like (t_1, t_2). Here, t_1 and t_2 are integers and $t_1 = t_2$, which represents the time that the object appears and disappears in the temporal axis, respectively. The values of t_1 and t_2 are the absolute time points of temporal feature. Any object, which have temporal constraints, own a special attribute, which is temporal interval attribute I, and the value of this attribute is an element in T, (t_s, t_e).

Let's write $A = \{a_1, a_2, \ldots, a_n\}$ for a set of all possible attributes of all the objects, and write V for the value set, which is consisted of all the elements subsets in $C \cup ID \cup T$. Here, C is a set of constants, and $C \in 2^{\aleph} \cup 2^{\Re} \cup 2^{\Im}$. Here, $\aleph$ is the set of all possible natural numbers, $\Re$ is the set of all possible real numbers, and $\Im$ is the set of all possible characters.

Definition 1. A **Multimedia Object**, *mo*, is a tuple *(oid, v)*, in which *oid* is an element of set *ID*. Function *New(void)* can generate a new object identifier, which is different from other objects. And v is an n-tuple $(a_1 : v_1, \ldots, a_n : v_n)$, here, object attribute $a_i (1 = i = n)$, which means the attribute type of this object, is an element of A. Function *Attributes(oid)* can get all attributes of the object whose identifier is *oid*. $v_i (1 = i = n)$, which is an element in set V, is the value of a_i. In the rest of this paper, we write $oid.a_i$ for the value of attribute a_i for the object with identifier *oid*.

The following figure describes some multimedia objects in a video data of a football game, especially, the temporal features and spatial features in some absolute time points (spatial features are fixed by MBR, which is a minimum bound rectangle framing spatial coordinate of an object. MBRs are represented in this figure by write rectangles.). The object O_1 represents the goal object that appears at absolute time point 67 and disappears at time point 81. O_2 represents the Argentina sporter A. O_3 represents a England sporter, who disappears at time point 51. O_4 represents the Argentina sporter B. O_5 represents the goalkeeper appears at 60. The object O_6 represents the football. We can notice that at time point 70, this object will overlap the goal object in the spatial relationship.

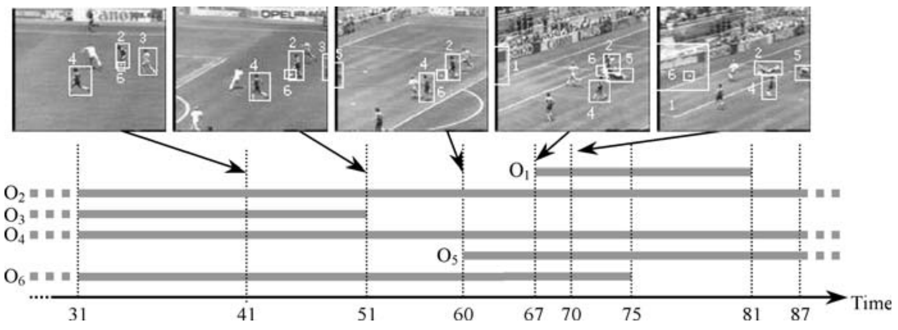


Figure 1. An example of the temporal features in a video data.

2.2. Multimedia Description Unit and Temporal Constraints

We represent a group of objects, which have certain relationships with each other, by a data structure named Multimedia Description Unit (MDU). The following is the formal definition of MDU.

Definition 2. A **Multimedia Description Unit** (MDU) is a triplet u: $\langle O, C, R\rangle$. Here, O is a set of all objects that this MDU needs to describe. C is a set of all possible relationship types that maybe exist among objects. R is a subset of the mapping set $2^O \times 2^O \rightarrow C$, which represents all possible relationships between objects.

Following Allen's definition, C consists of at least the following 13 temporal relationship elements, *equal, starts, finishes, meets, overlaps, before, during, starts*$^{-1}$, *finishes*$^{-1}$, *meets*$^{-1}$, *overlaps*$^{-1}$, *before*$^{-1}$, *durings*$^{-1}$. Temporal constraints exist between the temporal intervals of multimedia objects and the temporal relationships of those objects. For example, two objects that have same attribute values in temporal interval must have a temporal relationship called *equal* between them, which means that they appear and disappear at the same time. If there exists some certain relationship c between object X and object Y, the constraints between temporal intervals and temporal relationships is defined in Table 1.

Additional six temporal relationships like c^{-1} can be gotten by exchanging object X and object Y in upper table.

Following this formal definition, the video in Figure 1 can be described by the following MDU.

$$\begin{aligned}
u_{video} = \langle &\{O_1, O_2, O_3, O_4, O_5, O_6\},\\
&\{equal, starts, finishes, overlaps, during\},\\
&\{(O_6, O_1) \rightarrow \textit{ overlaps}, (O_5, O_2) \rightarrow \textit{finishes}, (O_2, O_4) \rightarrow \textit{equal},\\
&(O_3, O_2) \rightarrow \textit{starts}, (O_6, O_2) \rightarrow \textit{starts}, (O_5, O_1) \rightarrow \textit{during}\}\rangle
\end{aligned}$$

We can find that u_{video} describes six multimedia objects and their temporal relationship, and objects are described by their attributes. These attributes can be gotten from image processing and target identifying algorithms, and different applications focus in different attributes. For example, we can get the object description of the football game video

Table 1. Temporal constraints between Allen's temporal relationships and temporal intervals

Temporal interval c	Temporal relationship
X equal Y	$X.t_s = Y.t_s \wedge X.t_e = Y.t_e$
X starts Y	$X.t_s = Y.t_s \wedge X.t_e \neq Y.t_e$
X finishes Y	$X.t_s \neq Y.t_s \wedge X.t_e = Y.t_e$
X meets Y	$X.t_e = Y.t_s$
X before Y	$X.t_e < Y.t_s$
X overlaps Y	$X.t_s < Y.t_s \wedge X.t_e < Y.t_e$
X during Y	$X.t_s > Y.t_s \wedge X.t_e < Y.t_e$

as following.

$$O_1 = (oid_1, (NAME\text{: gate, } TEAM\text{: england, } I\text{:}(67,81)))$$
$$O_2 = (oid_2, (NAME\text{: sporterA, } TEAM\text{: argentina, } I\text{:}(31,51)))$$
$$O_3 = (oid_3, (NAME\text{: sporter, } TEAM\text{: england, } I\text{:}(67,81)))$$
$$O_4 = (oid_4, (NAME\text{: sporterB, } TEAM\text{: argentina, } I\text{:}(31,87)))$$
$$O_5 = (oid_5, (NAME\text{: goalkeeper, } TEAM\text{: england, } I\text{:}(60,87)))$$
$$O_6 = (oid_6, (NAME\text{: football, } I\text{:}(31,75)))$$

3. SEMANTIC ABSTRACTION IN MULTI-GRANULARITIES TEMPORAL INTERVAL

3.1. Partitions of Temporal Interval Domain

Temporal interval domain T can be divided into several partitions $T(G_0)$, $T(G_1)$, $\ldots$, $T(G_n)$ according to different granularities $G_0, G_1, \ldots, G_n$. Under one granularity, there is a partition, and all the different partitions form a layered structure.

Definition 3. The **Comparison of Granularity** exists in any two different granularities G_i and G_j of temporal interval domain T. For any temporal interval I, which belongs to $T(G_j)$, if we can find a set of temporal interval $\{I_1, I_2, \ldots, I_n\}$ in $T(G_i)$, which contains I, then we say that granularity G_i is **Finer** than granularity G_j, which is represented by $G_i \prec G_j$. Formally, it can be defined as the following:

$$\forall I \in T(G_j), I = \bigcup_{k=1}^{n} I_i, \{I_1, I_2, \ldots, I_n\} \subseteq T(G_i) \tag{1}$$

For any $I_l, I_k \in \{I_1, I_2, \ldots, I_n\}$, $I_l \cup I_k = (\min(I_l.t_s, I_k.t_s), \max(I_l.t_e, I_k.t_e))$

Such mappings exist between different granularities of temporal interval domain. The map from coarse granularity to fine granularity is called interval extension, which means using temporal interval set on G_i to express some temporal interval on G_j; while the map from fine granularity to coarse granularity is called interval approximation, which means using a bounder temporal interval on G_i to express a temporal interval on granularity G_i.

Definition 4. For the temporal interval on granularity G_j, the mapping $Exp_{G_j}^{G_i} : T(G_j) \to T(G_i)$ of interval extension on granularity G_i is a mapping set described by (2). Similarly, for the temporal interval on granularity $G_{i,}$, the mapping $App_{G_i}^{G_j} : T(G_i) \to T(G_j)$ of interval approximation on granularity G_j is a mapping set described by (3):

$$\forall I \in T(G_j), Exp_{G_j}^{G_i}(I) = \{I' \mid I' \in T(G_i), I' \subseteq I\} \tag{2}$$

$$\forall I' \in T(G_i), App_{G_i}^{G_j}(I') = I, I \in T(G_j) \wedge I \subseteq I' \tag{3}$$

The map of extension and approximation of temporal interval between different granularities is showed as following.

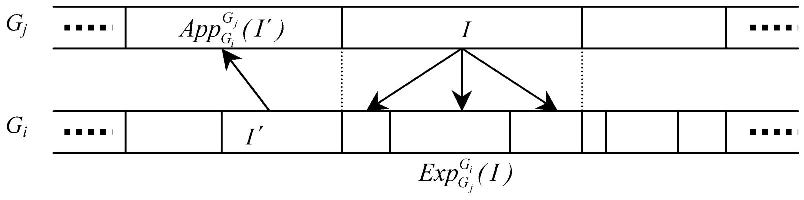


Figure 2. The mappings between two granularities G_i and $G_j (G_i \prec G_j)$.

3.2. Temporal Semantic of Generalization

In the temporal queries of multimedia data, besides the querying about thirteen basic temporal relationships by Allen, it is always necessary to answer the queries on generalization temporal semantics. Here, generalization is to abstract a more normal and higher layer temporal relationship from two or more basic temporal relationships. For example, the query *to find an object set that appears at the same time with object A*, or *to find an object set that appears after object A*. These two queries do not need strict basic temporal relationships. It only needs to query through generalized temporal semantics relationship between objects. Temporal generalization is different from one application to another application.

Figure 3 describes a hierarchy of one kind temporal generalization, in which the relationship *before* and *meets* are abstracted as *ahead*, which represents the predecessor relationship between objects. While the relationship *before*$^{-1}$ and *meets*$^{-1}$ are abstracted as *behind*, which represents the successor relationship between objects. The relationship *Ahead* and *behind* have been abstracted as *sequential*, which represents the sequential relationship between objects.

Allen's basic temporal relationships exist in the multimedia data that is described by MDU. For the queries with generalization temporal relationship, we have to get abstract temporal relationships between MDU objects according to the different generalization hierarchy.

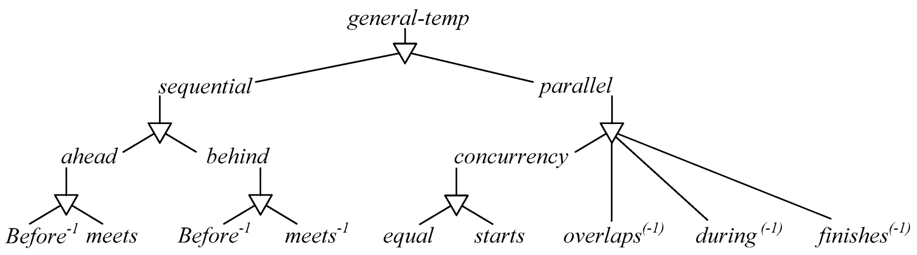


Figure 3. An example of *temporal* generalization hierarchy ($c^{(-1)}$ means $c \cup c^{-1}$).

Table 2. Temporal generalization abstraction $hierarchy_i$ and temporal constraints

Temporal Generalization Abstraction	Abstract Temporal Relationships	Basic Temporal Relationships	Temporal Interval
X ahead Y	*null*	*{before, meets}*	$X.t_e \leq Y.t_s$
X behind Y	*null*	*{before^{-1}, meets^{-1}}*	$Y.t_e \leq X.t_s$
X concurrency Y	*null*	*{equal, starts}*	$X.t_s = Y.t_s$
X sequential Y	*{ahead, behind}*	*null*	$X.t_e \leq Y.t_s \vee Y.t_e \leq X.t_s$
X parallel Y	*{concurrency}*	*{overlaps$^{(-1)}$,during$^{(-1)}$ finishes$^{(-1)}$}*	$[X.t_s, X.t_e] \cap [Y.t_s, Y.t_e] \neq null$
X general-temp Y	*{sequential , parallel}*	*null*	*true*

The generalization abstraction of temporal relationship needs to judge new temporal interval constraints. For the relationship between object X and object Y as described in Figure 3, Table 2 defines its temporal interval constraints of generalization temporal relationships.

Definition 5. **Temporal relationship generalization** of a multimedia description unit u is the generalization abstraction of temporal relationship between objects in u, written as $Gen_T(u, hierarchy_i(H))$, which means a new multimedia description unit u' that is got from u through temporal generalization. Here, $hierarchy_i$ means a certain temporal generalization abstraction i in the form of a tree; H means a certain abstract node in this generalization abstraction. If all children of H are basic temporal relationship, we call H **direct generalization** of basic temporal relationship; if not, we call **indirect generalization**.

The temporal direct generalization is defined as:

$u' = Gen_T(u, hierarchy_i(H)) \Leftrightarrow u'.O = u.O, u'.C = u.C \cup H, u'.R = u.R \cup R'$, in which:

$R' = \{(O_i, O_j) \rightarrow H \mid \forall O_i, O_j \in u'.O, \exists c \in C, ((O_i, O_j) \rightarrow c) \in R \wedge H$ is the direct generalization abstraction of $c\}$

If abstract node H can be got through many times abstraction of basic temporal relationship c, such as the indirect generalization *sequential* of the basic temporal relationship $before^{(-1)}$ and $meets^{(-1)}$, temporal relationship generalization of MDU can be got through by a recursive algorithm. To a known generalization abstract structure, we can abstract the basic temporal relationship to every corresponding node in generalization structure; therefore it has been extending the description ability of MDU in the system. The following is a completed description algorithm for temporal relationship generalization semantic.

```
procedure gen_extend(MDU $u$, HIERARCHY $h$)
    //the root note of generalization layer in form of
    tree is n
    gen_node($u$, $n$)
end procedure
function gen_node(MDU $u$, NODE $n$)
return MDU
    if ($n$ is the directed generalization note of basic
       temporal relatioship)
    then return $u=Gen_T(u, n)$
    else begin
            find all generalization abstraction child
            note of $n$ , signed as $n_1$, $n_2$, … ,$n_k$
            for $i$=1 to $k$ do $u_i= u_i \cup$ gen_node($u$, $n_i$)
              //the combination among object set,
              relationship type and object relationship
              is the combination of MDU data type.
            return $u_i$
         end
end function
```

The definition of temporal relationship generalization allows to do abstract temporal expression for temporal relationship between multimedia objects according to the required abstract method, which extends the expression capability of MDU in temporal relationship. For example, we can use the generalization abstract layer in Figure 3 to extend MDU u_{video}(use an abstract hierarchy sub-tree whose root is *parallel*) of the video section in Figure 1 to get a new MDU u_{gen_video} :

$$\begin{aligned}
u_{gen\_video} = \langle &\{O_1, O_2, O_3, O_4, O_5, O_6\},\\
&\{\textit{equal, starts, finishes, overlaps, during, parallel, concurrency}\},\\
&\{(O_6, O_1) \rightarrow \textit{overlaps}, (O_6, O_1) \rightarrow \textit{parallel}, (O_5, O_2) \rightarrow \textit{finishes}, (O_5, O_2) \rightarrow \\
&\textit{parallel},\\
&(O_2, O_4) \rightarrow \textit{equal}, (O_2, O_4) \rightarrow \textit{concurrency}, (O_2, O_4) \rightarrow \textit{parallel},\\
&(O_3, O_2) \rightarrow \textit{starts}, (O_3, O_2) \rightarrow \textit{concurrency}, (O_3, O_2) \rightarrow \textit{parallel},\\
&(O_6, O_2) \rightarrow \textit{starts}, (O_6, O_2) \rightarrow \textit{concurrency}, (O_6, O_2) \rightarrow \textit{parallel},\\
&(O_5, O_1) \rightarrow \textit{during}, (O_5, O_1) \rightarrow \textit{concurrency}, (O_5, O_1) \rightarrow \textit{parallel}, \}\rangle
\end{aligned}$$

It is obviously that the description capability of u_{gen_video} is higher than u_{video}, which provids data description for temporal relationship queries like *parallel* and *concurrency*.

3.3. Temporal Aggregation Semantic

In the data management of multimedia information system, another important semantic abstraction, aggregation semantics, is needed, and this semantic abstraction can aggregate some parts to a whole. For example, there is a video section, which is about the best scenes in a football game, and has described every team member as an object by MDU, including the temporal relationship between objects. However, if we want to answer the query like *to find all scenes including shooting team members and resist team member*s, we have to abstract some objects and their temporal relationship to a certain group, so we can make the objects like shooting team members and resist team members as a new shooting scene object. The essence of temporal aggregation semantics is to construct temporal relationship in higher layer granularity from temporal relationship in some lower layer granularity on temporal interval. Here, object in coarse granularity has maps to a group objects in fine granularity and the temporal relationship between them.

Definition 6. **Temporal relationship aggregation** in a multimedia description unit u is the aggregation abstraction of a set of object' sets $\{O^1, O^2, \dots, O^m\}$ *on* u, (in which $\bigcup_{i=1}^{m} O^i \in u.O$), written as $Agg_T(u, \{O^1, O^2, \dots, O^m\})$, which represents the new multimedia description unit u' gotten from temporal aggregation on u. If the set $\{O^1, O^2, \dots, O^m\}$ only includes one set of objects, it is named as **basic aggregation**; otherwise, it is named as **complex aggregation**. To the basic aggregation, if $O^1 = \{O_1, O_2, \dots, O_n\}$, then we can define:

$$u' = Agg_T(u, \{O_1, O_2, \dots, O_n\}) \Leftrightarrow u'.O = u.O \cup O'$$

Here, the object O' is a new object aggregateed from the set of objects $\{O_1, O_2, \dots, O_n\}$, whose object identifier, which is different from other identifiers, is created by the system, and the set of attributes, which is shown as following, is gotten from objects in $\{O_1, O_2, \dots, O_n\}$:

$$O'.ID = New(MO),\ Attibutes(O'.ID) = \bigcup_{i=1}^{n} Attibutes(O_i.ID)(O_i \in \{O_1, O_2, \dots, O_n\})$$

Temporal interval attribute I in the new multimedia object O' can be gotten from granularity approximation operation. Let G_i be the granularity before aggregation; and G_j be the granularity after that. The formal calculation is defined as following:

$$O'.I = \bigcup_{i=1}^{n} App_{G_i}^{G_j}(O_i.I)(O_i \in \{O_1, O_2, \dots, O_n\})$$

New object will be created after aggregation semantic abstraction, and has extended the former temporal relationship set of multimedia description unit. Therefore, some operation is required to get the temporal relationship type set C and temporal relationship set R' of u'. The detailed operation is shown as following:

$$C' = \{aggregate, aggregate^{-1}\},$$

$$R' = \{(O_i, O') \rightarrow aggregate, (O', O_i) \rightarrow aggregate^{-1} \mid O_i \in \{O_1, O_2, \dots, O_n\}$$

Then, we can get a new complete multimedia description unit u' as following:

$$u'.C = u.C \cup C', u'.R = u.R \cup R'$$

The following is a descriptive algorithm for temporal relationship generalization semantics:

```
procedure agg_extend(MDU $u$, OBJ-SETs $\{O^1,O^2,\ldots,O^m\}$)
    for $i$=1 to $m$ do
        $u=Agg_T(u, O^i)$
end procedur
```

Applications should appoint the object set needed to be aggregateed for temporal aggregation abstraction, and this appointment is decided by the semantics of multimedia objects. For example, shoot scenes can be calculated by football shooting to the gate in a football match. Therefore, goal scenes should include the scene subset of football and gate. We can aggregate set $\{O_1, O_6\}$ on u_{video} to a new MDU u_{agg_video} as following:

$$\begin{aligned}
u_{agg\_video} = \langle &\{O_1, O_2, O_3, O_4, O_5, O_6, O_{new}\},\\
&\{\textit{equal, starts, finishes, overlaps, during}\},\\
&\{(O_6, O_1) \rightarrow \textit{overlaps}, (O_5, O_2) \rightarrow \textit{finishes}, (O_2, O_4) \rightarrow \textit{equal},\\
&(O_3, O_2) \rightarrow \textit{starts}, (O_6, O_2) \rightarrow \textit{starts}, (O_5, O_1) \rightarrow \textit{during},\\
&(O_1, O_{new}) \rightarrow \textit{aggregate}, (O_6, O_{new}) \rightarrow \textit{aggregate},\\
&(O_{new}, O_1) \rightarrow \textit{aggregate}^{-1}, (O_{new}, O_6) \rightarrow \textit{aggregate}^{-1}, \}\rangle\\
O_{new} = (&oid_{new}, (\textit{NAME: shoot, TEAM: argentina, I:}(60,87)))
\end{aligned}$$

From above, it can be concluded that the descriptive capability of u_{agg_video} is stronger than u_{video}, which has provided data description for temporal relationship query, such as shooting.

Additionally, temporal aggregation abstraction is especially suit for management of multimedia file on Internet. The access to a web site is always processed according to right figure. The click to different entry would lead to different processes, which can be viewed as sequence of multimedia file in temporal constrains. If application focuses on the content of multimedia file, the creation of MDU should be fined to each object's granularity in the file. If application needs to manage the right figure of the wed site, then it is required to aggregate the file related to one subject to multimedia description unit on file granularity.

4. CONCLUSION

Multimedia temporal features management and temporal semantic abstraction are the focal problems that need to be solved in the research domain of multimedia data management. Based on absolute temporal management and temporal relationship research, this paper proposes a multimedia data model based on multimedia object description, and discusses the descriptive algorithm of generalization and aggregation, which are the two important temporal semantic abstractions. The table below describes a comparison between the model in this paper and other related researches:

From this table, it is obviously that MDU model, can describe absolute temporal features and temporal relationships at the same time. In semantic abstraction, MDU can perfectly describe generalization and aggregation, the two important temporal semantic abstractions, as well as support multimedia data independence, which mean it can realize

Table 3. The comparison between related multimedia data models (+ Means support, - means unsupported)

	Timeline	OCPN	TIB	SMA	*CoPaV*^{2}	MDU
Absolute temporal	+	-	+	+	+	+
Temporal relationship	-	Partly +	+	-	-	+
Temporal semantic abstraction	-	-	-	+	-	+
Data independence	-	-	-	-	+	+

multimedia index and reform based on temporal features. Our future researches will be focused on the realizing of the above model, including temporal query processing, query optimizing and the design of query language of multimedia data.

REFERENCES

1. M. Tamer Özsu, Issues in Multimedia Database Management, in: *IDEAS'99. Proceedings of the International Symposium on Database Engineering and Applications* (IEEE-CS Press, Montrea, 1999), pp. 452–459.
2. C. Breiteneder, S. Gibbs, and D. Tsichritzis, Modeling of Audio/Video Data, in: *Proceedings of the 11th International Conference on the Entity-Relationship Approach* (Karlsruhe, Germany, October 1992), pp. 322–339.
3. T. D. C. Little, A digital On-Demand Video Service Supportin Content-based Queries, in: *Proceedings of ACM Multimedia* (USA, August 1993), pp. 427–436.
4. J. F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM* **26**(11), 832–843 (November 1983).
5. G. A. Schloss and M. J. Wynblatt, Building Temporal Structures in a Layered Multimedia Data Model, in: *Proceedings of ACM Multimedia* (San Francisco, USA, August, 1994), pp. 271–278.
6. T. Little and A. Ghafoor, Interval-based conceptual models for time-dependent multimedia data, *IEEE Transaction on Knowledge and Data Engineering* **5**(4), 551–563 (1993).
7. T. Amagasa, M. Aritsugi, and Y. Kanamori, An Implementation of Interval-Based Conceptual Model for Temporal Data, *IEICE'99, Transactions on Information & System, Special Issue on Next Generation Database Technology, Jan.* **E82-D**(1), 136–146 (1999).
8. D. Zhang, D. Gunopulos, V. J. Tsotras, and B. Seeger, Temporal and spatio-temporal aggregations over data streams using multiple time granularities, *Information Systems* **28**(1–2), 61–84 (2003).
9. E. Megalou and T. Hadzilacos, Semantic abstractions in the multimedia domain, *IEEE Transactions on Knowledge Data Engineering* **15**(1), 136–160 (2003).
10. M. Hacid, C. Decleir, and J. Kouloumdjian, A database approach for modeling and querying video data, *IEEE Transactions on Knowledge and Data Engineering* **12**(5), 729–750 (2000).
11. J. F. Yan, Y. Zhang, and Z. H. Li, A Multimedia Document Database Model Based on Multi-Layered Description Supporting Complex Multimedia Structural and Semantic Contents, in: *MMM'04, Proceedings of the 10th International Multimedia Modelling Conference* (Brisbane, Australia., Jan. 2004), pp. 33–37.

NEW METHODS FOR ENHANCING THE EFFECTIVENESS OF THE DUBLIN CORE METADATA STANDARD USING COMPLEX ENCODING SCHEMES

István Szakadát, László Lois, and Gábor Knapp*

1. PROBLEM OUTLINE

After several pilot projects performed in the previous years (e.g.[1, 2]), the Hungarian Government decided in 2002 to start a new initiative for enhancing the digitization process of Hungarian cultural contents within the framework called the National Digital Archive (NDA). The most important question of the NDA concept was how the digitized cultural content can be integrated at as high a level as possible. In order to integrate the contents of the different archives within the NDA framework, two basic standards were offered to the participants: the architecture described by the Open Archives Initiative (OAI) helped the content integration, and the Dublin Core Metadata Standard was proposed for the data exchange format. During the implementation process we were faced with some problems when we wanted to apply qualified DC schemas.

Since the first workshop of the Dublin Core Metadata Initiative (DCMI), several authorities (including ISO,[3] NISO[4] and CEN[5]) standardized the basic 15 elements of the schema.[6] The Dublin Core Metadata Element Set was originally agreed at a workshop convened by the Online Computer Library Centre (OCLC) and the National Centre for Supercomputing Applications (NCSA) at Dublin, Ohio in March 1995. This invitational workshop convened selected librarians, archivists, humanities scholars, geographers and standards makers, with the specific aim of achieving consensus on a list of metadata elements which could produce basic descriptions of data in a wide range of subject areas. Due to its simplicity, the schema is widely accepted and used as the metadata exchange format among different digital object domains. The Dublin Core metadata are usually expressed

* Budapest University of Technology and Economics, Budapest, Hungary, syi@axelero.hu, lois@hit.bme.hu, knapp@mokk.bme.hu,

using XML syntax. The basic elements are represented as tags, the values are as simple character strings, no attributes were defined. The resulting record can be easily parsed and validated by standard XML tools, however it remains human readable.

The Dublin Core as a whole is not so stable as some established metadata standards (like MARC) because it is a 'loose implementation model' which is evolving as a variety of people continue to refine it. The most important layer, the Elements of the Dublin Core standard are fixed. Interoperability was made possible by the stable and well defined list of 15 elements.

> "The simplicity of Dublin Core can be both a strength and a weakness. Simplicity lowers the cost of creating metadata and promotes interoperability. On the other hand, simplicity does not accommodate the semantic and functional richness supported by complex metadata schemes."[5]

For more sophisticated document description and search support the schema had to be extended. As a first step, DCMI qualifier attributes were introduced to refine the meaning of the basic terms and type attributes to specify the source or syntax of the value.[7] Qualified DC schemas appeared for a more precise description of different document types in several institutes (e.g. in the European Broadcasting Union).[8] Although the recommended set of qualifiers was published quite early, different needs resulted in different element and qualifier sets, and the semantic unity could only be sustained with difficulty, and the syntax of records became more and more confused.

The publication of DCMI Metadata Terms in 2003[9] can be considered as an attempt to make the Dublin Core based metadata schemas manageable, and support interoperability and semantic consistency, while retaining the simplicity and flexibility. The terms for various purposes were organized into a linear list. Application Profiles for the description of resource types using the terms defined by DCMI became consistent and they now support interoperability and novel terms can be easily defined. However, the association of elements and different extended term types (for refinement, encoding scheme or vocabulary term) is not complete and not always consistent.

Identifying the role and possibilities of current encoding schemes recommended for particular elements and extending their usage to all elements makes Dublin Core based description more expressive.

2. INCONSISTENCY PROBLEMS OF CURRENT SOLUTIONS

Generally, in order to characterize a resource, a set of statements can be used, each consists of a property name and a value. The Dublin Core Metadata Initiative defined a simple system that could be accepted by most repository owners for metadata exchange. In Dublin Core based schemas property names are represented by a set of Elements and Element Refinements (formerly called qualifiers), property values can be simple character strings or can be controlled by Encoding Schemes.

Elements and Element Refinements always used to express semantics, the properties of the resource. The Encoding Schemes have a more complex role. They are essentially used for syntax description, but sometimes they may have semantic role either. Although all terms are defined in a single namespace, it is worth to discuss the two basic types separately.

2.1. The First Interpretation of the Qualified DC: Qualifiers as Attributes

If we would like a more sophisticated description we need a structure, a hierarchy of properties. Adding new elements to the basic 15 items can be easily done in a technical sense, but if the new element is not widely accepted, interoperability remains restricted. Better solution is to retain basic elements and use terms that narrows the meaning of the related element. How it is possible technically?

The first attempt was to qualify DC elements with attributes. Syntactically it can be represented by an attribute called 'qualifier' within the XML tag.

```
<dc:creator qualifier="director">John Smith</dc:creator>
```

The advantage of this method is, that the record remains human readable, and the relation between the element and the qualifier appears in the metadata record itself. Using the qualifier attributes the application of the "dumb-down" principle (i.e. ignoring the qualifiers not known by the client) is easy. However, in this case the qualifiers are built in the schema and can not be changed separately. It is especially confusing in the world of audiovisual documents, since the number of contributor or creator roles is more then 300 (according to the European Broadcasting Union's list), which is changing rapidly and permanently. So there are qualifiers that change independently of the remaining part of the schema. The changes are usually initiated by different organisations than the board responsible for the schema itself. Another disadvantage of such a method is, that a repository owner with special needs either has to build his own schema (define, publish elements and harmonize them with standards etc.), or he has to use an already accepted, but not customized schema. That's why, the qualifier as an attribute is not recommended by the DC community any more.

2.2. The New Interpretation of the Qualified DC: DCMI Metadata Terms

The new recommendation is the set of the DCMI Metadata Terms, which consists of a well defined, standardized set of the following types of terms:

- original DC Elements (like Title, Creator, Relation etc.),
- other Elements (like Audience).
- Element Refinements (like Abstract, dateAccepted etc.),
- Encoding Schemes (like ISO3166, TGN etc.),
- Vocabulary Terms (like Image, MovingImage, Text, etc.).

Using this set anyone can assemble an Application Profile by selecting proper items. The elements, the commonly used element refinements, and the other types of terms are collected in a common namespace, so selecting items from this controlled list ensures, that all terms are defined and understandable for all partners in a unified way. Since both elements and element refinements form a single namespace, the relation between them can be defined at the namespace level, and need not appear in the metadata exchange records. The element refinement term inherits all properties from the element that it refines.

> "However, since Element Refinements are properties of a resource (like Elements), Element Refinements can alternatively be used in metadata records independently of the properties they refine. In DCMI practice, an Element Refinements refines just one parent DCMI property."[11]

For example, the element refinement 'DirectorOfPhotography' inherits all the properties of the 'Creator' element, but the 'DirectorOfPhotography' term has a narrower meaning then the 'Creator' term, so the former can be considered as a generic subordinate of the latter. The usage of the two terms is identical, the difference between them reveals only in the relation of the terms described in the namespace. It is an important shift comparing to the original qualified DC interpretation, when the qualifier of an element had different role in the DC based description. The difference between the two interpretations is shown by the changed naming convention:

> "A shift from the former view to the latter is reflected in the names assigned by the Usage Board to Element Refinements, with a move away from adjective-like names such as 'created' (approved in July 2000) towards noun-phrase-like names such as 'dateCopyrighted' (approved in July 2002)."[11]

The consequence of this approach is that an element refinement can have just one parent: the 'DirectorOfPhotography' element refinement can refine only the 'Creator' element, but it can not be used to refine the 'Contributor' element, or conversely. It means that the implicit organizing principle among the DCMI terms is the so called mono-hierarchy (this term is borrowed from librarian heritage).

However this simple solution can cause some confusion. If we have an element refinement term (for example the 'DirectorOfPhotography'), this term can be subordinated to the 'Creator' element if we would like to describe a photo as a resource, but the same element refinement term should belong to the 'Contributor' element in the case of a bibliographic metadata schema. The same term (with the same meaning) can have two superordinated terms. If one would use one general schema for these two cases, the element refinements can have two (or more) parents i.e. the refinement relation is poly-hierarchical. However, this option is unacceptable if we would like to maintain the validity of the dump-down principle, because in this case we could not know which parent elements ('Creator' or 'Contributor') should be used if the element refinement term ('DirectorOfPhotography') can not be parsed.

It has to be notified, that the basically two-level hierarchy in DCMI terms can be easily expanded to more levels. The DCMI element refinement term has a 'Refines' property that defines the generic relation to the parent element. There is no any practical example among the DCMI Terms, but theoretically it is conceivable, that an element refinement term refines another element refinement term. In this case we can have three or maybe more levels of a hierarchical structure.

Using DCMI terms applications made for the handling of pure Dublin Core metadata can be adapted easily, because the exchange format is unchanged (e.g. eprints[14]).

Comparing the two approach:

Property	DC Qualifiers	DCMI Terms
Adding new refinement element	The whole schema has to be changed (versioned)	Only the new element refinement has to be defined and added
Metadata record readability	The two-level hierarchy appears in the record itself	The hierarchy appears only in the namespace
Element refinement restrictions	Element refinement with the same name can refine multiple elements	Element refinements has to be unique for all the schema and restricts only one element
Special requirements	New schema has to be created for different document types	The application profile for different resource types can be assembled using the DCMI terms
Expanding hierarchy	Requires well defined use of attributes	Easy, but full understanding of metadata exchange records assumes the knowledge of the application profile
Using "dumb-down" principle	Easy (only the exchange record is required)	Requires information from the DCMI term namespace
Application support	Special application has to be developed for qualified search	Applications developed for pure DC can be used

2.3. The Two Basic Types of Encoding Schemes

The next important DCMI metadata term types are the encoding schemes which are used to control the values of the properties of the resources. An encoding scheme can define the parsing rules, the interpretation of the value, or even the semantics. Encoding schemes usually appear as the value of the Type attribute in the original qualified DC schemas. The DCMI defines only two types of encoding schemes "officially" (the first two items of the following list); practically, however, there are three:

- Vocabulary encoding schemes,
- Syntax encoding schemes,
- Semantically structured values with optional syntax restrictions.

Looking at the list above, it can be seen that the last version promises the most flexible description (and the most complicated as well).

'Vocabulary encoding schemes' are used to indicate the simple fact, that the property value is derived from a controlled vocabulary. Using terms from the same origin is essential to ensure integration and interoperability (e.g. the Library of Congress or Dewey Decimal Classification). Referencing to another scheme (beyond DCMI terms), however, makes the development of the collection management and search application more complicated. Other vocabulary encoding schemes satisfying special requirements can be easily added.

(It is worth mentioning, that one of the DCMI vocabulary encoding schemes is principally different from the others. The Getty Thesaurus of Geographic Names is a system of proper names, while all others are composed of common nouns.)

The basic ‘Syntax encoding schemes’ can help metadata harvesting clients to interpret and validate values. For example, the W3C-DTF scheme defines date formats, supported ISO and RFC specification the country codes, or URI syntax.

2.4. Vocabulary Terms

Among the DCMI terms there are some vocabulary terms, which are all elements of the DCMIType encoding scheme. These terms can be the values of the ‘Type’ DC element, and this simple fact clearly shows that these terms have different role in the DC description: a vocabulary term can be the value of a property (an element or an element refinement). The class of the elements of the DCMIType encoding scheme is the only one example among the DCMI Terms for the ‘Vocabulary Terms’, but the functionalities of the DCMIType encoding scheme and the other vocabulary encoding schemes (like LCSH, TGN) are the same: providing uniform semantics for all terms what we would like to describe the values of the DC properties with. From this point of view we can say, that semantically vocabulary terms have the similar role as ‘Element refinements’ (to provide more exactly, more refined description), however their grammatical role differs.

Although ‘Vocabulary Terms’ appear as identical parts of the term system, their functionality significantly differs from the ‘Element’ and the ‘Element Refinement’ terms. The narrowing of semantics in this case is obtained by restricting the values of a property for a controlled list. All vocabulary terms currently registered among DCMI terms are related exclusively to the ‘Type’ element, however all controlled lists could be treated this way. The DCMI give preference to the ‘Type’ element over other elements (e.g. ‘Subject’). The Vocabulary terms fit into a semantic hierarchy (using the ‘narrower’ and ‘broader’ properties of the terms), and can be expanded easily if required. For example we can say, that the ‘type’ (‘Element’) of a resource is an ‘Image’ (‘Vocabulary term’), or a ‘Moving Image’ (‘Vocabulary term’ with narrower semantics as ‘Image’), or a ‘Silent Moving Image’ (‘Vocabulary term’ with narrower semantics as ‘Moving Image’). The latter term is not included into the DCMIType encoding scheme, but theoretically we can easily add it to this class.

2.5. The Third Type of Encoding Scheme

In a previous section we already mentioned, that in spite of the belief of the DC community we can differentiate three types of encoding schemes. The interesting exceptions among encoding schemes are the Box and Period terms which are intended to specify the syntax of the Coverage and the Date Elements, and the Spatial and Temporal Element Refinements. Conventional syntax encoding schemes are restricted to formal syntax, thus no more information is supplied about the document. These exceptions however provide more semantics as well.

Let us see the Period term![10] Using the structured element value, it is possible to determine a period without definite Start or End point, or a period with uncertain bounds characterized by a name (e.g. ‘Middle ages’).

```
<dcterms:date xsi:type="dcterms: Period"> Start=1991; End=2000;
Scheme="W3CDTF"</dcterms:date>
```

This – DCMI supported – encoding scheme is called DCSV (Dublin Core Structured Value) scheme[8] where punctuation characters are used in the element value to describe a structured value as follows:

- equals-signs (=) separate plain-text labels of structured value-components from the values themselves,
- semi-colons (;) separate (optionally labelled) value-components within a list,
- dots (.) indicate hierarchical structure in labels, if required.

If we use the Period encoding scheme to describe the value of the Date property of a resource (as in the above example), we can know more about the resource. We could express the same knowledge if we would define two new Element Refinements (dateStart and dateEnd) within a NDA namespace recording the following two DC-sentences:

```
<NDA:dateStart> 1991 </NDA:dateStart>
<NDA:dateEnd> 2000 </NDA:dateEnd>
```

These two DC-records and the previous record expressed with the help of the Period Encoding Scheme are semantically identical. The syntax of the Period term allows us to refine the Date property of the resource within the structure of the value of this property. It means that the Period term has not only a syntactic controlling power, but it has a semantic refinement ability as well.

The DCMI-DCSV recommendation intends to define a compact human-readable data-structuring method avoiding certain punctuation characters which can cause difficulties in some encoding environments. The DCMI-DCSV notation should normally be used only when no other suitable scheme is available. Note that some other more simple methods exist i.e. the comma-separated value (CSV) and tab separated value (TSV) schemes but these schemes are not human interpretable since the order of the values is fixed but not indicated textually. The XML provides a general solution using tags contained within angle brackets (<, >) to indicate the structure hence this approach covers the DCMI-DCSV but the human-readability is worse.

The speciality the DCMI-DCSV scheme is that it has a semantic message besides syntax definition as well. Note, that a syntax encoding form is embedded in the data structure. This encoding formalism is not familiar with the original DC concepts. However it is accepted, because closely linked semantic terms can not be expressed using unstructured values.

3. COMPLEX ENCODING SCHEMES FOR CONSISTENT DESCRIPTION

The XML attributes are commonly used to provide information that is not relevant to the value i.e. the used encoding scheme, the unique identifier within a global system etc. We show that the XML attribute is only slightly capable of element refinements. To demonstrate this we show the possibilities of the refinement encoding on a simple example.

Let us imagine that we have to describe a Contributor property of a resource (its value let it be ‘Rex Gleam’) and we would like to refine the Contributor element itself, i.e. we would like to express Rex Gleam’s role subordinated to Contributor (let it be ‘lighting

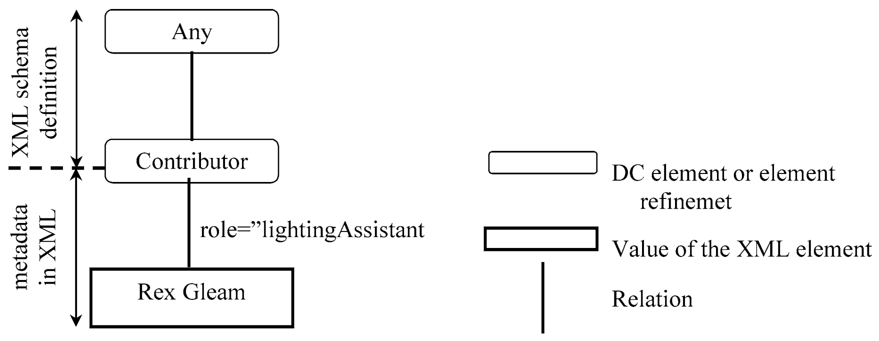


Figure 1. Refined description of a property by using XML attributes.

assistant'). Assume that one has defined a 'qualifier' attribute for all DC elements within the NDA namespace. By using this attribute to indicate the 'role' refinement we have to introduce one more attribute to describe the content of the role since an attribute can not contain structure or multiple values (see Figure 1):

```
<NDA:contributor NDA:qualifier="role"
NDA:role="lightingAssistant">Rex Gleam</NDA:contributor>
```

Alternatively, we get the same result by avoiding the qualifier attribute:

```
<NDA:contributor NDA:role="lightingAssistant">
Rex Gleam</NDA:contributor>
```

On the other hand, the original schema definition files of the dublincore.org (dc.xsd and dcterms.xsd) do not enable us to extend the schema of the DC elements with attributes not specified by them hence only the common XML attributes are useable i.e. xsi:type or xml:lang attributes. Hence the example above assumes that the original DC XML namespace is overridden with a new XML schema. The name 'lightingAssistant' could be a name in a namespace, but the attribute value validation of this is difficult in XML.

When we avoid using the attributes, we have two other solutions to encode the person and his/her role:

- *new term approach*: a new term for each role is introduced and the value of the term describes the person,
- *structured value (or mixed encoding scheme) approach*: a structure within the XML element value is used where both the person and the role is described (see Figure 2.)

An example for the first approach could be any *dcterm* element, but to solve our current problem the result of the first approach is the following:

```
<NDA:lightingAssistantContributor>Rex Gleam
</NDA:lightingAssistantContributor>
```

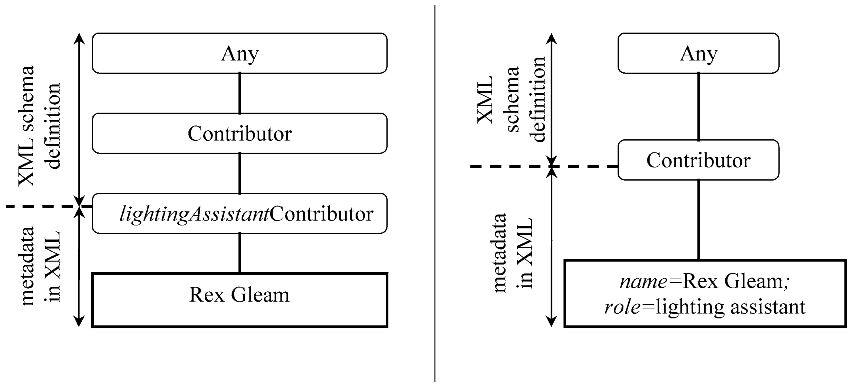


Figure 2. Comparing refinement using a new DC Term (left) and using structured value (right).

The second approach is based on a new XML element data type similarly to the Period encoding scheme. The DCMI-DCSV version is the following:

```
<NDA:Contributor xsi:type="NDA:personDCSVType">
name=Rex Gleam; role=lightingAssistant </NDA:Contributor>
```

while an XML friendly version of the above DCMI-DCSV scheme can also be used:

```
<NDA:Contributor xsi:type="NDA:personComplexType">
<name>Rex Gleam</name>
<role>lightingAssistant</role>
</NDA:Contributor>
```

Since the original *dc* and *dcterms* XML namespaces do not allow to extend the types within the namespace we have to derive a new Contributor element within a new namespace called NDA. Of course, we can "refine" the Creator element in the same way, and we can handle the problem mentioned earlier, that on the one hand a role, i.e. a photographer, can belong to the Creator element (in the case of a photo schema), while on the other it can refine the Contributor element (within a bibliographic schema). If we would like to resolve this problem with refinement elements, we would be faced with the problem of poly-hierarchy, which is not allowed within the DCMI, but if we use the offered DCSV-like solution, we can semantically refine both the Creator and the Contributor elements (even the Publisher one, also), and we do not harm the dump-down principle as well.

Using the structured value approach the role could be taken from a simple controlled list or from a thesaurus (i.e. from a new Vocabulary Encoding Scheme), and the – semantic – poly-hierarchy is also allowed, similarly to the avoided XML attribute-based approach. Furthermore, the *structured value approach* enables the schema designer to extend the structure with more sub-elements and within the XML based version the 'role' sub-element could be imported from a namespace:

```
<NDA:Contributor xsi:type="NDA:personComplexType">
<name>Rex Gleam</name>
```

```
<agent>Person</agent>
<agentID xsi:type="NDA:agentNameSpace">02x3h54f5f2</agentID>
<role xsi:type="NDA:roleType">lightingAssistant</role>
</NDA:Contributor>
```

The DCMI-DCSV version of the last example is a bit more complicated:

```
<NDA:Contributor xsi:type="NDA:personDCSVType">
name=Rex Gleam;
agent=Person;agentID=02x3h54f5f2; agentNS="NDA:agentNameSpace";
role=lightingAssistant; roleNS="NDA:roleType";
</NDA:Contributor>
```

The XML-based version could be more easily validated with an XML based system and the validation of the DCMI-DCSV version could be validated with an application outside the XML domain. Both versions are human-readable, but in this aspect the DCMI-DCSV scheme is the favourable.

Extending the DC refinement formalism to the other elements, the problem of the several possible subtypes (refinements) of the Contributor (Creator and Publisher) element can be handled. Defining a semantic-syntactic structured Role scheme, combined with an embedded RoleType Vocabulary Encoding Scheme containing the list of possible roles, and namespace containing the enumerated type of the proper names of agents, makes possible to accurately express the role-agent relation without leaving the domain of DCMI concepts.

4. CONCLUSIONS

The mission of the Hungarian National Digital Archive (NDA) is the enforcement of the digitization of the documents of cultural heritage. Digitization does not only mean the production of the digital representation of physical objects. The main purpose is to make accessible cultural values, and to achieve as high a level of integration among the different archives as possible. Based on some offered standards and solutions, the metadata of the different archives can be freely harvested which makes simultaneous searches possible.

However, at this point an important problem arises from the differences between the requirements of the collection management of the archives and the searching expectations or intentions of the users, or of the searching service providers. The generalization of the structured DCMI encoding scheme, like the term Period, will make it possible to satisfy both the professional's requirement of sophisticated description and the support of user search as well.

The choice of the best solution depends on the application. First, the solution must conform the DCMI recommendations as closely as possible hence we do not recommend using the XML attributes for DC element refinements. When one does not have to introduce many element refinements, and the mono-hierarchy is held, we recommend introducing them as a direct descendant of the DC elements like the so called *dcterms*. When the new refinements require a poly-hierarchical relation to the DC elements, like the set of the role refinements, we recommend using a child element structure within the appropriate descendant of the DC or *dcterms* element. The structure could be based on either the

DCMI-DCSV approach which can be validated against a syntax rule, or on the XML-based approach where not only the syntax but also the content of a sub-element can be validated. The XML-based version also enables us to use external namespace referencing, controlled lists etc., but it is important to underline that the overall metadata structure should not be too large or complicated since one must create and maintain the metadata and the human-readability must be held as much as possible.

REFERENCES

1. G. Knapp, G. Magyar, and G. Németh, Building an Open Document Management System with Components for Trust (Twelfth International Conference on Information Systems Development, Melbourne, 2003).
2. G. Magyar, G. Knapp, and I. Szakadát, Information Systems Development in the e-Age, conference paper (Tenth International Conference on Information Systems Development, London, 2001).
3. Dublin Core Metadata Element Set – Reference Description – Version 1.1 (CEN, 2000).
4. The Dublin Core Metadata Element Set, ANSI/NISO Z39.85-2001 (ANSI, 2001).
5. Information and documentation – The Dublin Core metadata element set (ISO, 2003); http://www.niso.org/international/SC4/n515.pdf.
6. Dublin Core Metadata Element Set, Version 1.1: Reference Description (2003); http://dublincore.org/documents/2003/06/02/dces/.
7. Dublin Core Qualifiers (2000); http://dublincore.org/documents/2000/07/11/dcmes-qualifiers/.
8. EBU Core Metadata Set for Radio Archives, Tech 3293 (European Broadcasting Union, 2001).
9. DCMI Metadata Terms, DCMI Usage Board (2001); http://dublincore.org/documents/2003/11/19/dcmi-terms.
10. DCMI DCSV: A syntax for writing a list of labelled values in a text string; http://dublincore.org/documents/dcmi-dcsv/.
11. DCMI Grammatical Principles (2003); http://dublincore.org/usage/documents/principles.
12. S. Cox, DCMI Period Encoding Scheme, specification of the limits of a time interval, and methods for encoding this in a text string (2000); http://dublincore.org/usage/documents/2000/7/11/dcm-period/index.shtml.
13. Powell, M. Nilsson, A. Naeve, P. Johnston, DCMI Abstract Model (2004); http://www.ukoln.ac.uk/metadata/dcmi/abstract-model.
14. http://www.eprints.org/.

FROM ANALOG INFORMATION TO DIGITAL DATABASES – DOES IT KEEP EVERYTHING INTACT?

Hagai Kirshner and Moshe Porat*

Abstract Information systems encapsulate digital data although the origin of many sources of information is analog or continuous. Digital signal representation has been widely used since Shannon formulated his sampling theorem. Still, several questions regarding digital information processing remain unsolved. One of the most relevant up-to-date goals is data storage and mining. The purpose of this work is to analyze the relation between continuous signals, or data sources, and their digitized representation. We concentrate on the operation widely used to compare and find similarities in vector representation, the inner-product. Applying the sampling scheme to continuous information sources that do not satisfy Shannon conditions (non band-limited), but with higher enough sampling rate, is assumed to yield only small approximation errors. In this work it is shown, however, that this assumption should be made with much prudence. In some cases, the result is likely to differ from that of the original continuous signals. We provide an analytic estimation to this error of digitization, and several applications are considered, including medical records. Our results provide a quantitative tool for calculating sampling errors, thus affording a useful means in evaluating the ongoing process worldwide of digitizing analog information sources.

1. INTRODUCTION

Information systems contain mainly digital data although the origin of many sources of information is analog or continuous. This is the case in medical applications for example, where continuous signals - physical or biological - are digitized by sampling procedures.[1] As a result, one might seek for suitable digitization tools and processes, such as storage, transmission, retrieval or classification. Digital signal representation has been widely used since Shannon formulated his sampling theorem. In its foundation, the sampling theorem reveals an explicit relation between a continuous signal and its samples. His major contri-

* Department of Electronics, Computers and Communication, Faculty of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel. kirshner@tx.technion.ac.il, mp@ee.technion.ac.il.

bution in the field of information representation is the basis for many present achievements in information technology. Still, several questions regarding digital information processing remain unsolved. One of the most relevant up-to-date goals is data storage and mining. With the ever-increasing amount of stored data, one needs reliable tools to compare and retrieve relevant data from available databases.

The purpose of this work is to characterize the relation between continuous signals, or data sources, and their digitized representation. We concentrate on the operation widely used to compare and find similarities in vector representation, the inner-product, denoted here as $\langle a, b\rangle$, where a and b are two vectors or signals (including images) in the database. This measure of inner-product, which acquires high values if two vectors are similar, and a zero value when they are totally different or orthogonal, is a useful tool in comparing images, speech signals, medical records, and the like. In this paper we will refer to the most common definition of inner-product: for two continuous data sources $f(t)$, $g(t)$, the inner product is defined as $\langle f(t), g(t)\rangle = \int f(t)g(t)dt$, and for two digitized sources $a[n]$, $b[n]$, the inner product is defined as $\langle a[n], b[n]\rangle = \sum a[n]b[n]$.

For example, consider three images containing two portraits and a tree (Figure 1). Examination of the ensued inner-product values enables one to classify them properly, similarly to what a human being would agree with:

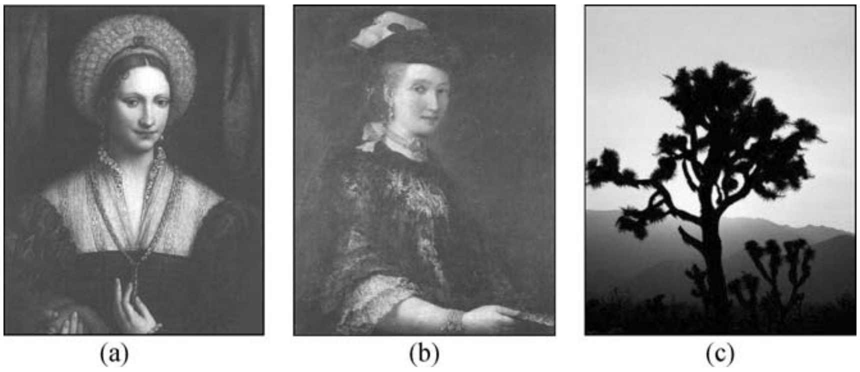


Figure 1. Two portraits (a) and (b), and a landscape image (c). As expected, the inner-product provides a measure of similarity, as judged by the human observer: $\langle a, b\rangle = 0.80$, $\langle a, c\rangle = 0.55$, $\langle b, c\rangle = 0.62$. These numbers are normalized, such that $\langle a, a\rangle = \langle b, b\rangle = \langle c, c\rangle = 1$.

The inner-product serves additional purposes, such as signal representation. In this context, this work can also contribute to situations in which sampled information is used for calculating representation coefficients of a signal with respect to a given set of basis functions.

2. THE PROBLEM

Practically, the inner-product measures the similarity of two data sources (continuous or digitized), thus giving rise to applications of data retrieval, data compression and pattern recognition.[2, 3] The question raised in this work is whether the inner-product as performed

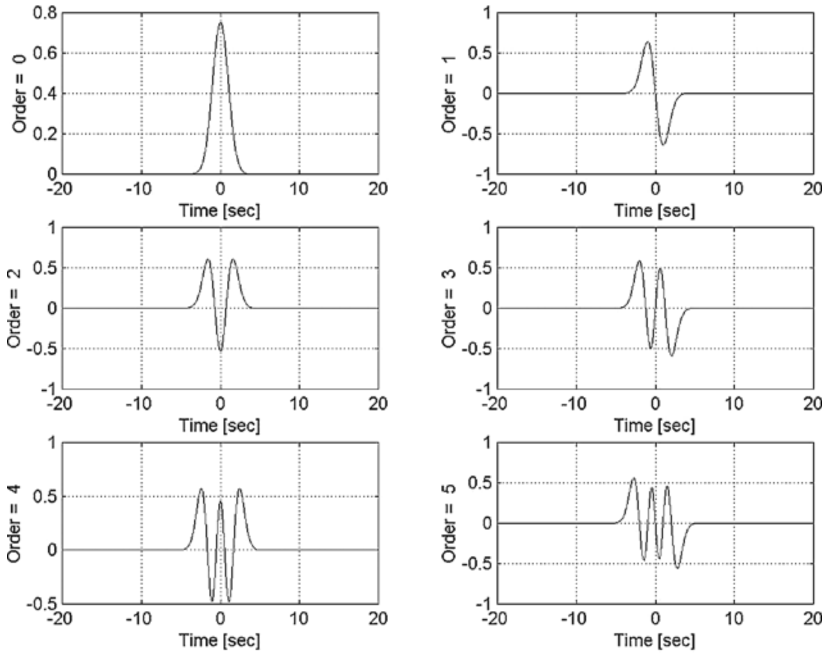


Figure 2. Several orthogonal Hermite functions.

in the analog or continuous domain is equal, or at least similar, to the result one may get in the digital domain of the same two entities, assuming that the digital signals are the sampled version of the continuous signals. The following two examples provide some insight into the problem.

2.1. Example 1: Digitizing MEG Signals

The Hermite functions constitute a useful basis for the vector space of continuous-time function known as L_2 (Figure 2). These functions are most suitable for describing continuous data sources that have finite time support. Such signals are MEG (myoelectrograph), ECG (electrocardiogram) and EEG (Electroencephalogram) signals, widely used in medical examination, storage and analysis. This set of functions is given by:

$$h_n(t) = \frac{H_n(t) \cdot e^{-t^2/2}}{\pi^{1/4} 2^{n/2} (n!)^{1/2}}$$

where $H_n(t)$ are the Hermite polynomials, defined by a recursive formula:

$$\begin{cases} g_t \cdot H_n(t) = \frac{1}{2} \cdot H_{n+1}(t) + n \cdot H_{n-1}(t) \\ H_0(t) = 1,\ H_1(t) = -2t,\ H_2(t) = 4t^2 - 2,\ H_3(t) = -8t^3 + 12t, \\ \quad H_4(t) = 16t^4 - 48t^2 + 12 \ldots \end{cases}$$

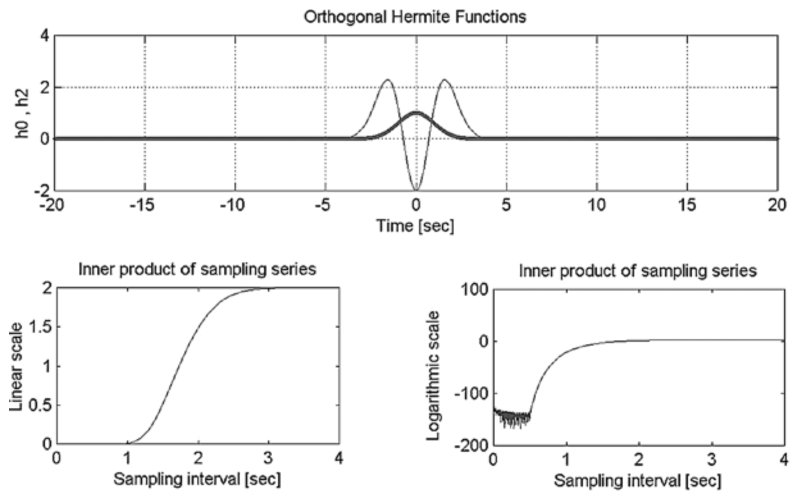


Figure 3. The inner product of two Hermite orthogonal functions $h_0(t), h_2(t)$ (top) in the continuous-time space is calculated and shown in the digital domain. Although the inner-product is supposedly zero, it is shown that in a digital database this inner-product is dependent on the sampling rate. Shown is the result as a function of the sampling interval in linear (bottom, left) and logarithmic (bottom, right) coordinates.

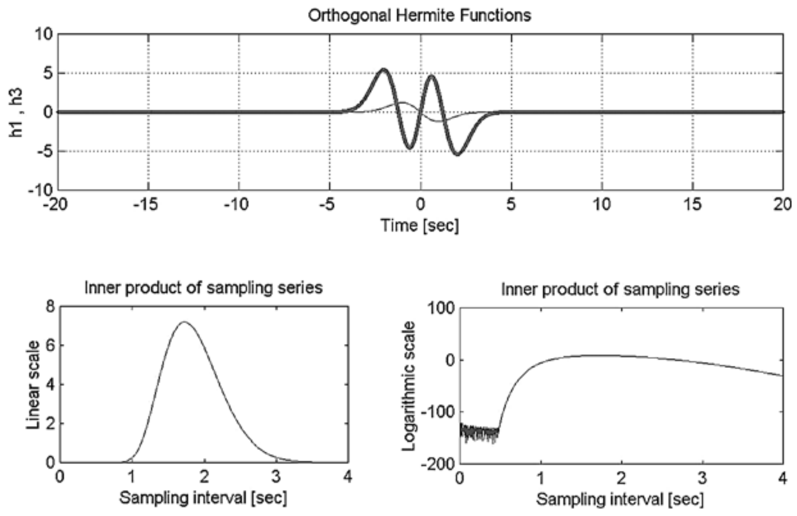
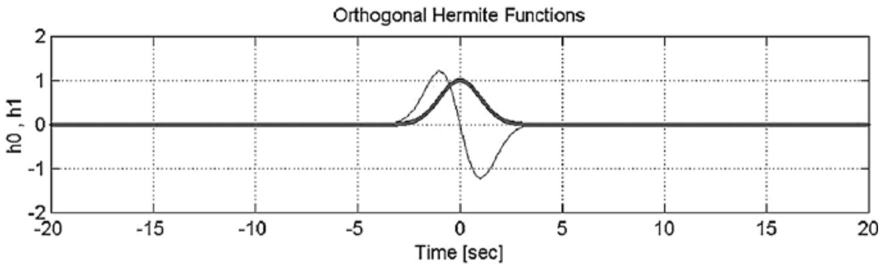


Figure 4. Similar to Figure 3, for two different orthogonal functions $h_1(t), h_3(t)$.

The inner-product of any two Hermite functions is zero, making them a set of orthogonal functions. Their sampled sequences, however, are not necessarily orthogonal, making these basic functions a suitable example to analyze and consider. Figures 3–5 illustrate the variation of the ensued (digital) inner-product as a function of the sampling interval. Some numerical results are also shown in Table 1.



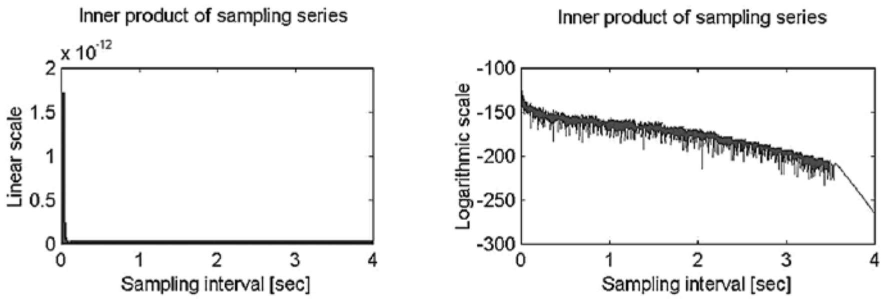


Figure 5. Similar to Figure 3, for the orthogonal functions $h_0(t), h_1(t)$ of Figure 2. This figure exhibits orthogonality of the sampled sequences regardless of the sampling interval. This is due to the symmetry and anti-symmetry properties of the sampled functions.

Table 1. Comparison of analog- vs. digital inner product at several sampling intervals

Inner Product	Analog	Digital				
		$T = 0.1$	$T = 0.5$	$T = 0.8$	$T = 0.9$	$T = 1$
$\langle h_0, h_2 \rangle$	0	5×10^{-16}	7.9×10^{-16}	9×10^{-6}	2×10^{-4}	0.0014
$\langle h_1, h_3 \rangle$	0	5×10^{-17}	2×10^{-14}	1×10^{-4}	0.0015	0.01
$\langle h_0, h_1 \rangle$	0	1×10^{-16}	1×10^{-13}	5×10^{-17}	5×10^{-17}	7×10^{-17}

Now, consider a simulated MEG signal $g(t)$, composed here of five Hermite functions:[1] $g(t) = 0.03h_0(t) - 1.076h_1(t) - 0.012h_2(t) + 0.34h_3(t) - 0.089h_4(t) - 0.122h_5(t)$. When incorporating this signal into a medical database, digitization is a necessary step. The most convenient way, thus common, is simply record $g(t)$ at discrete, equally spaced set of points. Upon acquisition, one may ask: "how much is the first Hermite function $h_0(t)$ dominant within this signal?". A natural way of answering this question is by applying vector-like inner product calculation. However, it can be shown that the ensued value is dependent upon the sampling interval T. For example, Figure 6 describes the calculated expansion coefficient of $g(t)$ with respect to $h_0(t)$ at various sampling intervals.

Since the correct value is known in this example (should be 0.03) it can be concluded that the approximation error is as depicted in Figure 7. Some numerical results are shown in Table 2.

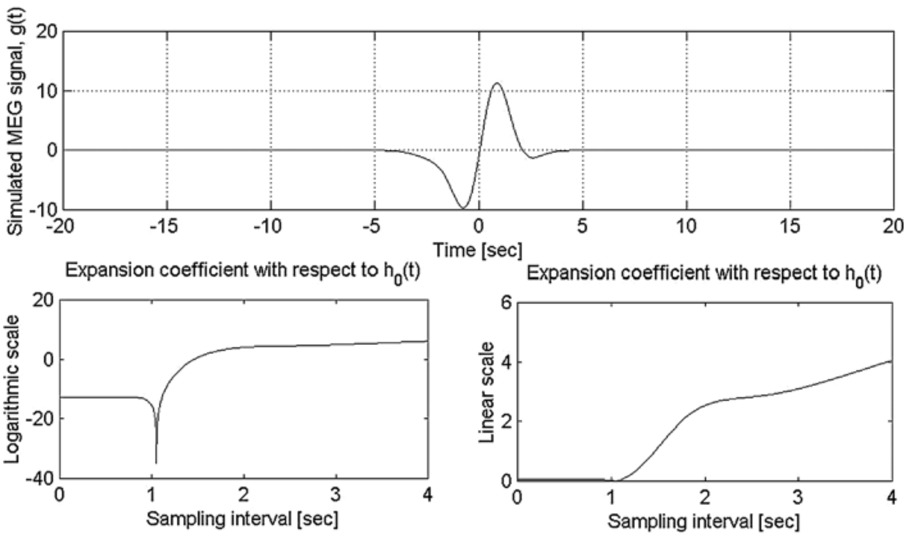


Figure 6. MEG signal (top) and its estimated component $h_0(t)$ as a function of its sampling rate, using logarithmic (bottom left) and linear (bottom right) coordinate systems.

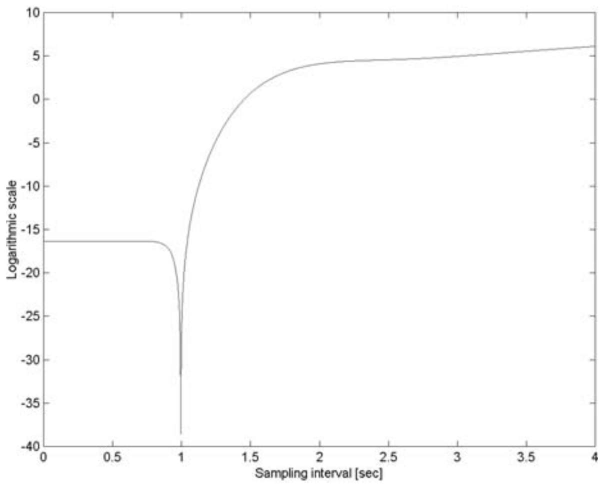


Figure 7. E stimation error of simulated MEG signal with respect to $h_0(t)$ at various sampling intervals.

Table 2. Comparison of analog- vs. digital inner product at several sampling intervals

Inner Product	Analog	Digital			
		$T = 0.1$	$T = 0.5$	$T = 1$	$T = 1.088$
$\langle g, h_0 \rangle$	0.03	0.0532	0.0532	0.0278	0.0304

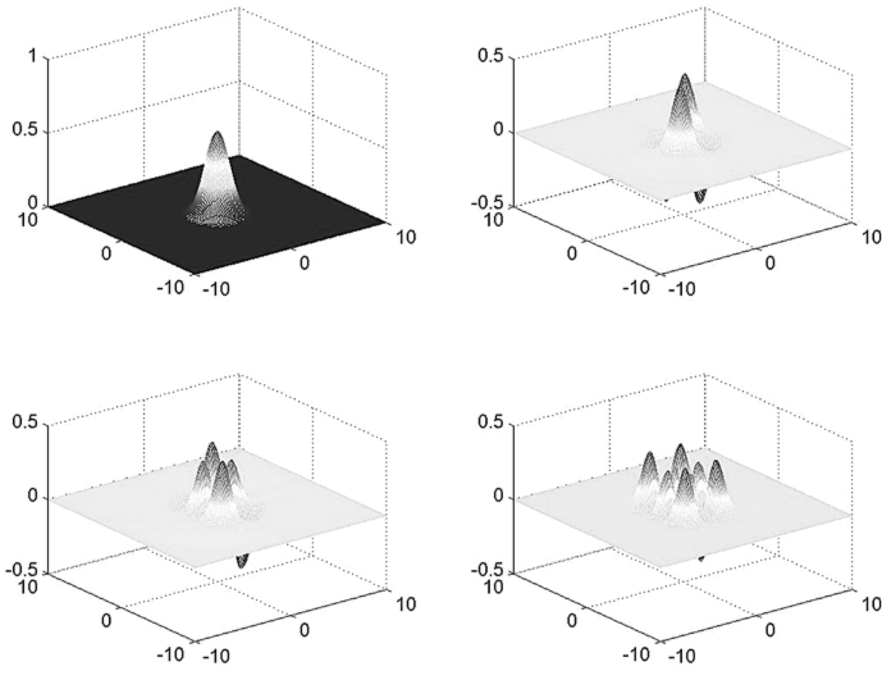


Figure 8. Two-dimensional (orthogonal) Hermite functions of order (0,0) (1,1) (3,1) & (4,2) used in analog image representation.

2.2. Example 2: Digitizing an Image

Suppose that one needs to acquire a two-dimensional data source such as a medical image. Again, digitization is a necessary step, and is done by recording the image at equally spaced grid of sampling points. Now, consider a simple image $g(x, y)$ composed of four two-dimensional functions:

$$g(x, y) = 0.5h_{0,0}(x, y) + h_{1,1}(x, y) - 0.8h_{3,1}(x, y) + 1.5h_{4,2}(x, y)$$

where $h_{m,n}(x, y)$ are two-dimensional Hermite functions (Figure 8).

Assuming that the relative contribution of $h_{4,2}(x, y)$ to the acquired image is needed, the calculated coefficient based on sampled values is again dependent on the sampling interval as shown in Figure 9. Furthermore, here, in the two-dimensional case, it differs from the known value of 1.5 even when the sampling resolution is relatively high (Figure 9 and Table 3).

In both examples, it is of major interest to predict these errors beforehand, i.e., prior to sampling, where essential information may be lost. Shannon's sampling theorem is of limited use here since most signals, including medical records, do not satisfy Shannon's conditions with regard to being band-limited, like in fact *all* finite signals widely used in databases and information systems.

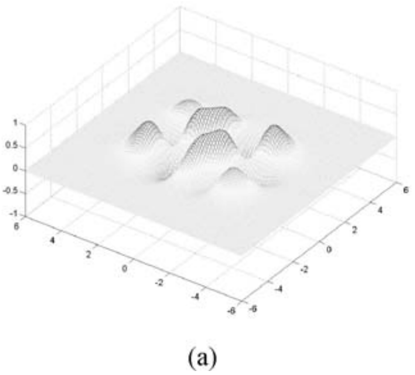
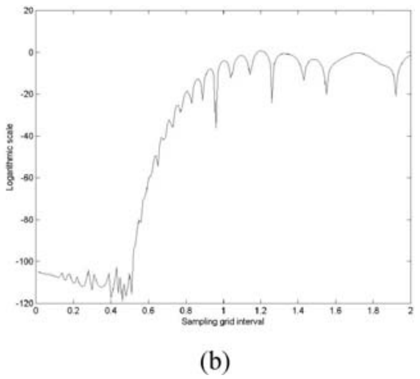



Figure 9. A test image (a) and the expansion coefficient error (b) with respect to at various sampling grid resolutions.

Table 3. Comparison of analog- vs. digital inner product at several sampling intervals

Inner Product	Analog	Digital				
		$T = 0.1$	$T = 0.5$	$T = 0.8$	$T = 0.9$	$T = 1$
$\langle g, h_{4,2} \rangle$	1.5	1.49	1.5	1.48	1.44	1.88

3. THE PROPOSED SOLUTION

Having considered these differences between digital and analog calculations of inner-product, we introduce in this paper two theorems that provide an analytic prediction of the result, based on typical characteristics of the signals involved. The first theorem deals with the less prevailing band-limited signals. Although not very common, this is the basis for the second, more practical theorem:

Theorem 1. Let $f(t), g(t)$ be two band-limited data sources. Then, digitizing them does not affect their inner-product if multiplied by the sampling interval T. i.e. the relation $\langle f(t), g(t)\rangle = T \cdot \langle f[n], f[n]\rangle$ holds for any T that satisfies the sampling theorem.

The more advanced Theorem deals with the more practical situation of general continuous data sources:

Theorem 2. Let $f(t), g(t)$ be any two continuous-time data sources. Then, digitizing them does affect their inner-product value. i.e. $\langle f(t), g(t)\rangle \neq T \cdot \langle f[n], g[n]\rangle$. Given their spectral transform[4] $\hat{f}(\nu)$ and $\hat{g}(\nu)$ of the continuous-time signals $f(t)$, $g(t)$, respectively, the digital inner-product is calculated as follows:

$$\langle f[n], g[n]\rangle = \frac{1}{T} \cdot \sum_{n=-\infty}^{\infty} \int \hat{f}(\nu) \cdot \overline{\hat{g}(n/T + \nu)} d\nu,$$

where an over-bar denotes complex conjugate. The proof for this theorem is given in the appendix.

The use of these theorems, in particular Theorem 2, can be helpful in evaluating the required sampling rate of analog signals. The only required information is the spectral Fourier statistics of the digitized signals, which is readily available in most cases of continuous-time data sources. It can easily shown that Theorem 2 can predict all the experimental results encountered in the above two examples of the previous section.

4. CONCLUSIONS

In this work, we have introduced and analyzed a source of errors in the commonly performed process of digitization, where analog information is transformed into digital records to be used in information systems. The situation was analytically investigated using the tool of inner-product, which is useful in pattern recognition, data retrieval, storage, compression and classification applications. It has been shown that the inner-product is not digitization-invariant. Although frequently used, applying vector-like inner product calculation is erroneous in many cases.

Using our proposed theorem, we provide an analytic estimation of this digitization error, using a general sampling scheme, and derive our result within the spectral domain. Several applications have been considered, including the case of medical records. Since our results provide a quantitative tool for calculating sampling errors, they afford a useful means in evaluating the ongoing process worldwide of digitizing analog information sources.

APPENDIX

Definition. Let $g(t) \in L_2$. Given any $\tilde{\varphi}(t) \in L_2$, *the sampling functional by interval T* with respect to $\tilde{\varphi}(t)$ would be the series $g[n] = \langle g(t), (1/T)\tilde{\varphi}\ (t/T - n)\rangle$ $n = 0, \pm 1, \pm 2 \ldots$.

Lemma 1. It is guaranteed that $\{g[n]\}_{n=-\infty}^{\infty} \in l_2$. However, in order to include $\tilde{\varphi}(t) = \delta(t) \notin L_2$ within this sampling scheme, another constraint must be imposed on the function to be sampled.

Lemma 2. Let $g(t), g'(t) \in L_2$. If $\bar{\tilde{\varphi}}(\nu) < \infty$ then$\{g[n]\}_{n=-\infty}^{\infty} \in l_2$. For a proof, see [5, Appendix C].

Definition. Let $\varphi(t) \in L_2$ such that $\{\varphi(t-n)\}_{n=-\infty}^{\infty}$ fulfills Riesz condition. Given the space $V_T(\varphi) = Span\{\varphi(t/T - n)\}_{n=-\infty}^{\infty}$, the function $\{Q_T g\}(t) = \sum_{n=-\infty}^{\infty} g[n] \cdot \varphi(t/T - n)$ would be *the approximation of $g(t)$ within $V_T(\varphi)$ by the sampling functional* $\{g[n]\}_{n=-\infty}^{\infty}$.

Definition. The autocorrelation series of $\varphi(t)$ is given by $a_\varphi[n] = \langle\varphi(t-n),\varphi(t)\rangle = \langle\hat{\varphi}(\nu)e^{-2\pi\iota n\nu}, \hat{\varphi}(\nu)\rangle = \int_{-\infty}^{\infty} \hat{\varphi}(\nu)\overline{\hat{\varphi}(\nu)}e^{-2\pi\iota n\nu}d\nu$. Its Fourier transform is then,

$$\tilde{A}_\varphi(\theta) \quad = \quad \sum_{n=-\infty}^{\infty} a_\varphi[n]\, e^{-\iota\theta n} = \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{\varphi}(\nu)\,\overline{\hat{\varphi}(\nu)}e^{-2\pi\iota n\nu} d\nu e^{-\iota\theta n}$$

$$
\begin{aligned}
&= \int_{-\infty}^{\infty} \hat{\varphi}(\nu)\,\overline{\hat{\varphi}(\nu)} \cdot \left[\sum_{n=-\infty}^{\infty} e^{-2\pi\iota n\nu} e^{-\iota\theta n}\right] d\nu \\
&= \int_{-\infty}^{\infty} \left|\hat{\varphi}(\nu)\right|^2 \cdot \left[\sum_{n=-\infty}^{\infty} \delta\left(\nu + \frac{\theta}{2\pi} - n\right)\right] d\nu \\
&= \sum_{n=-\infty}^{\infty} \left|\hat{\varphi}\left(n - \frac{\theta}{2\pi}\right)\right|^2 = \sum_{n=-\infty}^{\infty} \left|\hat{\varphi}\left(-\left(\frac{\theta}{2\pi} - n\right)\right)\right|^2 \\
&= \sum_{n=-\infty}^{\infty} \left|\hat{\varphi}\left(-\left(\frac{\theta}{2\pi} + n\right)\right)\right|^2
\end{aligned}
$$

Example. Let $g(t), g'(t) \in L_2$, and the series $g[n]$ its sampling functional by interval T_0 with respect to $\tilde{\varphi}(t) = \delta(t)$. The approximation of $g(t)$ within the space of finite-energy, bandlimited functions $[-\pi/T_1, \pi/T_1]$ is

$$\{Q_T g\}(t) = \sum_{n=-\infty}^{\infty} g(nT_0) \sin c(t/T_1 - n).$$

Exact reconstruction occurs when $T_0 = T_1$. If $T_0 > T_1$, aliasing occurs. One should also note that the autocorrelation series of $\varphi(t) = \sin c(t)$ is $a[n] = T\delta[n](\tilde{A}_\varphi(\nu) = T)$.

Lemma 3. Given non-complex analysis ($\tilde{\varphi}$) and synthesis (φ) functions,

$$\langle Q_T g, Q_T h\rangle = \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{g}(\nu)\,\overline{\hat{h}\left(\frac{n}{T} + \nu\right)} \cdot \overline{\hat{\tilde{\varphi}}(\nu T)\,\hat{\tilde{\varphi}}(n - \nu T)} \cdot \overline{\hat{A}_\varphi(2\pi\nu T)}\,d\nu.$$

Proof.

$$
\begin{aligned}
&\{Q_T g\}(t) \\
&= \frac{1}{T}\sum_{n=-\infty}^{\infty}\left[\int_{-\infty}^{\infty} g(\tau)\cdot\overline{\tilde{\varphi}\left(\frac{\tau}{T} - n\right)} d\tau\right] \cdot \underbrace{\varphi\left(\frac{t}{T} - n\right)} \\
&= \frac{1}{T}\sum_{n=-\infty}^{\infty}\left[\int_{-\infty}^{\infty} g(\tau)\cdot\overline{\tilde{\varphi}\left(\frac{\tau}{T} - n\right)} d\tau\right] \cdot \overbrace{T\int_{-\infty}^{\infty} \hat{\varphi}(\nu T)\, e^{-2\pi\iota\nu T} e^{2\pi\iota\nu t} d\nu} \\
&= \int_{-\infty}^{\infty} \hat{\varphi}(\nu T)\, e^{2\pi\iota\nu t} \cdot \underbrace{\left\{\sum_{n=-\infty}^{\infty}\left[\int_{-\infty}^{\infty} g[\tau]\cdot\overline{\tilde{\varphi}\left(\frac{\tau}{T} - n\right)} d\tau\right]\cdot e^{2\pi\iota(-\nu)T}\right\}} d\nu \\
&= \int_{-\infty}^{\infty} \hat{\varphi}(\nu T)\, e^{2\pi\iota\nu t} \cdot \overbrace{\sum_{n=-\infty}^{\infty} \hat{g}\left(\frac{n}{T} + \nu\right)\cdot\hat{\tilde{\varphi}}(n - \nu T)}\, d\nu \\
&= Fourier^{-1}\left\{\sum_{n=-\infty}^{\infty} \hat{g}\left(\frac{n}{T} + \nu\right)\cdot\hat{\tilde{\varphi}}(n - \nu T)\cdot\hat{\varphi}(\nu T)\right\}
\end{aligned}
$$

Now, due to invariance of the inner product under Fourier transform, one can write:

$$\begin{aligned}
\langle Q_T g, Q_T h\rangle &= \left\langle \hat{Q}_T g, \hat{Q}_T h \right\rangle \\
&= \int_{-\infty}^{\infty} \hat{Q}_T g\,(\nu)\,\overline{\hat{Q}_T h\,(\nu)}d\nu \\
&= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \hat{g}\left(\frac{n}{T}+\nu\right)\cdot \hat{\tilde{\varphi}}\,(n-\nu T)\cdot \hat{\varphi}\,(\nu T) \\
&\quad \overline{\cdot \sum_{n=-\infty}^{\infty} \hat{h}\left(\frac{n}{T}+\nu\right)\cdot \hat{\tilde{\varphi}}\,(n-\nu T)\cdot \hat{\varphi}\,(\nu T)}d\nu \\
&= \int_{-\frac{1}{2T}}^{\frac{1}{2T}} \sum_{n=-\infty}^{\infty} \hat{g}\left(\frac{n}{T}+\nu\right)\cdot \hat{\tilde{\varphi}}\,(n-\nu T) \\
&\quad \overline{\cdot \sum_{n=-\infty}^{\infty} \hat{h}\left(\frac{n}{T}+\nu\right)\cdot \hat{\tilde{\varphi}}\,(n-\nu T)}\cdot \sum_{n=-\infty}^{\infty} \left|\hat{\varphi}\,(\nu T+n)\right|^2 d\nu \\
&= \int_{-\frac{1}{2T}}^{\frac{1}{2T}} \sum_{n=-\infty}^{\infty} \hat{g}\left(\frac{n}{T}+\nu\right)\cdot \hat{\tilde{\varphi}}\,(n-\nu T) \\
&\quad \overline{\cdot \sum_{n=-\infty}^{\infty} \hat{h}\left(\frac{n}{T}+\nu\right)\cdot \hat{\tilde{\varphi}}\,(n-\nu T)}\cdot \hat{A}_{\varphi}\,(-2\pi\nu T)\,d\nu \\
&= \int_{-\infty}^{\infty} \hat{g}\,(\nu)\,\hat{\tilde{\varphi}}\,(-\nu T)\cdot \overline{\sum_{n=-\infty}^{\infty} \hat{h}\left(\frac{n}{T}+\nu\right)\hat{\tilde{\varphi}}\,(n-\nu T)}\cdot \hat{A}_{\varphi}\,(-2\pi\nu T)\,d\nu \\
&= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{g}\,(\nu)\,\overline{\hat{h}\left(\frac{n}{T}+\nu\right)}\cdot \hat{\tilde{\varphi}}\,(-\nu T)\,\overline{\hat{\tilde{\varphi}}\,(n-\nu T)}\cdot \hat{A}_{\varphi}\,(-2\pi\nu T)\,d\nu
\end{aligned}$$

Restricting ourselves to non-complex analysis ($\tilde{\varphi}$) and synthesis (φ) functions yields:

$$\begin{aligned}
&\langle Q_T g, Q_T h\rangle \\
&= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{g}\,(\nu)\,\overline{\hat{h}\left(\frac{n}{T}+\nu\right)}\cdot \overline{\hat{\tilde{\varphi}}\,(-\nu T)}\cdot \overline{\hat{\tilde{\varphi}}\,(n-\nu T)}\cdot \overline{\hat{A}_{\varphi}\,(2\pi\nu T)}d\nu
\end{aligned}$$

which proves the Lemma. Applying $\tilde{\varphi}\,(t) = \delta\,(t)$ and $\varphi\,(t) = \sin c\,(t)$:

$$\begin{aligned}
&\left\langle \{T\cdot g\,(nT)\}_{n=-\infty}^{\infty}, \{T\cdot h\,(nT)\}_{n=-\infty}^{\infty} \right\rangle \\
&\quad = \langle Q_T g, Q_T h\rangle = \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{g}\,(\nu)\,\overline{\hat{h}\left(\frac{n}{T}+\nu\right)}\cdot 1\cdot 1\cdot T d\nu
\end{aligned}$$

proves the theorem.

ACKNOWLEDGMENTS

This research was supported in part by the HASSIP Research ProgramHPRN-CT-2002-00285 of the European Commission, and by the Ollendorff Minerva Center. Minerva is funded through the BMBF.

REFERENCES

1. L. R. Lo Conte, R. Merletti, and G. V. Sandri, Hermite expansions of compact support waveforms: Applications to myoelectric signals, *IEEE Trans. Biomed. Eng.* **41**, 1147–1159 (1994).
2. R. O. Duda, P. E. Hart, and D. Stork, *Pattern Classification*, Second Edition (2001).
3. S. R. Kulkarni, G. Lugosi, and S. S.Venkatesh, Learning Pattern Classification, *IEEE Trans. Info. Theory* **44**(6), 2178–2206 (1998).
4. R. Bracewell, *The Fourier Transform and Its Applications* (McGraw-Hill, Singapore, 1986).
5. T. Blu and M. Unser, Approximation error for quasi-interpolators and (multi) wavelet expansions, *Applied and Computational Harmonic Analysis* **6**, 219–251 (1999).

MODEL AND KNOWLEDGE MANAGEMENT IN DISTRIBUTED DEVELOPMENT: AGREEMENT BASED APPROACH

Darijus Strašunskas and Yun Lin*

1. INTRODUCTION

Models are built to share knowledge or definitions with other people, and this application is especially directed to people that want to share knowledge or define knowledge in co-operation with others. Modeling is seen as "the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication",[1] and is applied in the early phases of information system analysis and design. However, many problems are encountered when building models. It is conceivable that a variety of different versions of models will be used in different stages of the development process; in general, it is difficult to develop a model that can be acceptable for all participants in a development project. Furthermore, it is known that different people usually present different models given the same domain and the same problem. The same information about system can be modeled at various levels of abstraction and from different viewpoints considering different aspects. Variations among models generally appear due to the creative nature of the modeling activity, as well as other factors such as the richness of the modeling language,[2] the ambiguities of modeling grammars, and others.

This problem becomes more evident when the process is distributed. Then the variability of the model versions increases due to the highly interactive and iterative nature of the development process and to the different, sometimes conflicting, angles to the problem and solution taken by the different stakeholders. Therefore, modeling process can be viewed as three dimensions of requirements engineering:[3] agreement, representation and specification dimension. The agreement dimension should be based on common understanding about problem domain, organizational strategy; the representation dimension is based on the essential semantic aspects of system analysis; the specification dimension

* Darijus Strašunskas, Dept. of Computer and Information Science, Norwegian Univ. of Science and Technology, NO-7491 Trondheim, Norway and Department of Informatics, Kaunas Faculty of Humanities, Vilnius University, 44280 Kaunas, Lithuania, dstrasun@idi.ntnu.no. Yun Lin, Dept. of Computer and Information Science, Norwegian Univ. of Science and Technology, NO-7491 Trondheim, Norway, yunl@idi.ntnu.no.

bases on the implementation oriented system development aspects. The most difficult in modelling is to arrive at a coherent, complete and consistent description of the problem and domain. Description should be shared and agreed between all stakeholders. Therefore, we focus on the agreement dimension that deals with the consolidation of logically and geographically distributed views.[3]

In this paper, we discuss distributed modeling, focusing on support for individual developers and allowing them expressing their view and perception of the Universe of Discourse (UoD) in model fragments, which are integrated based on common agreement among them. The paper is further structured as follows. Section 2 overviews related work. In Section 3, we further elaborate complexity of distributed modeling, discuss settings for our approach and present the approach. In Section 4, we illustrate our approach using example from a travel domain. Finally, Section 5 concludes the paper and lays down future work.

2. RELATED WORK

Many proposals on model composition are available in the literature. Model composition in a distributed heterogeneous environment has been the subject of a few recent research activities. Namely, the merging of ontologies is one of the recent model merging scenario. Collaboration during the modeling is one of the most important features of the ontology building tools, as ontology is seen as an explicit representation of a *shared* conceptualization.[4] However, less than half of ontology building tools surveyed in [5] have a multi-user support. Even the tools supporting collaborative ontology development still do it in the old-fashioned way, i.e. do not allow multiple accesses to concept (object) by different developers at the same time. The work that has been done so far in the area of collaborative work with ontologies mainly has focused on one ontology which is edited by the developer. I.e., the web-based Ontosaurus* supports collaboration and allow developers to edit ontology only when consistency is retained within the ontology as a whole.

Environments like Protégé† or Chimaera‡ offer sophisticated support for ontology engineering and merging of ontologies, but lack sophisticated support for collaborative engineering. Chimaera is build on top of Ontolingua Server[6] and, therefore, has the same support for collaborative engineering, i.e. read and write access rights to ontologies are controlled by the ontology owner; users are able to join a session and work simultaneously on the same ontology.

Some tools provide advanced support for communication between users contributing to better collaboration during ontology engineering, e.g. Tadzebao[7] supports both asynchronous and synchronous discussions on ontologies; Apecks[8] aims to support discussion about ontologies and allows different conceptualizations of a domain to co-exist.

In[9] we found the first attempt to use a totally distributed environment to work with ontologies. They present their work with the peer-to-peer Semantic Web. It allows users

* http://www.isi.edu/isd/ontosaurus.html.

† http://protege.stanford.edu/.

‡ http://www.ksl.stanford.edu/software/chimaera/.

to create, maintain, and control sharing of ontologies in a P2P environment. Although it allows users to add parts to ontologies, but it mainly seems to be built for maintaining, sharing and retrieving other ontologies.

WebOnto* is a web-based tool for developing and maintaining ontologies. It includes functions such as visualization, browsing and editing ontologies. The tool includes functionality for sharing changes between users. Mintra et al. [10] present a toolkit called Onion. It is a toolkit to help domain expert bridge the gap between smaller domain specific ontologies by creating links between ontologies based on their context. Before Onion was developed most research on ontology construction focused on tools for building a single global ontology.

Hozo[11] environment for distributed ontology development is focused on building a single ontology by on splitting it into components and establishing dependency between them. The target ontology is obtained by compiling the component ontologies. System does not allow multiple accesses to a concept by different developers at the same time, as developers have assigned a particular component ontology to develop. OntoEdit[12] allows multiple user access to the same ontology to build it collaboratively. It provides namespace mechanism allowing splitting ontologies into modules.

In summary, most tools provide the collaborative facilities by supporting basic requirements for distributed development, e.g. rights- and user management, locking mechanism, communication and notification means. Even most sophisticated modeling environments do not provide means for development of the *shared conceptualization*, i.e. allowing users to develop overlapping fragments of models based on their own understanding and perception of the real world.

3. DISTRIBUTED MODELING

3.1. Complexity of Distributed Modeling

Model development is a complex and difficult task. It is usually a creative and collaborative process, during which different stakeholders are focusing on various aspects, expressing them at different levels of abstraction, and producing several variants of each.

Different levels of abstraction of the same system enable to deal with complexity by removing details from model. The model must be able to act as an efficient and effective communications medium between the different parties involved in development project. Usually, models are augmented by details on each development step.

A system can be described from many viewpoints. Each viewpoint defines what characteristics should be included in its views and what issues should be ignored or treated as transparent. A view is, therefore, a piece of the model that is small enough to comprehend but that also contains all relevant information about a particular concern. Variants dimension is more concerned with different versions and configurations.

The success of distributed project depends on how well "laissez-faire" rule is obeyed, meaning that developers should be allowed to express what they want in whatever form.

* http://webonto.open.ac.uk/.

More precisely, Farshchian[13] emphasized a list of requirements for development environments to enable collaboration in geographically distributed developments. Here we adopt the requirements (**Req.n**) as follows.

Req.1 - *Unrestricted product object types* – a development environment should allow the developers to share any type of object that they might find useful for supporting their cooperation.

Req.2 - *Unrestricted relation types* – a development environment should allow the developers to create any type of relation between any two objects.

Req.3 - *Incremental product refinement* – a product development environment should provide the developers with flexible mechanisms for incrementally refining the product. The developers should be allowed to start with vague products, and to refine them into more complete and formal ones.

Concurrent engineering changes old practice, when all the required objects were locked during the whole change/modification activity. Each software engineer should have direct access to all needed objects. But changed version should be kept with access forbidden for other developers during modification, because the state of fragment is inconsistent in a modification phase. If n engineers change the same object concurrently, this object should have $n + 1$ different copies.[14] It means that each developer needs the private copies of fragments. On the other hand, the colleagues know that other changes possibly are done on the same fragments/objects and want to be incorporated when relevant. In summary, collaborative distributed development needs tools that allow the creation and access to a central composite product, and at the same time support development in local workspaces.

3.2. Knowledge Preservation

In a collaborative environment where different users work on models, it is important that there is a way of sharing own views, and step by step achieving agreement and common conceptualization. This is usually called model integration and can be accomplished by merging or term alignment. It is important to keep term merging separated from the term alignment. Merging means that one new model is created from n existing models. Model alignment is when links are created between models, so that the models can be used as one.

Although, the initial goal is usually to develop a single model of the UoD, it turns out to be very important to preserve and model the various "views" of the information seen by different stakeholders and participants during the system analysis phase. Usually, different developers might have different vocabulary to express their perception of the world. It is important to preserve knowledge of developer that is expressed in the model fragment she has developed. We need to ensure that developer's work will not be disturbed, for instance, if a developer uses term 'aircraft' referring to 'airplane', this term should be preserved in her local view, otherwise after several changes it will be difficult to continue.

3.3. An Approach

Underlying hypothesis of our approach is that given the same problem domain to reason about, the model developed by different stakeholders will not only differ, but as well will have some overlapping parts, i.e. some parts (views) in different models are com-

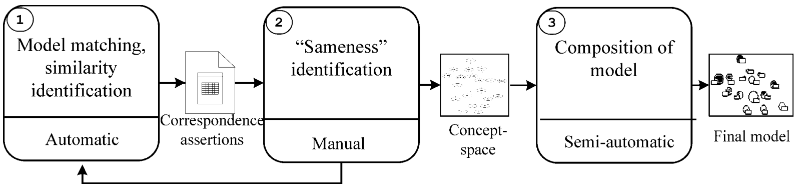

1
Model matching, similarity identification
Automatic
Correspondence assertions
2
"Sameness" identification
Manual
Concept-space
3
Composition of model
Semi-automatic
Final model

Figure 1. Functional view of our approach.

monly shared. In order to integrate the distributed models, these commonalities should be captured.

Our approach consists of 3 basic steps (see Figure 1):

Step 1 - Model matching and similarity identification. Model integration typically involves identifying the correspondences between two models, determining the differences in definitions, and creating a new model that resolves these differences. Four types of view differences are described in[15], which were paraphrased by Hefflin and Hendler:[16]

- **terminology**: different names are used for the same concepts;
- **scope**: similar categories may not match exactly; their extensions intersect, but each may have instances that cannot be classified under the other;
- **encoding**: the valid values for a property can be different, even different scales could be used;
- **context**: a term in one domain has a completely different meaning in another.

Some of the above listed problems might be found and resolved automatically, i.e. scope, context. At present, there exists a number of automated schema and model matching systems, for instance,[17–25] which produce correspondence suggestion between elements of different models. In general, match algorithms developed by different researchers are hard to compare since most of them are not generic but tailored to a specific application domain and model types. Usually, use of only one approach will not produce good enough matching; for better results we need to combine several schema matching approaches. Currently, we are investigating which combination of techniques is most applicable for our approach.

Step 2 - "Sameness" identification. Given current techniques for correspondences assertion between models, it is possible to calculate quite precise similarity of the concepts. However, it is impossible to identify the "sameness" of concepts without knowing authors' intention. Therefore, manual intervention and agreement is necessary at this step. The authors of all model fragments are notified about the results of the model matching and are asked to verify them by pointing the same concepts and achieving agreement about the concept name, if they use different terms.

These two steps (step 1 and step 2) are iterated as many times as new fragments are signed-in to the repository. Step 2 results in the common knowledge layer or so called "concept-space", where the commonly agreed concepts and relations between them are placed. This layer is used to differentiate from the local namespace, which is kept unique for each developer allowing to use own vocabulary. Figure 2 exemplifies the idea behind our approach. I.e. after having identified the concepts being the same, despite of different

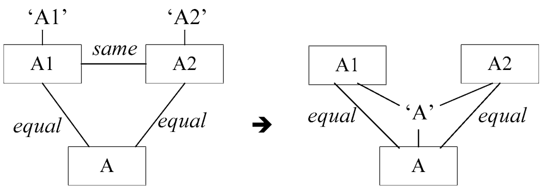


Figure 2. Management of "sameness".

term used to name them, the "equality" relationship is established between local concepts named '*A1*' and '*A2*', and the agreed concept named '*A*'.

Step 3 - composition of models. In this step a final application dependent model is produced based on agreed view and formed model in the common concept space.

3.4. Further Elaboration

Since distributed models are built by different modelers having various modeling purposes and viewpoints, the perspectives of different models may be far from each other. Context similarity is considered during the agreement and identification of the same concepts. Some constraints in different local models may conflict with each other, even being agreed and integrated in common models. Furthermore, some concepts and relationships in the integrated models may be redundant or may need to be further specified, i.e. what to do with derived relationships or model fragments at different abstraction levels. Further, we introduce more rules for model refinement (step 3).

Here, we mainly focus on static (class) diagram which presents concepts and their relationships. Hence, the integration refining issues include abstracting concepts and refining concepts, adding and deleting properties of concepts, adjusting types of properties, abstracting transitive relationships into high level relationships and refining relationships into low level relationships. We define a set of generic rules for the above mentioned refinement transformations. Before formulizing those refining issues, we make some definitions. Let UoD_I be universe of discourse for integrated model, and UoD_D– universe of discourse for particular local model fragments. Then, C_D is a concept used from a local model fragment and C_I is the concept in the integrated model. $P(c)$ is the set of properties of concept C and p is a property. While, $R(C_i * C_j)$ is the relationship between concepts C_i and C_j.

Rule 1. Abstraction of concepts. Concepts used in local models are usually more concrete. Often, during the integration, super concepts are needed to generalize those sub concepts, or even replace sub concepts if the sub concepts are not important in an integrated model.

Let, C_{Di} and C_{Dj} be two concepts from a model i and model j. Both concepts are elements from the same domain (UoD). Then a concept C_I from the domain of integrated model will be a super concept of C_{Di} and C_{Dj} in the integrated model.

$$\begin{aligned} &C_{Di} \in UoD_I \wedge C_{Dj} \in UoD_I \wedge \exists C_I(C_I \in UoD_I \wedge C_{Di} \subseteq C_I \wedge C_{Dj} \subseteq C_I) \\ &\quad \Rightarrow C_I = Abstract(C_{Di}, C_{Di}) \qquad (1) \end{aligned}$$

Rule 2. Refinement of concepts. There is a need to create new concepts, when UoD_I of an integrated system is broader than the one considered in the local model fragments. Some of such concepts are created based on a relationship between existing concepts.

$$R(C_{Di} * C_{Dj}) \in UoD_I \wedge \exists C_I (C_I \in UoD_I \wedge C_I \notin UoD_{Di} \wedge C_I \notin UoD_{Dj}) \\ \Rightarrow Create(C_I, R(C_{Di} * C_{Dj})) \qquad (2)$$

Rule 3. Addition and/or deletion of properties of concepts. Certain properties of concepts are ignored in the distributed model fragments as being not important in a limited scope or in a certain viewpoint, but they might be critical for an integrated system. On the other hand, certain concepts contain too many details which are necessary in some isolated models, but inessential for the integrated system. Properties need to be further edited according to requirements for new integrated system.

$$\exists p (p \in UoD_I \wedge p \notin P(C_D)) \Rightarrow AddProp(p, C_I) \qquad (3)$$

$$\exists p (p \in P(C_D) \wedge p \notin UoD_I) \Rightarrow DelProp(p, C_D) \qquad (4)$$

Rule 4. Adjustment of types of properties. Types of properties usually concern implementation oriented aspects, and have little effects on the semantics of models. Meaning that possibly the same property has different types in different models. In order to keep the consistency of integrated model, types of the same property should be unified obeying implementation requirements of system.

Let *Sem*(p) be the semantics of property p and $T(p)$ be the type of property p.

$$Sem(p_{Di}) = Sem(p_{Dj}) \wedge T(p_{Di}) \neq T(p_{Dj}) \Rightarrow Adjust(T(p_{Di}), T(p_{Dj})) \qquad (5)$$

Rule 5. Abstraction of transitive relationships into higher level relationships and refinement of relationships into lower level relationships. A transitive relationship is the semantic equivalent of a collection of normal relationships.[26] The transitive abstraction relationship is the high level relationship and a direct relationship which can not be refined is low level relationship. With different requirements, perhaps only high level relationship is enough while on other cases low level relationship is necessary. There are three generic relationships – generalization, aggregation and association, which are supported by most modeling languages. The transitive abstraction rules for different combination of three generic relationships are different. In[27], they developed a set of transitive abstraction rules for inference of transitive relationships (e.g., classA-*association*-> classB<- *aggregation*-classC ⇒ classA-*weakAssociation*->classC, meaning that, if classA has association relation with classB, and classB is aggregated into classC, then the resulting abstraction would be weak association between classA and classC), which we do adopt for our purposes. Given the combination of $R(C_{Di} * C_{Dj})$ and $R(C_{Dj} * C_{Dk})$ satisfies one of transitive rules, the result would be $R(C_{Di} * C_{Dk})$, while R here is specified as either generalization (R_{Ge}), aggregation (R_{Ag}) or association (R_{As}) and parameters are non-transitive.

$$R(C_{Di} * C_{Dj}) \cup R(C_{Dj} * C_{Dk}) \in RuleSet \Rightarrow R(C_{Di} * C_{Dk}) \qquad (6)$$

The refinement process based on rules is semi-automatic. Developers need to make decision on what concepts and what properties are important, at what kind of granularity concepts and relationships should kept as they depend on the requirements of integrated system.

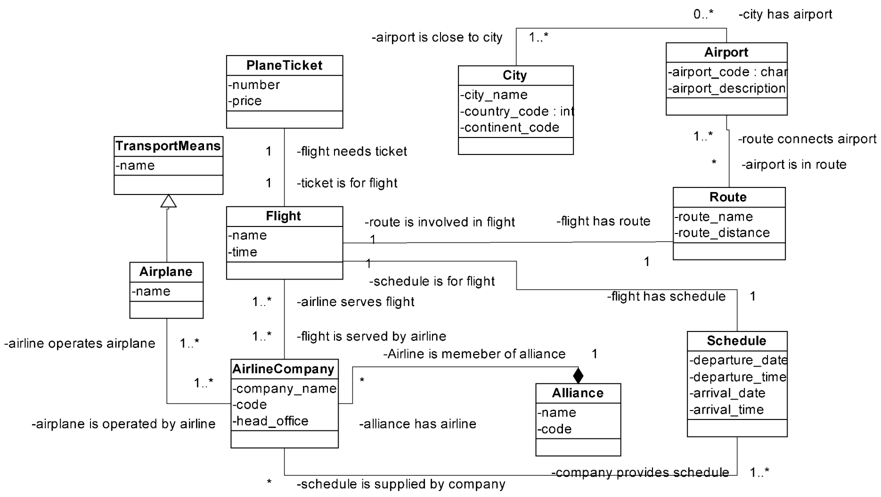


Figure 3. Airplane transportation model fragment.

4. APPLICATION OF THE APPROACH

In this section, we exemplify our approach using a case from a travel domain. There are requirements to build a travel agency system which provide airplane and train ticket, and hotel booking services as well to provide other tourism information. Separate models are made by different modelers and later they are integrated into one model. To illustrate our approach, we focus on two model fragments: airplane and train transportation model fragments. Then these two models will be integrated as a part of the whole travel agency system model.

Airplane transportation model fragment describes basic concepts and their relationships about flight. Figure 3 shows UML class diagram for airplane transportation. Train transportation model fragment contains concepts and relationships about train transportation information and is depicted in Figure 4, using UML class diagram as well.

Step 1. Model matching and similarity identification. Because both two models are built in UML and they are quite similar in structure and in context, the model similarity can be identified by current available schema and model matching systems.[17–25] In this particular case, we have adopted iMapper,[25] developed in our group. Most similar concepts pairs from the two models are {*Schedule*, *Timetable*}, {*Flight*, *Trip*}, {*City*, *City*} and {*Route*, *Route*}* (see Figure 5).

Step 2. “Sameness” identification. With the list of similar concept pairs, modelers should reach agreements on whether two concepts are same or not (see Figure 5). Concepts ‘*Schedule*’ and ‘*Timetable*’ are regarded as being the same, only different terminology used. ‘*Timetable*’ is decided as a common concept name for this concept, so ‘*Timetable*’ is

* The first concept in parentheses is from airplane transportation ontology model fragment and the second one is from train transportation ontology model fragment.

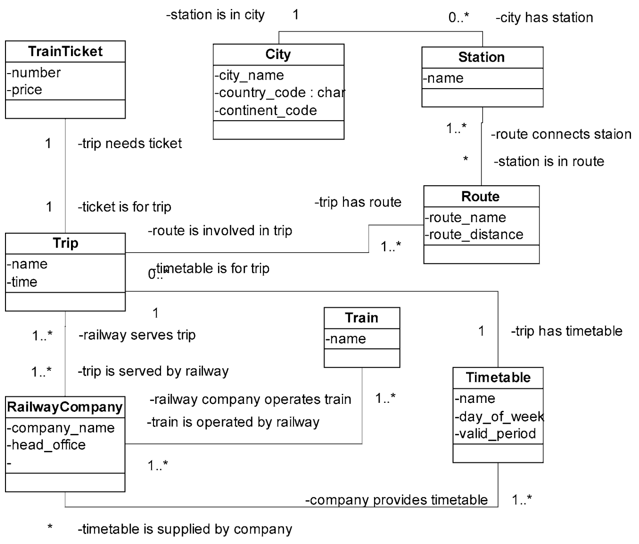


Figure 4. Train transportation model fragment.

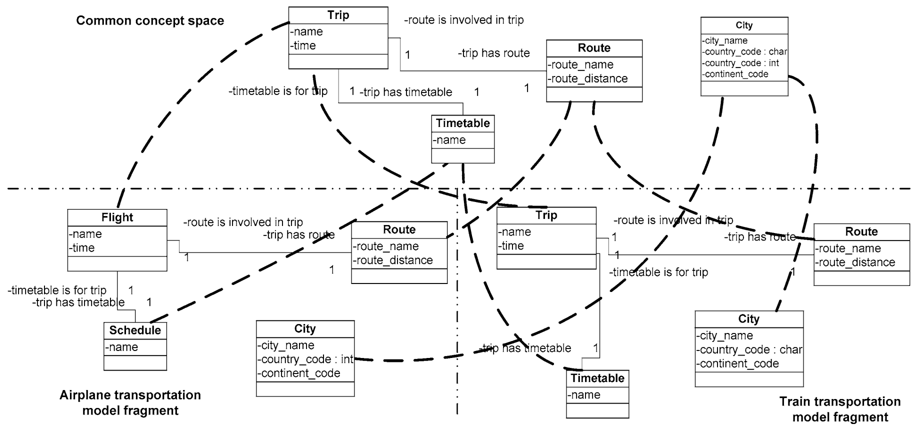


Figure 5. Explanatory visualization of mappings between models in local namespaces (bottom part) and concept space (upper part).

put in the common concept-space and is referred by *'Schedule'* in airplane transportation model fragment and by *'Timetable'* in train transportation model fragment. *'Trip'* is put in the common concept-space as the reference of *'Flight'*. Trip is chosen because the name of *'Flight'* is more specified to airplane but the structure of it is same as *'Trip'* concept. Two *'Routes'* concepts are regarded as the same. Two *'City'* concepts look almost the same, but the type of property *'country_code'* in two models are different: one is *'int'* and the other is *'char'*. Such difference is kept in common concept space and will be resolved in step 3 as it depends on an application.

Step 3. Composition of model. As *'AirlineCompany'* and *'RailwayCompany'* refer to different entities, the way to integrate them is by generalizing and relating them by more abstract concept. Therefore, we apply Rule 1 (see Eq. 1):

$$\begin{aligned}&(AirlineCompany \in TravelDomain) \wedge (RailwayCompany \in TravelDomain)\wedge \\ &(TransportCompany \in TravelDomain) \wedge (AirlineCompany \subseteq TransportCompany)\wedge \\ &(RailwayCompany \subseteq TransportCompany)\end{aligned} \tag{7}$$

And as result we introduce an abstract concept *'TransportCompany'* being the super concept of *'AirlineCompany'* and *'RailwayCompany'*.

The generation links between *'AirlineCompany'*, *'RailwayCompany'* and *'Transport-Company'* should be added into integrated model. When the generation links are added in the model, other relationships related to *'AirlineCompany'* and *'RailwayCompany'* should be checked if they are consistent with *'AirlineCompany'* and *'RailwayCompany'* or link them directly to their super concept *'TransportCompany'*.

Applying Rule 4 (Eq. 5): Concept *'City'* in Figure 3 is considered the same as in Figure 4, but the type of property *'country_code'* in Figure 3 is *'int'* while in Figure 4 it is *'char'*. Type *'int'* is adjusted into *'char'* in the integrated model.

Applying Rule 5 (Eq. 6): *'TransportTicket'* is inserted as super concept (Rule 1) of *'PlaneTicket'* (Figure 3) and *'TrainTicket'* (Figure 4). *'PlaneTicket'* has a relationship with concept *'Flight'*, as well as *'TrainTicket'* is related to concept *'Trip'*, and during sameness check we have agreed on that *'Flight'* is same as *'Trip'*. When integrating models, we should remove all the relationships connected with *'Flight'* to *'Trip'*. The link between *'PlaneTicket'* and *'Flight'* is removed and changed into relationship between *'PlaneTicket'* and *'Trip'*. Such relationship is semantically equivalent to the one between *'TrainTicket'* and *'Trip'*. The two relationships could be abstracted to the relationship between *'Trip'* and *'TransportTicket'* because of transitive relationship rule:

$$\begin{aligned}&R_{Ge}(TransportTicket * PlaneTicket) \cup R_{As}(PlaneTicket * Trip) \\ &\quad \Rightarrow R_{As}(TransportTicket * Trip)\end{aligned} \tag{8}$$

Finally, the integrated model is shown in Figure 6. It is based on the concepts identified being the same (i.e., Figure 5) and the rules for model refinement. It should be noted that the local distributed model fragments are still kept unchanged.

5. CONCLUSIONS AND FUTURE WORK

We have outlined a framework to support management of distributed modeling activities by distinguishing 2 main layers: the local namespace; the shared and agreed concept space. The local namespace allows the developers to model their views as they perceive and use their preferable terminology, i.e. providing full "laissez-faire" for their creativity. The concept space is used for sharing of conceptualization. The concept-space, or common knowledge layer, results into the target model. The most important contribution of this paper is that separation between these two "conceptual" spaces provides means for preserving knowledge of each developer, allowing them to use their own terminology, agree about individual conceptualization and still refer to the common concept space for agreement purposes.

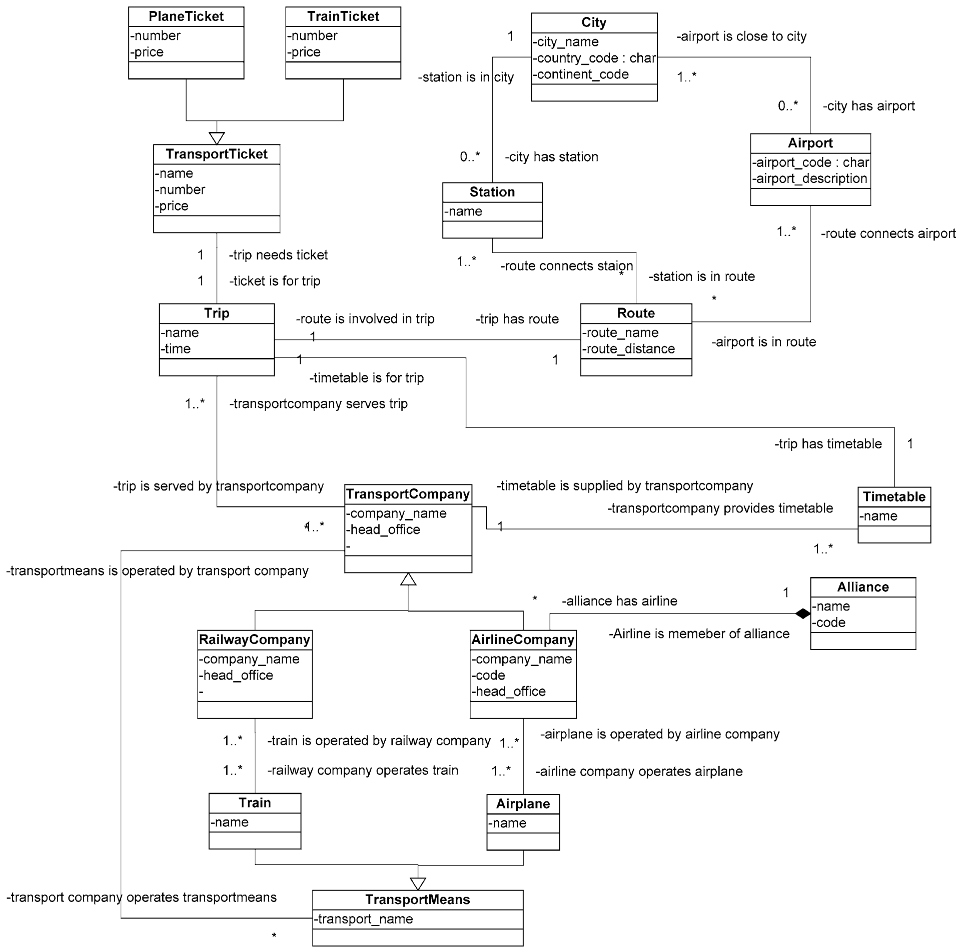


Figure 6. Integrated model for travel domain.

Model fragments alignment in the concept space provides a good means for learning while modeling. For instance, having identified parts of models being the same, the developers are notified about certain mismatches (i.e. class concepts are the same, but relationship type between them differs), and discuss the difference. Therefore, we consider it being important to stepwise integrate model fragments during the development time, not only the products (models) themselves. That is the main difference from current state-of-the-art, where existing methodologies focus on models, as final product, integration.

The approach has limitation as it has not been yet tested in real distributed settings. Having different people involved it may be difficult for them to agree about whether some concepts are the same. But we believe that model integration during the development activities having authors present will produce better results, than any other post-development based integration, when authors of constituent model fragments are not available.

The approach is first step towards implementing the environment for collaborative modeling considering other aspects of collaboration, e.g. user awareness, support for opportunistic communication. Future work mainly concerns developing mechanism for recording all operations performed, tracing the information and decisions based on which concepts were added into the common knowledge layer. The challenge is creating an algorithm for automatic update of the models in the concept-space based on observed changes in the local model fragments.

REFERENCES

1. J. Mylopoulos, Conceptual modeling and Telos. Chapter 2, in: *Conceptual Modeling, Databases, and CASE*, edited by P. Loucopoulos and R. Zicari (Wiley, 1992), pp. 49–68.
2. T. Moriarty, The importance of names, *The Data Administration Newsletter* **15** (2000).
3. K. Pohl, The three dimensions of requirements engineering, in: *Proceedings of 5th Intl. Conf. on Advanced Information Systems Engineering* (CAiSE'93), edited by C. Rolland, F. Bodart, and C. Cauvet (Springer-Verlag, Paris, France, 1993), pp. 275–292.
4. T. R. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition* **5**(2) (1993).
5. M. Denny, Ontology Building: A Survey of Editing Tools (2002); http://www.xml.com/lpt/a/2002/11/06/ontologies.html.
6. A. Farquhar, R. Fikes, and J. Rice, *The Ontolingua Server: A Tool for Collaborative Ontology Construction* (KSL Stanford University, USA, 1996).
7. J. Domingue, Tadzebao and WebOnto: discussing, browsing, and editing ontologies on the Web, in: *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop* (Kanff, Canada, 1998).
8. J. Tennison, and N. R. Shadbolt, APECKS: a tool to support living ontologies, in: *Proceedings of 11th Knowledge Acquisition for Knowledge-Based Systems Workshop* (Banff, Canada, 1998).
9. M. Arumugam, A. Sheth, and B. Arpinar, Peer-to-Peer Semantic Web: a distributed environment for sharing semantic knowledge on the web (2002); http://lsdis.cs.uga.edu/lib/download/ASA02-WWW02Workshop.pdf.
10. P. Mitra, M. Kersten, and G. Wiederhold, Graph-oriented model for articulation of ontology interdependencies, Stanford University Technical Note, CSL-TN-99-411, (1999) and in: *Proceedings of the 7th Intl. Conf. on Extending Database Technology* (EDBT 2000).
11. E. Sunagawa, K. Kozaki, Y. Kitamura, and R. Mizoguchi, An environment for distributed ontology development based on dependency management, in: *Proceedings of 2nd Intl. Semantic Web Conf. (ISWC2003)*, edited by D. Fensel et al. (LNCS 2870, Springer-Verlag, Berlin, Heidelberg, 2003), pp. 453–468.
12. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke, OntoEdit: collaborative ontology development for the Semantic Web, in: *Proceedings of the 1st Intl. Semantic Web Conf. (ISWC2002)* (Sardinia, Italy, 2002).
13. B. A. Farshchian, *A Framework For Supporting Shared Interaction in Distributed Product Development Projects* (PhD thesis, IDI-NTNU, Trondheim, Norway, 2001).
14. J. Estublier, Objects control for software configuration management, in: *Advanced Information Systems Engineering, proceedings of 13th Intl. Conf. CAiSE*2001*, edited by K. R. Dittrich, A. Geppert, and M. C. Norrie (Interlaken, Switzerland, LNCS 2068, Springer-Verlag, 2001), pp. 359–373.
15. G. Wiederhold, An algebra for ontology composition, in: *Proceedings of 1994 Monterey Workshop on Formal Methods* (1994), pp. 56–62.
16. J. Hefflin, and J. Hendler, Dynamic ontologies on the Web, in: *Proceedings of 17th National Conf. on Artificial Intelligence* (AAAI-2000).
17. H. H. Do, and E. Rahm, COMA – A system for flexible combination of schema matching approaches, in: *Proceedings of 28th Intl. Conf. on Very Large Databases* (VLDB, Hong Kong, 2002).
18. A. Doan, J. Madhavan, P. Domingos, and A. Halvey, Learning to map between ontologies on the semantic web, in: *Proceedings of WWW-02, 11th Intl. WWW Conf.* (Hawaii, 2002).
19. W. Li, and C. Clifton, SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks, *Data & Knowledge Engineering* **33**(1), 49–84 (2000).

20. J. Madhavan, P. A. Bernstein, and E. Rahm, Generic schema matching with Cupid, in: *Proceedings of 27th Intl. Conf. on Very Large Databases (VLDB)* (Roma, Italy, 2001), pp. 49–58.
21. S. Melnik, E. Rahm, and P. A. Bernstein, *Rondo: A Programming Platform for Generic Model Management* (SIGMOD, 2003), pp. 193–204.
22. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi, Observer: an approach for query processing in global information systems based on interoperability between preexisting ontologies, in: *Proceedings 1st Intl. Conf. on Cooperative Information Systems* (Brussels, 1996).
23. R. A. Pottinger, and P. A. Bernstein, Merging models based on given correspondences, in: *Proceedings of the 29th VLDB Conference* (Berlin, Germany, 2003).
24. E. Rahm, and P. A. Bernstein, A survey of approaches to automatic schema matching, *The VLDB Journal* **10**(4), 334–350 (2001).
25. X. Su, J. A. Gulla, Semantic enrichment for ontology mapping, in: *Proceedings of the 9th Intl. Conf. on Applications of Natural Language to Information Systems (NLDB'04)* (Manchester, UK, Springer-Verlag, 2004).
26. A. Egyed, Compositional and relational reasoning during class abstraction, in: *Proceedings of the 6th Intl. Conf. on the Unified Modeling Language (UML)* (San Francisco, USA, 2003), pp. 121–137.
27. A. Egyed, and P. Kruchten, Rose/Architect: a tool to visualize architecture, in: *Proceedings of the 32nd Hawaii Intl. Conf. on System Sciences (HICSS)* (1999).

RENAISSANCE OF BUSINESS PROCESS MODELLING

Marite Kirikova and Janis Makna*

1. INTRODUCTION

Business process modelling (BPM) became popular more than a decade ago, when the activities and hopes related to business process reengineering where one of the most popular issues to be researched and applied.[1] However, business process reengineering was not as successful as was expected. Therefore its popularity dimmed and interest from BPM switched to other means of business process improvement. In the early years of BPM the main purpose of this activity was to prove that the new business process to be introduced is more efficient than the previous one. Therefore modelling was mainly concerned with assigning values to economic and time attributes of the process and simulation of the defined processes in order to obtain data for process evaluation. Most of BPM tools were autonomous[2] based on concepts of data flow diagrams or Petri nets and usually extended by built-in or related business process simulation modules.

Over a decade later the number of BPM tools exceeds 300. Among the most popular are those tools, which are an integrated part of enterprise modelling environments.[3] Use of BPM varies from simple models to very complex systems of models. Business process re-engineering is only one of the current areas of BPM application. Quality management, knowledge management, change management, continuous business process improvement, development of different types of information systems, and enterprise integration are other areas of BPM application. The high level of BPM popularity has led also to efforts to standardize BPM notations.[4]

The goal of the paper is to develop a comprehensive survey of several approaches to business process model descriptions, to analyse the role of BPM and business process models in different areas of application, and to draw conclusions regarding the requirements for business modelling tools (software systems for business process modelling) in those areas.

Different concepts used in business process analysis are described in Section 2. Capabilities of BPM tools are analysed in Section 3. Role of BPM in several application areas

* Department of Systems Theory and Design, Riga Technical University, 1 Kalku, Riga, LV-1658, Latvia, marite@cs.rtu.lv, promis@apollo.lv.

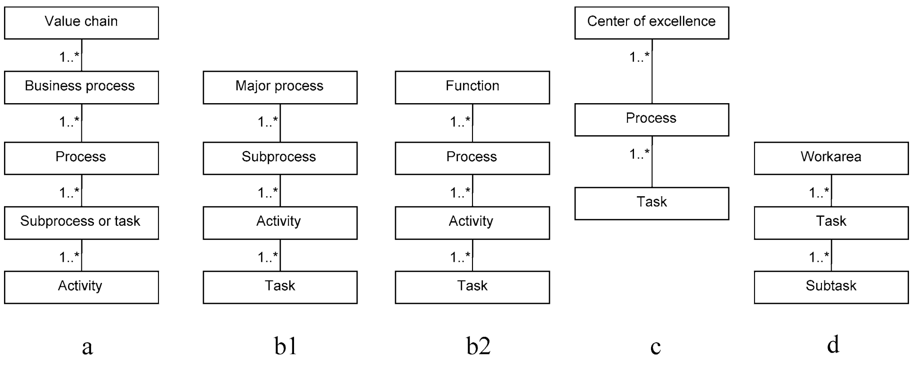


Figure 1. Use of different notations in business process description.

is illustrated in Section 4. Suitability of BPM tools to organisational BPM needs is discussed in Section 5. Brief conclusions and directions of future investigations are given in Section 6.

2. CONCEPTS USED FOR BUSINESS PROCESS DESCRIPTION

From the point of view of systems theory, business process is the set of activities that transform particular inputs into particular outputs. This implies that what is seen as a process in one level of abstraction could be regarded as an activity at a higher level of abstraction. Therefore it is not surprising that the use of such notions as business process, activity, task, etc. varies from one business process oriented community to another. In Figure 1 some examples of notations are given. The notations are used for business process descriptions, which may or may not include graphical business process models.

The set of notations shown in Figure 1*a* comes from P. Harmon's book "Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes"[5] and is supported by Business Process Trends portal.[6] Both sets of notion reflected in Figure 1 *b*1 and *b*2 appear in Business Process Improvement Workbook,[7] the set of notions given in 1*c* is supported by META Group.[8, 9] The set in Figure 1*d* comes from the practical application of BPM in requirements engineering.[10] Figure 1 shows that there is no a standard approach of how to call the phenomena at different levels of abstraction. However, when it comes to business modelling tools usually one and the same notations are used at all abstraction levels, because the number of levels is not limited.[11–14]

In this paper we will use term "process" at any level of abstraction i. We consider a process as a system of subprocesses (where the number of subprocesses is equal to or more than 1). Each subprocess at level i becomes a process at level $i - 1$. Notions "activity" and "task" will be used as synonyms for the notion "subprocess". This is a simplification that suits the purpose of the paper. However it has to be taken into consideration that contents of description in several approaches vary for process representations at different levels of abstraction. Examples of such approaches are given in Figure 2. In case reflected in Figure 2*a*,[10] processes at each level have unique attributes, in META group's case[8] shown in

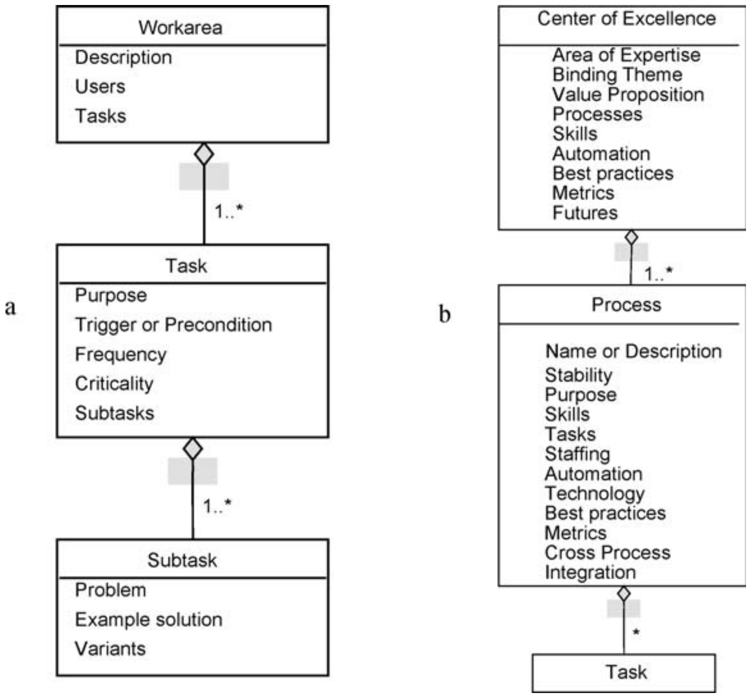


Figure 2. Detailed descriptions at different levels of abstraction.

Figure 2*b*, some of attributes (e.g., *Skills* and *Metrics*) are common for two levels, but some (e.g., *Area of Expertise* and *Integration*) are unique for each particular level of abstraction.

Depending on the purpose of process descriptions several business process classifications are used. From the point of view of substance of performer of the task three types of processes usually are considered:[15]

- Manual processes
- Semi-automated processes
- Automated processes

In the context of Web applications, processes are divided in the following three groups:[16]

- Private processes
- Abstract processes
- Collaboration processes

Private (internal) business processes are internal to a specific organisation. Abstract (public) business processes represent interactions between private business processes and other processes or participants. Only those activities that are used to communicate outside the private business processes are included in the abstract processes. Collaboration processes depict the interaction between two or more business entities. These interactions are defined as a sequence of activities that represent the message exchange patterns between the entities involved.

From the point of management level, the processes may be divided in operational and strategic processes.[17] Processes may be classified also according to their functional role or other selected criteria.[18–21]

Each of process classes mentioned above may require specific details in their descriptions. However, there are attributes that are common for all process classes. We will focus on those common attributes when discussing BPM tools and applications in further sections of the paper.

3. CAPABILITIES OF BPM TOOLS

By notion BPM tool we refer here to software systems meant for BPM. Number of such tools exceeds 300.[3] Therefore it is a challenge for the organisation to make informed choice of the BPM tool. In this section we will refer to features of the tools that, if considered, could be helpful when deciding whether to use the tool, and, if yes, which tool to choose. Popkin Software and Systems Inc.[22] suggest the following 11 criteria for choice of the business process reengineering tools:

- A common repository
- Ability to modify the repository
- Integrated process and data models
- Multiple approaches to modelling
- Comprehensive support in all modelling areas
- Flexible rule enforcement
- Ability to create detailed reports
- Full extensibility
- Complete functionality
- Multi-user product
- Low price with high functionality

The above-mentioned characteristics of tools are important not only for business process reengineering, but also for business process modelling. However they are quite vague and may be interpreted differently by various tool vendors. In addition, it is necessary to consider some other issues that may help to evaluate BPM tools. Advanced BPM tools fall into the category of CASE (computer aided systems engineering) tools and enterprise modelling tools. Issues of selection and effective use of those tools have been quite widely analysed.[23, 24] The results of the analysis comply and go beyond the criteria mentioned above and are applicable to BPM tools. However, they do not concentrate on the specifics of BPM. Therefore in this paper we will consider such features as expressiveness of the tool, conceptualisation capability, modelling flexibility, and representational effectiveness, which are essential in contemporary BPM applications. Those features are not independent properties of the tools, they are related each to other. Therefore equilibrium of these features is an integrated or emergent property of the BPM tools that actually helps to evaluate the suitability of a particular tool for a particular BPM situation.

3.1. Expressiveness of BPM Tools

Expressiveness of a BPM tool is characterised by scope and number of attributes to be reflected by business process model (diagram). However, comparison of tools on the basis of represented attributes is very difficult, because the names and meaning of the attributes vary from tool to tool according to business process description languages chosen (see also Figure 1 and Figure 2). Different researchers to evaluate expressiveness of the tools have used the following approaches:

- Evaluation on the basis of key attributes
- Evaluation on the basis of generic business process model
- Evaluation on the basis of distinguished features of a particular tool

Evaluation of BPM tools Axiom-SYS, BP Win, COSA Workflow, GRADE, Oracle Designer, Scitor Process, Silverrun-BM, and Workflow BPR on the basis of such key attributes as (1) multilevel representation, (2) simulation possibilities, and (3) possibility to reflect external entities, subprocesses, information flows, material flows, control flows, timers, triggering conditions, stores, performers, cost, and duration is presented in context of workspace modelling.[25] Attempts to provide a generic BPM framework are presented by several researchers.[16, 26, 27] Generic framework based comparison of business modelling methodologies GERAM, ARIS, CIMOSA, GRAI/GIM, IEM and PERA is given by K. Kosanke.[28] Comparative original metamodel based overview of business modelling languages EPC, BML, and SD is presented by a group of Swedish researchers.[27] Generic framework based comparison of business modelling techniques IDEF0, IDEF1, IDEFX, RAD, REAL, DM, OO, AI, and MAIS are described by F.-R. Lin, M.-Ch. Yang, and Y.-H. Pai.[26] The BPM tool comparisons based on distinguished features of a particular tool are usually given by the tool vendors.[22]

The evaluations mentioned above mainly focus on the presence of particular attributes of business process in the BPM languages or tools. But the presence may occur differently – the attribute may be given as a *text description* (T), it can be a *built in attribute* with a possibility to directly assign its value during the modelling (A), there may be a possibility to *establish link* to another business model where the attribute is reflected as an object (L), and there may be a possibility to *navigate* to the business model where the attribute is represented as an object (N). T and A here characterise direct expressiveness of the models while L and N characterise extended or indirect expressiveness of the models. Indirect expressiveness is a key for the next feature of BPM tools, namely cotextualisation capability.

3.2. Contextualisation Capabilities

By conceptualisation capability we understand here an extent to which the tool can show the business process in different contexts, such as strategy of organisation, organisational knowledge, computerisation of information processes, etc. This capability is directly related to the indirect expressiveness of the tool. We can talk about two levels of conceptualisation capability (1) manual conceptualisation capability, and (2) automatic conceptualisation capability. Manual conceptualisation capability means that there is a possibility to find relationships between business process elements and related organisational issues, i.e., it is possible to establish a link between the business process element and the issue be it reflected in another model or the document, or marked by pointer to a particular person. Automatic conceptualisation means that by clicking on a particular attribute of the business process the user navigates to the graphical business model where the attribute is represented as an object and highlighted among other objects included in this model.

Advanced business modelling environments, such as Casewise, ARIS, GRADE, ADONIS[11–13, 29] provide at least the so-called 3D automatic conceptualisation capability. 3D automatic conceptualisation capability means that navigation between organisation process model, organisational structure model, and data model is established. However, in different applications of BPM other than mentioned above automatic navigation capabilities may be needed. More details about these needs are given in Section 4.

Conceptualisation capability is closely related to the next feature of the BPM tools, namely – modelling flexibility.

3.3. Modelling Flexibility

Modelling flexibility is a term that may be interpreted in many different ways. In this paper by this term we mean a possibility to choose between several basic perspectives in BPM. Since the times of functional, mechanistic organisations it is common to view organisational structure as a more stable component than organisational process. Many BPM tools are built with this assumption implied. This means that, in case the performers of the process are shown in the graphical representation of the process, the performers are defined first and then processes assigned to them.[13] However in contemporary turbulent environment not always the performers of the process are known in advance. Therefore it is important to have a possibility to model process first and then assign performers. The result of the modelling is the same as in the previous case, but the process of obtaining the model requires different sequence of steps in knowledge development about business processes and different modelling capabilities of BPM tool. There are several tools that permit the structure-independent process modelling and still have a capability to reflect performers in the model. ARIS[11] and GRADE[12] are two examples of such tools.

3.4. Representational Effectiveness

Representational effectiveness shows how much information may be included in one screen or one page of business process model. Figure 3 illustrates this feature by comparing two possible representations of one and the same information. In case *a* the representation uses at least 2 times more space than in the case *b*. Case *a* does not focus on the process view, i.e., all the details are represented graphically. It is not the best way of information representation from the cognitive psychology point of view.[30] For easier and faster comprehension it is better to represent graphically only the main objects and add the details in the form of text (case *b*). There is a tendency to represent all business process information, including event branching, graphically in several BPM languages and tools.[11, 16] However, from the point of view of comprehension and effective use of modelling space, balanced approaches to information representation as well as several representations of one and the same modelling information are necessary. In this regard Leibniz prize winner

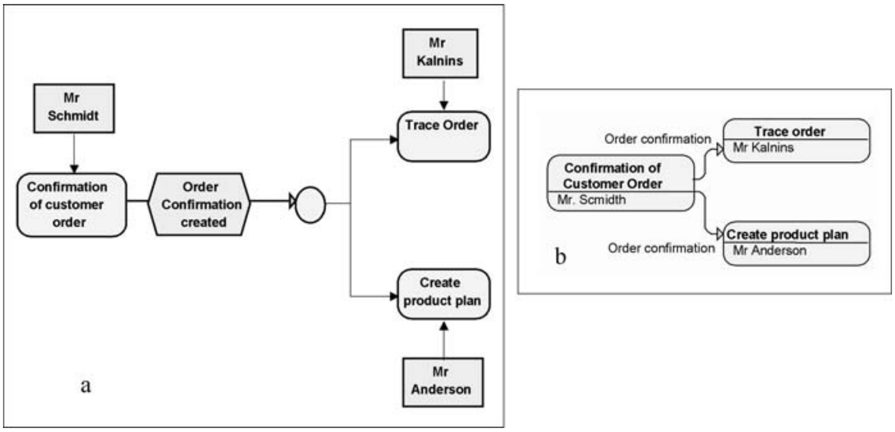


Figure 3. Different representations of one and the same information.

business modelling tool GRADE[12] is a good example of effective use of modelling space and multiple representations of modelling information.

Business process information may be needed at different levels of abstraction and detail, as well as focus on different issues relevant to organisations. Therefore capability of the tool to represent this information effectively is one of the essential properties to be considered when choosing the tool for organisational needs.

4. APPLICATIONS OF BPM

BPM first became popular during the era of business process reengineering. Today, over a decade later, BPM is used not only in business process reengineering but also in many other different areas.[5] In this section we will briefly discuss BPM in contexts of knowledge management, business process improvement, change management and information systems development.

4.1. BPM in Knowledge Management

The role of BPM in Knowledge Management is threefold. First business processes, if modelled, are a part of codified structural intellectual capital of the organisation.[31] Second, knowledge processes applied in an organisation are a part of business processes to be modelled.[21] Third, business process models may be used as a means for facilitation of knowledge processes, such as knowledge creation.[32] Knowledge management requires knowledge as one of contextual dimensions to be available for BPM. This dimension has two subdimensions, namely, the subdimension of tacit knowledge and subdimension of codified knowledge. Tacit knowledge is related to organisational structure, which may be used in parallel or sequentially with the knowledge dimension. Regarding codified knowledge subdimension, documents that comprise information regulating the business processes have a special role. Navigation possibilities from business model to these documents are one of desirable properties of BPM tools used in knowledge management.[33, 34]

The above mentioned three roles of business process modelling in organisational knowledge management require tools that provide possibility to model at different levels of abstraction and at different levels of detail. Modelling of knowledge also requires a capability of the tool to aggregate knowledge in lower levels of abstraction and represent it as a meta-knowledge at higher levels of abstraction. BPM tools rarely meet this requirement because they are mainly design for top-down modelling.

4.2. BPM in Business Process Improvement and Change Management

Some recent surveys show that about 83% of companies are engaged in business process improvement and redesign.[5] This is one of the main reasons of renaissance of the business process modelling tools. To improve the process, it is necessary to discuss it. Graphical representations are helpful tools for sensemaking regarding the processes.[32] On the other hand, changes in one process may cause changes in many other processes. When business processes are not modelled, those additional changes in many cases are unexpected by organisation and require use of resources for firefighting. Sophisticated BPM tools give an opportunity to make business processes transparent for decision makers and consider possible consequences of particular changes.[35] One more factor that facilitates BPM activities in organisations is quality standards that require business process descriptions. When descriptions are already there, the process of actual graphical modelling is less time consuming and uncommon than in situations when the models must be built from scratch.

Globalisation, extensive use of information technologies, and other present day phenomena suggest that business process improvement and redesign are going to be continuous phenomena in organisational activities. Business process improvement is one of the ways in which organisations can stay competitive in a turbulent environment. Business process redesign is needed in each case when internal or external changes hinder current organisational work distribution and established flows of activities. Therefore contemporary BPM tools are expected not only to reflect the business process at several levels of decomposition and simulate it but also reveal such information as conformance to standards, viewpoint diversity, complexity, etc. We surveyed nine popular BPM tools according to more than 30 properties and found that they differ considerably in their capabilities to represent business process estimates and contexts while almost all of them provide all basic BPM facilities (see Section 3.1).

4.3. BPM in Information Systems Development

One more area where business modelling becomes increasingly popular is information systems development. Types of business process models used in information systems development depend on development methodologies. In agile systems development methods easy to manage tools or even none software tools are used. Conversely, model driven approaches rely on complex, expressive modelling environments. BPM is essential in the following information systems development cases:

- Enterprise Resource Planning (ERP) systems acquisition
- Business process driven information systems development
- Use of BPM for requirements analysis

- Requirements generation from business process models

Each ERP system has a business process model behind it. It is beneficial to be able to compare organisational processes with this model when deciding which ERP system to acquire. This comparison is possible only if business process models of both ERP system and the organisation are available. Textual descriptions of the processes in this case do not provide effective means for decision making.[36]

Model driven development, which is based on sophisticated systems development tools capable to perform many automatic transformations of diagrams, is one of the recent tendencies in the information systems development.[37] The most advanced and challenging approach among the model driven ones is business process model driven information systems development.[38]

One of the desirable properties of information systems requirements is their consistency. If requirements are generated from business process models then the request for consistency is satisfied. However, generation of requirements in terms of graphical program specifications,[12] use cases[12, 29] or generation of XML code[16] from the business process model requires BPM languages and tools that are capable to represent all details needed for requirements or XML code generation. Another way of supporting the consistency of requirements is to represent them in business process model for consistency checking.[39, 40]

5. HOW TO CHOOSE THE BPM TOOL

Having more than 300 commercial BPM tools the choice of the right one for organisational needs is a task of high complexity. Sections 3 and 4 of this paper were written to introduce a spectrum of issues to be considered in tool acquisition process and to provide references to sources of more detailed information. We suggest the following 3-step procedure for BPM tool acquisition:

1. Decide whether organisation needs the BPM tool at all
2. Define required functionality of the tool
3. Choose between tools that satisfy defined functionality taking into consideration such factors as pricing policy, vendor support, and organisational context and consequences of tool acquisition.[24]

In deciding whether the BPM tool is needed or not, several threats of BPM tool use[41] are to be considered. Table 1 in column 1 reflects the list of threats highlighted by agile systems development community.[41] In column 2 we give some comments, which are based on our theoretical research and ten-year experience of BPM tool use.

Required functionality of the tool may be defined on the basis of knowledge of BPM tool capabilities in general and particular needs of organisation (see Sections 2 and 3). It is a big difference whether organisation needs just a tool for drawing boxes and lines between them in brainstorming sessions, or it wants to monitor its business processes by maintaining and analysing their models.

The third step is fully common sense based. BPM tools are like people – each has its own ‘personality’. It is almost impossible to compare many tools to one another without introducing subjective bias. Therefore, when analysing suitability of available tools to or-

Table 1. Comments on BPM tool acquisition threats

N	Threat	Comment
1.	Initial training and education	Tools differ in skills needed for their use. Education in tools use is provided by several universities[42], hence more and more business and information specialists are familiar with BPM tools.
2.	Evaluation costs	Vendors of tools differ in their strategies that allow tool evaluation. Some provide demo and evaluation versions[11, 12, 13]; others allow evaluating the tool only in costly presence of consultants [29].
3.	Maintenance of the model over time	There are two issues to be considered: (1) conceptual maintenance, i.e. the organisational effort is needed to maintain business process model consistent with the current business processes; (2) technical maintenance – a tool shall have a model maintenance capability. Almost all modelling environments support version control of models, but simple drawing tools may not have this feature.
4.	Upgrade costs of the tool	Depends on the vendor. They may vary from 0 EUR to thousands of EUR.
5.	Ongoing usage/maintenance fees	Depend on the vendor.
6.	Time lost waiting for the tool to do its job	Only for low quality tools. Such tools as ARIS[11], Casewise[13], GRADE[12], etc. do their job at the speed of word processors.
7.	Time used over-using the tool	Many tools provide facilities for effective model design. It is easier to change the model electronically than to do it manually. Time depends also on users' reasonability and skills.
8.	Migration costs to port models to another tool	Should be compared with costs of building models from scratch.
9.	Increased effort to synchronise models with other artifacts, such as source code	Depends on the tool and modelling skills. Many BPM tools have built-in synchronisation modules that considerably decrease not increase the synchronisation efforts
10.	CASE tools often promote syntax over communication between developers	Yes, if the tool has no capability to represent business process at several levels of abstraction. Otherwise it depends only on the skills of the user. On the other hand the tool with its built in expressiveness and contextualisation capability may promote consideration of forgotten aspects during the modelling sessions
11.	Generated code often too simplistic, or cluttered with extraneous information required by the tool	(1) Code generation not always is needed in BPM. (2) Different tools have different code generation capabilities.
12.	Poor user interfaces often hamper the modelling effort	There are tools with poor interfaces. But there are also tools with very good interfaces and even with distinguished interfaces (e.g., GRADE[12])
13.	Inadequate integration with other tools reduces productivity and/or requires integration work	This is the question to be answered: "What is better for the organisation - 'other tools' integrated with developers brain or 'other tools' integrated with business process models?"
14.	Complex tools often prevent the inclusion of non-developers in modelling efforts	We shall distinguish between complexity of the tool and complexity of use of the tool. Many complex BPM tolls, such as GRADE[12], Casewise[13], etc. do not prevent inclusion of non-developers. They promote rather than hinder the inclusion of non-developers by capability to provide navigable HTML versions of models.

ganisational needs, it is better to consider match between organisation and tool rather than comparative characteristics of the tools. Still we suggest studying at least one of advanced business modelling tools before making any decisions. ARIS,[11] Provision,[14] Casewise,[13] and GRADE[12] are good choices because they have high capabilities and helpful free evaluation versions.

6. CONCLUSIONS

In this paper we amalgamated some issues on our current research on BPM tools. Renaissance of business process modelling, exposed by rich variety of BPM application areas and large number of BPM application cases, which we are experiencing in the beginning of the new millennium, is caused, on one hand, by organisational needs for business modelling in order to survive in culturally, economically and technologically turbulent environment; on the other hand, it is caused by the fact that the quality of commercial tools for BPM has considerably increased during the last decade. Availability of many commercial BPM tools calls for methods and methodologies for tool acquisition. The paper suggests some issues to be considered and guidelines to be applied in tool acquisition. However, it is not yet a methodology. Our further research is aimed at development of BPM tools knowledge base and ontology of tools evaluation. Both of them could form the basis of tool acquisition methodology.

REFERENCES

1. M. Hammer and J. Champy, *Re-engineering the Corporation: A Manifesto for Business Revolution* (Ballinger, Cambridge, MA, 1993).
2. List of CASE tools. Available at http://www.cc.queensu.ca/Software-Engineering/tools.html (accessed June 25, 2004).
3. Briefing Business Process Modelling Tools. Available at http://www.bpmg.org/classic/Articles_CaseStudies/Briefing-ProcessModellingTools.htm (accessed June 25, 2004).
4. Business Process Modelling and Standardisation. Available at http://www.cimosa.de/Standards/BPM_and_Standardisation.pdf (accessed June 25, 2004).
5. P. Harmon, *Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes* (Morgan Kaufman Publishers, 2003).
6. Business Process Trends portal. Available at http://www.bptrends.com (accessed June 25, 2004).
7. H. J. Harrington, E. K. C. Esseling, and H. van Nimwegen, *Business Process Improvement Workbook: Documentation, Analysis, Design, and Management of Business Process Improvement* (1997).
8. A. Bruce and D. Kutnick, *Building Operational Excellence: IT People and Process Best Practices* (Pearson Educational, 2002).
9. META Group portal. Available at http://www.metagroup.com/us/home.do (accessed June 25, 2004).
10. S. Lausen, Task descriptions as functional requirements, in: *IEEE Software* (March/April, 2003), pp. 58–65.
11. ARIS toolset. Available at http://www.ids-scheer.com (accessed June 25, 2004).
12. GRADE tools. Available at http://www.gradetools.com (accessed June 25, 2004).
13. Casewise software. Available at http://www.casewise.com (accessed June 25, 2004).
14. ProVision enterprise modelling software. Available at http://www.proformacorp.com/ (accessed June 25, 2004).
15. Jenz and Parner portal. Available at http://www.jenzundpartner.de (accessed June 25, 2004).
16. Business Process Notation, (Draft 0.1), Business Process Management Initiative, 2003. Available at http://www.bmi.org (accessed June 24, 2004).
17. R. Normann, *Reframing Business: When the Map Changes the Landscape* (John Wiley and Sons, Ltd., 2001).
18. APQC Process Classification Framework. Available at http://www.apqc.org/portal (accessed June 25, 2004).
19. Th. W. Malone, K. Crowston, and G. A. Herman Eds., *Organizing Business Knowledge: The MIT Process Handbook* (MIT Press, 2003).
20. P. Loucopoulos, The S3 (Strategy-Service-Support) framework for business process modelling, in: *Proceedings of CAiSE'03 Workshops, in connection with The 15th Conference on Advanced Information Systems Engineering*, edited by J. Eder, R. Mittermeir, and B. Pernici (University Maribor Press, Klagenfurt/Velden, Austria, June 16–20, 2003), pp. 378–382.
21. R. Woitsch and D. Karagiannis, Process-oriented knowledge management systems based on KM-services: The PROMOTE approach, in: *The Practical Aspects of Knowledge Management, Proceedings of the*

4th International Conference, edited by D. Karagiannis and U. Reimer (Springer, Vienna, Austria, December 2002), pp. 398–412.

22. The 11 Most Important Features to Look For in a BPR tool, Popkin Software & Systems; http://www.popkin.com (accessed June 25, 2004).
23. J. Stirna, Choosing Strategy for Enterprise Modelling Tool Acquisition, Department of Computer and Systems Sciences, Stockholm University and Royal Institute of Technology, Report Series No. 99–102, 1999.
24. W. J. Orlikowski, CASE tools as organisational change: Investigating incremental and radical changes in systems development, *Management Information Systems Quarterly* **17**(3) (1993).
25. M. Kirikova, Modelling the boundaries of workspace: A business process perspective, in: *Information Modelling and Knowledge Bases XIII*, edited by H. Kangassalo, H. Jaakkola, E. Kawaguchi, and T. Welzer (IOS Press, Ohmsha, Amsterdam, Berlin, Oxford, Tokyo, Washington, DC, 2002), pp. 266–278.
26. F-R. Lin, M-Ch. Yang, and Y-H. Pai, A generic Structure for business process modeling, *Business Process Management Journal* **8**(1), 19–41 (2002).
27. E. Söderström, B. Andersson, P. Johannesson, E. Perjons, B. Wangler, Towards A framework for comparing process modelling languages, in: *The Fourteenth International Conference on Advanced Information Systems Engineering* (CAiSE02, Toronto, Springer LNCS, 2002).
28. K. Kosanke, Comparision of Modelling Methodologies, 1996. Available at http://cimosa.cnt.pl/Docs/cmm.htm (accessed June 25, 2004).
29. ADONIS – the business process management tool. Available at http://boc-eu.com/index.html (accessed June 25, 2004).
30. J. R. Anderson, *Cognitive Psychology and Its Implications* (Freeman and Champy, 1995).
31. D. Apshvalka and J. Grundspenkis, Making organisations to act more intelligentlyin the framework of organisational knowledge management system, in: *Scientific Proceedings of Riga Technical University on Computer Science*, Vol. 17 (RTU, Riga, 2003), pp. 72–82.
32. A. Persson, and J. Stirna, Creating an organisational memory through integration of enterprise modelling, patterns and hypermedia: The hyperknowledge approach, in: *Information Systems Development: Advances in Methodologies, Components, and Management*, edited by M. Kirikova, J. Grundspenkis, W. Wojtkowski, W. G. Wojtkowski, S.Wrycza, and J. Zupancic (Kluwer Academic/Plenum Publishers, 2002), pp. 181–192.
33. M. Kirikova, Facilitating comprehension of normative documents by graphical representations, in: *Practical Aspects of Knowledge Management*, edited by D. Karagiannis and U. Reimer (Springer Verlag, Berlin Heidelberg, 2002), pp. 369–376.
34. M. Kirikova and J. Vanags, A systemic approach for managing normative acts: a business modelling perspective, in: *Transactions in International Information Systems: Systems Analysis and Development Theory and Practice*, edited by A. Nowicki and J. Unold (Wroclaw University of Economics, Wroclaw, 2001), pp. 39–58.
35. M. Kirikova, Conversion of inventions into requirements for computer based information systems, in: *Scientific Proceedings of Riga Technical University, Series: Computer Science, Applied Computer Systems – 4th Thematic Issue* (RTU, Riga, 2003), pp. 55–61.
36. K. Slavinska, Introduction of ERP system. Engineer thesis (Riga Technical University, 2003) (in Latvian).
37. O. Nikiforova and M. Kirikova, Two-hemisphere model driven approach: engineering based software development, in: *The proceedings of the 16th Conference on Advanced Information Systems Engineering*, (CAiSE'2004, Riga Latvia, June 7–11, 2004) (to be published).
38. ArcStyler MDA-Business Transformer Modelling Style Guide for ARIS, Interactive Objects, 2002.
39. M. Kirikova, Business Modelling and Use Cases in Requirements Engineering, in: *Information Modelling and Knowledge Bases XII*, edited by H. Jaakkola, H. Kangassalo, and E. Kawaguchi (IOS Press, Ohmsha, Amsterdam, Berlin, Oxford, Tokyo, Washington, DC) pp. 410–420.
40. G. Sparks, The Business Process Model, Enterprise Architect, 2000. Available at http://www.sparxsystems.com.au (accessed June 25, 2004).
41. Sc. W. Ambler, It's "Use the Simplest Tool" not "Use Simple Tools". Available at http://www.agilemodeling.com/essays/simpleTools.htm (accessed June 25, 2004).
42. M. Kirikova, Bridging the educational gap between requirements holders and requirements engineers, in: *Proceedings of the 4th IEEE Internacional Baltic Workshop on Databases and information Systems*, Vol. 2, edited by A. Caplinskas (Technika, Vilnius, 2000), pp. 35–46.

DOES THE PERCEIVED QUALITY OF AN ELECTRONIC GROCERY STORE EXPLAIN THE BUYING BEHAVIOR OF ITS CUSTOMERS?

An exploratory field study

Osmo Kurkela and Juhani Iivari*

1. INTRODUCTION

Grocery shopping has been suggested as one potential application area for e-commerce (Kutz, 1998). Yet, groceries are one of the most difficult areas of trade for e-commerce, because they are local, the physical delivery aspect is critical, an average purchase basket consists of many items and the value-to-weight ratio of purchases is low (Heikkilä et al., 1998; Raijas and Tuunainen, 2001). On the other hand, groceries form the largest section of retailing, shopping is frequent, buying patterns are fairly stable, and the shopping behavior of most customers is more or less habitual and automatic, being based on earlier experiences (Raijas and Tuunainen, 2001). Assuming that grocery shopping is quite routine, it probably does not have the same enjoyment aspect (Dawson et al., 1990) as shopping in fashion boutiques, electronics shops, or antique bookstores, for example. Therefore there is a great potential for electronic grocery stores (Raijas and Tuunainen, 2001). Nevertheless, Geuens et al. (2003) found that Belgian consumers are still very conservative as far as their grocery shopping is concerned, although they do not like it. Grewal et al. (2004) also report that despite a prominent start, home grocery deliveries now appear to have somewhat fizzled out. All these factors make online groceries intellectually an interesting application area for e-commerce, although perhaps not economic successes.

The purpose of this paper is to analyze the real buying behavior of electronic grocery store customers, and especially the impact of perceived quality of a grocery store on this behavior. Electronic grocery shopping may be interpreted quite widely as a facility enabling consumers to order groceries from home electronically (by phone, fax or Internet)

* Osmo Kurkela, Oulu Polytechnic, Raahe Institute of Computer Engineering and Business Rantakatu 5A, P.O. Box 82, FIN-92101 Raahe, Finland, osmo.kurkela@ratol.fi. Juhani Iivari, Department of Information Processing Science, University of Oulu, P.O. Box 3000, 90014 Oulun yliopisto, Finland, juhani.iivari@oulu.fi.

and to obtain subsequent delivery to their home (Verhoef and Langerak, 2001). This paper applies the concept in a more specific meaning to refer to electronic grocery shopping using the Internet. On the other hand, we do not require delivery to the customer's home, as a customer may also pick the products up from the store. The paper is an exploratory field study. The results are based on experiences with one electronic grocery store launched in a major city in Finland.

2. PREVIOUS RESEARCH

Since electronic grocery shopping on the Internet forms quite a new phenomenon, it is understandable that there is not much empirical research available on it (Verhoef and Langerak, 2001), especially from the customer's viewpoint. Most articles on the topic are descriptive by nature (Heikkilä et al., 1998; Ahola, 1999; Heikkila et al., 1999; Ahola et al., 2000; Morganosky and Cude, 2000; Oinas-Kukkonen, 2000; Palmer et al., 2000; Raijas and Tuunainen, 2001; Rohm and Swaminathan, 2004).

There is a dearth of more theoretically oriented studies of electronic grocery shopping. Henderson et al. (1998) investigated the association between a number of variables (perceived usefulness, perceived enjoyment, peer-group norms, usability, perceived shopping experience) on the intention to use such a system in the future, and found only perceived enjoyment and peer-group norms to be significant predictors of this intention. Verhoef and Langerak (2001) examined (1) the relationship between three (dis)advantages of electronic grocery shopping (i.e. physical effort, time pressure and shopping enjoyment), as compared with traditional in-store shopping, and consumers' perceptions of innovation characteristics (i.e. relative advantage, compatibility, and complexity), and (2) the relationship between perceptions of innovation characteristics and consumers' intentions to adopt electronic grocery shopping. They found physical effort and time pressure to have a significant effect on the relative advantage and time pressure to affect compatibility. Further, all three innovation characteristics significantly affected the intention to adopt electronic grocery shopping. Childers et al. (2001), in their analysis of attitudes towards online shopping in the case of groceries, found perceived usefulness, ease of use and enjoyment to be significant predictors of this attitude. Furthermore, navigation, convenience, and the substitutability of the electronic environment for personal examination of the products were found to be important predictors of the three antecedents of attitudes.

3. THEORETICAL BACKGROUND AND RESEARCH MODEL

There is no generally accepted measure of the quality of an electronic grocery store, although there are an increasing number of validated instruments for measuring the quality of a website (Agarwall and Venkatesh, 2002; Aladwani, 2002; Muylle, 2003). All these instruments reflect only the viewpoint of a website user, however, not of a consumer of the products and services sold through the website. Our basic assumption is that the decision to use or not use a website for shopping does not depend on the quality of the website alone but also on the quality of the products and services sold through it.

3.1. The Concept of the Quality of a Grocery Store

At the time we initiated the study reported in this paper, none of the above instruments had been published. To start with, we decided to consider an electronic grocery store as an IT artifact and its use as an innovation as far as the customer is concerned. These views led us to focus on previous research into the quality of information systems (DeLone and McLean, 1992), characteristics of innovations (Moore and Benbasat, 1991; Rogers, 1995) and more indirectly the Technology Acceptance Model (TAM) (Davis, 1989; Davis et al., 1989).

DeLone and MacLean (1992) propose a framework for identifying six success measures: system quality, information quality, use, user satisfaction, individual impact and organizational impact. Because our focus lies in electronic grocery stores, which are mainly B2C commerce, we did not consider organizational impact/effectiveness as relevant here.* We therefore focus only on system quality, information quality, user satisfaction and individual impact as perceived by customers.† Our assumption is that the technical aspects of system quality are not of interest to customers, and thus we focus only on the usability aspect of system quality, and more specifically on perceived ease of use (Davis, 1989).

In the case of user (information) satisfaction, we decided to apply the concept liberally to capture satisfaction with a number of dimensions of an electronic grocery store. A list of quite conventional aspects of grocery stores (product assortment, quality of fresh products, prices, service, presentation of products, opening times) was complemented by more specific aspects of electronic trading (method of payment, Internet use, safety and privacy). Many of these dimensions have also been reported as relevant in the context of electronic grocery stores (Henderson et al., 1998; Raijas and Tuunainen, 2001.)

DeLone and MacLean (1992) characterize individual impact as "an indication that an information system has given a user a better understanding of the decision context, has improved his or her decision-making productivity, has produced a change in user activity, or has changed the decision maker's perception of the importance or usefulness of the information system" (p. 69). Davis (1989) defines perceived usefulness in quite a analogous way: "the degree to which a person believes that using a particular system would enhance his or her job performance can be interpreted as users' beliefs or expectations about the individual impact of the system, focusing specifically on the impact on individual job performance."(??)

If an innovation is interpreted as "an idea, practice, or object that is perceived as new by an individual or other unit of adoption" (Rogers, 1995, p. 11), electronic grocery stores in the current stage of their diffusion can be considered an innovation as far as their customers are concerned. Based on the DOI literature, we decided to focus on relative advantage (perceived usefulness) and complexity (perceived ease of use). Even though potentially relevant to the adoption of (IT) innovations, we did not consider the other innovation characteristics to describe the quality of an electronic grocery store as such. More specifi-

* One could, of course, have considered the impact of the family as a kind of organization to which most customers belong.

† We exclude use here because we consider it a dependent variable possibly affected by the quality of the electronic grocery store rather as an aspect of its quality.

cally, we excluded compatibility, because it is unclear whether this should be evaluated in terms of traditional grocery shopping or in terms of the electronic way of doing it.

To sum up, both the literature on IS quality and the literature on innovation characteristics led us to identify perceived usefulness and perceived ease of use, as defined in TAM, as relevant aspects of the perceived quality of an electronic store. This was complemented with perceived satisfaction, extended to cover the quality of the products and services, for the case of a grocery store.

3.2. The Research Model

The model employed for the present research is depicted in Figure 1. In addition to the perceived quality of an electronic grocery store, it identifies a number of household and customer characteristics as control variables which may affect the adoption of an online grocery store. The adoption units in the present study are individual consumers. Rogers views adoption events as dichotomous decisions that do not take into consideration the extent of the adoption. Cooper and Zmud (1990) introduce the concept of "infusion" to address the problem of the extent of adoption. They interpret infusion of the product as the extent to which the innovation is applied in terms of its fullest potential.

In view of the novelty of electronic grocery stores, we decided to consider a customer's buying behavior as an indicator of adoption and its infusion. One can distinguish two aspects in customers' buying behavior: buying frequency and the average sum spent. Even though these can be multiplied together to give the total sum spent, we will keep them as separate indicators of buying behavior, because it is unclear whether they are positively or negatively correlated with each other or for practical purposes uncorrelated. It may also be that the frequency of shopping in an electronic grocery store and the average sum spent are affected by different factors.

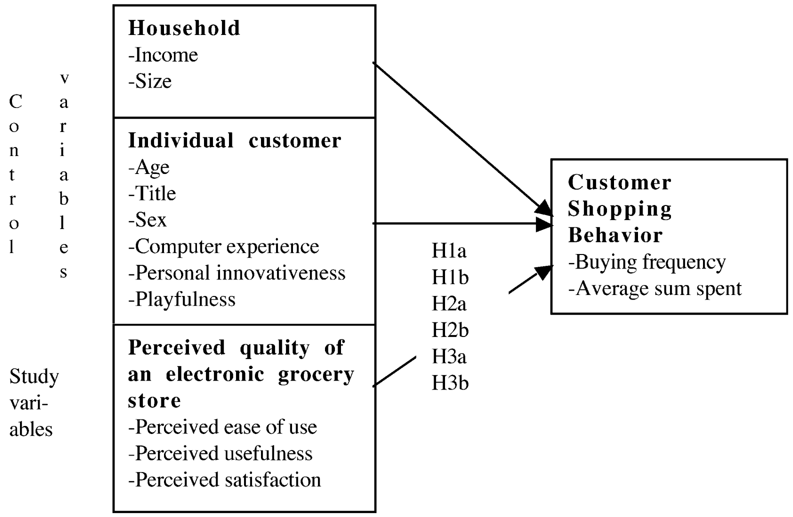


Figure 1. The research model explaining electronic grocery shopping behavior.

As explained above, DOI theory (e.g. Rogers, 1995; Torkzadeh and Klein, 1980) and its application to electronic grocery stores (Verhoef and Langerak, 2001) are suggestive of innovation characteristics, most notably relative advantage (perceived usefulness) and complexity (perceived ease of use), and compatibility, all of which affect the adoption of an innovation. This theory leads to four hypotheses:*

H1a The perceived usefulness of an electronic grocery store is positively associated with buying frequency.
H1b The perceived usefulness an electronic grocery store is positively associated with the average sum spent.
H2a The perceived ease of use an electronic grocery store is positively associated with buying frequency.
H2b The perceived ease of use an electronic grocery store is positively associated with the average sum spent.

These hypotheses are also partly supported by the Technology Acceptance Model (TAM), which was developed to predict the adoption (use) of IT innovations (Davis, 1989; Davis et al., 1989). One should note, however, that the purpose of the present study is not to test TAM, even though there is some overlap because of its overlap with DOI theory (Moore and Benbasat, 1991).

Our hypotheses concerning the association between perceived satisfaction with an electronic grocery store and customer shopping behavior are based on research into customer satisfaction in marketing, where it is assumed to affect the likelihood of repeated purchasing of a product or service. The meta-analysis of customer satisfaction by Szymanski and Henard (2001) reports a positive relationship between the two.

H3a Perceived satisfaction with an electronic grocery store is positively associated with buying frequency.
H3b Perceived satisfaction with an electronic grocery store is positively associated with the average sum spent.

To test the possible impact of the perceived quality of an electronic grocery store on customers' shopping behavior it is necessary to control other variables that could potentially affect this behavior. Income and family size are clearly such variables at the household level (Morganosky and Cude, 2000; Raijas and Tuunainen, 2001), and age, sex and position are also potentially relevant at the individual level (Morganosky and Cude, 2000; Raijas and Tuunainen, 2001). We also included personal innovativeness (Agarwal and Prased, 1998) and computer playfulness (Webster and Martochhio, 1992).

* We referred above to the problem that it is unclear whether compatibility should be evaluated in terms of traditional way of grocery shopping or in terms of electronic way doing it. The short-form instrument for compatibility proposed by Moore and Benbasat (1991) includes items such as "Using a XXX is compatible with all aspects of my work", "I think that using XXX fits well with the way I like to work" and "Using XXX fits into my work style". If we substitute "shopping" for "work" in the items, it is unclear especially in the case of the first and the last item whether the referent should be grocery shopping in the traditional way or using an electronic option. As a consequnce it is unclear whether compatibility should be hypothesized to affect the shopping behavior positively or negatively.

4. THE RESEARCH METHOD

4.1. The Field Study

The model of Figure 1 was tested in a field study analyzing the use of an electronic grocery store (EGS) in a major Finnish city. The choice of one organization controls for the possible confounding effects of organizational level variables such as institutional constraints and infrastructure arrangements, which may have an influence on individual adoption and acceptance, making it more likely that micro-level effects will be detected (Karahanna et al., 1999).

Data about the independent variables in Figure 1 were collected using a questionnaire survey. The first online survey, administrated in November and December 1999, was answered by a total of 56 customers, 30% of the total active customer base of the EGS. Of these, 15 answered through the Internet and 41 by mail. The second online survey, administrated in two phases, in April and May 2001 and November and December 2001, elicited 50 replies, giving a total number of 106. Participation was voluntary, of course, and people were assured that their individual replies would be treated as confidential. All the customers had prior experiences of the EGS, which had been initiated as a value-added service for an existing store and was operated as a retail outlet for one of the major wholesale companies in Finland. To purchase online, a customer has to have the wholesaler's membership card. The charge for home delivery was 13.46 Euro, and the cost of picking the goods up from the store was 6.73 Euro.

4.2. Measurement of the Variables

Perceived quality of the electronic grocery store

Perceived ease of use was measured on a Likert scale, using six items modified from Davis (1989). The reliability of the scale was 0.87. Perceived usefulness was similarly measured using six items modified from Davis (1989), the reliability of the scale being 0.81.

The customer's satisfaction with the electronic shopping was measured with eleven items, with factor analysis by the principal components method with varimax rotation to reduce the original number of variables to a smaller set. This gave four factors:

1. Satisfaction with safety, information privacy, and network use.
2. Satisfaction with methods of payment, presentation of products, and opening times.
3. Satisfaction with the fresh of products, service, and delivery.
4. Satisfaction with assortment and prices.

The measures for satisfaction and its four factors are formative (see Section 4.3), which means that reliability measures such as Cronbach alpha are not meaningful in this case.

Buying behavior

We had access to data on the actual buying behavior of the EGS customers covering the period from 20.11.1999 to 20.2.2002, which made it possible to analyze their frequency of shopping, average sums spent and total sums spent. We used data for a six-month period to represent buying behavior during each round of the survey.

Control variables

Family income, size of the household, and respondent's age, position and sex were measured by one item. Computer experience was measured in terms of skills in word processing and spreadsheet use, e-mail, and electronic banking through the Internet.

The measure of personal innovativeness was adopted from Agarwal and Prased (1998), who define this construct in the domain of information technology as "the willingness of an individual to try out a new information technology". The reliability of the four-item measure was 0.88. The measure of playfulness was adopted from Werbster and Martochhio (1992), adapting it to playfulness with the World Wide Web (WWW) rather than microcomputers in general. The reliability of the scale was 0.90.

4.3. Data Analysis

The hypothesized relationships among the study variables depicted in Figure 1 were tested by the PLS method, which is particularly well suited for predictive applications and theory building (Chin and Newsted, 1999). It does not imply parametric assumptions of multivariate normal distribution, and the sample size can be small, the minimum being ten times the number of items in the most complex construct in the model (Chin, 1998; Gefen et al., 2000).

PLS recognizes two components of a causal model: a measurement model and a structural model. A structural model consists of the unobservable, latent constructs and the theoretical relationships among them. Testing this includes estimating the path coefficients, which indicate the strengths of the relationships between the independent and dependent variables. Furthermore, for each construct in a structural model, there is a related measurement model which links the latent construct in the diagram with a set of observed items.

PLS distinguishes two types of measures (Chin, 1998): reflective indicators and formative indicators. Reflective indicators measure the same latent variable. Good reflective indicators should be highly correlated, as implied by Cronbach alpha. Formative indicators are not necessarily highly correlated, but instead they are viewed as cause variables that provide the conditions under which the latent variables they are connected with are formed (Chin, 1998). Computer experience and satisfaction (factors 1–4) are formative measures in our model, whereas perceived ease of use, perceived usefulness, playfulness and personal innovativeness are reflective measures.

More specifically, a molar approach (Bagozzi, 1985; Chin and Gopal, 1995) was adopted to test the model in Figure 1. Total satisfaction was considered a second order concept influenced by four factors of satisfaction (see Figure 2). To measure total satisfaction, the original eleven items referring to satisfaction were used.

5. RESULTS

5.1. Measurement Model

The model of Figure 1 was tested using PSL-Graph, version 03.00 software. To test the measurement model, we examined (1) individual item loadings, (2) internal consistency (reliability of measures), (3) convergent validity, and (4) discriminant validity.

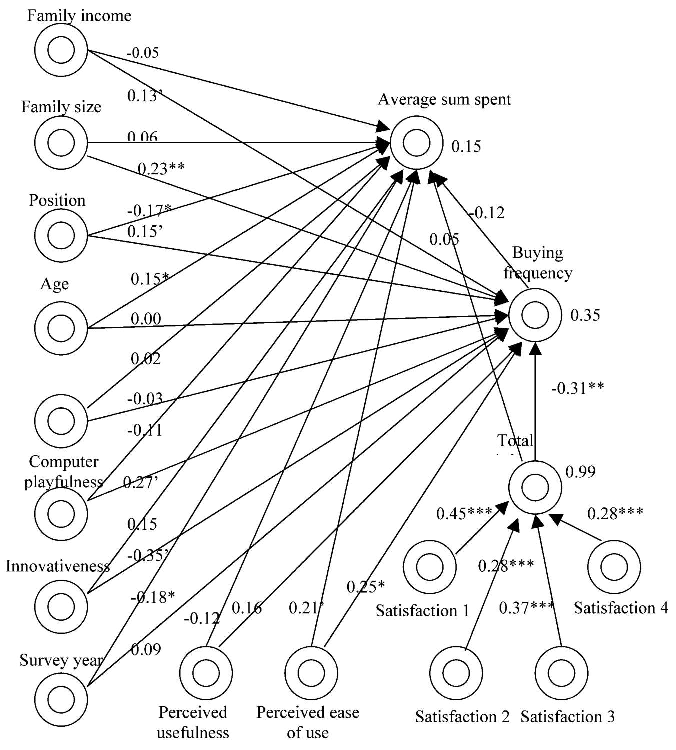


Figure 2. Results of PLS (Partial Least Square) structural analyses.

Of the 48 item loadings, only 22 had absolute values above the threshold of 0.7. Of the 26 violations, eleven concerned total satisfaction, five computer playfulness, two personal innovativeness, two software experience, one perceived usefulness, three satisfaction 2, one satisfaction 3, and one satisfaction 4. Many of these 15 violations were close to 0.70, so that only eight had loadings lower than 0.65. The internal consistencies of all the reflective measures clearly exceeded the cut-off value of 0.70. Convergent validity is considered adequate when the average variance extracted is 0.50 or more, and this condition was satisfied with four exceptions: total satisfaction (average variance extracted 0.30), computer playfulness (0.38), computer experience (0.44) and satisfaction 2 (0.47). For satisfactory discriminant validity, the average variance shared between a construct and its measures should be greater than the variance shared by the construct and other constructs in the model (Chin 1998). Four violations were discovered (between total satisfaction and the four factors of satisfaction).

5.2. Structural Model

The tests performed on the structural models gave the results depicted in Figure 2. The bootstrap resampling technique (500 resamples) was used to determine the significance of the paths within the structural model.

As shown in Figure 2, perceived usefulness is not a significant predictor either of buying frequency or of the average sum spent. Perceived ease of use, on the other hand, is a significant predictor of buying frequency ($\beta = 0.25$, $p \leq 0.05$) and an almost significant predictor of the average sum spent ($\beta = 0.21$, $p \leq 0.10$). Total satisfaction with electronic grocery shopping is a significant predictor of buying frequency ($\beta = -0.31$, $p \leq 0.01$), but surprisingly the relationship is negative. On the other hand, satisfaction does not have any significant impact on the average sum spent.

Satisfaction consists of four factors: satisfaction with safety, information privacy, and network use (satisfaction1) ($\beta = 0.45$, $p \leq 0.001$), satisfaction with methods of payment, presentation of products, and opening times (satisfaction 2) ($\beta = 0.28$, $p \leq 0.001$) and satisfaction with fresh products, service, and delivery (satisfaction 3) ($\beta = 0.37$, $p \leq 0.001$) and satisfaction with assortment and prices (satisfaction 4) ($\beta = 0.28$, $p \leq 0.001$).

Among the control variables, personal innovativeness is an almost significant predictor of buying frequency ($\beta = -0.35$, $p \leq 0.10$), but not of the average sum spent, as also is computer playfulness ($\beta = 0.27$, $p \leq 0.10$ for buying frequency). Computer experience has no significant influence on either indicator of buying behavior, while age has a significant influence on the average sum spent ($\beta = 0.15$, $p \leq 0.05$), but not on buying frequency. Position has a significant impact on the average sum spent ($\beta = -0.17$, $p \leq 0.05$), and an almost significant influence on buying frequency ($\beta = 0.15$, $p \leq 0.10$), whereas family size has a significant influence on buying frequency ($\beta = 0.23$, $p \leq 0.01$), but not on the average sum spent. Income has only an almost significant influence on buying frequency ($\beta = 0.13$, $p \leq 0.10$). Finally, survey year has a significant association with the average sum spent ($\beta = -0.18$, $p \leq 0.05$).

Overall, the model explains a considerable portion of the variance in buying frequency (34.9%), but only a minor portion of that in the average sum spent (14.9%).

6. DISCUSSION

The findings are partly as expected and partly surprising. Of the six hypotheses proposed in Section 3.2, the data supported only hypothesis H2a. Contrary to our expectations, perceived usefulness did not predict customers' buying behavior, although perceived ease of use did predict buying frequency and to a lesser extent the average sum spent. This suggests that those who perceive an electronic grocery store to be easier to use are apt to use it more frequently and also to buy slightly more.

Quite surprisingly, a marked negative association existed between total satisfaction and buying frequency, the likely explanation for which is the reciprocal interdependence between satisfaction and use (DeLone and MacLean, 1992). While satisfaction influences use (buying behavior), it is also influenced by the buying behavior experience. In our case it seems that those who shopped more in the electronic grocery store in question became more dissatisfied. The explanation for this may be the novelty of electronic grocery shopping for customers, which may underline the influence of buying behavior on satisfaction rather than vice versa. In fact, the EGS studied here had opened just before our field study was initiated. To test this we revised the model of Figure 2 to include all eleven dimensions of satisfaction as dependent variables, with buying behavior and the average sum spent as their predictors. The analysis showed that satisfaction with product presentation was significantly dependent on buying frequency ($\beta = -0.17$, $p \leq 0.05$) and satisfaction with service almost significantly so ($\beta = -0.16$, $p \leq 0.10$). These negative associations can be explained by increased dissatisfaction with product presentation and service among more frequent customers of ESG. The former finding is consistent with the problem of finding products and information experienced in two electronic grocery stores, as reported by Raijas and Tuunainen (2001).

On the other hand, satisfaction with the assortment of goods, satisfaction with opening times and satisfaction with safety were positively dependent on the average sum spent ($\beta = 0.11$, $p \leq 0.05$, $\beta = 0.14$, $p \leq 0.001$, $\beta = 0.15$, $p \leq 0.05$, respectively) and satisfaction with the freshness of the products and product presentation almost significantly so ($\beta = 0.10$, $\beta = 0.15$, $p \leq 0.10$ on both cases). The positive associations between the average sum spent and satisfaction with the assortment and satisfaction with the freshness of the products are easy to understand, while the satisfaction with opening times may reflect convenience shopping. Morganosky and Cude (2000) report that over 70% of grocery consumers shopping online report convenience and saving of time as their primary reason for this preference.

The above positive associations can also be interpreted, however, as suggesting that satisfaction with these dimensions explains the average sum spent. Unfortunately, our data ($n = 106$) do not allow a more complete testing of the model (so that some individual dimensions of satisfaction were independent variables explaining buying behavior while others were dependent on it). Thus our results concerning the relationship between satisfaction and buying behavior are not conclusive.

The fact that perceived usefulness was found to be a poor predictor of customers' buying behavior is consonant with the finding of Henderson et al. (1998) that perceived usefulnesss did not emerge as a significant predictor of intention to use of an electronic supermarket, but contradicts Childers et al. (2001), who found perceived usefulness to be a significant predictor of attitude. The weak significance of perceived usefulness is also in line with Gefen and Straub (2000), who found this factor to be a significant predictor of intended purchasing behavior in their free simulation experiment of online shopping for books. This study differs, however, in the sense that behavior was measured in objective terms rather than as an intention to behave, as in Gefen and Straub (2000), or as self-reported behavior, as is common in TAM research (see Straub et al., 1995). On the other hand, perceived ease of use was a significant predictor of buying frequency and an almost significant predictor of the average sum spent. This is in line with Childers et al. (2001), who found perceived ease of use to be a significant predictor of attitude.

Of the control variables, family size was found to affect the frequency of shopping in an electronic grocery store more than family income. Perhaps surprisingly, position was a significant, but negative predictor of the average buying spent, i.e. those in a higher position tended to buy less. On the other hand, they used the facility almost significantly more frequently. Age was a positive predictor of average buying behavior.

In view of the novelty of electronic grocery stores and the finding of Raijas and Tuunainen (2001) that users are more willing to try something new than non-users, it is surprising that personal innovativeness had an almost significant negative association with buying frequency. On the other hand, the influence of computer playfulness on buying frequency was almost significantly positive.

7. FINAL COMMENTS

This study has clear practical implications for practitioners responsible for the further development of the electronic grocery store concept. Firstly, one should pay proper attention to the perceived ease of use of an online electronic store, as this seems to promote the acceptance in terms of buying frequency and the average sum spent. The results also suggest that more attention should be paid to the reasons for satisfaction or dissatisfaction with grocery stores. Even though it may be that those who bought things more frequently became dissatisfied with the system, it is likely, or at least possible, that this dissatisfaction may influence customers' buying behavior in the future.

The study has its limitations. It focused only on one electronic grocery store, which has the benefit that many potentially extraneous variables were controlled, but at the same time limits the generalizability of the results. The number of respondents (106) was also low, which means that one must interpret the statistical results with care. The low number of respondents is explained by the fact that the electronic grocery store in question was very new, having just been launched. Thus the customers included are 'early adopters' in the sense of Rogers (1995), and it is an open question to what extent the results can be generalized to later adopters. It may also be that these early adopters' buying behavior has not stabilized. Our data also suffered from certain validity problems.

The above findings clearly justify continued research. As pointed out above, the relationship between satisfaction and buying behavior clearly requires additional research. Research should also be extended to cover a larger number of electronic grocery stores, in the form of either focused studies of single electronic grocery stores, as here, or cross sectional surveys. In both cases it would be useful if the studies were comparable with each other. There is also a need for longitudinal studies of customers' acceptance of electronic grocery stores.

REFERENCES

Agarwal, R., and Prased, J., 1998, A conceptual and operational definition of personal innovativeness in the domain of information technology, *Information Systems Research* **9**(2):204–215.

Agarwal, R., and Venkatesh, V., 2002, Assessing a firm's Web presence: A heuristic evaluation for the measurement of usability, *Information Systems Research* **13**(2):168–186.

Ahola, H., 1999, Marketing in www context – A case study of a pilot web-based supermarket, *Twelfth International Bled Electronic Commerce Conference*, Bled, Slovenia, June 7–9.

Ahola, H., Oinas-Kukkonen, H., and Koivumäki, T., 2000, Customer delivered value in a Web-based supermarket, *Proceedings of the the 33rd Hawaii International Conference on Systems Sciences*.

Aladwani, A. M., 2002, The development of two tools for measuring the easiness and usefulness of transactional Web sites, *European Journal of information Systems* **11**(3):223–234.

Childers, T. L., Carr, C. L., Peck, J., and Carson, S., 2001, Hedonic and utilitarian motivations for online retail shopping behavior, *Journal of Retailing* **77**:511–535.

Chin, W. W., 1998, The Partial Least Squares approach to structural equation modelling, in: *Modern Methods for Business Research*, G. A. Marcoulides, ed., Lawrence Erlbaum Associated, Mahwah, NJ, pp. 295–336.

Chin, W. W., and Gopal, A., 1995, Adoption intention in GSS: Relative importance of beliefs, *The Data Base for Advances in Information Systems* **26**(2&3):42–63.

Chin, W. W., and Newsted, P. R., 1999, Structural Equation Modeling analysis with Small Samples Using Partial Least Squares, in: *Statistical Strategies for Small Sample Research*, R. Hoyle, ed., Sage Publications, pp. 307–341.

Cooper, R. B., and Zmud, R. W., 1990, Information technology implementation research: A technological diffusion approach, *Management Science* **36**(2):123–139.

Davis, F. D., 1989, Perceived Usefulness, Perceived ease of use and acceptance of information technology, *MIS Quarterly* **13**(3):319–340.

Davis, F. D., Bagozzi, R. P., and Warshaw, P. R., 1989, User acceptance of computer technology: A comparison of two theoretical models, *Management Science* **35**(8):982–1003.

DeLone, W. H., and McLean, E. R., 1992, Information systems success: the quest for the dependent variable, *Information Systems Research* **3**(1):60–95.

Gefen, D., and Straub, D., 2000, The relative importance of perceived ease of use in IS adoption: A study of e-commerce adoption, *Journal of Association for Information Systems* **1**(8).

Gefen, D., Karhanna, E., and Straub, D., 2003, Trust and TAM in online shopping: An integrated model, *MIS Quarterly* **27**(1):51–90.

Geuens, M., Brengman, M., and S'Jegers, R., 2003, Food retailing, now and in the future. A consumer perspective, *Journal of Retailing and Consumer Services* **10**:241–251.

Grewal, D., Iyer, G. R., and Levy, M., 2004, Internet retailing: enablers, limiters and market consequences, *Journal of Business Research* **57**(7):703–713.

Hart, C., Doherty, N., and Ellis-Chadwick, F., 2000, Retailer adoption of the Internet, Implications for retail marketing, *European Journal of Marketing* **34**(8):954–974.

Heikkilä, J., Kallio, J., Saarinen, T., and Tuunainen, V. K., 1998, Analysis of expectations on electronic grocery shopping for potential customer segments, *Australian Journal of Information Systems* **6**:56–69.

Heikkilä, J., Kallio, J., Saarinen, T., and Tuunainen, V. K., 1999, EC of groceries for elderly and disabled; a comparison of alternative service models, *Information Technology & People* **12**(4):389–402.

Henderson, R., Rickwood, D., and Roberts, P., 1998, The beta test of an electronic supermarket, *Interacting with Computers* **10**:385–399.

Karahanna, E., Straub, D. W., and Chervany, N. L., 1999, Information technology adoption across time: A cross-sectional comparison of pre-adoption and post-adoption beliefs, *MIS Quarterly* **23**(2):183–213.

Kutz, K., 1998, On-line grocery shopping on track for rapid growth, *Andersen Consulting Newsletter*, January 20, Chicago.

Moore, G. C., and Benbasat, I., 1991, Development of an instrument to measure the perceptions of adopting information technology innovation, *Information Systems Research* **2**(3):192–222.

Morganosky, M. A., and Cude, B. J., 2000, Consumer response to online grocery shopping, *International Journal of Retail & Distribution Management* **28**(1):17–26.

Muylle, S., Moenart, R., and Despontin, M., 2004, The conceptualization and empirical validation of web site user satisfaction, *Information & Management* **41**(5):543–560.

Oinas-Kukkonen H., 2000, Balancing the vendor and consumer requirements for electronic shopping systems, *InformationTechnology & Management* **1**:73–84.

Palmer J., Kallio, J., Saarinen, T., Tinnila, M., and Tuunainen, V. K., 2000, Online grocery shopping around the world: Examples of key business models, *Communications of Association for Information Systems* **4**(3).

Raijas, A., and Tuunainen, V. K., 2001, Critical factors in electronic grocery shopping, *Int. Rev. of Retail, Distribution and Consumer Research* **11**(3):255–265.

Rogers, E. M., 1995, *Diffusion on Innovation*, Fourth edition, Free Press New York.

Rohm, A. J., and Swaminathan, V., 2004, A typology of online shoppers based on shopping motivations, *Journal of Business Research* **57**(7):748–757.

Straub, D., Limayem, M., and Karahanna-Evaristo, E., 1995, Measuring systems usage: Implications for IS theory testing, *Management Science* **41**(8):1328–1342.

Szymanski, D. M., and Henard, D. H., 2001, Customer satisfaction: A meta-analysis of the empirical evidence, *Journal of Academy of Marketing Science* **29**(1):16–35.

Verhoef, P. C., and Langerak, F., 2001, Possible determinants of consumers' adoption of electronic grocery shopping in the Netherlands, *Journal of Retailing and Consumer Services* **8**:275–285.

Webster, J., and Martocchio, J. J., 1992, Microcomputer playfulness: Development of a measure with workplace implications, *MIS Quarterly* **16**(2):201–224.

ONTOLOGY-BASED DECISION SUPPORT SYSTEM FOR CRIME INVESTIGATION PROCESSES

Dale Dzemydiene and Egle Kazemikaitiene*

1. INTRODUCTION

The characteristics of complex and dynamic crime investigation domain require new ways of information extraction and knowledge representation. The crime analysis information system, based on ontology, ensures the proper application of the structural model of crime information, determines the main rules, how to acquire the important forensic and crime investigation information about crime from the primary sources.

The goal of developing a decision support system (DSS) is to act "helpfully" in terms of reliably reproducing the cognitive behaviour of crime investigators demands thoroughly considered concepts, principles, and crime scenarios.

We can classify the problems arising in the investigation of a crime into two main groups: the problems of a purely criminal nature and those associated with the criminal law, penal process, organization, and criminology. One can solve the first group of problems by employing the crime analysis information, while for the problems of the other group we need information on the criminal law, penal process and the like, that would be significant for the crime investigation.

The crime analysis information, first of all, is data that pattern the criminal event and its investigation.[1, 5, 7, 18] Naturally, the data that serve for revealing the criminal offence are of great importance as well.

The target of creating an advisory information system in order to render an opportunity for the investigator of using the auxiliary and consultative information is connected with a lot of complex technological problems. The requirement posed to the crime analysis information system, possessing the primary investigation information, must render an opportunity to present inquiries to other required information systems, to receive their replies and, based on them, to provide already processed further information, versions of the investiga-

* Dale Dzemydiene, Law University of Lithuania, Ateities str. 20, LT-08303 Vilnius, Lithuania; Institute of Mathematics and Informatics, Akademijos 4, LT-08303 Vilnius, Lithuania, daledz@ktl.mii.lt. Egle Kazemikaitiene, Law University of Lithuania, Ateities str. 20, LT-08303 Vilnius, Lithuania, eglek@ltu.lt.

tion course and possible ways of decision.[6] In his turn, the investigator, basing on already processed information and on his own logical thought, would make a proper decision.

Crime registration is a very significant stage in the development of crime investigation practice, i.e., criminal records, automated and partially automated recording, card files, collections, and databases.[11, 13] However, the systems of a recording nature are not sufficient in providing complicated information supply for crime investigation.[11] It would be reasonable to have information on the crime scene observing practice of different crime types, i.e., knowledge of the crime scene investigation tactics and strategies of various types of crimes and their peculiarities, where to look for traces, what investigation plan to make up, and what problems to solve.[5, 12] One more peculiarity as to the information acquisition on the crime scene observation is the place from which the investigator could send a request to the crime analysis system.

Solution of the second group problems is no less important when building a crime analysis system. For instance, after receiving a report or application on the event, the investigator must judge whether there are any criminal indications. Only such an information system that contains information with evident criminal indications could be of use to him. There are, e.g., particular signs of violence or other body injuries typical of various cases of violence. Each crime bears signs typical of this kind of crime that ought to be related in the crime analysis information system.

Another task for the investigator is the proper qualification of a criminal act. Therefore the investigator should have access to the corresponding information, i.e., information on the criminal law: penal laws, their comments and explanations, court practice.

In the analysis of the components of intellectual information systems, it is important to consider how to represent crime investigation knowledge and by what methods to build decision preparation systems. To this end, an attempt is made to describe the sense of application of these methods, by pointing out the decision-making problems, creating the crime investigation ontology.

The main structural requirements for developing the ontology-based crime analysis information system are considered.

2. RELATED WORKS

There are a number of projects currently developed in areas, as diverse as psychology, medicine, crime analysis and computing, which focus on the analysis and management of images with some support for language engineering techniques. A large volume of data related to the modus operandi (MO) is being collected. The force linked intelligence system FLINTS[9] combines forensic and physical evidential "hits" in order to display links between criminals and the evidence, and produces a profile of offenders and of crimes committed. Such systems as IMPRESS, FLINTS, LOCARD and imaging workflow systems can all be used as systems for building the profile of a habitual criminal.[12, 9, 16] There are evidence tracking systems that deal with the movement of crime-related exhibits (Locard Evidence Tracking System) from the crime scene through to the court; this tracking is again performed through a class description, much like that used by freight handling organizations.[1, 15]

A workflow system should be able to process and fuse together the impression evidence in the two modalities: free text descriptions and images. Given a large volume of impression evidence, it is important for such a system to learn to fuse the different items of impression evidence in a coherent whole. There are problems related to the variance in the ways other than Police Forces describe and image a scene of crime and indeed there are variations within a large Force. Nevertheless, there are similarities in descriptions as manifested by terminology, reflecting the conceptual structure of forensic science and criminology, on the one hand, and the specialized nature of the images of impressions.[15, 16, 5] There is a need to have a holistic view of the impression evidence: different types of evidence and two different modalities. Training of forensic scientists and officers should reflect this. An intelligent workflow system for impression evidence, that pro-actively fuses, and learns to fuse, descriptions and images, will require an active cooperation between the end-users (the Police Forces), the academic researchers in information extraction, in text and image mining, and in Grid technologies, and software vendors specialized in working with the Police Forces.

One of the important research problems is the use of manually created ontology and thesauri. The IMPRESS project will benefit from the lessons learned in these projects on the management of image repositories and will inform them in return about image-text interactions and how adaptive learning systems can be beneficial.[12] The research work undertaken already in building image management systems that beneficially use texts collateral to the image has been documented.[18] The key problem of inter-indexer variability has not been extensively discussed.[8] Multiple classifier systems have been used to deal with properties of an individual image (colours, shapes, texture) and there are some results on relating images to sound.[18]

3. KNOWLEDGE REPRESENTATION TECHNIQUES USED IN THE DECISION SUPPORT SYSTEM FOR CRIME INVESTIGATION PROCESSES

3.1. An Approach for Integrating Ontology into the Scenario Generation Algorithm

The development of application ontology helps to create the framework and thus to ensure the collection, accumulation, storage, treatment, and transmission, in a proper form, of important investigation information, which establishes conditions to make optimal decisions in the investigation of crimes.

The integration of ontology, meta-modelling, and crime pattern recognition is very important in this DSS. We use the model-based and the case-based reasoning techniques, derived from the existing technology of compositional modelling, and integrate reasoning about evidences based on ontology.[4]

Scenarios are modelled as the causes of evidences and they are inferred based on the evidences they may have produced.

The goal of the DSS is to find a set of possible alternatives of crime causes following from scenarios that support the entire set of available evidence. This set of possible alternatives of crime modus operandi can be defined as:

$$A_C = \{a \in A \mid \exists s \in S, (\forall c \in C, (S \Rightarrow c)) \wedge (S \Rightarrow a)\},$$

where A_C is the set of all possible alternatives of crime modus operandi (e.g., suicide, accident, or murder), S is the set of all consistent crime scenarios, C is the set of all collected pieces of crime evidence.

The methodology of implementing the decision support system is based on the created ontology.

Ontology refers to engineering artefacts, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary worlds.[10] We must consider the language $\boldsymbol{L}$ with the vocabulary V, and $I: V \rightarrow D \cup \boldsymbol{R}$ as a function assigning the elements of D to constant symbols of V, and the elements of $\boldsymbol{R}$ to predicate symbols of V. The role of ontology can be considered as a set of logical axioms designed to account for the intended meaning of vocabulary. In general, it is not easy to find the right set of axioms. Ontology may be informal, for example, specified by a catalogue of types that are definitions stated in the natural language, or formal, for example, specified by axioms and definitions stated in the formal language.[3, 17]

The applied ontology of the crime investigation subject area is created by making use of a unified modelling language (UML), applying object-oriented designing methods,[2] and it is translated into the XML language.

The creation and implementation of ontology enable us to systematize the collected facts, to evaluate structures of knowledge.[10] Creation of ontology is an interactive process. The methodologies proposed for ontology development are: M.Uschold's methodology, M.Grüninger's and M.S.Fox's (TOVE) methodology; KACTUS approach, On-To-Knowledge (OTK) methodology.[14]

An example of description of the object class diagram for revealing the criminal character of crime by UML is presented in Figure 1.

The description of such schemas are translated into XML, XMI language.

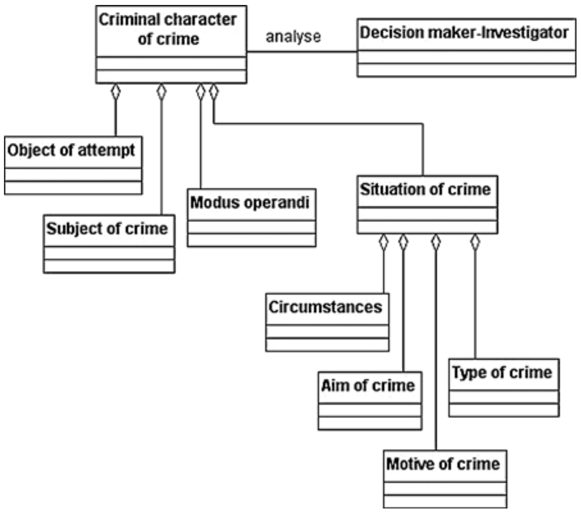


Figure 1. Example of the object class diagram of the crime character in UML.

While considering the examples of ontology development,[3, 10, 14, 17] we can classify opportunities of developing different kinds of ontology according to their level of generality:

- Top-level ontology describes very general concepts, such as space, time, matter, objects, etc., which are independent of a particular problem or domain;
- Domain ontology and task ontology describe the vocabulary related to the generic domain (like law, medicine) or generic task or activity (like forensic process, diagnosis), by specializing the terms introduced in the top-level ontology.
- Application ontology describes concepts depending both on a particular domain and task, which are often specializations of both the related ontology. These concepts often correspond to the roles played by domain entities while performing a certain activity.

The crime analysis information system should supply qualitative processed information for the investigator and help to make decisions on all important points of investigation: to produce versions, to choose the tactics and strategy of investigation, etc. Because of that the crime analysis information system becomes an instrument for crime investigation in practice.

3.2. Description of Forensic Evidence Investigation Processes

The scenarios of revealing the forensic evidence are different and depend on the type of crime. We choose the type of crime against a person's life to illustrate the representation of the scenario and collaboration activities of forensic evidence investigation processes.

The main processes are revealed:

- The scene of crime is observed without touching anything.
- A group of investigators inspect the surroundings of the crime event, by fixing all the possible traces from the outlying areas to the centre (inspection of fingerprints, cigarette-butts, blood, etc.) – this is the stage of a primary observation process.
- A corpse in the centre and its examination (a doctor examines the body, diagnoses the cause of death by the signs of violence, he proves it to be not a suicide), and the time of murder by the death marks, the site (the corpse could be transferred).
- The investigator examines the body and clothes in order to identify the person, sex, fingerprints, blood, saliva, sperm or other traces on the body or clothes.

After the all-round inspection/examination everything is fixed by taking pictures and drawing up the report.

An example of the sequence diagram is presented in Figure 2.

According to the primary information some versions are raised and a further plan of investigation is drawn up. This is a first important decision for further actions.

The group of investigators must reveal all the possible versions, which are: suicide, premeditated murder, and manslaughter, and propose the decision.

The maintenance of crime investigation information is a complicated system that can be described by the following specific features:

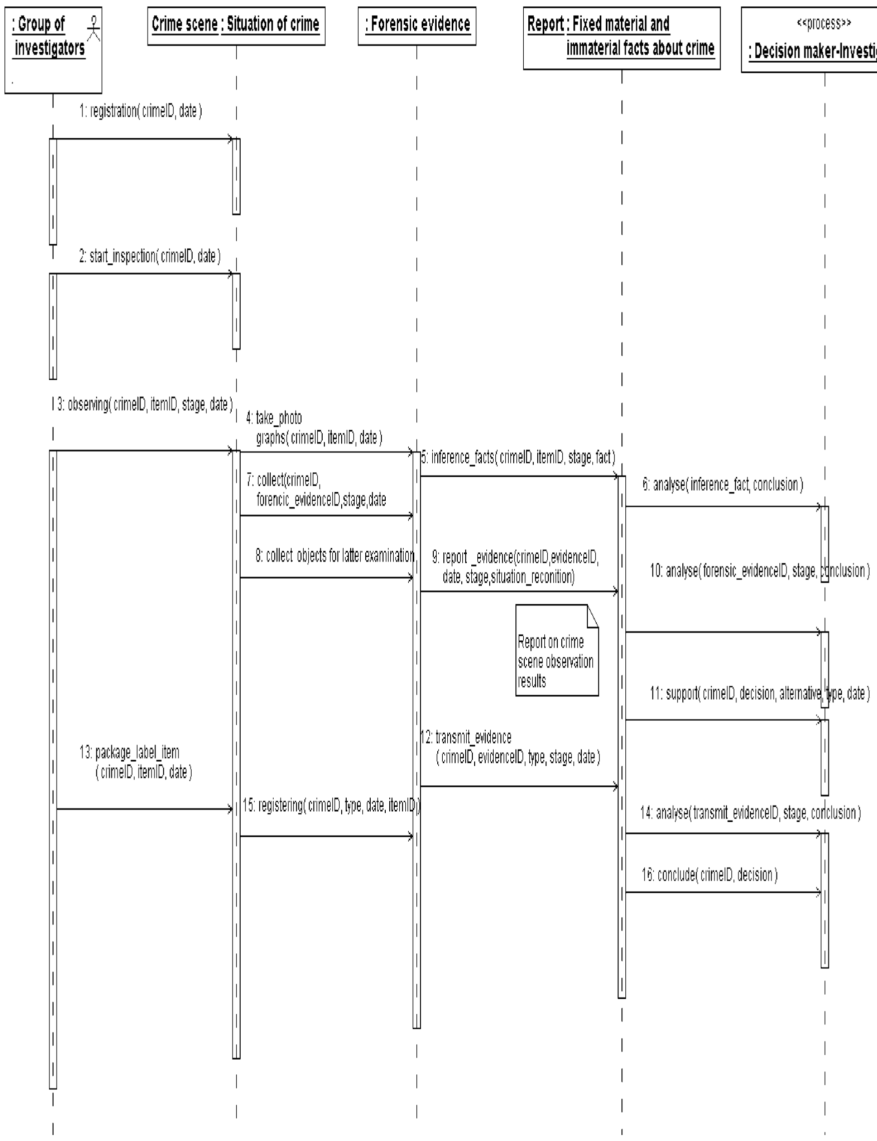

: Group of investigators
Crime scene : Situation of crime
: Forensic evidence
Report : Fixed material and immaterial facts about crime
<<process>>
: Decision maker-Investigator
1: registration(crimeID, date)
2: start_inspection(crimeID, date)
3: observing(crimeID, itemID, stage, date)
4: take_photo graphs(crimeID, itemID, date)
5: inference_facts(crimeID, itemID, stage, fact)
6: analyse(inference_fact, conclusion)
7: collect(crimeID, forencic_evidenceID,stage,date
8: collect objects for latter examination
9: report _evidence(crimeID,evidenceID, date, stage,situation_reconition)
10: analyse(forensic_evidenceID, stage, conclusion)
Report on crime scene observation results
11: support(crimeID, decision, alternative, type, date)
12: transmit_evidence (crimeID, evidenceID, type, stage, date)
13: package_label_item (crimeID, itemID, date)
14: analyse(transmit_evidenceID, stage, conclusion)
15: registering(crimeID, type, date, itemID)
16: conclude(crimeID, decision)

Figure 2. Sequence diagram of the forensic evidence investigation process.

- The aim of the crime analysis information system is to help the investigator to quickly and effectively investigate a crime, by doing different hard jobs, and to create presumptions for purposive investigation;
- The creation of a crime analysis information system is based on the recommendations of criminality and other sciences, aimed mainly at the investigation and prevention of crime;

- The crime analysis information system has to be regulated;
- The information has to be accessible just for a limited count of users to ensure the security of it, so the crime analysis information system is a system of limited use;
- The treatment of information in the crime analysis information system has to conform to the purposes of crime investigation;
- The optimisation of information processes is realized by their automation and technological implementation.

The automation of information processes is realized by a strict implementation of necessary procedures that include:

- Determination of type, object and character of information used in the crime investigation process;
- Analysis of investigation information in order to determine disadvantages of information supplying as well as possible automation tendencies of investigator's work;
- Design of the crime analysis information system
- Control of the crime analysis information system and its the improvement.

The crime analysis information system should secure the component integrity and could operate as an integral system. Consequently, the requirements posed to it are those typical of such a system:

- Independent of individual components, quality is a typical feature of the crime analysis information system of a special purpose and kind, and it may not be reduced only to the features of elements that compose the system;
- Qualitative features of the crime analysis system depend on the qualitative features of the elements that compose it;
- There is a close connection among the elements of the crime analysis information system. These are so close that a change in one component of the system stimulates changes in other components or sometimes even in the whole system;
- The correct order of the constituent elements as well as of the links among them characterizes the crime analysis information system as an integral system. This is mostly evident in the structure and organization of the system;

Just like any other integral system, the crime analysis information system cannot exist being isolated, without any link with the exterior world. The crime analysis information system interacts with the real world and other information systems. It is a dynamic and developing information system.

The subject area considered by us could be limited up to the consideration of crime investigation as an ontological point of view based on building of the exterior world model.

Criminality information should describe the circumstances of the crime event, the mechanism of crime, and particular regularities of modus operandi. The facts about criminals, traces and evidence, and about modus operandi and versions make the basis for creating the crime analysis information system.

The important knowledge in the crime investigation is knowledge of crime investigation methods, criminal procedure, and other knowledge of special scientific methods and

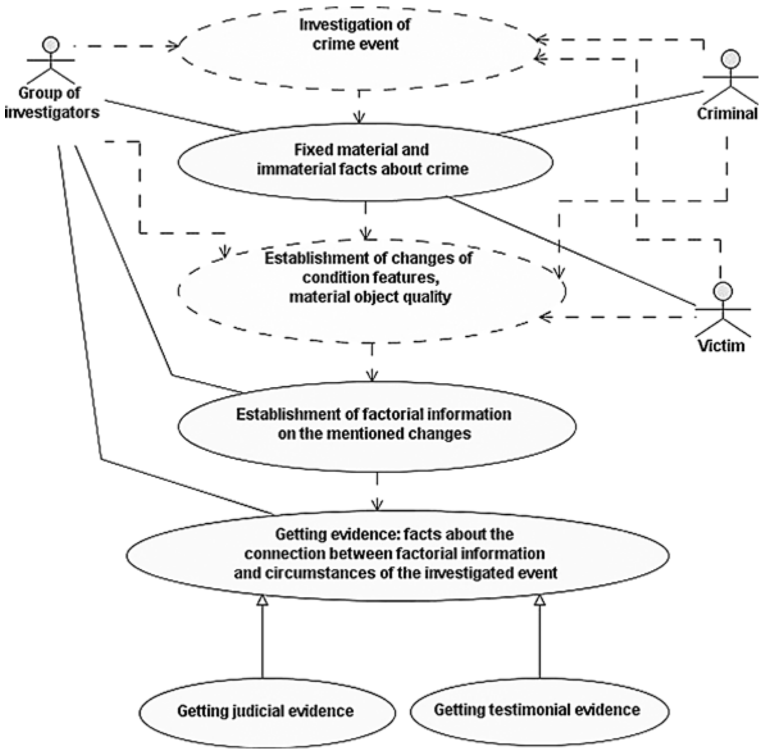

Investigation of crime event
Group of investigators
Criminal
Fixed material and immaterial facts about crime
Establishment of changes of condition features, material object quality
Victim
Establishment of factorial information on the mentioned changes
Getting evidence: facts about the connection between factorial information and circumstances of the investigated event
Getting judicial evidence
Getting testimonial evidence

Figure 3. Example of the use case diagram of the investigation stages of crime evidence information.

tactics for investigation. Investigators and judges take part in the knowledge process, the essence of which is a reconstruction of the crime event by the evidence and traces left in the surroundings. These traces are fixed and recognized, that for unknown reasons become evidences.

The creation, removal, and transformation of evidence information take place step by step and reflect multi-stage and structured causative connections that describe the role of actors in different criminal activities and use-cases (Figure 3).

Major data flows are considered by separating a criminal event as a process, and models of data flows as well as of the processes transforming them are constructed.

The main algorithm of our decision support system is presented in Figure 4. It shows the integration of basic investigation processes and case-based reasoning mechanisms applied in our DSS. The components of this algorithm are the databases of solved crimes cases that are helpful in such a reasoning mechanism.

The investigation of criminal event could be detailed as follows:

- Establishment of the sources of the investigated case;
- Extraction of information on the investigated event from the sources;
- Implementation of particular circumstances of the investigated event;

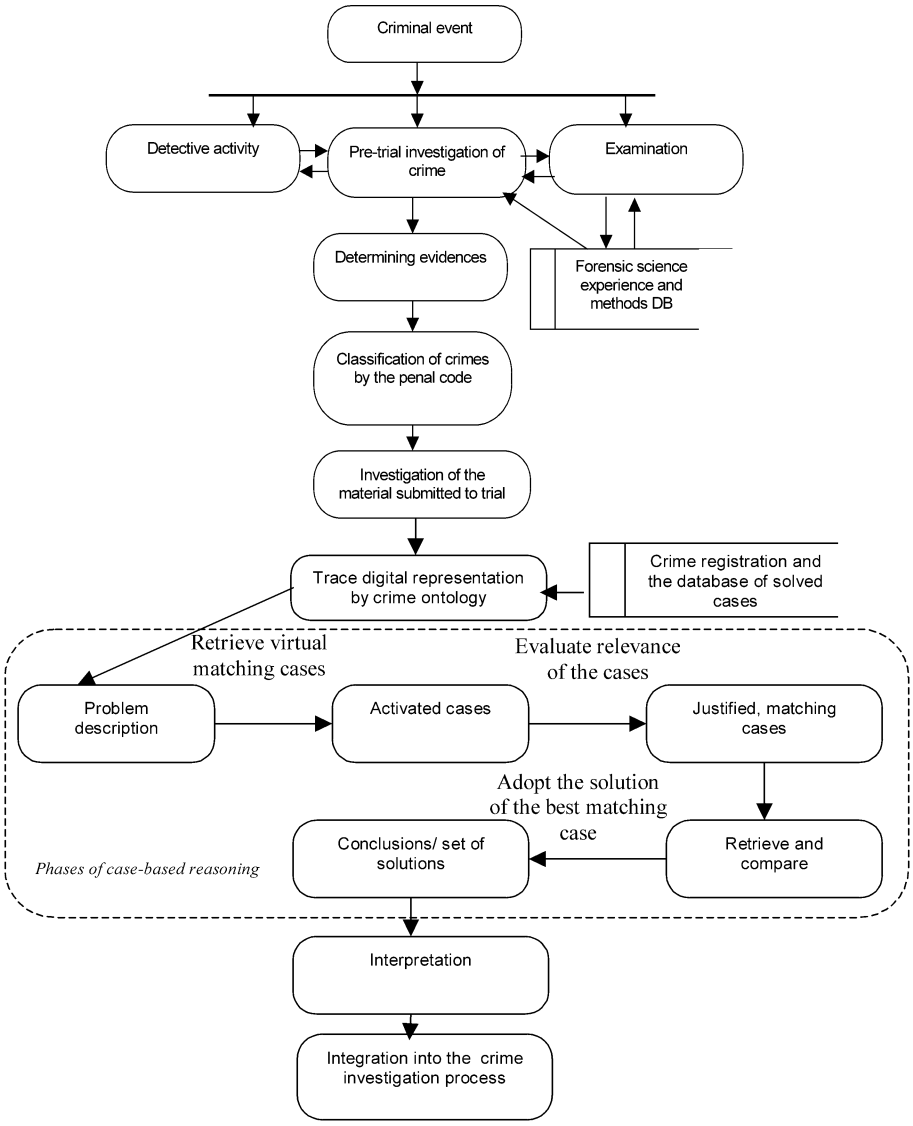

Criminal event
Detective activity
Pre-trial investigation of crime
Examination
Determining evidences
Forensic science experience and methods DB
Classification of crimes by the penal code
Investigation of the material submitted to trial
Trace digital representation by crime ontology
Crime registration and the database of solved cases
Retrieve virtual matching cases
Evaluate relevance of the cases
Problem description
Activated cases
Justified, matching cases
Adopt the solution of the best matching case
Conclusions/ set of solutions
Retrieve and compare
Phases of case-based reasoning
Interpretation
Integration into the crime investigation process

Figure 4. The algorithm of the main processes of model-based and case-based reasoning for pre-trial crime investigation.

- Development of a complete information system and determination of the actual structure of the investigated event.

Thus, all the activities connected with the investigation of crime are aimed at the final result – to define the structure (pattern, model) of a criminal event.

Pre-trial crime investigation is based on crime investigation science knowledge about the crime investigation, planning, investigation situations, versions, etc. Here information

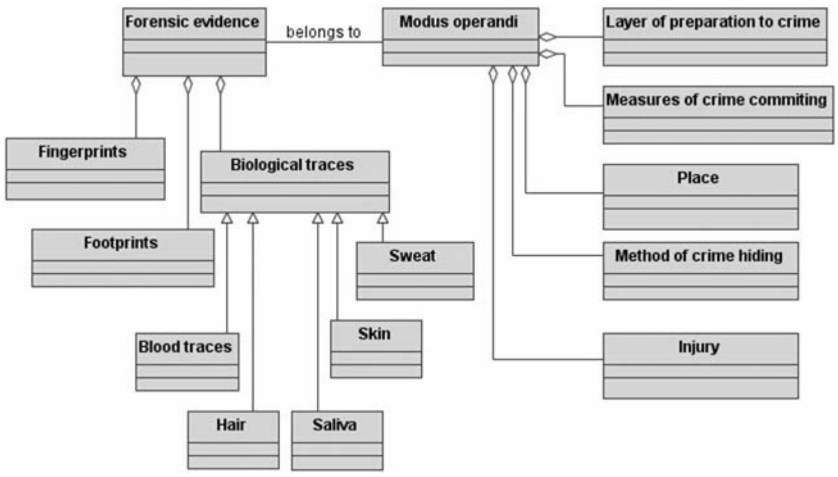


Figure 5. An example of class diagram of forensic evidence.

on archives of penal crimes as well as statistical information on criminology databases is of great importance.

The ontology of crime analysis information first of all is the reflection of a crime and its investigation; information which describes the circumstances of a crime, its mechanism and particular regularities of crime committing, information on criminals, traces as well as modus operandi (Figure 5), versions, and knowledge about the methods of crime investigation, penal procedure, and other methodological and tactical recommendations of criminality and forensic science.

4. GENERATION OF CRIME MODUS OPERANDI ALTERNATIVES ACCORDING TO ONTOLOGY

According to the crime classification in the penal code (for example in Lithuania*), there are: crimes against the humanity (genocide, etc.), crimes against person's life (murder, etc.), crimes against property (thefts, material injury, etc.), economic crimes (burglary, etc.), – in total all about two hundred different types of crime (Figure 6 illustrates hierarchy of the main types of crimes). In its own turn, the creation of information on different types of crime should be based on the crime character and its elements.[5, 13]

Crime analysis information should be dynamic, complex, operative, true, trustful, available, safe, ergonomic, and scientific. The sources of criminality information could be material and immaterial objects that are connected with the circumstances of the crime event and contain the information on these circumstances.

The available evidence supports every alternative of crime modus operandi that follows from a plausible scenario. The rules are constructed by the ontology presented in the

* Lithuanian Penal Code, 2000 09 26 d. Act of Law No. VIII-1968. – 99–329 articles.

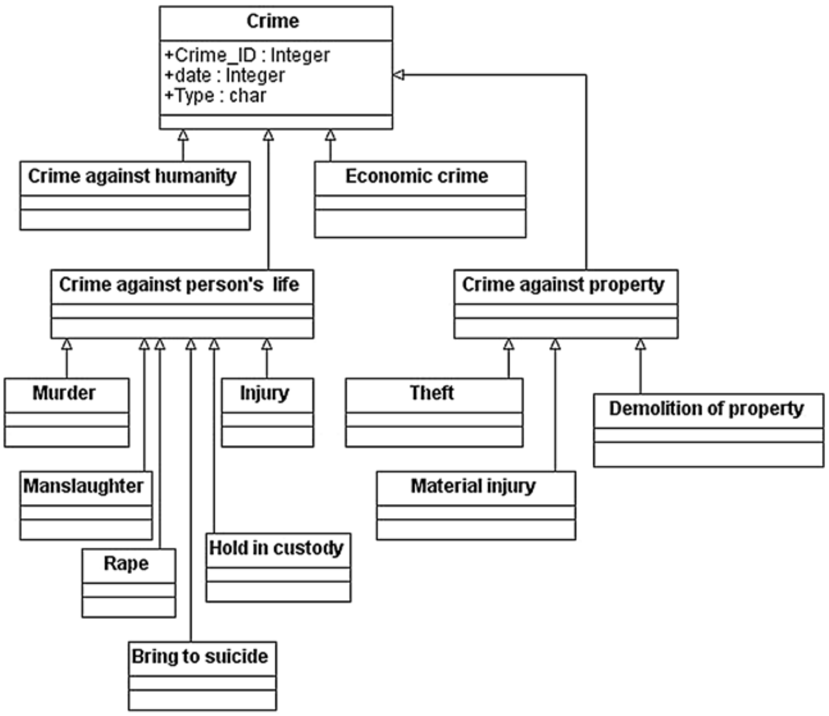


Figure 6. The example of crime classification.

previous examples, and the scenario space is revealed:

C_1 = {*violence_injuries(p), ballistic_report_of_shoot_mechanism(p), bullets_and_cartridges(p), blood_traces_analyses(p)* $\Rightarrow$ *diagnosis(murder(p))*}.

C_2 = {*medical_report_of_occasional_death(p), lack_of_collagen(p), accidental_blood_vessel_rupture(p))* $\Rightarrow$ *diagnosis(accidental_death(p))*}.

In a possible world described by crime event environment C_1, *murder(p)* is true, and in the one described by *C2*, *accidental_death(p)* is true.

The knowledge base of the DSS is developed by aforementioned rules. The development of application ontology helps to create the framework and thus to ensure the collection, accumulation, storage, treatment, and transmission, in a proper form, of important investigation information, which establishes conditions to make optimal decisions in the investigation of crimes.

5. CONCLUSIONS

The main advantage of the approach presented here is a possibility to integrate ontology into the scenario space generation algorithm. The knowledge representation methods play an important role in solving decision-making problems for the development of an advisory system in crime investigation processes. The approach for developing a decision support system is based on model-based and case-based reasoning techniques and use cases of crime investigations. The decision support system allows combinations of crime events and states that produce a given set of evidence from the knowledge base and ontology, helpful in generating scenario fragments from inconsistent situations. The proper crime investigation depends on the quality of the advisory crime analysis information system based on ontology, converting the first outside and inside facts in to criminality analysis processes, rendering opportunities of optimal decision making for the investigator. The main purpose of such a decision support system is supplying of information on the crime investigation activity in the crime analysis. The methodology for development of the decision support system relies upon the case-based reasoning mechanism and cases of use of crime investigations.

REFERENCES

1. K. Ahmad, B. Vrusias, and M. Tariq, Cooperative neutral networks and integrated classifications, *Proc. 2002 Int. Joint Conf. On Neutral Networks* (IEEE Press, Piscataway, 2002).
2. G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide* (Addison Wesley, 1999).
3. A. Čaplinskas, A. Lupeikienė, and O. Vasilecas, The role of ontologies in reusing domain and enterprise engineering assets, *Informatica* **14**(4), 455–470 (2003).
4. J. Keppens and Q. Shen, On compositional modelling, *Knowledge Engineering Review* (16), 157–200 (2001).
5. D. Dzemydiene, E. Kazemikaitiene, and R. Petrauskas, Knowledge representation in advisory information system of crime investigation domain, in: *Databases and Information Systems II*, edited by Hele-Mai Haav and Ahto Kalja (Kluwer Academic Publishers, 2002), pp. 135–148.
6. D. Dzemydiene and V. Rudzkiene, Data analysis strategy for revealing multivariate structures in social-economic data warehouses, *Informatica* **14**(4), 471–486 (2003).
7. D. Dzemydiene and V. Rudzkiene, Multiple regression analysis of crime pattern warehouse for decision support, Lecture Notes in Computer Science 2453, in: *Database and Expert Systems Applications*, edited by A. Hameurlain, R. Cicchetti, and R. Traunmuller (Springer, 2002), pp. 249–258.
8. J. P. Eakins and M. E. Graham, Content-based Image Retrieval: A Report to the JISC Technology Applications Programme (Image Data Research Institute Newcastle, Northumbria, 1999).
9. Force Linked Intelligence System – FLINTS – Linking Police Work with Science and Technology; http://www.west-midlands.police.uk/flints/background.htm.
10. N. Guarino, Formal Ontology and Information Systems, in: *Formal Ontology in Information Systems. Proceedings of FOIS'98*, edited by N. Guarino (IOS Press, Italy, Amsterdam, 1998), pp. 3–15.
11. B. Hebenton and T. Terry, *Criminal Records* (Brookfield, USA, 1993).
12. IMPRESS – IMPRession Evidence and Serial crime profiling System; www.computing.surrey.ac.uk/ai/impress/full.html.
13. G. L. Kovacich and W. Boni, *Hig Technology- Crime Investigator's Handbook: Working in the Global Information Environment* (Butterwoth-Heinemann, 2000), pp. 115–136.
14. S. Maskeliūnas, Ontological engineering: common approaches and visualisation capabilities, *Informatica* **11**(1), 41–48 (2000).
15. K. Pastra, H. Saggion, and Y. Wilks, Intelligent indexing of crime-scene photographs, *IEEE Intelligent Systems, Special Issue on "Advances In Natural Language Processing"* **18**(1), 55–61 (2003).
16. F. Roli and J. Kittler, eds., Multiple Classifier Systems: Proceedings (*LNCS*), Third International Workshop, MSC 2002 (Cagliari, Italy, June 24–26, 2002).
17. J. F. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations* (Brooks/Cole, Thomson Learning, Pacific Grove, CA, 2000).
18. R. K. Srihari and Z. Zhang, Show & Tell: a semi-automated image annotation system, *IEEE Multi Media* **7**(3), 61–71 (2000).

VARIATION IN STUDENTS' CONCEPTIONS OF OBJECT-ORIENTED INFORMATION SYSTEM DEVELOPMENT

Ilona Box and Raymond Lister*

1. INTRODUCTION

CRUD (create, read, update, delete) analysis in object-oriented information system development (OOISD) is recommended as a means to improve the quality of the resulting system.[1–9] Box[10] went further by stating that it also improved students' learning of OOISD. She stated, based on anecdotal evidence, that "the earlier [students] can detect errors of omission the more likely they are to succeed at "good" analysis and design; the more confident they feel about learning and doing OOISD and the better their object thinking".

If OOISD students did attempt CRUD analysis in the manner presented by Box & Ferguson,[9] early, during analysis, would Box's statement, in part or whole, be supported? In the first instance, would it be possible to detect the conceptions students have about OOISD by examining students' CRUD matrices? In this paper, we explore the question of what are the variations in the conceptions constituted in CRUD matrices that are an outcome of CRUD analysis after the development of high-level use cases and the initial class diagram. We use a phenomenographic research approach to constitute the variation in conceptions. Our results are categories describing what types of errors the students make in the CRUD matrices. Based on 57 student assignments, we identify three categories representing various types of errors made by students. The categories of description, the outcome space of the research, contribute to the ISD community by providing a taxonomy of errors to inform teaching practice of OOISD and gives modest conditional support to Box's statement.

1.1. Phenomenography

Phenomenography is a research approach focusing on the qualitatively different ways people experience, understand, perceive, or conceptualise a phenomenon.[11] The underpinning philosophy is that there are a limited number of qualitative ways of experiencing

* Both authors, University of Technology, PO Box 123, Broadway, NSW 2007. Ilona Box, also University of Western Sydney, Locked Bag 1797, PENRITH SOUTH DC, NSW 1797, ilona@cit.uws.edu.au.

phenomena. Phenomenographers usually collect their data by recording and transcribing interviews with a small number of interviewees. The transcripts are analysed to identify one or more dimensions of variation; a dimension of variation is a set of categories somehow related, e.g. linearly or hierarchically, with a small number of categories in the set. Since phenomenographers wish to capture the variation in experiences, and not quantify the popularity of each experience (though this can be done in follow-up work), they can work with small numbers of interviewees.

Bruce[12] has presented a synopsis of phenomenography in information technology research. Booth[13] conducted the seminal phenomenographic work in computing, "what does it mean and what does it take to learn to program?". Also in 1992, Gerber, Buzer, Worth and Bruce asked educators and researchers of geographical information systems (GIS) their views and experience of GIS.[12] Academics', students', and practitioners' concepts of information systems have also been explored.[14] Cope's[15] later study identified students' different ways of seeing information systems, providing insights into how students' ways of seeing differs from the views of experts in the field. Bruce notes, "The differences identified are educationally critical". Other studies in the information technology discipline published in recent years include: Berglund[16] on the understanding of computer networking protocols; McDonald[17] on the nature and acquisition of algorithm understanding; Klaus and Gable[12] on the understanding by senior managers of knowledge management in the context of enterprise systems; Stewart and Klaus[12] on the experience of the business-IT relationship. Other areas of research underway are learning to program and learning about computer networking and data communications;[12] and how students, who take their first programming course, understand objects and classes.[18]

These works are leading the research in computing using phenomenographic studies. Bruce[12] states education research like this is critical to the design of effective professional education, and of some importance to information technology educators.

In this paper, we report upon our own phenomenographic study, to investigate the qualitatively different understandings students have of OOISD as constituted in students' CRUD matrices.

2. METHOD

Our data came from one source, 57 assignments completed by students as partial fulfilment of the first OOISD subject in an undergraduate computing degree. The assignments are work not collected by students at the end of semester and represent the five grades awarded for this assessment task. The assignment was an optional assessment tasks. Students chose to do the assignment to try for a credit or distinction grade.

The assignment was based on a case study (Appendix A) and required the students to correct and complete 12 high-level use cases, draw a class diagram, and do a CRUD analysis among a number of other tasks. The students were explicitly instructed to do the CRUD analysis before finalising their high-level use cases and class diagram and to follow the software development method as described in Box & Ferguson [9] Templates of the CRUD matrices, as discussed in Box & Ferguson, are shown in Appendix B. As well as the discussion provided in the text, the students received instruction and practice in the use of the CRUD matrices.

The CRUD matrices from the assignments were analysed in the phenomenographic style. Our focus for analysis was drawn to the errors the students had made in the CRUD matrices. The analysis was an iterative process. We did not begin with the categories; we formed the categories from what we found in the data. The types of errors were identified and then the set of CRUD matrices from each assignment were placed into one category or between categories. The categories were revised. The placing of assignments and revision of categories was iterated until we reached a consensus of what were the categories and the relationships between the categories. We only added a category when we could identify CRUD matrices in support of that category. The outcome space is the qualitative description of each category. The qualitative descriptions are of the predominant types of errors in a category and descriptive exemplars of how that error manifested in the CRUD matrices in the assignments.

It is important to understand that a single assignment is not designated to a single category. Students naturally have several conceptions about OOISD, although they may understand some conceptions better than others. Therefore, one assignment may fall across more than one of the following categories.

3. RESULTS

From the data, we identified three categories. The categories were constituted as aggregations of the types of errors (or lack thereof) we found in the CRUD matrices the students created.

3.1. Fragmented and Unstructured Conceptions

The first category is fragmented and unstructured conceptions. Here, it is difficult to identify conceptions that are correct. There are many conceptions that are incorrect. Often, one or more conceptions are in contradiction or conflict with other conceptions. We regard this as the less powerful understanding of OOISD. The various types of errors found in the CRUD matrices and which constitute this category are:

1) Object-oriented principles (encapsulation, data abstraction, inheritance):
   a) Identifying a class that only represents a chunk of data in the system. For example, an abstract class called *Data* [Assignment 6], a persistent class called *Records* with the attribute *Unit Outlines* [Assignment 7], a persistent class called *Record* [Assignment 45].
   b) Including a class name as an attribute in another class. For example, in the CRUD attribute matrix for the class *Unit Outline, unitCoordinator* is listed as an attribute*, Outline-Coordinator* is listed in the CRUD Association matrix, and *Unit Coordinator* is listed in the CRUD class matrix [Assignment 8].
2) Persistence (identifying persistent and transient objects, and abstract classes):
   a) Classes at the same level or specialisation in an inheritance structure are indiscriminately identified as persistent, transient, or abstract. For example, the classes listed in the CRUD class matrix: *Unit Coordinator, Team leader, Administrator*, and *Head of School* (as specialisations of person) are identified as persistent, transient, abstract, and abstract respectively [Assignment 1].

b) Classes with objects that come into existence as part of the new system are identified as persistent; classes with objects that exist before the new system is built are identified as abstract. For example the classes: *Discipline team, Student, Team Leader, Unit coordinator*, and *Unit* exist in other software systems and were identified as abstract. The classes: *Outline, Summative assessment*, and *Terms and choices* do not exist in other software systems, are considered "new" classes and are labelled persistent [Assignment 7].

c) Classes are not identified as persistent, transient or abstract. For example, the classes: *Unit Outline, Unit Coordinator, Team Leader, University*, and *Administrator* are listed in the CRUD class matrix and the "P" column is left blank [Assignment 8].

d) Classes that are roles or actors are classified as persistent; classes that represent things are classified as transient. For example, the classes: *Team leader* and *Coordinator* are typed as persistent; the classes: *Unit outline* and *Calculate summative assessment* are typed as transient [Assignment 3].

3) Traceability (among CRUD matrices, and between use cases or the class diagram and CRUD matrices):

a) The number of classes shown in the class diagram is not the same number listed in the CRUD class matrix. For example, the class diagram contains 11 classes and the CRUD class matrix lists only five classes, [Assignment 3], the class diagram contains seven classes and in the CRUD class matrix 49 classes are listed [Assignment 6].

b) The number of associations shown in the class diagram is not the same number listed in the CRUD association matrix. For example, the class diagram contains 10 associations and the CRUD association matrix lists only six [Assignment 1].

c) The names of the use cases in the CRUD matrices are not the same as in the use case diagram or high-level use cases. For example, the use cases: *Create unit outline, Add new team leader, Calculate summative assessment*, and *Record summative assessment*, are shown in the use case diagram and high-level use cases; in the CRUD class matrix the use cases are listed as: *A Unit outline, New team leader*, and *Calculate/record Summative assessment* [Assignment 16].

d) The identification of classes is inconsistent between matrices. For example, in the CRUD class matrix classes are identified as: *Unit Coordinator* and *Unit Outline*, and in the CRUD association matrix the classes are identified as: *Coordinator, Outline*, and *UnitOutline* [Assignment 8].

e) The attributes in the class in the class diagram are not the attributes listed in the CRUD attribute matrix for the class. For example, the class *Unit outline* has the attributes: *coarse title, course ID*, and *semester* in the class diagram; in the CRUD attribute matrix the attributes listed are: *Name, Idnumber, Address*, and *Phone number* [Assignment 3].

f) Only some of the attributes in the class diagram for a class are listed in the CRUD attribute matrix for the class. For example, the class *Unit Outline* in the class diagram has the attributes: *Title Page, Main Page*, and *Summative Assessment*; in the CRUD attribute matrix for the class the attributes are: *Titlepage, Main Heading, Unit Number, Unit Name*, and *Summative assessment* [Assignment 4].

g) The use cases identified in the CRUD class matrix to create, read, update, or destroy are not listed in the CRUD attribute matrix where they would be expected. For example, in the CRUD class matrix, the class *Unit Outline* is created during the use case *Create Unit Outline*; in the CRUD attribute matrix for the class *Unit Outline* all the attributes are initialised at creation by the use case *Define Outline* [Assignment 8].

4) Notation conventions (the notation of class, association, attribute, and use case names):
a) Little or no consistency between the notation for the names of classes, attributes and/or use cases between the class diagram, use case diagram, or high-level use cases, and the CRUD matrices. For example, five out of six class names in the class diagram has each word in title case, and in the CRUD class matrix, the four classes listed are written in sentence case [Assignment 21].
b) Little or no conformity to the notation for the names of classes, attributes, and/or use cases stipulated in the software process. For example, in the process described by Box & Ferguson, which conforms to UML version 1.4, class names are written in upper camel case (e.g. UnitOutline), attribute names are written in lower camel case (e.g. unitNumber), and use cases are written in sentence case (e.g. Create unit outline).

5) Process (the functions the system needs to perform):
a) For the classes that can be matched to people, a description or N/A rather than the use case is entered in the CRUD class matrix. For example, in the CRUD class matrix, the class *Unit Coordinator* has for create *Unit Outline*, has for read *N/A*, has for update *Make changes in unit outline*, and has for delete *Delete data from unit outline*; the class *Team Leader* has for create *N/A*, has for read *Read the unit outline create by unit coordinator*, has for update *Approval unit outline and update it*, and has for delete *Send data to unit coordinator for delete data from unit outline* [Assignment 17].

6) Design (the representation of design decisions relating to the identification of classes, associations, attributes, and use cases in CRUD matrices):
a) Identifying more than one use case that performs the same function. For example, objects in the class *Unit Outline* are read during the use case *Read unit outline*; objects in the class *Archive* are read during the use case *Access unit outline* [Assignment 1]. b) Classes and use cases identified using the same nomenclature. For example, in the CRUD class matrix the class is named *Calculate summative assessment* and the use case to create objects in this class is named *Calculate assessment* [Assignment 3], in the CRUD association matrix the associations are identified as *Approve unit outline, Modify unit outline*, and *Submit unit outline* [Assignment 21].
c) The attributes do not belong to objects in the class in a CRUD attribute matrix. For example, the class *unit outline* has the attributes: *Name, Idnumber, Address*, and *Phone number* [Assignment 3].
d) Identifying and naming individual parts in whole-part associations when the parts have common attributes. For example, listing in the CRUD class matrix the classes *AssumedKnowledge, ClassHours, Component, Content, Cover, Cov-*

erPage, Disabilities, Exclusion, Introduction, LearningSkillsUnit, Malpractice, Method, Note, OutOfClassHours, Practice, Prerequisite, Presentation, RecommendedText, References, StaticNote, StudentLearningOutcome, and *SummativeAssessment* as classes that have objects created during the *Create unit outline* use case [Assignment 6].
e) Identifying separate classes for the same data. For example, *Outline* and *Records* [Assignment 7], *Unit Outline* and *Archive* [Assignment 5].
f) Classes and associations are identified as separate due to the timing or sequence of events. For example, the class *Unit outline* is created during the *Create unit outline* use case, the class *Approval* is created during the *Submit unit outline* use case, and the *Approval-ApprovalResult* association is created during the *Approve unit outline* use case [Assignment 15].

3.2. Conceptions About Process Override Conceptions About Objects

Here, conceptions about what the system needs to do override conceptions about object-orientation. Where the processing of the system and the CRUD analysis are aligned the CRUD matrices are correct. There are fewer errors than in the previous category. The various types of errors (or lack thereof) found in the CRUD matrices and which constitute this category are:

1) Object-oriented principles (encapsulation, data abstraction, inheritance):
a) The majority of needed classes are listed, however, the understanding of object-oriented principles lacks the awareness to separate data appropriately. For example, the class *UnitOutline* is not separated into classes of *Unit* and *UnitOutline* [Assignment 2], the class *Unit Outline* is duplicated as the class *Template* [Assignment 13], the class *Unit* is listed, though struck out, in the CRUD class matrix and does not appear in the class diagram [Assignment 46].
b) There is little, if any, use of inheritance.
2) Persistence (identifying persistent and transient objects, and abstract classes):
a) The class that is a role or actor that could be seen as outside the system is classified as abstract; all other classes are correctly classified as persistent. For example, only the class *Administrator* is typed as abstract [Assignment 43], only the class *User* is identified as abstract [Assignment 46].
b) Incorrect identification of persistence is rare.
3) Traceability (among CRUD matrices, and between use cases or the class diagram and CRUD matrices):
a) The trace errors are minor inconsistencies. For example, for the minority of classes in the class diagram the order of the attributes is not the same as the order of the attributes in the corresponding CRUD attribute matrix for these classes [Assignment 2], the class in the class diagram is named *Explanations* and in the CRUD class matrix is listed as *Explanation* [Assignment 13].
4) Notation conventions (the notation of class, association, attribute, and use case names):
a) Consistency between the notation for the names of classes, attributes and/or use cases between the class diagram, use case diagram, or high-level use cases,

and the CRUD matrices is good though some errors still occur. For example, the use of abbreviations in the use case name, such as, *Calculate sum. assess. total* [Assignment 42].
b) Conformity to the notation for the names of classes, attributes and/or use cases stipulated in the software process is good though some errors still occur. For example, attribute names are written in upper camel case [Assignment 2], class names and attribute names include spaces between words [Assignment 13].

5) Process (the functions the system needs to perform):
a) For classes representing actors, the use cases chosen to complete the CRUD class matrix are those with which the actors are associated in the use case diagram. For example, the class *Administrator* has for create, read, update, and destroy the use case *Administer users* [Assignment 2].
b) The class is being considered in terms of its part in the process rather than the objects belonging to the class. For example, the class *Team Leader* has for create *Add new unit coordinator*, and for read *Approve unit outline* [Assignment 37].
c) The identification of the class is a step in the process rather than one to which objects would belong. For example, the classes *Approval* and *Submit* [Assignment 2], *Calculation* and *Submission* [Assignment 43], *Report* [Assignment 47], and *Submission* [Assignment 50].

6) Design (the representation of design decisions relating to the identification of classes, associations, attributes, and use cases in CRUD matrices):
a) The choice of a particular use case is not in keeping with many of the other use cases chosen in the CRUD matrices. For example, the class *Summative assessment* has for create *Calculate sum. assess. total*, and has for read *Create unit outline*, the class *Unit Outline* has for create *Create unit outline*, and has for read *Approve unit outline* [Assignment 42].
b) Only a few classes are identified beyond the initial, easily identified set of classes. For example, the initial, easily identified, set of classes includes: Administrator, TeamLeader, UnitCoordinator, UnitOutline, DisciplinedTeam, and SummativeAssessment, beyond this set the classes listed in CRUD matrices are: Help, School, School Archive [Assignment 46], Explanatory Document [Assignment 50], Unit, Unit Offering, and Campus [Assignment 19].

3.3. Conceptions of Design Decisions Appropriate Within the Object-Oriented Paradigm

This category shows the more powerful understanding of OOISD. The types of errors are predominantly about incorrect or weak design choices. The various types of errors (or lack thereof) found in the CRUD matrices and which constitute this category are:

1) Object-oriented principles (encapsulation, data abstraction, inheritance):
a) There is more and accurate use of inheritance. For example, the classes *FormativeAssessment* and *SummativeAssessment* inherit from the class *Assessmen*t, and the classes *Administrator, TeamLeader*, and *UnitCoordinator* inherited from the class *Person* [Assignment 24].

2) Persistence (identifying persistent and transient objects, and abstract classes):
   a) Incorrect identification of persistence is rare. For example, the classes *Assessment* and *Person* are correctly, and the only classes, identified as abstract classes [Assignment 24].
3) Traceability (among CRUD matrices, and between use cases or the class diagram and CRUD matrices):
   a) Trace errors do not occur.
4) Notation conventions (the notation of class, association, attribute, and use case names):
   a) Consistency between the notation for the names of classes, attributes and/or use cases between the class diagram, use case diagram, or high-level use cases, and the CRUD matrices is good though some errors still occur. For example, the use of abbreviations in the use case name, such as, *Calculate sum. assess. total* [Assignment 42].
   b) Conformity to the notation for the names of classes, attributes and/or use cases stipulated in the software process is good though some errors still occur. For example, attribute names are written in upper camel case [Assignment 2], class names and attribute names include spaces between words [Assignment 13].
5) Process (the functions the system needs to perform):
   a) The object-oriented paradigm is at the fore. The processes performed by the system are presented as use cases that determine the creation, reading, updating, and destruction of objects within classes.
6) Design (the representation of design decisions relating to the identification of classes, associations, attributes, and use cases in CRUD matrices):
   a) Indications of consideration of the consequences of making design decisions. For example, the notes: *Should TeamLeader and UnitCoordinator objects be destroyed, if they contain information on which session they apply to?* [Assignment 24] and *Summative assessment is part of the unit outline and cannot be destroyed* [Assignment 9].
   b) An awareness that use cases have a limit to the functions for which each is responsible. For example, the case study provided 12 use cases; in a CRUD class matrix listing 19 classes, seven Create/Class cells, two Read/Class cells, 10 Update/Class cells, and 14 Destroy/Class cells were not assigned one of the 12 provided use cases. [Assignment 36].
   c) The identification of a singleton object. For example, the note *A single Unit Outline Handler instance must be created when the system is installed… should never be destroyed* [Assignment 10].
   d) Objects that are associated as actors to a use case need be read. For example, objects in the class *Team Leader* are read during the use case *Approve outline* [Assignment 12].

4. DISCUSSION

In considering these results, we need to keep in mind two mistakes that can arise from a misunderstanding of phenomenography. First, phenomenography is a qualitative method of research, not quantitative. Hence we draw no conclusions about the number of assignments that fall into any of the above categories or the frequency with which students fit into a category. To make such conclusions would require significantly more data and a different research approach. The aim of phenomenographic research is to capture diversity. Second, the categories do not represent a single assignment. Typically, if an individual is shown the categories generated from phenomenographic research, they will identify with more than one position. There may be some positions to which they identify very strongly, and some positions to which they do not identify at all, but it is rare for a person to identify with only one category.

The assignments provided 57 separate sources of data, which is a relatively high number of sources for a phenomenographic study. Phenomenographers often continue to collect data until they believe they have reached "saturation". That is, they collect data and analyse it concurrently, ceasing to collect data when they have several consecutive interviews that do not lead to the identification of new categories. From our 57 sources, we do not claim to have reached saturation because the assignments were those not collected by students even though the assignments span the range of grades awarded. However, it was felt that within this data set the categories are a reasonable outcome space, i.e. categories of description of the variation in students' conceptions of OOISD constituted in aggregations of the types of errors or lack thereof made in CRUD matrices.

Examining more CRUD matrices in more assignments may add more categories, but is unlikely to invalidate the categories we have identified in this paper. Students chose to do the assignment to try for a credit or distinction grade. All students were required to attempt a final exam of 60 multiple-choice questions for a pass grade. The students whose assignments were used as the data set for this phenomenographic study received scores for the final exam ranging from 20 to 46, where 19 was the lowest score and 52 the highest for all students. The assignments are therefore a reasonable representation of the diversity of CRUD matrices in assignments. However, the study could benefit from follow-up work such as interviews where students are questioned about their understandings while undergoing the experience of doing a CRUD analysis.

Phenomenographers do not necessarily identify a unique set of categories from the same data. For example, if Cope[15] examined our data set, he may find evidence for the same categories he identified in his study of students' different ways of seeing information systems. Or if Eckerdal[18] were to examine our data set, she may find evidence for the same categories she identified in her first major study of students understanding of the concepts object and class. The categories identified in any study are to some extent dependent on the intent of the phenomenographer. Our intent was to identify the types of errors students made, and we chose our categories accordingly.

If phenomenographers do not necessarily identify a unique set of categories from the same data, is phenomenographic work therefore reliable and valid? Phenomenographic work can be considered valid and reliable in the following sense. If two people were given the description of the focus of the study, some categories, and some quotes from data, those people would usually place the quotes into the same categories. The readers can determine for themselves whether they would place most of the above examples into the same categories as those into which the authors have placed them.

In constituting our categories, we wanted to focus on the CRUD matrices in an assignment that also contained high-level use case descriptions, a use case diagram, and a class diagram. An analysis of these models could reveal different categories or dimensions of variation in students' conceptions of OOISD. We acknowledge this, but we regard it as separate to our concern. By focusing on the CRUD matrices and limiting our consideration of the other models to their relationship with the CRUD matrices, we identified a taxonomy of errors to inform teaching practice of OOISD. If the way OOISD is taught can be informed by this study, then a taxonomy of errors adds value to the Box & Ferguson CRUD analysis method.

For one of the authors, an unexpected insight to emerge from this study is the potential for CRUD analysis to be used as the first instrument for teaching OOISD. Until this study, the author had not intuited or reasoned about this possibility. We believe that it is possible to identify students' conceptions of OOISD by providing OO analysis models, asking the students to complete a CRUD analysis, and then design interventions to correct the students' conceptions of OOISD. This falls in line with the arguments of teaching students to read program code before writing program code. We would be asking students to read OO analysis models, complete CRUD matrices as a demonstration of their understanding of the models, before asking them to create OO analysis models.

5. CONCLUSIONS

It was asserted that CRUD matrices improve students learning of OOISD (Box, 2003). We doubt that CRUD matrices, of themselves, can improve students learning of OOISD. However, students do have varying conceptions about OOISD. We conducted a phenomenographic study that identified three categories of students conceptions about OOISD constituted as aggregations of types of errors or lack thereof made in CRUD matrices. From an analysis of 57 data sources, the categories are: fragmented and unstructured conceptions, conceptions about process override conceptions about objects, conceptions of design decisions appropriate within the object-oriented paradigm. These categories are probably not an exhaustive list, but we believe that further data gathering will not invalidate these categories.

These categories can be used to inform the teacher of students' affinities with the categories, from strongest to weakest. We believe that the teaching and learning of OOISD can be improved with the early use of CRUD matrices and at the same time informing students of the taxonomy of errors. Then introducing interventions to help make the students conceptions of OOISD more powerful. Our hope is that educators will use the categories we have identified to make explicit to students the errors that are known to occur if their understanding of OOISD is weak.

Beyond CRUD matrices, this paper demonstrates how phenomenography can be used as a tool for constituting categories of students' understandings of ISD. It can be used to define least powerful to most powerful understandings, before deciding on subject design or during the formative evaluation of subject design. We found the effort of analysing our data led to a reflection, or summative evaluation, of the subject design. By the time we finished the analysis, we saw merit in the information revealed in the categories, not just the more powerful category to which we had ascribed the most value. Indeed, we found variations of which we were not fully aware prior to this study. Beginning with a phenomenographic study may therefore lead to a more comprehensive approach to subject design in general.

REFERENCES

1. E. Gottesdiener, OO Methodologies: Process and Product Patterns, *Component Strategies* **1**(5) (1998).
2. J. Satzinger, R. Jackson, and S. Burd, *Systems Analysis and Design in a Changing World*, 2nd ed. (Course Technology, Boston, MA, USA, 2002).
3. L. A. Maciaszek, *Requirements Analysis and System Design: Developing Information Systems with UML* (Addison-Wesley, Harlow, England, 2001).
4. D. W. Brown, *An Introduction to Object-Oriented Analysis, Objects, and UML in Plain English*, 2nd ed. (John Wiley & Sons, 2002).
5. A. Dennis, B. H. Wixom, and D. Tegarden, *Systems Analysis and Design: An Object-oriented Approach with UML* (John Wiley & Sons, Inc., 2002).
6. D. Brandon, Jr., CRUD matrices for detailed object oriented design, *The Journal of Computing in Small Colleges* **18**(2), 306–322 (2002).
7. F. Armour and G. Miller, *Advanced Use Case Modelling: Software Systems* (Addison-Wesley, 2001).
8. AMS, *AMS best practices use cases: Advanced use case modeling* (AMS, 2003).
9. I. Box and J. Ferguson, *Object Oriented Software Development: Step by Step* (Pearson Education, Sydney, 2002).
10. I. Box, Old Trick, New Dogs: Learning to use CRUD matrices early in object-oriented information system development, in: *Constructing the Infrastructure for the Knowledge Economy: Methods & Tools, Theory & Practice*, Twelfth International Conference On Information Systems Development, Editor (Kluwer, Melbourne, 2003).
11. F. Marton, Phenomenography-a research approach to investigating different understandings of reality, *Journal of Thought* **21**(3), 28–49 (1986).
12. C. Bruce, Phenomenography in the Centre for Information Technology Innovation (CITI), in: *Current Issues in Phenomenography Symposium* (Canberra, Australia, 2002).
13. S. Booth, *Learning to program: A phenomenographic perspective (Dissertation abstract)* (1992).
14. C. Cope, P. Horan, and M. Garner, Conceptions of an Information System and Their Use in Teaching about IS, *Informing Science* **1**(1), 9–22 (1997).
15. C. Cope, *Educationally critical aspects of the experience of learning about the concept of information systems* (La Trobe University, Melbourne, 2000).
16. A. Berglund, On the Understanding of Computer Network Protocols, in: *Department of Information Technology* (Uppsala University, Uppsala, 2002), p. 76.
17. P. McDonald, The nature and acquisition of algorithm understanding: a phenomenographic investigation, in: *Current Issues in Phenomenography Symposium* (Canberra, Australia, 2002).
18. A. Eckerdal, *Resources for Learning Object-oriented Programming* (SIGCSE 2003 Doctoral Consortium, 2003).

APPENDIX A: UNIT OUTLINE CREATION AND APPROVAL CASE STUDY

Students were provided with a detailed description of the following case study and the assignment requirements, which included CRUD matrices as described by Box and Ferguson.[9] (A unit is synonymous with subject, course, or paper.)

This study is about the creation and approval of unit outlines within a school of computing and IT (SCIT). The system includes the creation of outlines by unit coordinators, the approval of the outlines by team leaders, and the creation of reports by administration staff.

Currently, the unit coordinator of each unit writes the unit outline, usually based on the previous version, rarely from scratch. After a team review, the team leader approves the outline, indicated by signing the cover page, and forwards it to the administration. The outline is made available online and archived in the SCIT's records. The current system is segregated and requires too much manual intervention. The desired system should reduce the time to enter and maintain outlines and derive reports.

An outline needs to be created each time a unit is delivered and to standards specified by the school and university. All outlines must have a consistent layout. An outline contains information that is the same for all outlines for a semester and information that varies from unit to unit, such as the content and learning objectives. The outline will start with a title page and contain sections such as: prerequisites, exclusions, assumed knowledge, introduction, student learning outcomes, content, delivery mode, practices of the school concerning assessment, method of assessment, clauses regarding academic malpractice, disabilities, and the learning skills unit, the recommended text and readings, unit coordinator, lecturer, and tutor contact details. The user interface needs to behave in such a way, as much as possible, so the coordinator can select options. The following limited set of high-level use cases were provided for correction and completion by the students.

Use case: Approve unit outline
Category: Core
Actors: Team Leader
Trace: <enter the traces to the business functions>
Description: This use case begins when a Team Leader chooses to approve a unit outline. The authority of the Team Leader is verified and the unit outline is marked as approved. On completion, the team leader and unit coordinator are notified of the approval, the unit outline is made available to the students and a copy is kept in the school archive.

Use case: Calculate summative assessment total
Category: Core
Actors: Team Leader
Trace: <enter the traces to the business functions> Included by: Record summative assessment
Description: This use case begins when a unit coordinator has completed the entry of all summative assessment. The summative assessment total is calculated by adding together the value of each summative assessment. On completion, the summative assessment total is shown onscreen.
Notes: The summative assessment total must equal 100.

Use case: Record summative assessment
Category: Core
Actors: Team Leader
Trace: <enter the traces to the business functions> Includes: Calculate summative assessment total
Description: This use case begins when a unit coordinator commences the entry of a summative assessment. The summative assessment details are recorded. On completion, the unit coordinator verifies that all summative assessment is complete.

Use case: Add new unit coordinator
Category: Core
Actors:
Trace: <enter the traces to the business functions> Extend from: Create unit outline, Administer users
Description: This use case begins when.... ... The use case ends when....

Use case: Modify existing unit coordinator
Category: Core
Actors:
Trace: <enter the traces to the business functions> Extend from: Create unit outline, Administer users
Description: This use case begins when.... ... The use case ends when....

Use case: Create unit outline
Category: Core
Actors:
Trace: <enter the traces to the business functions> Includes: Record summative assessment
Description: This use case begins when.... ... The use case ends when....

Use case: Provide explanation
Category:
Actors:
Trace: <enter the traces to the business functions> Extend from: Create unit outline
Description: This use case begins when.... ... The use case ends when....
Notes: Explanations of terminology and choices available when creating the unit outline.

Use case: Add new team leader
Category:
Actors:
Trace: <enter the traces to the business functions> Extend from: Administer users
Description: This use case begins when.... ... The use case ends when....

Use case: Modify existing team leader
Category:
Actors:
Trace: <enter the traces to the business functions> Extend from: Administer users
Description: This use case begins when.... ... The use case ends when....

Use case: Administer users
Category: Core
Actors: Administrator
Trace: <enter the traces to the business functions> Extends to:
Description: This use case begins when.... ... The use case ends when....

Use case: Modify unit outline
Category:
Actors:

Trace: <enter the traces to the business functions>
Description: This use case begins when.... ... The use case ends when....

Use case: Submit unit outline
Category: Core
Actors:
Trace: <enter the traces to the business functions> Extend from: Create unit outline, Administer users
Description: This use case begins when.... ... The use case ends when....
Notes: A unit coordinator must indicate that the unit outline is ready for the team leader's approval.

APPENDIX B: TEMPLATES FOR THE CRUD MATRICES

The following templates for the CRUD matrices are reproduced from Box and Ferguson (2002, p. 122, 123 & 124)

CRUD Class Analysis

Class	**P**	**Create**	**Read** (utilise)	**Update** (maintain)	**Destroy**
ClassName	*P/T/A*	*Use case name*	*Use case name*	*Use case name*	*Use case name*

Notes: ______________________________

Figure 1. CRUD class matrix format from Box & Ferguson, 2002, p. 122.

CRUD Association Analysis

Association	**Create**	**Read** (utilise)	**Update** (maintain)	**Destroy**
ClassAName-ClassBName	*Use case name*	*Use case name*	*Use case name*	*Use case name*

Notes: ______________________________

Figure 2. CRUD association matrix format from Box and Ferguson, 2002, p. 123.

CRUD Attribute Analysis – Class: *ClassName*

Attribute	**Initialise at Create**	**Read** (utilise)	**Update** (maintain)	**Reset**
attributeName	*Use case name*	*Use case name*	*Use case name*	*Use case name*

Notes: ______________________________

Figure 3. CRUD attribute matrix format from Box & Ferguson, 2002, p. 124.

END-USER COMPUTING IN BANKING INDUSTRY
A case study of a large Slovenian bank

Janko Hriberšek, Borut Werber, and Joze Zupancic*

1. INTRODUCTION

The term "end-user computing" (EUC) refers to people developing software applications for themselves or for others even they are not trained MIS professionals (Kreie et al., 2000). When compared to organizational computing, EUC has the tendency to be closely related to individual user tasks and motivations. For example, a financial analyst can create a spreadsheet to analyse and graph discrepancies between budget and actual performance numbers. A project manager can develop a small database to track the progress of the project and employee assignments.

Many user can not wait any more that the IS department produces some of the needed applications, in particular small applications to support their decision making, and develop the required systems by themselves. So, they can also avoid the delays and the time consuming procedures needed to obtain formal approval and funding for systems developed by the IS department. Often, users prefer to analyse data stored in corporate database in their own way. In this case the IS department can provide a facility for them to copy data from the database into a spreadsheet, and refresh it on demand.

From the start in the late 1970, EUC developed tremendously. Now, end-user development of applications forms a significant part of organizational systems development, and the ability to develop small applications is often a part of job requirements for many positions. EUC can also be considered as one of the emerging IT management practices that are changing the roles and responsibilities of IS specialists (Martinsons and Cheung, 2001).

Because EUC has changed considerably in the last few years, authors of a widely used text book on EUC, Regan and O'Connor (2000), suggested a new term, "end-user information systems" (EUIS) instead of end user computing. Edberg and Bowman (1996)

* Janko Hriberšek, Nova Ljubljanska banka d.d., Trg republike 2, 1520 Ljubljana, Slovenia. Borut Werber and Joze Zupancic, University of Maribor, School of Organizational Science, 4000 Kranj, Slovenia.

suggested the term "user-developed applications" (UDA). In most of the current literature, the term end-user computing (EUC) is still used.

Our study presents the results of an empirical investigation of EUC in the major bank in Slovenia, with emphasis on end-user support. The goal of the investigation was to identify and evaluate key factors of successful end-user support, in particular factors related to skills and knowledge needed for successful use of computers. Directions for further development of users' knowledge and skills are discussed in the paper.

2. MANAGEMENT AND SUPPORT OF EUC

It is difficult to estimate the entire contribution of end user developed applications to the corporate information system. A study from 13 years ago (Zupancic and Leskovar, 1991) showed that – according to the estimates of IS managers – the contribution of EUC to corporate IS was on average 10%: professional IS staff would need 10% more time to develop all the applications that have been developed by end users. Most likely, the contribution was even higher, since IS managers lacked a comprehensive overview of end-user activities. Several recent investigations indicate that in many companies up to 50% of the total IS budget is used for EUC (for example McGill, 2002).

Since end-users usually lack technical knowledge regarding software and hardware, they require adequate management and support to realize the productivity gains of EUC. Review of literature reveals that the following three forms of end user support are commonly found in most large organizations:

The Information Centre (IC). The basic goal of an IC is to help users help themselves. An important task of the IC is to maintain balance between support and control. The services offered by the IC can be grouped into the following six categories (Govindarajulu and Reithel, 1998):

- Hardware support (demonstration and standardization of hardware, outlining formal procedures for getting hardware approved, support of telecommunication hardware, . . .)
- Software support (assisting with application maintenance, variety of software supported, outlining formal procedures for getting software approved, support telecommunication software, . . .)
- Data support (assisting users in locating data, assisting with data transfer, providing backup, recovery and archiving, facilitating data sharing among users, maintaining subject data bases, . . .)
- Functional support (assisting in problem specification, designing the application, assisting in choosing techniques, developing applications with users, generating prototypes, . . .)
- Training and education
- Other (help desk, hot-line telephone assistance, literature, . . .)

Local MIS staff. These staff members exclusively support the users of a specific department and typically report to their functional department manager. Most routine IC functions are often assigned to local MIS staff, while ICs are mainly responsible for standardization, communication technology training, and data backup and recovery.

Informal support: Users generally seem to maintain an informal network for their support needs. The main sources of informal support are peers, friends/colleagues, and lead users – users who have more experience in and knowledge of IT than other users.

Many studies demonstrated that users are different and require different types and forms of education and training and different support when using various programming tools. Considerable difference was found in the needs for support and among departments in the same company. According to Speier and Brown (1997) the level and sophistication of EUC depends on (1) the attitude of the department manager towards EUC, (2) perception of EUC risks by the department manager, and (3) departmental EUC policy.

Several studies address the organization and management of users' support (e. g. Winter et al., 1997; Speier and Brown, 1997; Govindarajulu and Reithel, 1998; Govindarajulu et al., 2000; Govindarajulu 2002). Questionnaires were developed to measure the support from different sources, such as IC, local MIS staff, IT vendors and informal support, to investigate why users prefer a particular form of support over the others. The investigations showed following most important factors that impact the attitude of users towards different forms of support:

- Level of support the user received in the area of user's interest
- The proximity of the support
- Quality of the support staff
- Quality of user-developed applications

According to Govindarajulu et al., (2000) users preferred the support in advice of local MIS staff, their next choice was the IC, while informal forms of support were ranked the lowest. The following factors might impact the selection:

- Local support is easily accessible
- Local MIS staff deals with a relative small number of users, so they can devote more time and attention to the users
- Local MIS staff is familiar with the departmental business process and related problems.

The benefits that have been claimed for end user development of applications include better access to information and improved quality of information, leading to improved employee productivity and performance. However, realization of these benefits may be put at risk because information produced by user developed applications may be incorrect and applications may be poorly tested and poorly maintained (McGill, 2002).

Many recent studies address the quality of end-user developed applications. There are considerable differences between the system quality assessments by end-users developers and independent expert assessors. In particular, users with little experience may erroneously consider the applications they develop to be of high quality (McGill 2002). Training in system analysis and design methods was suggested to improve the application quality (Kreie et al., 2000). Because spreadsheets are the most commonly used tool for end-user development, many studies focus on the system quality of spreadsheets developed by end-users. McGill and Klobas (2004) investigated the role of spreadsheet knowledge in the successful use of spreadsheet applications. They demonstrated that spreadsheet knowledge not only influences the quality of the system being developed, but also acts directly upon individual impact of the application: successful use of spreadsheet applications appears to require sufficient knowledge to understand and, if necessary, to alter the user-developed spreadsheet application. Other studies (e.g. Morison et al., 2002; Kruck et al., 2003) found that electronic spreadsheet models to enhance decision making frequently contain errors that have significant negative effects on the ultimate quality of the decisions. An experimental study with 152 students who developed a simple spreadsheet (Panko and Sprague, 1998) showed that 35% of the spreadsheets (and 2% of cells) contained at least one error. By testing and inspecting the code users mostly were not able to detect the errors that may considerably impact the quality of the decisions. Therefore, Teo and Tan (1999) suggested that education and training of users should more emphasise the designing, testing and maintaining the spreadsheets than the various functions and possibilities offered by spreadsheets. The study (Kruck et al., 2003) suggests that training in cognitive skills and logical reasoning may help the users to develop better spreadsheets with less errors.

The list format appears to be the more natural way of representing data for people unfamiliar with dealing with abstractions of data and logical relationship between them. The study (Hobbs and Pigott, 2001) presents two case studies in which the first stage of a data base development process was completed entirely by the end user, making use of their own understanding of the dataset, the problem domain and tools that were familiar to them. An IT expert then facilitated the conversion of the dataset to relational database, with participation of end users throughout the process. So, the end users were able to see the concepts of database design emerge naturally from a problem that was already familiar to them, and to understand their importance in a practical manner.

3. THE INVESTIGATION

Our investigation, focusing on UEC support, was carried out in the largest bank in Slovenia, Nova Ljubljanjska banka d.d. (NLB). Nearly the entire business process in the bank is supported by computer systems, and most of the approximately 4000 employees use computers. Because most of the employees are involved in various forms of data and information analysis, we expected that many users will be have the propensity to develop their own applications that may support them in decision making and keeping their records. To justify the investment into HW and SW installed on the PCs, the available tools should be used efficiently and effectively. The goal of the investigation was to identify and evaluate key success factors of end-user support, in particular how users gain skills and knowledge needed for successful use of computing tools.

We developed a questionnaire based on the one used by Govindarajulu and Reithel (1998). The basic 6 groups of questions related to forms of support were left unchanged. We added questions that reflected the specifics of the business environment, to gain some additional insight into EUC in the bank, and identify needs for additional education and training of the users. Respondents rated the listed items on scale 1 to 7 where 1 denoted "not adequate" (or "not interesting", "not important", . . .), and 7 indicated "very adequate" (or "very interesting", "very important", . . .). They were also asked add comments to each group of questions. Among the key success factors of end-user support we focused on the following three forms of support:

- Information centre (called also “PC-team”, and included a “help desk”)
- Locals MIS staff responsible for the PCs (called also PC caretakers)
- Informal forms of support (friends, colleagues, relatives . . .).

A limited number of selected respondents reviewed the initial version of questionnaire and suggested several improvements. Data for the needs of our investigation was collected from employees that were the most likely developers of end user applications: (1) department and branch managers, (2) business analysts, (3) employees responsible for the organization and evaluation of business processes, and (4) sales personnel (commercialists). Based on an analysis of managerial and professional work assignments, we developed a list of 24 job titles, and a list of employees holding these job titles. All IS professionals involved in the development of corporate IS were eliminated from the list. The questionnaire was mailed to the remaining 496 employees.

The questionnaire was mailed to potential respondents in November 2001, and 97 of them were returned by the specified due date. After a detailed examination of responses, three questionnaires were eliminated because answers to some of the key questions were missing, yielding a 19.6% response rate. This is comparable to similar investigations published in the literature. Data from the remaining 94 (19.0%) questionnaires was analysed using the statistical package SPSS for Windows.

4. OVERVIEW OF THE RESULTS

About 75% of the employees of NLB are women; their portion among respondents was 67%. The average age of the respondents was somewhat less than the overall average age of the employees: 41.5% of them were between 31 and 40 years old, and 31.9% between 41 and 50. They have been using a computer from 2 to 23 years; the average was 10.6 years.

The users rated the complexity of applications that they developed on average by 2.47 (standard deviation 1.57) on scale 1 to 7, where 1 denoted “very simple”, and 7 “very complex”. This self-assessment indicates that users developed rather simple systems. The self-assessment of their computing skills was relatively high: they average rating on scale 1 to 7 was 4.0 (standard deviation 1.1)

Respondents rated the adequacy of the three forms of support (IC, local MIS staff, and informal) related to:

- Hardware (3 questions)
- Software (3 questions)
- IT purchase (4 questions)
- Data bases (5 questions)
- Quality of the support (6 questions)
- Application development (5 questions).

Table 1 presents a summary of their responses. Differences in average ratings of the three forms of support were relatively small, but it is evident that respondents rated informal support in most of the categories the highest, and support from the IC the lowest. The same ranking applies also to the overall quality of support.

Table 1. Average ratings of the three forms of support by category of support

	HW support	*SW support*	*HW/SW purchase*	*Data support*	*Appl. devel.*	*Quality of support*	*Total*
Information centre (IC)	4.10	4.30	4.03	4.25	3.88	4.75	4.30
Local MIS staff	5.03	4.87	4.87	4.84	4.30	5.16	4.88
Informal support	4.94	4.92	4.77	5.08	4.63	5.61	5.07

For most of the categories *hardware support* (demonstration of HW, standardisation of HW, use of telecommunication technology . . .) local MIS staff was rated as the most adequate, while standardization of HW was mostly considered to be the responsibility of the IC. Some respondents commented that the opportunities IC support is often not used enough, particularly when dealing with telecommunication technology, which is relatively complex and unfamiliar to most of the users.

The differences in ratings of the three forms of *software support* (SW maintenance, diversity of SW, use of communication SW . . .) were even lower than for hardware support, while the informal support was still rated the highest.

Only less than a half of the respondents answered the questions related to purchase support (selection and purchase of SW and HW, assistance in obtaining approval for the purchase . . .), and the responses indicate that users gave priority local MIS staff and informal support. The likely reason for the missing responses is that the bank standardised most of the HW and SW. When a user needs a computer, a workstation with all the standard software – purchased and in-house developed – is installed and delivered to his or her desk. There were only a few advanced users among the respondents who needed more sophisticated and sometime untypical SW and/or HW, and used the support of vendors when selecting and installing this kind of technology. Most the employees didn't need special programming tools, and will most likely never need tools such as ARIS, SPSS, or Rational Rose that are used only by small groups of users. This type of software can be ordered and installed only on the request by a specific user.

Data related support (locating data, backup, recovery and archiving, maintaining subject data bases . . .) was used only by about a half of respondents. Again they rated the informal support the highest. Not many users used informal support, but those who used this form of support rated it high.

Ratings of the *application development support* (definition of the problem, program design, choice of the programming technique, prototype development . . .) were in general lower than the ratings for other categories of support, and the ranking of the three forms of support was the same as for other categories.

Quality of the support (computing and business knowledge, ability to deliver support, communications skills, attitude towards the user . . .): respondents emphasised the quality of informal support, although we expected that the local MIS staff would be able to provide the best support for the local users. A possible explanation for this is that respondents rely more on their own networks than on services offered by the bank.

Table 2. Average ratings of the quality of different aspects of support

	Availa-bility	*Satisfaction*[†]	*Underst. the user*	*Education /training*[‡]	*Internet search*[§]	*Frequency of use*[**]
Information centre (IC)	4.29	4.83	5.15	4.34	4.35	3.12
Local MIS staff	4.96	5.25	5.70	5.00	4.98	4.47
Informal support	5.49	4.97	5.87	5.00	5.45	4.44

[†] Completeness, usability and importance of the information.
[‡] Basic and advanced training in computing and data transfer.
[§] Supporting the user when they searched for an information on the internet.
[**] The rating 1 indicates “never” and 7 “very frequently”.

Table 3. Average ranks of the sources knowledge needed for application development

Source of knowledge	*Average rank*
Co-workers from NLB	5.11
Literature	4.99
Internal courses	4.85
Friends acquaintances outside NLB	4.73
Internet	3.67
External courses	3.36
Regular study	3.09

Table 4. Perceived needs for additional education and training

Subject area	*Average rating*	*Standard deviation*
Spreadsheets (MS Excel, ...)	5.36	1.78
Using Internet (browsers, ...)	4.83	2.05
Text processing (MS Word, ...)	4.80	2.08
Presentation graphics (PowerPoint, ...)	4.73	2.06
Data bases (MS Access, ...)	4.71	2.09
Use of e-mail	4.70	2.16
Project planning (MS Project, ...)	4.19	2.27
Data base design and application design	3.62	2.32
Application testing methods	3.49	2.29

In addition to six groups of questions from the questionnaire used in (Govindarajulu and Reithel, 1998) we asked the respondents questions related to the quality of the support. Responses are summarized in Table 2.

Table 3 presents the average ranks of different sources of knowledge: a list of the seven sources was included in the questionnaire and respondents were asked to rank them by importance, where 1 indicated the least important and 7 the most important source.

Respondents were also asked to indicate tools and subjects where they needed additional education and training. Table 4 presents the average ratings: 1 denotes “not interested”, and 7 “highly interested”.

Because spreadsheets are the most widespread EUC programming tool in the bank, it is not surprising that users expressed the highest interest for additional knowledge on this subject. On the other hand, application and data base development methods and application testing which are essential for the quality of user-developed application (e. g. Kreie et al. 2000; McGill, 2002) ranked the lowest.

Often, some users spend a considerable amount of time for application development, which may result in lack of time for their regular assignments. Therefore, respondents were asked to rate the impact of their EUC system development activities on their professional work. Rating 1 denoted, "my professional work is not affected at all", and 7 denoted, "EUC strongly hinders my professional work". The average rating was 2.02, indicating only a minor negative impact on the regular work.

5. DISCUSSION

The investigation showed that users – application developers in NLB – prefered informal sources of support to the local MIS staff and the IC. The only exceptions were some specific areas, such as HW standardization, and approval of HW and SW purchases, where respondents rated the IC higher than the other two forms of support. Despite of the possibilities offered by the bank (IC, local MIS staff) users in most cases still favoured the informal sources of support. A detailed examination of the responses revealed that they used informal sources less frequently as the other two sources of support, but they rated them very high. The local MIS support was rated lower than the informal support. When the respondents were asked the question: "Which of the support sources would you recommend your co-workers?" they favoured local MIS staff (average rating 5.51) over the informal sources (4.87) and the IC (4.62).

A possible explanation for such attitude of the users towards different forms of support is that IC represents a remote support, which cannot help the users with their professional work, and only takes care of the functioning of the IS. Local MIS staff is familiar with the business processes and related problems in the departments, and are therefore rated higher, while emotional relations with friends, acquaintances and relatives may impact the rating of informal sources of support. On the other hand, IC is overloaded with administrative tasks such as recording users' problems and forwarding them to experts from the relevant subject area, which extends the response time to user's request. A computer application that would enable users to enter their problems into the system would save time to the staff of the "help desk" and allow them to focus more on users' problems and needs.

Unlike in our investigation, users in the study (Govindarajulu and Reithel, 1998), carried out in large companies in the USA, rated the support by the local MIS staff the highest, followed by the IC and informal support. This difference may indicate a lack of organized help factors: instead turning to professional employees, users are seeking help from people who are closest to them. Possible reasons for such behaviour are (1) lack of local MIS staff, (2) insufficient knowledge and skills of local MIS staff, and (3) lack of "formal help seeking" culture.

Ratings of the support quality (Table 2) do not indicate a shortage local MIS staff, because their availability was rated higher than the availability of informal support, and users were more satisfied with the local MIS staff support than with informal support. This indicates also that they did not perceive a lack of knowledge on the side of local MIS staff. A case study of a large manufacturing company in Slovenia from 13 years ago (Zupancic, 1993) showed the same as our study: the most important source of support for users was informal, colleagues, friends, acquaintances and literature. This supports the assumption of the lack of "formal help seeking" culture. A wider investigation would be necessary to reject or accept this assumption.

6. CONCLUSION

Adequate user support is essential for successful development of EUC, to fully exploit the potentials of EUC. Because EUC is based on non-professional programmers, the probability of mistakes and malfunctions in user-developed applications is relatively high. Increased level of computing knowledge and skills, resulting from adequate training of potential users – application developers, and different forms of user support can maximize the advantages of EUC and minimize its risks.

Based on the results of our investigation, we suggest the following measures to the management of NLB:

- Strengthen the role and redefine the function of the IC, so that it will be able to provide quick responses to concrete questions from the users, as basic precondition for successful development of EUC
- Provide a variety of training courses for PC tools
- Prepare additional training based on the wishes and interests of users (Table 4)
- Increase the level of knowledge and skills not only in using the PC tools, but also in application and data base design and application testing, particularly for the advanced users
- Develop training focusing on improvement of the quality of end user developed applications, particularly spreadsheets as the most common EUC tool.
- Consider the development and implementation of distance training for basic training, particularly to cover the topics that many end user programmers are interested in, such as spreadsheet development and testing
- Implement computer application that would enable users to enter their HW and SW related problems without directly contacting the support staff.

The study could be expanded by considering a broader range of support factors: in addition to the three form of support, user groups, and SW and HW suppliers should be also considered. Several organizations from the banking and financial sector and other organizations should be included, so that the conclusions from the study could be generalized to the financial and banking industry in Slovenia.

Another aspect that might require further investigation is the quality of user developed applications, in particular spreadsheets. Like most other organizations, NLB relies on the users' (authors') subjective judgement of the quality and correctness of the spreadsheets and other applications. Several investigations (e.g. McGill, 2002; Teo and Tan, 1999) demonstrated that users are mostly not able to detect errors in spreadsheets by testing and inspecting the code. Therefore a policy of testing and documenting user developed application should be established that would consider several elements of quality, as suggested for example in (McGill, 2002). An investigation of weaknesses errors in user developed application may help to define this policy and identify areas and directions where additional training of the users would be necessary.

Our study showed that most users were developing rather simple applications, which indicates that they probably do not fully exploit the computing potential available on their desktops. A goal of such study could be to identify the forms of management and support that would encourage more extensive use of computing equipment and help the users to maximize the productivity gains of EUC.

REFERENCES

Edberg, D. T., and Browman, B. J., 1996, User-developed applications: An empirical study of application quality and developer productivity, *Journal of Management Information Systems* **13**(1):167.

Govindarajulu, C., Reithel, B. J., and Sethi, V., 2000, A model of end user attitudes and intentions toward alternative source of support, *Information & Management* **37**:77.

Govindarajulu, C., and Reithel, B. J., 1998, Beyond the information centre: An instrument to measure end-user computing support from multiple sources, *Information & Management* **33**:241.

Hobbs, V. J., and Pigott, D. J., 2001, Facilitating end user database development by working with users' natural representations of data, in: *Managing Information Technology in a Global Economy: IRMA International Conference*, M. Khosrowpour, ed., Idea Group Publishing, Hershey, PA, pp. 650–656.

Kruck, K. E., Maher J. J., and Barkhi R., 2003, Framework for cognitive skill acquisition and spreadsheet training, *Journal of End User Computing* **15**(3):20.

Kreie, J., Cronan P. T., Pendley, J., and Renwick, J. S., 2000, Application development by end-users: Can quality be improved? *Decision Support Systems* **29**:143.

Martinsons, M. G., and Cheung C., 2001, The impact of emerging practices on IS specialists: Perceptions, attitudes and role changes in Hong Kong, *Information & Management* **38**:167.

McGill, J. T., 2002, User developed applications: Can end users assess quality? *Journal of End User Computing* **14**(3):1.

McGill, T. J., and Klobas, J. E., 2004, The role of spreadsheet knowledge in user-developed application success, *Decision Support Systems*, article in press available online at www.sciencedirect.com (April, 30, 2004).

Morrison, M., Morrison, J., Melrose, J., and Wilson, E. V., 2002, A visual code inspection approach to reduce spreadsheet linking errors, *Journal of End User Computing* **14**(3):51.

Panko, R. R., and Sprague, R. H., Jr., 1998, Hitting the wall: Errors in developing code and inspecting a 'simple' spreadsheet model, *Decision Support Systems* **22**:337.

Regan, E., and O'Connor, B., 2000, *End-user Information Systems: Implementing Individual and Work Group Technologies*, Prentice Hall, New York.

Speier, C., and Brown, C. V., 1997, Differences in end-user computing support and control across user departments, *Information & Management* **32**:85.

Teo, T. H. S., and Tan, M., 1999, Spreadsheet development and 'what if' analysis; quantitative versus qualitative errors, *Accounting, Management & Information Technology* **9**:141.

Winter, S. J., Chudoba, K. M., and Gutek, B. A., 1997, Misplaced resources? Factors associated with computer literacy among end-users, *Information & Management* **32**:29.

Zupancic, J., and Leskovar, R., 1991, An analysis of key issues in information systems development, in: *Proceedings of the 13th international Conference on Information Technology Interfaces*, Cavtat, Croatia, June, pp. 3–8.

Zupancic, J., 1993, End-user computing: A case study, in: *Proceedings of the United Kingdom Systems Society Conference on Systems Science: Addressing Global Issues*, F. A. Stowell, D. West, and J. G. Howell, eds., Plenum Press, New York, pp. 499–504.

WORKING WITH ALTERNATIVE DEVELOPMENT LIFE CYCLES: A MULTIPROJECT EXPERIMENT

Darren Dalcher, Oddur Benediktsson, and Helgi Thorbergsson*

1. MOTIVATION

Should the Waterfall model be dropped in favour of more modern approaches such as incremental development and eXtreme Programming? Many developers appear to show a preference for such modern approaches but there appears to be very little non-anecdotal data that substantiates such choices. IS, software development and software engineering books discuss the alternatives but offer little in the way of direct comparison and explicit metrics that address the impacts on the product, project and people.

The classic Waterfall model was refined in the early 1970s to cope with the larger and more demanding software projects characterised by a growing level of complexity. It was heavily influenced by early attempts to introduce structure into the programming process[1–7] and therefore offers a systematic development process leading to the orderly definition of artefacts. Many adaptations and adjustments have been made to the model leading to a variety of representations. Recent evidence suggests that it is still used extensively in many software development projects.[8–13] However, the proliferation of alternative models now offers a wide choice of perspectives and philosophies for development. Chief among them are incremental approaches,[14, 15] evolutionary approaches,[16] and more recently, Extreme Programming and agile methods.[17–19] Extreme Programming is often viewed as a lightweight methodology focused on the delivery of business value in small fully integrated releases that pass tests defined by the clients. Reduced focus on documentation and control allows development teams to produce more creative solutions leading to greater satisfaction all around.

Given the benefits of Extreme Programming in terms of creativity, value delivery and higher satisfaction levels, it is not surprising that many managers and developers have adopted such practices. To date however, there is not much information on the direct comparisons between the different approaches in terms of the quality, functionality and scope

* Oddur Benediktsson and Helgi Thorbergsson, University of Iceland, Dunhaga 5, IS-107, Reykjavik, Iceland. Darren Dalcher, Software Forensics Centre, Middlesex University, Trent Park, Bramley Road, London N14 4YZ, UK.

of the resulting products. Nor is there much information regarding the likely impact on the project, the intermediary products and the development team.

The aim of this work was to investigate the impacts of different approaches. Fifteen teams working on comparable applications made use of four development approaches ranging from a sequential, via incremental and evolutionary development to Extreme Programming. The resulting data on the relative merits, attributes and features of the products, the time spent in each stage and the measures of requirements, design and programming outputs provide a start towards understanding the impact of selecting a programming and management approach and towards making informed decisions about which one to apply under different circumstances.

2. BACKGROUND

The skillset focusing on the life cycle of IS projects is critical to both understanding and practising sound development and management. Indeed life cycles are prominently featured within the Bodies of Knowledge of many different disciplines (including the PMBoK, SWEBOK and APM BOK).[20–22] A life cycle represents a path from the origin to completion of a venture. Phases group together directly related sequences and types of activities to facilitate visibility and control thus enabling the completion of the venture. The project life cycle thus acts as a framework focusing on the allocation of resources, the integration of activities, the support of timely decision making, the reduction of risk and the provision of control mechanisms. The benefits of using a life cycle approach identified by[23] include attaining visibility, providing a framework for co-ordinating and managing, managing uncertainty, and providing a common and shared vocabulary.

However the divergent choice in approaches leads to a dilemma when it comes to selecting the most suitable approach for a project. At the beginning of every project the manger is expected to commit to a development approach. This is often driven by past experience or other projects that are, or have been, undertaken by the organisation. In practice, little work has been conducted in this area and it is unusual to find studies comparing empirical result (with very few notable exceptions e.g. specification vs. prototyping).[24, 25]

Typical approaches include sequential, incremental, evolutionary and agile approaches. Each is likely to be better suited to a particular scenario and environment and to result in certain impacts on the overall effort and the developed products. These approaches as discussed in[23] are highlighted below.

Sequential Approaches. Sequential approaches refer to the completion of the work within one monolithic cycle leading to the delivery of a system.[10] Projects are thus sequenced into a set of steps that are completed serially typically spanning determination of user needs to validation of its achievement. Progress is carried out in linear fashion enabling the passing of control and information to the next phase when pre-defined milestones are reached and accomplished.

Incremental Approaches. Incremental approaches emphasise phased development by offering a series of linked mini-projects (referred to as increments, releases or versions).[15, 18, 26, 27] Work on different parts and phases, is allowed to overlap throughout the use of multiple mini-cycles running in parallel. Each mini-cycle adds functionality and capability. The approach is underpinned by the assumption that it is possible to isolate meaningful subsets that can be developed, tested and implemented independently.

Evolutionary Approaches. Evolutionary approaches[16, 28] recognise the great degree of uncertainty embedded in certain projects and allow developers and managers to execute partial versions of the project while learning and acquiring additional information. Evolutionary projects are defined in a limited sense allowing a limited amount of work to take place before making subsequent major decisions.

Agile Approaches. Agile development has proved itself as a creative and responsive effort to address users' needs focused on the requirement to deliver relevant working business applications quicker and more cheaply.[17–19] The application is typically delivered in incremental (or evolutionary or iterative) fashion. Agile development approaches are concerned with maintaining user involvement[29] through the application of design teams and special workshops. Projects tend to be small and limited to short delivery periods to ensure rapid completion. The management strategy relies on the imposition of *timeboxing*, strict delivery to target which dictates the scoping, the selection of functionality and the adjustments to meet the deadlines.[30]

Each approach appears to have clear benefits, at least from a theoretical perspective. Project managers are expected to select the most suitable approach to maximise the chances of successfully delivering a product that will address the client's needs and prove to be both useful and usable. The choice should clearly relate to the relative merits of each approach. However, not enough is known about the impact of each approach and very little comparative studies that can of the relative merits have been conducted. Real data is needed in order to facilitate comparisons between the approaches. Indeed, such measurements are likely to lead to better informed choices in preference to the adoption of fads. The rest of the paper describes the experiment, reviews the results, frames them and draws the resulting conclusions.

3. THE EXPERIMENT

The experiment was designed to investigate the impact of a development approach on the product and its attributes. It involved 55 developers in fifteen teams developing comparable products from the same domain. The objectives of the experiments were to investigate the differences in resource utilisation and efficiency between the 15 solutions.

The product to be developed was an interactive package centred on a database system for home or family usage with an appropriate web-based user interface. The basic scope definitions of the projects were prepared in advance to ensure relative equality in terms of complexity and difficulty. Each product was slightly different using a different set of users. Users were consulted through the process, reviewed the system and came up with useful comments. Having fifteen distinct projects (as opposed to multiple versions of the same project) reduced the likelihood of plagiarised solutions across the different groups.

All groups were asked to use JAVA and J2EE for transaction processing. Most used Sun public domain server tools for the deployment of J2EE and the Cloudscape database. A small number of the groups used JBoss server, MySQL database system and Borland tools. Regardless of the platform they selected the groups were instructed to experiment their way through the technology with little formal help. The result of the effort was a working product that can be viewed as working usability or technology prototypes.

No preferences regarding the project, team or overall approach were accommodated. This is largely representative of real-life project environments. The 55 developers were randomly allocated to groups each consisting of three or four developers. Each project title (with its general definition) was also randomly allocated to a group. Finally, in order to remove any bias, each group was also allocated a random development life cycle (from a set of four).

The set of software development methods included one representative from each class of approaches. The V-model (VM) (see for example[31] or IEEE830) was used as an example of a Waterfall-type sequential approach. The Incremental Model (IM) and the Evolutionary Model (EM) were explained in advance and used in accordance with the description in[31]. The agile approaches were represented by Extreme Programming (XP) as defined by Beck.[19]

The experiment took place in 2003 as part of a full year, two semester project. All participants were Computer Science majors in the final year of their programme at the University of Iceland. Students had lectures twice a week and had one project meeting a week. Each method was designated an expert leader (method champion) from the faculty.

Data was collected regularly to ensure all stages of the processes were covered. Students were encouraged to keep logbooks to record measures and faults. Measures were checked and confirmed by the relevant method champion to ensure accuracy and correctness. Students understood that the grades were not linked to time and error measurements recorded in their logs and therefore had no reason to misrepresent activities or events. They were further encouraged to record data and events as they happened rather than resort to memory.

4. RESULTS

4.1. Time Spent on the Project

Table 1 shows the effort data as reported by the groups. Time spent was measured in hours and divided into the seven general activities giving an overall total. The total was converted into Project Months (PM) at 152 hours per PM. The groups are ordered by the development model used – V-model (VM), Evolutionary Model (EM), Incremental Model (IM), and Extreme Programming (XP). Sub-averages are given for each group as well as the over all average (OAve) and the over all distribution of time per activity.

One of the Incremental groups spent 8.4 PM (1275 hours) on the project. This group is taken to be a statistical outlier as their result is out of line with other groups and the overall time availability and is therefore dropped from the tables below. (Rationale: The value of 8.4 PM is well above the likely maximum of 6.6 PM as computed by the rule of thumb: $median + 1.5 Interquartile\_Range$.)

The V-model projects took longest with an average of 748 hours, followed by XP, Evolutionary, and Incremental (see[32]). The longest time spent by a group, 919 hours, belongs to a V-model group whilst the shortest period, 372.5 hours, was experienced by an XP group. The requirements activity was demanding for the V-model, Evolutionary and

Table 1. Effort in hours by activity

Group	*Reqs Spec*	*Design*	*Code*	*Integr. & Test*	*Review*	*Repai r*	*Other*	*Total hours*	*Total PM*
VM1	65	76	111	11	23	36	194	515	3.4
VM2	68	54	291	13	75	108	301	907	6.0
VM3	97	80	231	31	97	68	317	919	6.0
VM4	66	66	192	89	34	31	175	651	4.3
Ave	*73.9*	*68.6*	*206.1*	*35.6*	*56.9*	*60.4*	*246.5*	*748.0*	*4.92*
EM1	84	93	255	63	6	75	165	741	4.9
EM2	37	73	116	39	14	63	122	463	3.0
EM3	83	46	230	48	34	35	112	588	3.9
EM4	68	20	76	81	38	19	104	406	2.7
Ave	*67.8*	*58.0*	*169.1*	*57.8*	*23.0*	*48.0*	*125.6*	*549.3*	*3.61*
IM1	57	52	259	30	14	52	56	520	3.4
IM2	35	77	160	68	61	74	131	607	4.0
IM3	40	25	138	24	38	36	178	478	3.1
Ave	*43.8*	*51.2*	*185.7*	*40.7*	*37.7*	*53.8*	*121.6*	*534.5*	*3.52*
XP1	26	0	256	83	42	92	238	736	4.8
XP2	12	66	246	124	76	106	39	669	4.4
XP3	11	13	115	42	23	80	90	373	2.5
Ave	*16.2*	*26.2*	*205.6*	*82.7*	*46.9*	*92.7*	*122.4*	*592.5*	*3.90*
OAve	**53.3**	**52.8**	**191.1**	**53.1**	**41.0**	**62.4**	**158.6**	**612.1**	**4.03**
Tot %	**9**	**9**	**31**	**9**	**7**	**10**	**26**	**100**	

Incremental teams but was less so for the XP teams. A similar trend was identified during the design activity. Programming was led by the V-model teams, as was the Reviews activity, while integration and testing was led by the XP groups. Repair was also led by the XP teams.

4.2. Requirements Output

The requirements output (Table 2) was assessed in terms of the number of pages, words, lines, Use Cases, screens, Database diagrams, DB tables and data items in DB.

The Incremental model resulted in a leading average output of 3262 words. XP in contrast averaged 542 words. Similar trends were observed for pages and lines, but in both of these the V-model just managed to overtake the Incremental model. The number of screens was dominated by XP with 17.7, followed by Evolutionary with 11, V-model with 6.8 and Incremental with 4.5. It should be noted that the XP groups did not design Use cases but made up Stories. The Stories were counted as Use cases. Overall, the V-model appears to have necessitated the most time, to have used the most words and correspondingly to have provided the fewest diagrams (i.e. a picture is still worth a thousand words).

4.3. Design Output

Design output (Table 3) was evaluated in terms of the number of design diagrams produced including; architecture, class, sequence, state, and other diagrams.

As the number of diagrams is relatively small it is easier to focus on the total. The number of diagrams per group ranges between 1 and 22. The group producing 22 diagrams

Table 2. Metrics for requirements and database

Group	*Pages*	*Words*	*Lines*	*Use cases*	*Screens*	*DB diagrams*	*DB Tables*	*Data items*
VM1	26	3605	1046	17	12	1	4	22
VM2	25	2560	829	5	4	1	2	34
VM3	22	3641	890	2	2	1	3	18
VM4	15	1887	431	6	9	1	4	40
Ave	*22.0*	*2923*	*799*	*7.5*	*6.8*	*1.0*	*3.3*	*28.5*
EM1	11	1496	463	14	7	1	8	46
EM2	15	2564	600	14	14	1	9	44
EM3	29	3263	971	14	13	2	4	27
EM4	21	3051	821	14	10	1	6	37
Ave	*19.0*	*2594*	*714*	*14.0*	*11.0*	*1.3*	*6.8*	*38.5*
IM1	20	3377	720	16	2	1	6	22
IM2	27	3410	887	13	11	2	2	9
IM3	21	3000	564	15	2	1	5	29
Ave	*22.7*	*3262*	*724*	*14.7*	*5.0*	*1.3*	*4.3*	*20.0*
XP1	4	730	158	*9*	14	1	3	26
XP2	1	435	42	*10*	17	0	8	25
XP3	12	461	69	*6*	22	1	4	17
Ave	*5.7*	*542*	*90*	*8.3*	*17.7*	*0.7*	*5.0*	*22.7*
OAve	**17.8**	**2391**	**607**	**11.1**	**9.9**	**1.1**	**4.9**	**28.3**

was utilising the V-model, while an XP group was responsible for the single diagram (compared to other XP groups with 2). The average for the Incremental groups is 15.7 diagrams, the V-model 12.8, Evolutionary 8.5 and XP 1.7. XP teams clearly produce fewer diagrams during the design stage compared with the V-model and Incremental development.

4.4. Final Product

The product size (Table 4) was measured in terms of Java classes and the number of lines of code. Lines of code include lines produced in JAVA, JSP, XML and any other languages.

The number of JAVA classes varies between 7 and 50. XP teams averaged 27.7 classes, compared with Evolutionary with 26.8, Incremental with 20.7 and V-model with 8.5.

In terms of Java lines of code, the XP teams delivered an average of 4836 compared with 1032–1803 LOC from the other teams. The picture is mirrored in terms of XML lines of code. The comparison of the total lines of codes produced a striking result as the XP model teams delivered significantly more lines of code than any one else. The 3.5:1 range in product size between XP and the V-model is remarkable, considering that all teams worked on essentially similar projects. The results would suggest that XP has a higher productivity rate in terms of the size of the delivered product. It is worth noting that the V-model teams spent a lot of time and effort on the requirements specification and the design activities. The V-model requires sequential progression so that technology is only addressed during the implementation stage. As a result they did not start experimenting with the technology

Table 3. Design metrics

Group	*Overview diagrams*	*Class diagrams*	*Sequence diagrams*	*State diagrams*	*Other diagrams*	*Total*
VM1	1	4	2	0	3	10
VM2	1	4	2	0	0	7
VM3	1	1	3	14	3	22
VM4	1	1	2	7	1	12
Ave	*1.0*	*2.5*	*2.3*	*5.3*	*1.8*	*12.8*
EM1	1	1	3	2	1	8
EM2	2	1	4	0	2	9
EM3	1	2	1	0	2	6
EM4	1	2	7	0	1	11
Ave	*1.3*	*1.5*	*3.8*	*0.5*	*1.5*	*8.5*
IM1	1	7	5	0	2	15
IM2	4	1	14	0	1	20
IM3	2	3	4	0	3	12
Ave	*2.3*	*3.7*	*7.7*	*0.0*	*2.0*	*15.7*
XP1	2	0	0	0	0	2
XP2	0	0	0	0	1	1
XP3	1	0	0	0	1	2
Ave	*1.0*	*0.0*	*0.0*	*0.0*	*0.7*	*1.7*
OAve	**1.4**	**1.9**	**3.4**	**1.6**	**1.5**	**9.8**

until very late in the project. They were then forced to go back and modify the design thus affecting their overall time allocation.

Quality was assessed following ISO9126 measures and focusing on the five areas of Functionality, Reliability, Usability, Efficiency and Maintainability. The resulting differences are not significant enough to discuss in this paper, save to point out that for maintainability, the leading average was provided by the V-model with 8.9, followed by Evolutionary with 7.9, XP 7.7 and Incremental with 7.3.

Each team was meant to collect defect data and classify it into 7 types of defects according to the Orthogonal Defect Classification (ODC) scheme.[33, 34] This data is known to be heterogeneous and inconsistent and will therefore be ignored in the main. It is useful however to note that the V-model dominated most categories of defects resulting in an average of at least double the defect rate produced by other groups. Finally, the most **comprehensive** solutions, resulting in the highest level of **satisfaction** from the users, were provided by the XP teams.

5. CONCLUSIONS, OBSERVATIONS AND LIMITATIONS

The results of this experiment provide some useful quantitative information on the relative merits of different development methods and on the impact they can have on different activities. The experiment conducted here is viewed as a first step towards providing a clearer understanding of the issues related to the choice of development approaches. The

Table 4. Performance metrics: Classes and lines of code

Group	*Java classes*	*Java*	*jsp*	*XML*	*Other*	*Total LOC*
VM1	10	905	1105	0	46	2056
VM2	7	648	427	12	45	1132
VM3	8	1300	2600	40	16	3956
VM4	9	1276	511	0	0	1787
Ave	*8.5*	*1032*	*1161*	*13*	*27*	*2233*
EM1	50	1665	1309	0	0	2974
EM2	9	1710	2133	0	130	3973
EM3	31	1254	2741	0	0	3995
EM4	17	1278	1429	0	17	2724
Ave	*26.8*	*1477*	*1903*	*0*	*37*	*3417*
IM1	42	3620	0	73	0	3693
IM2	10	289	1100	36	0	1425
IM3	10	1501	1666	0	42	3209
Ave	*20.7*	*1803*	*922*	*36*	*14*	*2776*
XP1	16	1849	2105	0	638	4592
XP2	38	6028	1229	2379	124	9760
XP3	29	6632	1652	583	0	8867
Ave	*27.7*	*4836*	*1662*	*987*	*254*	*7740*
OAve	**20.4**	**2140**	**1429**	**223**	**76**	**3867**

results can be sensitive to the performance of individuals within groups and to the synergy created within any group. Indeed, one group included a number of accomplished programmers working for the Iceland Telephone company and a bank. Nonetheless, the results are significant in highlighting trends and providing a comparison between the different approaches.

5.1. Time Spent on Project

Figure 1 depicts the distribution of effort spent by the four categories of groups. Despite the fact that all projects addressed comparable problems, V-model projects took somewhat longer than the other projects. The other three models fell within 10% of each other. In terms of the significance of the results, XP only consumed a minor proportion of the effort during the requirements and design activities compared to the other models. It is interesting to note that XP teams took the lead in terms of hours spent in testing and code correction. The effort of the Waterfall teams dominated the requirements, design and even the programming activities. Integration and testing required relatively little effort from the Waterfall (and incremental) teams, presumably due to the level of detailed planning and additional time spent during the earlier activities of requirements and design, possibly also indicating earlier discovery of errors.[10, 37]

In terms of all projects, the activities of requirements and design were each responsible for 9% of the overall time spent. In XP however, requirements accounted for 2.7% and design for 4.4%. Overall, programming was responsible for 31% of the time, integration and

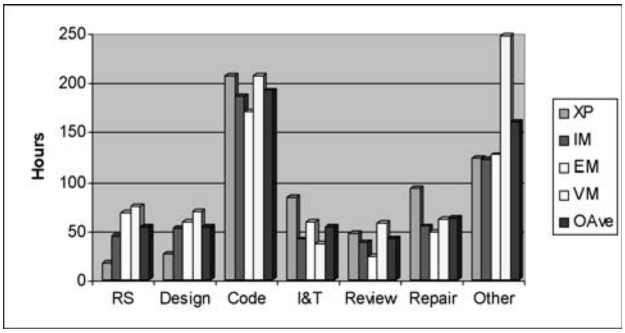


Figure 1. Average effort distribution by activity.

testing 9%, reviewing 7%, repair 10% and other activities for 26% of the time. Activities not directly considered as part of the life cycle (‘other’) still accounted for over a quarter of the time and must therefore be planned for. This is roughly in line with industrial rules of thumb for additional (and non-technical) activities. Note however that the time spent on ‘other’ activities by V-model teams was on average double the time spent by the other teams.

Excluding the non-technical activities produces an overall profile for all teams of 12% for requirements, 12% for design (against an expectation of 20%), 41% coding, 12% integration and testing, 9% review and 13.5% for repair. Indeed, one could argue that about a third of the technical effort was dedicated to quality activities including testing, integration, review and repair, which is roughly in line with the typical rule of thumb for quality activities. The early definition activities of requirements and design thus seem to add up to just under a quarter of the overall development effort. Overall, the figures are still slightly high in terms of coding (cf.[36], 15%), and marginally low in terms of design.

Figure 1 also reveals the familiar shape of a Rayleigh curve (with significant coding humps) for each of the four model types.[37] It is worth noting the slight drop between the requirements and design efforts in both the incremental and the evolutionary methods due to some of the preparatory work being completed upfront.

It is also interesting to compare the relative positioning of the XP model curve as the curve starts at much lower point for the early activities (representing less expended effort during these activities) but then rises to the maximum level of effort during the coding activity thus representing the steepest climb rate out of the four models presented. This supports the reported focus of XP and other agile methods on delivering useful code and spending less upfront effort through the avoidance of exhaustive analysis and documentation.[19, 29] It is also interesting to note the higher level of integration and testing effort that goes into XP. The lighter effort upfront, combined with the heavier loading in terms of coding, integration and testing make for a Rayleigh curve with a much steeper diagonal tilt.

The results seem to confirm the notion that XP requires less effort during the initial stages, in particular during the requirements activity. Repair activities in XP consumed more resources than in the incremental and evolutionary models. In closing it is also significant to note that 'other' activities were responsible for consuming significantly fewer hours in XP, than they did in the V-model.

5.2. Requirements and Design Outputs

The V-model and the incremental model produced a significant amount pf pages, words and lines of requirements. The Evolutionary method was not far behind. XP produced less than a quarter of the number of pages, roughly a sixth of the number of words and between a seventh to an eighth of the lines of specification produced by the V-model and the Incremental model. In other words, XP results in significantly less pages of requirements and in less words being used to describe these requirements.

XP produced significantly more screens than the other methods. In terms of Use Cases both the V-model and XP teams produced significantly less Use Cases than the Incremental and Evolutionary teams. XP produced in average less than two design diagrams compared with almost 16 produced by the Evolutionary and 13 by the V-model.

5.3. Product Size and Productivity

XP produced 3.5 times more lines of code than the V-type model, 2.7 times the LOC produced by the Incremental model, and 2.2 times more code than the Evolutionary teams. This significant trend is also reflected in terms of the number of LOC produced in Java and XML.

Another way of viewing the product size is through the classification of size categories provided by Boehm[35] and by Fairley.[38] Both asserted that a small product is in the range of 1–2K LOC. While Boehm suggested that a typical intermediate/medium product is the range of 8–32K LOC, Fairley adjusted the range to 5–50K LOC. The XP groups are the only teams to have produced products that would clearly qualify as medium-sized products according to both criteria. Significantly, these products were also viewed as the most comprehensive and the most satisfactory products. Given the same amount of time, XP teams were thus able to build on the feedback from small and frequent releases, resulting in an intermediate/medium product (compared to the smaller products delivered by the other teams). The ability to involve users coupled with regular delivery seems to have resulted in additional useful functionality and a greater level of acceptance and satisfaction thus making the resulting product more valuable.

XP teams displayed 4.8 times more productivity in LOC/PM than the V-model teams, 2.8 more than the Incremental teams and 2.3 more than the Evolutionary teams. So the difference in productivity is more pronounced than that observed for product size. The picture is repeated for the derived function of number of pages of LOC. The average productivity is 1077 LOC/PM with XP dominating with an average of 2262 LOC/PM. Industrial productivity is often taken to be in the range of 200 to 500 LOC/PM. With the exception of the V-model, all methods averaged above the upper range of the expected productivity level. However, XP clearly offered outstanding productivity. The average productivity is also computed at 5.5 classes/PM (with the V-model clearly trailing with 1.7 classes/PM). Kan[39] reports productivity in C++ and Smalltalk projects in the range 2.8 to 6 classes/PM. With the exception of the V-model teams, all models offered average performance at or above the top end of the productivity figure put forward by Kan. All V-model teams performed at or below the lower limit of Kan's range (in fact only one V-model team was within the range with the other three performing clearly below it).

5.4. The Experiment

The experiment yielded valuable results. The teams delivered working prototypes (not final quality products). The technology used may have played a role. The teams using Sun public domain server tools and Cloudscape were working with tools that were not fully mature products. The few teams utilised the JBoss server, MySQL and Borland tools which are more mature and seemed to work better.

The need to experiment with technology proved time-consuming for some of the groups. This became an issue for the V-type model teams as the experimentation was delayed until after the requirements were fully understood and specified and the design was stable. The impact of exploring the technology meant that these teams were forced to re-assess some of their design solutions in line with their emerging understanding the technical environment. Indeed, this touches on the relationship (and intertwining) between design and implementation environment and the need to integrate the two.[40] The discussion of a problem is often enhanced by the consideration of design and implementation concerns. The constraints imposed on a solution by later stages need to be acknowledged and addressed to reduce the conflicts that will need to be resolved at a later stage. Methods that create functional barriers and that do not allow a relationship, albeit rudimentary to the notion of the solution may thus play a part in delaying essential interactions thereby arresting progress and requiring subsequent rework cycles to rectify the effects of the separation.

5.5. Limitations

The experiment involved 15 teams working on comparable projects utilising four different models. As a result the sample size for each model is three to four groups (in line with other known experiments in this area using two, three or four groups in each category).[24, 25] Despite the limited number of data points the experiment offers a quantitative investigation of the extent to which the development approach affects the outcome of a project and an experimental comparison of the different software development life cycles and methods. The empirical findings are therefore viewed as a systematic interpretation offered as part of a more comprehensive study, rather than as conclusive answers.

Employing students as developers offers many benefits to all parties. Most significantly, it enables students to interact with real development tasks, to gain experience in teamwork and to work on non-trivial problems from beginning to end thus embracing the entire life cycle. From the point of view of the experiment, it enables direct experimentation in the development of similar or comparable products through the use of alternative methods and approaches. However, this also implies that the experiment was conducted in educational settings, rather than an industrial environment. Consequently, certain aspects of the environment which are normally associated with XP practices, such as sustainable pace, on-site customer, continuous communication and open workspace may have been slightly compromised. On the other hand the definition of the work and its relevant context meant that participants were stakeholders themselves. It is impossible to surmise whether the XP results would have been even more pronounced under the full conditions recommended as best practice for XP development.

5.6. Concluding Comments

Selecting the most suitable method is contingent on the context and participants. The direct comparison between the four approaches offers an interesting way of quantifying the relative impacts of the various approaches on the product. Whilst incremental and evolutionary approaches have been offered as improvements to sequential approaches, the added comparison with XP is instructive. XP is currently offered as a lightweight alternative to the sequential notion. The direct comparison between the four approaches therefore provides a quantitative basis for beginning to consider the impact of the different approaches thus bridging the gap between the descriptions of the different life cycles and the need to make an educated choice based on the likely impact of the approach. The comparison yields interesting results and comparative measures (e.g. V-model teams and XP teams produced significantly less Use Cases than incremental or Evolutionary teams).

In the experiment XP teams produced solutions that encompassed additional functionality. Indeed, in terms of significant results, their products can be characterised as consisting of the largest number of screens and the most lines of code. In terms of the process, they can be said to have spent the least time on requirements and consequently to have produced the least number of pages of requirements. They also generated significantly less diagrams. Early work on how programmers spend their time seemed to indicate that a very small proportion of time (13–15%) is spent on programming tasks.[38] Methodologies like XP attempt to overcome that by optimizing the time available for programming (i.e. minimal documentation) resulting in enhanced output (more code and more screens) that is delivered more rapidly.

Experience: The student experience was evaluated by way of a survey conducted at the end of the final semester that looked at the results. Most students working in Incremental, Evolutionary and XP teams were content with their model. Intriguingly, ALL students in groups utilizing the V-model indicated a strong preference towards using a different model to the one they were allocated and were therefore less satisfied with the process.

5.7. Future Work

It is intended to continue with the experiments and the group projects. The lessons learned from the first year of the experiment will result in a number of small changes:

- Following the strong preference against the use of a sequential model the groups will be allowed to choose their development model. It will be interesting to see how many of the teams will select some form of agile methods.
- The suite of tools selected will be based on mature and more unified technology. Groups will be instructed to use Eclipse platform-based Java and J2EE with integrated use of Ant, CVS, JUnit, and MySQL based on the JBoss server. An OO metrics plug-in for Eclipse will also be utilised.
- To ensure that all teams deliver useful products there will be firm dates for delivering partial increments (two in the first semester, three in the second)

It is clear that the large number of variables makes a simple decision about the 'ideal method' and the appropriate set of tools difficult, if not impossible. The different metrics reveal that each method has relative strengths and advantages that can be harnessed in specific situations. Experiments such as this make a significant contribution to understanding the relative merits and their complex relationships. As the experiment is improved, and hopefully repeated elsewhere, a clearer picture of the issues, merits and disadvantages is likely to emerge and a deeper understanding of the role and application of each life cycle method will hopefully ensue.

REFERENCES

1. R. G. Canning, *Electronic Data Processing for Business and Industry* (John Wiley, New York, 1956).
2. R. G. Canning, *Installing Electronic Data Processing Systems* (John Wiley, New York, 1957).
3. H. D. Bennington, Production of large computer programs, *Annals of the History of Computing* (4), 350–361 (1956) (5 Oct. 1983).
4. W. A. Hosier, Pitfalls and Safeguards in Real-Time Digital systems with Emphasis on Programming, *IRE Transactions on Engineering Management*, pp. 91–115 (1961).
5. W. W. Royce, Managing the development of large software systems: Concepts and techniques, in: *Proceedings, IEEE WESCON* (August 1970).
6. H. N. Laden and T. R. Gildersleeve, *System Design for Computer Applications* (John Wiley, New York, 1963).
7. L. A. Farr, Description of the Computer Program Implementation Process, SDC Technical Report, 1963.
8. C. J. Neill and P. A. Laplante, Requirements engineering: The state of the practice, *IEEE Software* **20**(6), 40–45 (2003).
9. P. A. Laplante and C. J. Neill, The demise of the waterfall model is imminent and other urban myths, *ACM Queue* **1**(10), 10–15 (2004).
10. R. S. Pressman and D. Ince, *Software Engineering: A Practitioner's Approach*, 5 ed. (McGraw-Hill, Maidenhead, 2000).
11. B. W. Chatters, Software Engineering: What do we know? in: *FEAST 2000* (Imperial College, London, July 2000).
12. D. Dalcher, Towards continuous development, in: *Information Systems Development, Advances in Methodologies, Components and Management*, edited by M. Kirikova et al. (Kluwer, New York, 2002), pp. 53–68.
13. M. A. Cusumano, et al., *A Global Survey of Software Development Practices* (MIT, Cambridge, Ma., 2003), pp. 1–17.
14. H. D. Mills, Incremental software development, *IBM Systems Journal* **19**(4), 415–420 (1980).
15. D. R. Graham, Incremental development and delivery for large software systems, *IEEE Computer*, pp. 1–9 (1992).
16. T. Gilb, Evolutionary development, *ACM SIGSOFT Software Engineering Notes* **6**(2), 17 (1981).
17. A. Cockburn, *Agile Software Development* (Addison-Wesley, Boston, MA, 2002).
18. C. Laraman, *Agile and Iterative Development: A Manager's Guide* (Addison-Wesley, Boston, MA, 2004).
19. K. Beck, *Extreme Programming Explained: Embrace Change* (Addison-Wesley, Boston, MA, 2000).
20. PMI, *A Guide to the Project Management Body of Knowledge*, 2000 ed. (Project Management Institute, Newton Square, PA., 2000).
21. M. Dixon, *APM Project Management Body of Knowledge*, 4th ed. (Association for Project Management, High Wycombe, 2000), p. 68.
22. P. Bourque and R. Dupuis, *A Guide to the Software Engineering Body of Knowledge SWEBOK* (IEEE Computer Society, Los Alamitos, CA, 2001).
23. D. Dalcher, Life cycle design and management, in: *Project Management Pathways: A Practitioner's Guide*, edited by M. Stevens (APM Press, High Wycombe, 2002).
24. B. W. Boehm, T. E. Gray, and T. Seewaldt, Prototyping vs. specifying: a multiproject experiment, *IEEE Transactions on Software Engineering* **SE-10**(3), 290–303 (1984).

25. L. Mathiassen, T. Seewaldt, and J. Stage, Prototyping vs. specifying: principles and practices of a mixed approach, *Scandinavian Journal of Information Systems* **7**(1), 55–72 (1995).
26. H. D. Mills, Top-Down Programming in Large Systems, in: *Debugging techniques in Large Systems*, edited by R. Ruskin (Prentice-Hall, Englewood Cliffs, New Jersey, 1971), p. 41–55.
27. C. Laraman, and V. R. Basili, Iterative and incremental development: A brief history, *IEEE Computer* **36**(6), 47–56 (2003).
28. T. Gilb, *Principles of Software Engineering Management* (Addison Wesley, Wokingham, 1988).
29. A. Alliance, *Agile Manifesto* (The Agile Alliance, 2001).
30. J. Stapleton, *DSDM Dynamic Systems Development Method* (Addison-Wesley, 1997).
31. S. L. Pfleeger, *Software Engineering: Theory and Practice*, 2 ed. (Prentice-Hall, Upper Saddle River, New Jersey, 2001).
32. O. Benediktsson and D. Dalcher, Effort estimation in incremental software development, *IEE Proceedings Software* **150**(6), 251–358 (2003).
33. R. Chillarege, et al., Orthogonal defect classification – a concept for in-process measurements, *IEEE Transactions on software Engineering* **18**(11), 943–956 (1992).
34. M. Butcher, H. Munro, and T. Kratschmer, Improving software testing via ODC: three case studies, *IBM Systems Journal* **41**(1), 31–44 (2002).
35. B. W. Boehm, *Software Engineering Economics* (Prentice Hall, Englewood Cliffs, 1981).
36. A. Macro and J. N. Buxton, *The Craft of Software Engineering* (Addison Wesley, Wokingham, 1987).
37. L. H. Putnam, A general empirical solution to the macro software sizing and estimating problem, *IEEE Transactions on Software Engineering* **SE-4**(4), 345–361 (1978).
38. R. Fairley, *Software Engineering Concepts* (McGraw-Hill, New York, 1985).
39. S. H. Kan, *Metrics and Models in Software Quality Engineering* (Addison-Wesley, Boston, 2003).
40. W. Swartout and R. Balzer, On the inevitable intertwining of specification and implementation, *Communications of the ACM*, pp. 438–440 (1982).

A MODEL FOR A METHOD ADAPTATION PROCESS

Mehmet N. Aydin, Frank Harmsen, Kees van Slooten, and
Robert A. Stegwee*

1. INTRODUCTION

Despite the best endeavours both in the area of Information Systems Research and in practice, effective use of method(logies) in Information Systems Development (ISD) stays as an issue on both communities' agendas. By an effective use of the method we mean considering ISD methods (ISDMs) as a means for problem-solving rather than as a constraint on practitioners' activities in work practices. Empirical findings indicate that information system development (ISD) methods are often adapted to different project contexts since the ways methods prescribe may not accommodate uniqueness of a project situation (Fitzgerald, Russo, and O'Kane, 2003).

In the 1980s and 1990s, rationales behind structured, brand-named conventional ISD methods have been questioned as being IT-oriented, complex, and rigid, especially, in new application domains (Cockburn, 2001). As opposed to conventional methods, Rapid Application Development (RAD) methods appear to be a promising ISD approach that supports work practice and fulfils practitioners' needs (Martin, 1991). One can characterize these methods as agile, business-, human-, process-oriented, and adaptive to different project situation (Cockburn, 2001).

Tailoring a method requires several compromises and trade offs between a tailored method and changes in a project situation. Clearly, project managers need guidance that facilitates their decision-making process leading to a tailored method (Aydin and Harmsen, 2002). In response to such a need, this paper proposes a model. Namely, the proposed model is aimed at facilitating project managers in adapting an ISD method to a project situation. In the next section, we give basic information of existing method adaptation approaches in method engineering in general and in situational method engineering in particular. Afterwards, we introduce our model and elaborate its components in detail. In Section 3 we instantiate the model for a RAD method (DSDM) with an illustrative project

* Mehmet N. Aydin, Kees van Slooten, and Robert A. Stegwee, University of Twente, Department of Business Information Systems, School of Business, Public Administration & Technology, P.O. Box 217 7500 AE Enschede, The Netherlands, m.n.aydin@utwente.nl, cvs@utwente.nl, r.a.stegwee@utwente.nl. Frank Harmsen and Robert A. Stegwee, Cap Gemini, The Netherlands, frank.harmsen@capgemini.com, robert.stegwee@capgemini.com.

situation. The conclusion section emphasizes our follow-up research to be undertaken in the near future.

2. INTRODUCING AN INTEGRATED ACTIVITY MODEL FOR METHOD ADAPTATION

2.1. The Essence of Method Engineering

Method Engineering (Kumar and Welke, 1992; Brinkkemper, 1996; Hidding, 1997) is the discipline to build project-specific information system development methods, called situational or situated methods, from parts of existing methods, called method fragments. Method Engineering includes route map configuration (Slooten and Brinkkemper, 1993), aiming at tuning and extending method fragments to obtain a method. A route map can be configured by considering the situation in which the resulting situated method will be applied and the success one wants to achieve with the situated method (Klooster, Brinkkemper, Harmsen, and Wijers, 1997). This so-called S3 model (Harmsen, Lubbers, and Wijers, 1995) provides heuristics that guide method engineers in constructing and tailoring a method.

An important aspect of Method Engineering is capturing and distributing knowledge and experience of practitioners, in particular project managers, to improve tailored methods and the process to construct tailored methods (Rolland and Prakash, 1996; Jarke, Pohl, Rolland, and Schmitt, 1994). Experience factory as a project-independent unit within an organization has been proposed a way to capture and process project experiences of all kind and to actively support the project by providing suitable experiences from the past projects (Basili, Caldiera, and Rombach, 1994). In that respect a case-based approach is used as a technique to employ accumulated project experiences for tailoring the software process (Althoff, Birk, Greese von Wangenheim, and Tautz, 1998). Eliciting project characteristics, combined with the required project success are identified as essential activities in such a tailoring process (Henninger and Baumgarten, 2001). All these approaches to Method Engineering stress the importance of a sound decision-making process leading to a tailored method. This decision-making process should be facilitated by knowledge support to improve both the efficiency of the process itself and the effectiveness of the resulting situated method.

2.2. Introducing an Integrated Activity Model

Our model, so-called an integrated activity model, is based on the configuration procedure for a scenario (Slooten and Brinkkemper, 1993) and the process of situational method engineering (Harmsen, Brinkkemper, and Oei, 1994).

Figure 1 depicts key activities needed to operationalize a method adaptation process. As we shall explain later on, different ISD levels can be considered while tailoring a method (Zachman, 1987). At the first level, usually called the project preparation level, practitioners start with an intention that reflects business opportunities to be seized or problems to be solved.

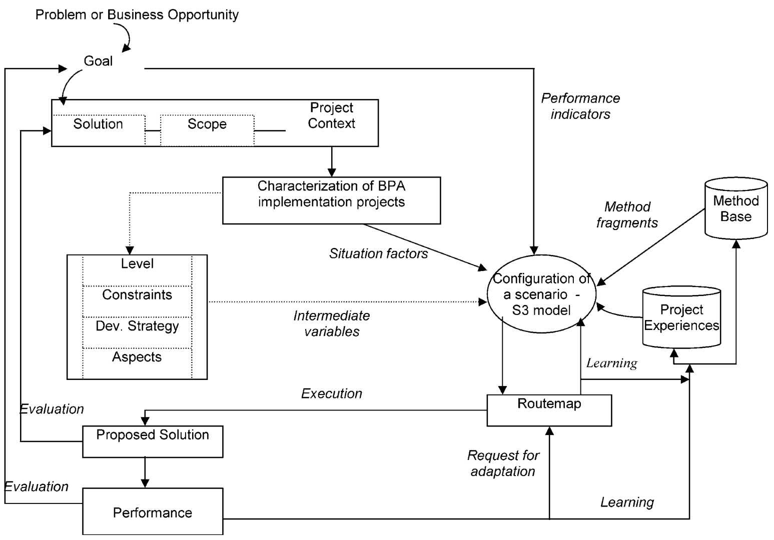


Figure 1. An integrated activity model for a method adaptation process.

Several ideas, options emerge before the actual start of the project and they are refined and formulated in terms of a project goal that includes the solution to be achieved in a relevant scope. Scope may refer to boundaries of a solution in given context. Context refers to the association of a situation and decisions made on it (Jarke et al., 1994). This level may result in a 'go or no go' decision for a project. A 'go' decision is then followed by a preparation for the project kick-off. In our model understanding project context is a key theme in the preparation phase of a project. A main purpose in this activity is to address business aspects of a project, to get into project surroundings and interact with a project environment in which different parties are involved and various expectations exist. As we shall see later on, in our sample RAD method (DSDM) this activity is placed in the pre-project phase. Basically, this activity helps project managers to figure out business values behind the service delivered, uncovering key financial issues, determining various architectures, such as information architecture, process architecture. Notice that this activity might be sufficient enough for a certain degree of clarity of a solution to be delivered. A solution develops along with the interactions between the project organization and the target organization.

So far we have explained a number of activities providing inputs to a scenario configuration activity. The configuration activity plays a central role in the model. With this activity situation factors from the characterization, performance indicators from the goal (Klooster et al., 1997), method fragments from the method base are used to configure a 'goal-oriented situation-specific' route map. In addition to mentioned inputs, intermediate variables mediate the configuration in such a way that the levels of systems development (Figure 1), the constraints derived from situations factors, the development strategy to be applied and aspects of modeling to be used are providing a refinement for dominant situation factors and suitability of method fragments. A final remark in this activity is on the use of project experiences for making better decisions.

In principle, a critical decision-making process at the heart of S3 Model may require sound argumentation and critique of a choice of method fragments (Vahidov and Elrod, 1999). In work practices, such a process incorporate heuristics, best practices, a plenty of common sense, yet practitioners might be pragmatic while selecting appropriate method fragments. Thus, configuration of a project scenario is an evolutionary and learning process. Several feedback loops are aimed at collecting project experiences. These are learning from clarification and improvement of route maps, learning from an execution of route map, evaluation of a realized solution by comparing with an intended solution, evaluation of specified route map execution with respect to predefined performance indicators.

3. ILLUSTRATION OF THE MODEL

3.1. DSDM as a Starting Point for Tailoring

DSDM (Dynamic Systems Development Method, 2000) is a method for RAD, Rapid Application Development (Martin, 1991; Boehm, 1999). In the UK and in the Benelux, DSDM, which is supported by a consortium of over 600 organizations, has become a de-facto market standard. The method highly emphasizes the notions of suitability and adaptability – DSDM is to a certain extent suitable for a project or organization and is adaptable if not completely suitable. For the purpose of this research, we have considered three components of DSDM: its underlying philosophy, its terminology and its essential techniques (Figure 2). In practice, each of these components can be applied separately, and subsets of the components can be applied on their own as well. DSDM framework suggests a complete project approach including key phases, products and roles, which should be customized according to a project situation. In this manner, DSDM is highly adaptable – it is possible to use full-fledged DSDM, but individual techniques or just terminology are still meaningful to be used.

3.1.1. Philosophy of DSDM

The underlying philosophy or way of thinking of DSDM is captured in nine principles (Stapleton, 1997), stressing typical RAD features like iterative development, inte-

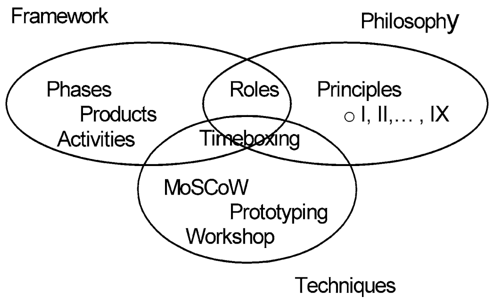


Figure 2. DSDM in a nutshell – three domains encapsulated in DSDM manual (adopted from (Aydin and Harmsen, 2002)).

grated testing, user involvement in the development team and fitness for business purpose (DSDM, 2000).

3.1.2. Terminology of DSDM

DSDM has a lifecycle model of which the core consists of five phases each of which delivers a number of products, of which characteristics and quality criteria are described in the manual (DSDM, 2000). Moreover, roles and team structure are defined as well.

One can see a possible mapping between DSDM phases and typical ISD project levels (Zachman, 1987). Although all levels are important for the model, this paper mainly focuses on scope, partly object systems analysis and design level (OSAD). Slooten and Brinkkemper (Slooten and Brinkkemper, 1993) relate models concerning at the business model level in (Zachman, 1987) to OSAD level and similarly at the design model level to information system analysis and design level (ISAD). More specifically, for the sake of simplicity this paper will instantiate the integrated activity model for the feasibility study phase and the business study phase in DSDM.

3.1.3. Techniques of DSDM

Important techniques of DSDM are timeboxing, facilitated workshops, prioritization and prototyping. Timeboxing refers to setting a deadline by which a predefined objective must be met, instead of describing when a task must be completed. MoSCoW, which is abbreviation for must, should, could have and want to have but won't have this round, is a way to prioritize requirements of the system. Modeling techniques are not included in DSDM, as they often are part of modeling tool sets, which are not part of the method. However, suggestions for modeling techniques to be used are described.

3.1.4. Suitability of DSDM

DSDM clearly states in which situations DSDM is applicable, and in which situations the method should be tailored. For this, an instrument called 'suitability filter' is included in the manual. The filter considers the critical success factors for DSDM and characteristics of projects that are especially effective for DSDM. Each potential project should be judged individually using the filter. If the project provides a good match against the filter, then DSDM can be considered the appropriate development approach. Inability to satisfy all of the criteria results in modification of the method.

3.2. Illustration of the Model by Using DSDM for a Sample Project Context

We presume that a pre-project phase is completed and several tasks including suitability of DSDM for the project, a preliminary route map based on DSDM method fragments are done. An instrument called suitability filter is used to assess appropriateness of DSDM for a project situation at hand. Basically, a number of questions are asked to figure out a degree of matching between the pre-conditions of DSDM to be a method of choice, which manifest themselves in principles of DSDM, and characteristics of the project along with situation factors for which a comprehensive list of factors available in the literature (Hoef, Harmsen, and Wijers, 1995; Slooten and Hodes, 1996; Klooster et al., 1997).

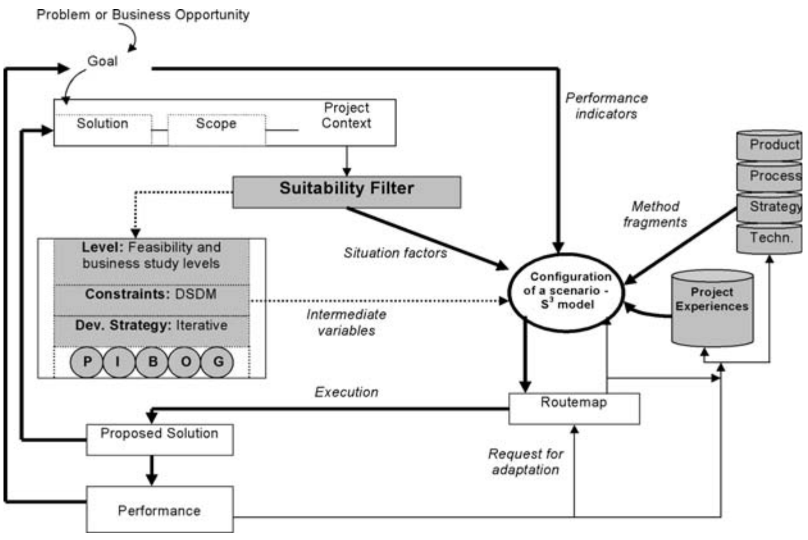


Figure 3. Instantiation of the model with DSDM (Abbreviations- P: process, I: information, B: behaviour, O: organization, G: goal.

Now we will show how to use the suitability filter enacted with the proposed model while tailoring DSDM at the feasibility and the business study level (Figure 3). The suitability risk list consists of a number of questions and for each of them an answer is expected as yes or no, comments for further explanations are encouraged, measures for controlling (prevention and/or correction of negative impacts of situation factors) and working instructions are available.

Basically, the suitability risk list helps project coaches and project managers characterizing the project so that they can make a better decision about which method fragments could be most appropriate and contribute to a successful execution. We identify four method fragment types that are building blocks for a route map. They are as follows.

- Process fragments include all details about activities, roles to be performed to attain desired products. They show how products are going to be delivered
- Since DSDM is in favour of a product-oriented approach, product fragments are essential in a route map. In that case, product fragment describes which products are going to be delivered.
- Strategy fragments indicate in which way(s) and to what extent a method fragment contributes to a certain aspect of situational method engineering.
- Technical fragments are including all job aids, tools, techniques to control and support systems development and project management.

3.2.1. Selection of Product and Process Fragments

In Table 1 and 2 we show key product fragments and roles (process fragments) and a number of indicators describing goals at the feasibility and business study level. A com-

Table 1. Examples of product and process fragments and some key indicators for goals at the Feasibility Level in DSDM

Feasibility Level			
Product fragments		Process fragments [a]	Goal (key indicators)
Main products	Models		
Feasibility Report Feasibility Prototype Outline Plan	Key Business Data Key Business Activities Key People/Users Key Events/Interfaces Business vision, scope, objectives	Visionary Technical Coordinator Executive Sponsor	To indicate scope of the solution and its impact (organizational fit), and to outline the problem to be addressed by the new system. To describe at a high level the business processes and contribute to business process improvement.

[a] We only provide roles-related fragments. See the complete list of fragments in (DSDM, 2000)

Table 2. Examples of product and process fragments and some key indicators for goals at the Feasibility Level in DSDM[a]

Business Study Level			
Product fragments		Process fragments [b]	Goal (key indicators)
Main products	Models		
Business Area Definition Outline Prototyping Plan System Arch. Defn.	Business Functions Data/Relationships/Rules Business Events Business Scenarios Business Architecture System Locations	Visionary Ambassador user(s) Project Manager	To identify business needs that should be supported by the proposed computer system To identify the business processes and business scenarios that need to change System acceptance and fitness for business purpose

[b] We only provide roles-related fragments. See the complete list of fragments in (DSDM, 2000)

plete list of products and process fragments is available in the DSDM manual. Goal related key indicators are also coming from DSDM. To give a flavour of how the goal drives selection of product fragment consider the following example: suppose we choose the first indicator in Table 1, i.e. ‘to indicate the scope of the solution and its impact (organizational fitness) and outline the problem to be addressed by the new system’, then the product fragment, ‘vision/scope/objectives model’ appears to be best suited to realize this goal. In that case ‘executive sponsor and visionary’ (process fragment) might be considered as targeted key roles for the acceptation of the chosen goal.

Now we shall simply illustrate how the modeling aspects can mediate the choice of suitable fragments. First of all, we provide some basic information on these aspects. Five modeling aspects can be distinguished (Slooten and Brinkkemper, 1993): Process (P) aspect refers to methods to model business process, activities and functions; Information (I) aspect refers to methods to model business information analysing message and data flows; Behaviour (B) aspect refers to methods to model business events and time; Organi-

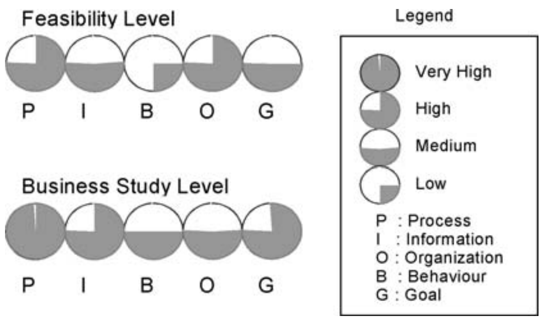


Figure 4. Visualization of the degree of importance of modeling aspects for the two levels in DSDM.

zation (O) aspect refers to methods to model structural and cultural aspects of the organization; and finally Goal (G) aspect refers to methods for articulating and solving the problem to reach certain organizational goals.

In Figure 4 we show a perceived degree of importance of each modeling aspect at two levels in terms of 'very high', 'high', 'medium' and 'low' (DSDM, 2000). Now, lets consider the first three indicators in Table 2 as the targeted indicators for the goal to be achieved, then, according to DSDM, we may notice that the process and the organization aspects of systems modeling could be more important than others. And, these two aspects will mediate the selection of process and product fragments, i.e. we will choose those fragments that can help us capturing such modeling activities. In summary, a degree of importance of each modeling aspect helps us better understanding on what extent a method fragment contributes to certain aspect of the situated method.

Now, we shall show how the suitability filter co-operates with the modeling aspects, the goal indicators and selection of method fragments. To do this we will use an illustrative project situation. For the sake of simplicity we will limit ourselves to two situational factors: clarity and stability.

Clarity refers to what extent the goals, needs, and desires of users are clear and coherent enabling a stable specification of functional requirements. Stability indicates to what an extent goals, needs, and desires of users will not change over time enabling a stable specification of functional requirements. Meanwhile, we assume that 'fitness for business purpose' and 'system acceptance' are considered as being two key goal indicators at the business study level (Table 1). At this stage project managers need to know which method fragments should be selected so that tailoring DSDM (selecting required method fragments) to a given situation and a given goal could be done appropriately. In our example we know that a type and a degree of user involvement is one of dominant strategy fragments that contributes to an achievement of predefined goal indicators (see Slooten and Hodes, 1996 for this heuristic). There is still a set of options for this strategy fragment. For instance, for the types of user involvement we have the following options provided by DSDM: ambassador role who is representative of the entire community of users, visionary role who has the high level view of the solution, and advisor role. The next step will be the use of modeling aspects for product fragments. So, since we focus on 'fitness for business purpose' and 'system acceptance'-related goal indicators we might more focus on process and goal modeling aspects at the business study level. This tells us that the 'business area definition' is the key product to achieve predefined goal indicators in a given situation.

4. CONCLUSIONS

Our proposed model is aimed at supporting practitioners for a method adaptation process. We should note that the proposed model employs key aspects of method engineering and consider the engineering perspective as a reference domain. However, method adaptation can also be elaborated from the viewpoint of what we call, the socio-organizational (Baskerville and Stage, 2001). This perspective appears to be concerned with 'soft and ill-structured issues' like culture and politics. Baskerville and Stage suggests method adaptation process as a 'complex socio-organizational phenomenon'. The meaning of the method fragment in (Baskerville and Stage, 2001) is different from its meaning in method engineering; it is not prescribed or coherent part of the method, it is rather invented on-the-fly, emerging in the work setting. We believe that two perspectives on method adaptation, i.e. the engineering perspective and socio-organizational perspective, are complementary rather than conflicting. Indeed, the engineering perspective, that we use in this paper, seems be to be useful for 'well-structured' method fragments for which we give some examples from the sample method (DSDM) and the socio-organizational perspective appears to provide handy mechanism to understand 'unstructured' fragments that deal with ill-structured issues. So, if we now consider our work from the engineering perspective, the proposed model seems to be useful for adapting ISD method. In this paper, for the sake of simplicity we rather use simple examples. One can argue that illustrative examples for the activities and inputs for S3 are not very challenging, yet we use these examples on the purpose of demonstrating how the model works. It is worth noting that this model extends, integrates and explicates two approaches to a method adaptation process proposed in method engineering. As we mentioned before the model might employ key aspects of the socio-organizational perspective as well and needs to be studied in work practice. Actually, we are currently working on this issue and the preliminary findings appear to support our model on the condition that the model should also accommodate unstructured method fragments. Indeed, this is the follow-up research issue that we are working on and aiming at testing our models in work practice.

REFERENCES

Aydin, M. N., and Harmsen, F., 2002, Making a method work for a project situation in the context of CMM, in: *Product Focused Software Process Improvement*, M. Oivo and S. Komi-Sirviö, eds., Springer LNCS 2559, pp. 158–171.

Althoff, K.-D., Birk, A., Greese von Wangenheim, C., and Tautz, C., 1998, Case-based reasoning for experimental software engineering, in: *Case-Based Reasoning Technology*, M. Lenz et al., eds., Springer Verlag, pp. 235–254.

Basili, V. R., Caldiera, G., and Rombach, H. D., 1994, Experience factory, in: *Encyclopedia of Software Engineering,* J. J. Marciniak, ed., John Wiley & Sons, pp. 469–476.

Baskerville, R., and Stage, J., 2001, Accommodating emergent work practices: Ethnographic choice of method fragments, in: *Realigning Research and Practice in IS Development: The Social and Organizational Perspective*, B. Fitzgerald, N. Russo, and J. DeGross, eds., Kluwer, New York, pp. 12–28.

Boehm, B. W., 1999, Making RAD work for your project, *IEEE Computer* **32**(3):113–114.

Brinkkemper, S., 1996, Method engineering: Engineering of information systems development methods and tools, *Information and Software Technology* **38**:275–280.

Cockburn, A., 2002, *Agile Software Development*, Reading Addison Wesley Longman, Massachusetts.

DSDM, 2000, Dynamic Systems Development Method Manual (June 1, 2000); http://www.dsdm.org.

Fitzgerald, B., Russo, N. L., and O'Kane, T., 2003, Software development method tailoring at Motorola, *Communications of the ACM* **46**(4):64–70.

Harmsen, F., Brinkkemper, S., and Oei, H., 1994, Situational method engineering for information systems project, in: *Methods and Associated Tools for the information Systems Life Cycle,* T. W. Olle and A. A. Verrijn Stuarts, eds., North-Holland, Amsterdam, pp. 169–194.

Harmsen, F., Lubbers, I., and Wijers, G., 1995, Success-driven selection of fragments for situational methods: The S^3 model, in: *Proceedings of the Second International Workshop on Requirements Engineering Requirements Engineering: Foundations of Software Quality*, K. Pohl and P. Peters, eds., Aachener Beitrage zur Informatik, Band 13, pp. 104–115.

Henninger, S., and Baumgarten, K., 2001, A Case-based approach to tailoring software processes, in: *Proceedings of the 4th International Conference on Case-Based Reasoning*, D. W. Aha, I. Watson, and Q. Yang, eds., Springer, Lecture Notes in Artificial Intelligence, Canada, pp. 249–262.

Hidding, G. J., 1997, Reinventing methodology, *Communications of the ACM* **40**(11):102–109.

Hoef, R. van de, Harmsen, F., and Wijers, G., 1995, Situation, scenario, and success, *Memoranda Informatica* 95–12, University of Twente, Enschede.

Jarke, M., Pohl, K., Rolland, C., and Schmitt, J. R., 1994, Experience-based method evaluation and improvement: A process modeling approach, in: *Methods and Associated Tools for the information Systems Life Cycle*, T. W. Olle and A. A. Verrijn Stuarts, eds., North-Holland, Amsterdam, pp. 169–194.

Klooster, M., Brinkkemper, S., Harmsen, F., and Wijers, G., 1997, Intranet facilitated knowledge management: A Theory and tool for defining situational methods, in: *Proceedings of the 9th International Conference CAiSE'97*, A. Olivé and J. A. Pastor, eds., Springer Verlag LNCS 1250, pp. 303–317.

Kumar, K., and Welke, R. J., 1992, Methodology engineering: A Proposal for situation-specific methodology construction, in: *Challenges and Strategies for Research in Systems Development*, W. W. Cotterman and J. A. Senn, eds., John Wiley & Sons, Chichester, pp. 258–269.

Martin, J., 1991, *Rapid Application Development*, Macmillan Publishing, New York.

Rolland, C., and Prakash, N., 1996, A proposal for context-specific method engineering, in: *Method Engineering: Principles of Method Construction and Tool Support*, S. Brinkkemper, K. Lyytinen, and R. J. Welke, eds., Chapman & Hall, Atlanta, pp. 191–208.

Stapleton, J., 1997, *Dynamic Systems Development Method – The Method in Practice*, Addison-Wesley.

Vahidov, R., and Elrod, R., 1999, Incorporating critique and argumentation in DSS, *Decision Suport Systems* **26**(3):249–258.

Van Slooten, C., and Brinkkemper, S., 1993, A Method engineering approach to information systems development, in: *Information System Development Process*, N. C. Prakash, C. Rolland, and B. Pernici, eds., Elsevier Science Publishers B.V., North Holland, pp. 167–186.

Van Slooten, C., and Hodes, B., 1996, Characterizing IS development projects, in: *Method Engineering: Principles of Method Construction and Tool Support*, S. Brinkkemper, K. Lyytinen, and R. J. Welke, eds., Chapman & Hall, Atlanta, pp. 29–44.

Zachman, J. A., 1987, A Framework for information architecture, *IBM Systems Journal* **26**(3):276–292.

ON THE USE OF INFORMATION SYSTEMS RESEARCH METHODS IN DATA MINING

Mykola Pechenizkiy, Seppo Puuronen, and Alexey Tsymbal*

1. INTRODUCTION

Information systems are powerful instruments for organizational problem solving through formal information processing (Lyytinen, 1987). Data mining (DM) and knowledge discovery are intelligent tools that help to accumulate and process data and make use of it (Fayyad, 1996). Data mining bridges many technical areas, including databases, statistics, machine learning, and human-computer interaction. The set of data mining processes used to extract and verify patterns in data is the core of the knowledge discovery process. Numerous data mining techniques have recently been developed to extract knowledge from large databases.

The area of data mining is historically more related to AI (Artificial Intelligence), pattern recognition, statistical, and database communities, though we think there is no objective reason for that. And nowadays, although the field of data mining according to the ACM classification system[†] for the computing field is a subject of database applications (H.2.8) that in sequence related to database management (H.2) and to information systems field (H.), there exists a gap between the data mining and information systems communities. Each of the two scientific communities publishes its own journals and books, and organizes different conferences that rarely cover the same issues. This situation is not beneficial since both communities share in common many similar problems being solved and therefore are potentially helpful for each other.

In this paper (in Section 2) we consider some existing frameworks for data mining, including database perspective and inductive databases approach, the reductionist statistical and probabilistic approaches, data compression approach, and constructive induction approach. We consider their advantages and limitations analyzing what these approaches account in the data mining research and what they do not.

* Mykola Pechenizkiy and Seppo Puuronen, Department of Computer Science and Information Systems, University of Jyväskylä, Jyväskylä, Finland. Alexey Tsymbal, Department of Computer Science, Trinity College Dublin, Dublin, Ireland.

[†] See http://www.acm.org/class/1998/ccs98.html.

The study of research methods in information systems by Järvinen (1999) encouraged us to analyse connections and appropriateness of them to the area of data mining. In Section 3 we are trying to view the data mining research as a continuous information system development process. We refer to the traditional framework presented by Ives et al. (1980) that is widely known and has been used in the classification of Information Systems research literature. The framework is a synthesis of many other frameworks considered before by other researchers and covers their main elements. For us this framework is more substantial than the others since it also focuses on the development of information systems.

Ives et al. (1980) considers an information system (IS) in an organizational environment that is further surrounded by an external environment. According to the framework an information system itself includes three environments: user environment, IS development environment, and IS operations environment. Drawing an analogy to this framework we consider a data mining system as a special kind of adaptive information system that processes data and helps to make use of it. Adaptation in this context is important because of the fact that the data mining system is often aimed to produce solutions to various real-world problems, and not to a single problem. On the one hand, a data mining system is equipped with a number of techniques to be applied for a problem at hand. From the other hand there exist a number of different problems and current research has shown that no single technique can dominate some other technique over all possible data-mining problems (Wolpert and MacReady, 1996). Nevertheless, many empirical studies report that a technique or a group of techniques can perform significantly better than any other technique on a certain data-mining problem or a group of problems (Kiang, 2003). Therefore viewing data mining research as a continuous and never-ending development process of a DM system towards the efficient utilization of available DM techniques for solving a current problem impacted by the dynamically changing environment is a well-motivated position.

In this paper we focus on the IS development process. We consider information systems development framework of Nunamaker (1990–91) adapted to data-mining systems development. We discuss three basic groups of information systems research methods. Namely, we consider theoretical, constructive and experimental approaches with regard to Nunamaker's framework in the context of data mining. We demonstrate how these approaches can be applied iteratively and/or in parallel for the development of an artefact – a data-mining tool, and contribute to theory creation and theory testing. We conclude with a brief summary and discussion of our further research in Section 4.

2. THEORETICAL FRAMEWORKS FOR DATA MINING

2.1. A Database Perspective and Inductive Databases

A database perspective on data mining and knowledge discovery was introduced in Imielinski and Mannila (1996). The main postulate of their approach is: "there is no such thing as discovery, it is all in the power of the query language". That is, one can benefit from viewing common data mining tasks not as dynamic operations constructing new pieces of information, but as operations finding unknown (i.e. not found so far) but existing parts of knowledge.

In Boulicaut et al. (1999) an inductive databases framework for the data mining and knowledge discovery in databases (KDD) modeling was introduced. The basic idea here is that data-mining task can be formulated as locating interesting sentences from a given logic that are true in the database. Then discovering knowledge from data can be viewed as querying the set of interesting sentences. Therefore the term "an inductive database" refers to such a type of databases that contains not only the data but a theory about the data as well (Boulicaut et al., 1999).

This approach has some logical connection to the idea of deductive databases, which contain normal database content and additionally a set of rules for deriving new facts from the facts already present in the database. This is a common inner data representation. For a database user, all the facts derivable from the rules are presented, as they would have been actually stored there. In a similar way, there is no need to have all the rules that are true about the data stored in an inductive database. However, a user may imagine that all these rules are there, although in reality, the rules are constructed on demand. The description of an inductive database consists of a normal relational database structure with an additional structure for performing generalizations. It is possible to design a query language that works on inductive databases (Boulicaut et al., 1998). Usually, the result of a query on an inductive database is an inductive database as well. Certainly, there might be a need to find a solution about what should be presented to a user and when to stop the recursive rule generation while querying. We refer an interested reader to the work of Boulicaut et al. (1999).

2.2. The Reductionist Approach

In Mannila (2000) two simple approaches to the theory of data mining are analysed. The first one is the reductionist approach of viewing data mining as statistics. Generally, it is possible to consider the task of data mining from the statistical point of view, emphasizing the fact that DM techniques are applied to larger datasets than it is in statistics. And in this situation the analysis of the appropriate statistics literature, where strong analytical background is accumulated, would solve most of the data mining problems. Many data mining tasks naturally may be formulated in statistical terms, and many statistical contributions may be used in data mining in a quite straightforward manner. The second approach discussed by Mannila (2000) is a probabilistic approach. Generally, many data mining tasks can be seen as the task of finding the underlying joint distribution of the variables in the data. Good examples of this approach would be Bayesian network or a hierarchical Bayesian model, which give a short and understandable representation of the joint distribution. Data mining tasks dealing with clustering and/or classification fit easily into this approach. However, it should be admitted that data mining researchers with computer science background typically have rather little education in statistics and this is a reason to the fact that achievements from statistics are used not to such an extent as could be possible.

A deeper consideration of data mining and statistics shows that the volume of the data being analysed and different background of researchers are, probably, not the most important ones that make the difference between the areas. Data mining is an applied area of science and limitations in available computational resources is a big issue when applying results from statistics to data mining. The other important issue is that data mining approaches emphasize database integration, simplicity of use, and understandability of results. Last but not least Mannila (2000) points out that the theoretical framework of statistics does not concern much about data analysis as a process that generally includes data understanding, data preparation, data exploration, results evaluation, and visualisation steps. However, there are persons (mainly with strong statistical background) who equate DM to applied statistics, because many tasks of DM may be perfectly represented in terms of statistics.

2.3. Data Compression Approach

A data compression approach to data mining can be stated in the following way: compress the dataset by finding some structure or knowledge for it, where knowledge is interpreted as a representation that allows coding the data by using fewer amount of bits. For example, the minimum description length (MDL) principle (Mehta et al., 1995) can be used to select among different encodings accounting to both the complexity of a model and its predictive accuracy.

Machine learning practitioners have used the MDL principle in different interpretations to recommend that even when a hypothesis is not the most empirically successful among those available, it may be the one to be chosen if it is simple enough. The idea is in trading between consistency with training examples and empirical adequacy by predictive success as it is, for example, with accurate decision tree construction. Bensusan (2000) connects this to another methodological issue, namely that theories should not be ad hoc that is they should not overfit the examples used to build them. Simplicity is the remedy for being ad hoc both in the recommendations of philosophy of science and in the practice of machine learning.

The data compression approach has also connection with the rather old Occam's razor principle that was introduced in 14th century. The most commonly used formulation of this principle in data mining is "when you have two competing models which make exactly the same predictions, the one that is simpler is the better".

Many (if not every) data mining techniques can be viewed in terms of the data compression approach. For example, association rules and pruned decision trees can be viewed as ways of providing compression of parts of the data. Clustering approaches can also be considered as a way of compressing the dataset. There is a connection to Bayesian theory for modelling the joint distribution – any compression scheme can be viewed as providing a distribution on the set of possible instances of the data.

However, in order to produce a structure that would be comprehensible to the user, it is necessary to select such compression method(s) that is (are) based on concepts that are easy to understand.

2.4. Constructive Induction Approach

Constructive induction is a learning process that consists of two intertwined phases, one of which is responsible for the construction of the "best" representation space and the second concerns with generating hypothesis in the found space (Michalski and Wnek, 1993). Constructive induction methods are classified into three categories: data-driven (information from the training examples is used), hypothesis-driven (information from the analysis of the form of intermediate hypothesis is used) and knowledge-driven (domain knowledge provided by experts is used) methods. Any kind of induction strategy (implying induction, abduction, analogies and other forms of non-truth preserving and non-monotonic inferences) can be potentially used. However, the focus usually is on operating higher-level data-concepts and theoretical terms rather than pure data. Michalski (1997) considers constructive (expands the representation space by attribute generation) and destructive (contract the representational space by feature selection or feature abstraction) operators that can be applied to produce a better representation space comparing to the original one. In Bensusan (1999) it was shown that too many theoretical terms could impair induction. This vindicates an old advise of the philosophy of science: avoid adding unnecessary metaphysical baggage to a theory. Theoretical terms are often contrasted with observational terms. It is generally accepted that the more data we have the better model we can construct. However, this is not true for higher-level concepts that constitute a theory.

Many data mining techniques that apply wrapper/filter approaches to combine feature selection, feature extraction or feature construction processes (as means of dimensionality reduction and/or as means of search for better representation of the problem) and a classifier or other type of learning algorithm can be considered as constructive induction approaches.

2.5. Conclusion on Considered Frameworks

The reductionist approach of viewing data mining in terms of statistics has advantages of the strong theoretical background and easy-formulated problems. The data compression and constructive induction approaches have relatively strong analytical background, as well as connections to the philosophy of science. In addition to the just-mentioned frameworks an interesting solution is proposed in the microeconomic view on data mining, introduced by Kleinberg (1998), where a utility function is constructed and trying to be maximized. The data mining tasks concerning processes like clustering, regression, and classification fit easily into these approaches.

The inductive databases framework suggests architecture for data mining systems and allow to view data mining as a process. Association rules and other simple pattern formalisms can be described by this approach. However, for example, clustering is harder to describe in a useful way within the inductive databases framework.

In one way or another, we can easily see the exploratory nature of the frameworks for the data-mining field. Different frameworks account different data mining tasks, allow preserving and presenting background knowledge. However, what seems to be lacking in most of the approaches, are the ways for taking the iterative and interactive nature of the data mining process into account (Mannila, 2000). Furthermore, none of the considered frameworks considers data mining in the context of an adaptive system that processes information.

In the next section we introduce an information systems development framework and then consider how data mining can be seen as an iterative and interactive development process within this framework.

3. DATA MINING AND INFORMATION SYSTEMS FRAMEWORK

3.1. Generations of DM Systems

Present history of data mining systems' development totals three main stages/generations (Piatetsky-Shapiro, 2000). Year 1989 can be referred to as the first generation of data mining/KDD systems when a few single-task data mining tools such as C4.5 decision tree algorithm (Quinlan, 1993) existed. They were difficult to use and required significant preparation. Most of such systems were based on a loosely-coupled architecture, where the database and the data mining subsystems were realised as separate independent parts. This architecture demands continuous context switching between the data-mining engine and the database (Imielinski and Mannila, 1996).

Then, the year 1995 can be associated with formation of the second-generation tools-suits. Data mining as a core part of KDD started to be seen as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" (Fayyad, 1996, 22). Some examples of the knowledge discovery systems that follow Fayyad's view on DM as the process are: SPSS Clementine, SGI Mineset (Brunk et al., 1997), and IBM Intelligent Miner (Tkach, 1998).

Numerous KDD systems have recently been developed. At the beginning of the millennium there exist about 200 tools that could perform several tasks (clustering, classification, visualization) each for specialized applications (therefore often called "vertical solutions") (Piatetsky-Shapiro, 2000). This growing trend towards integrating data-mining tools with specialized applications has been associated with the third generation of DM systems (Fayyad and Uthurusamy, 2002).

Because of increasing number of such "vertical solutions" and possibility to accumulate knowledge from these solutions, there is a growing potential for appearance of next-generation database mining systems to manage KDD applications. These systems should be able to discover knowledge by selecting and combining several available most suitable for specific domain KDD techniques. While today's algorithms tend to be fully automatic and therefore fail to allow guidance from knowledgeable users at the key stages in the search for data regularities, the researchers and the developers, who are involved into the creation of the next generation data mining tools, are motivated to provide a broader range of automated steps in the data mining process and make this process more mixed-initiative, in which human experts collaborate more closely with the computer to form hypotheses and test them against the data (Ankerst, 2002).

Since a data mining system is often aimed to produce solutions not to a single problem but rather to various real-world problems, it has to be armed with a number of techniques to be applied for a problem at hand. However, current research has shown that no single technique can dominate some other technique over all possible data-mining problems (Wolpert and MacReady, 1996). Nevertheless, many empirical studies report that a technique or a group of techniques can perform significantly better than any other technique on a certain data-mining problem or a group of problems (Kiang, 2003). Therefore a good data mining system should be adaptive for solving a current problem impacted by the dynamically changing environment and being continuously developed towards the efficient utilization of available DM techniques.

3.2. Information Systems Framework

The traditional framework presented by Ives et al. (1980) is widely known and has been used in the classification of IS research literature. We consider this framework because:

(1) it is a synthesis of many other frameworks considered before by other researchers and covers their main elements;
(2) it is helpful in drawing the analogy between the information systems and data mining systems as a special kind of adaptive information system that processes data and helps to make use of it;
(3) for us this framework is more substantial that the others since it also focuses on the development of information systems as we focus on the development of data mining systems.

Ives et al. (1980) considers an information system in an organizational environment that is further surrounded by an external environment. According to the framework an information system itself includes three environments: user environment, IS development environment, and IS operations environment. There are accordingly three processes through which an IS has interaction with its environments: the use process, the development process, and the operation process.

Analogously, a data-mining system that is equipped with a collection of DM techniques and knowledge how to utilize those for various tasks can be considered as a system with user environment, DM development environment, and DM operations environment. However, in this paper, we focus on the development process of an artefact for data mining and leave operation and use processes for further research.

In the information systems research a variety of research methods have been applied. Davis (2000, 80) expresses this saying that "the field has a richer set of views than other fields because the positivist philosophy that dominated the American research and the phenomenology philosophy that tended to dominate in Europe were both supported by the worldwide community". Even when there are still discussions going on about suitable research methods in the field, we share with many others the opinion that there is room for many research methods, both hard and soft.

Iivari et al. (1998) relate development process to the constructive type of research because of their philosophical belief that development always involves creation of some new artefacts – conceptual (models, frameworks) or more technical artefacts (software implementations). The research approach is classified as constructive where scientific knowledge is used to produce either useful systems or methods, including development of prototypes and processes. Iivari et al. (1998) argue the importance of constructive research especially for applied disciplines of information systems and computer science, and DM may be considered as such a discipline.

Nunamaker et al. (1990–91, 94) consider system development as a central part of a multi-methodological information systems research cycle (Figure 1). Theory building involves discovery of new knowledge in the field of study, however it is rarely contributing directly to practice. Nevertheless, the built theory often (if not always) needs to be tested in the real world to show its validity, recognize limitations and make refinements according

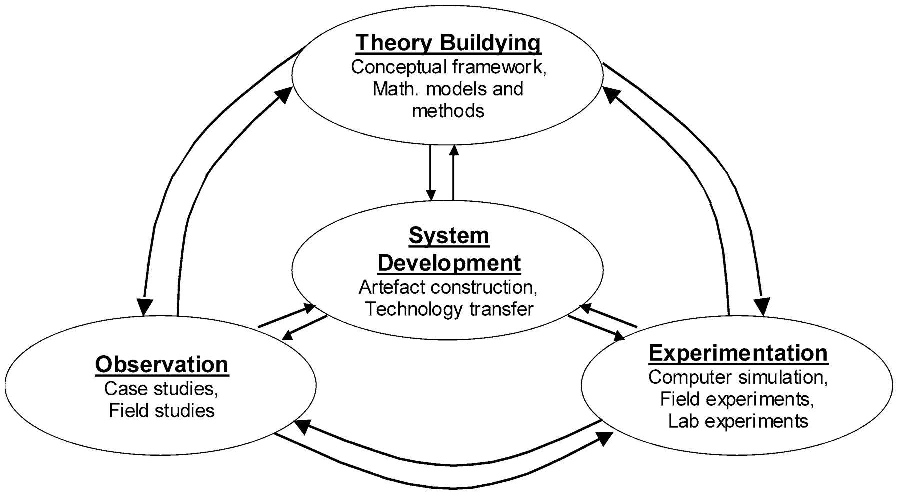

Theory Buildying
Conceptual framework, Math. models and methods
System Development
Artefact construction, Technology transfer
Observation
Case studies, Field studies
Experimentation
Computer simulation, Field experiments, Lab experiments

Figure 1. A multimethodological approach to the construction of an artefact for data mining (adapted from Nunamaker et al., 1990–91, 94).

to observations made during its application. Therefore research methods are subdivided into basic and applied research, as naturally both are common for any large project (Nunamaker et al., 1990–91). A proposed theory leads to the development of a prototype system in order to illustrate the theoretical framework from the one hand, and to test it through experimentation and observation with subsequent refinement of the theory and the prototype in an iterative manner. Such a view presents the framework of IS as a complete, comprehensive and dynamic research process. It allows multiple perspectives and flexible choices of methods to be applied during different stages of the research process.

In the following subsections we consider applying information systems research methods in the context of the data-mining field. we consider theoretical, constructive and experimental approaches with regard to Nunamaker's framework in the context of data mining. We demonstrate how these approaches can be applied iteratively and/or in parallel for the development of an artefact – a data-mining tool, and contribute to theory creation and theory testing.

Particularly, in the next section we consider the construction of an artefact for data mining as system development applying multi-methodological information systems research cycle presented in this section.

3.3. Construction of an Artefact for Data Mining

Can we build an artefact that would be useful? If a research question deals with the verbs like introduce, improve, maintain, cease, extend, correct, adjust, enhance an so on, the study according to Järvinen (1999, 59) likely belongs to the constructive research. Indeed these are the actions that researchers in the area of data mining perform when developing new theories and their applications.

From the data mining research point of view the constructive approach can be seen to help to manipulate and coordinate integrative work (selection and combination) of different data mining techniques, and to conduct the experimental approach. However, in this paper we emphasize the goal of a data mining artefact construction as the major one.

Development of an artefact for data mining can be described in terms of initial and target/final states and the building process itself that includes specification and implementation stages (and usually it is difficult to see if these stages are performed sequentially, iteratively or in parallel. The building process aims to minimize the difference between the target and final states. In our situation the initial state may be described in terms of existing (available in the system) sets of different data mining techniques, e.g. certain clustering, feature transformation and classification techniques. And the target state would be a system that has a possibility adaptively select/construct the most appropriate approach/solution for a given task according to the specificity of this given task.

It is obvious that in order to construct a good artefact with such adaptivity we need some background knowledge about the artefact's components (that is basic data mining techniques) and their appropriateness for certain dataset characteristics. Beside this we need also some background knowledge about the artefact's external environment that are different real-world problems, often called just datasets.

In data mining a dataset is usually characterised by analysis of its domain, statistical, information-theoretical properties and simple measures like the number of instances and attributes. And DM techniques are commonly specified with their requirements, capabilities and/or limitations. Examples of such characteristics are algorithm run-time parameters, ability of handling misclassification costs, and data types supported. Beside specifications, DM techniques may be characterised by many representational and functional characteristics associated either with the expert knowledge about the techniques or with the past learning experience of a corresponding DM technique. These various characteristics include attribute types supported, bias/variance profile, incrementality, cost handling support; efficiency characteristics: training and execution time, training and execution space; resilience characteristics: scalability, tolerance to missing values, tolerance to noise and irrelevant and redundant attributes; and finally practicality characteristics: runtime parameter handling, interpretability, and transparency (Hilario and Kalousis, 1999).

Thus, it is natural that the theory-creating research has to be performed during which the basics of available data mining techniques should be elaborated. For this purpose a literature survey and review commonly are undertaken. This helps to understand the background of the problem and analyse previous findings in the area. However, such theory-creating research can be supported also by meta-learning approaches that in (semi)-automatic way may help to state and check different hypothesis about the relations between technique's and dataset's characteristics. An overview of different meta-learning approaches can be found, for example, in Hilario and Kalousis (2000).

From the theory development point of view there are possibilities to apply either inductive or deductive approaches, and actually it is reasonable to try their combination in the sense that it is possible to use the findings from both approaches in order to check their consistency and guarantee more sophisticated completeness. Inductive theory building is based on search for trends, generalizations from experiments, whereas deductive approaches are based on logical inference on a set of axioms/hypothesizes.

It should be noticed that in some cases it is not possible just to adjust an existing program for someone's research purposes and program design and implementation are required. However it is reasonable to use existing libraries and appropriate tools when possible. In this situation it might be possible to use tested and validated tools as a core/backbone for a new tool and the development process can be focused on the new part of the desired tool.

There are two alternatives to create a tool: to develop it in whole or to develop one component of the tool after another. The second alternative has the following advantages: each component can be designed, implemented, tested, and refined independently before it is included into the meta-approach. The control over the individual components can be organized and the experiments can be easily performed on separate components also.

Evaluation process is an essential part of constructive research. Usually, experimental approach is used to evaluate a data mining artefact. We consider the experimental approach in the next subsection. We will try to show that the experimental approach, however, can be beneficial for theory testing and can be a means of constructing new pieces of knowledge and thus contributing to the theory-creating process.

3.4. Experimental Approach: Theory Testing and Artefact Evaluation

By the evaluation of artefact we understand first of all (1) the evaluation of learned models and meta-level models and (2) testing the constructed theory of different data processing and machine-learning techniques selection and combination.

As from the theory evaluation as from the artefact evaluation point of view, the general principle of evaluation – the new derivation or construct must be better that its best challenger – is applicable for data mining as well. 'Goodness' criterion of a built theory or an artefact is multidimensional and sometimes is difficult to be defined because of mutual dependence between the compromising variables. However, it is more or less easy to construct a criterion based on such estimates as accuracy (including sensitivity and specificity, and various costs matrices) of a built model and its performance (time and memory resources). On the other hand – it is more difficult or even impossible to include into a criterion such important aspects as interpretability of the artefact's output because such kinds of estimate usually are subjective and can be evaluated only by the end-users of a system.

Experimental study can be done in the 'field' or in the 'laboratory'. In the first case different approaches are tested on so-called real-world datasets with real users. In the second case systematically controlled experiments can be organized. Controlled experiments sometimes might produce more beneficial results for theory creating, since unlike real world datasets, synthetically generated data allow to test exactly the desired number of characteristics while keeping all the others unchanged.

Theory testing might be seen here at different levels. A low-level task is to evaluate how well a built model works. The other task is to analyse how the built model performs comparing to the other models. Then it is usually necessary to compare the algorithm selected to build the models with other algorithm(s). Finally, when 'laboratory' experiments and evaluation are finished (that are experiments on synthetic datasets in our situation), it is necessary to go to the field and organize 'field' experiments (that would be experiments on real-world or benchmark datasets).

When testing and validating a model, data miners use several techniques. They include sampling, validation, cross-validation, stratification, Monte Carlo methods, division a dataset into training, validating and testing sets etc. There are two of the most essential elements of any experimental design, namely randomization and experimental control (of all nuisance variables or it is better to say possibility to control adjustable variables and restrictions of known factors).

The evaluation of a selected approach can be provided either based on the filter paradigm, when evaluation process is independent from a learning algorithm and the most appropriate approach is chosen from available ones according to certain data characteristics before the algorithm starts, or based on the wrapper paradigm (Kohavi, 1998) that assumes interaction between the approach selection process and the performance of the integrative model. In order to compare the two approaches Student's t-test and McNemar's test are used as standard de facto (Dietterich, 1998).

However, the experimental approach benefits not only for the artefact evaluation and theory testing that has been used for artefact construction but also it can contribute to the knowledge by producing new pieces of theory about selection and/or combination of DM techniques for a given dataset. Meta-learning approaches is one good example of such attempts to contribute to new pieces of theory induction.

In conclusion we would like to notice that it is reasonable to consider how the results achieved through different research approaches relate to each other and search for contradictions in the results. It can be expected that such joint use of these approaches will give a better understanding of the introduced research goal and benefit in a more significant and sophisticated contribution to the knowledge in the area.

4. CONCLUSION

In this paper we considered several frameworks for data mining. These frameworks are based on different approaches, including inductive databases approach, the reductionist statistical approaches, data compression approach, constructive induction approach and some others. We considered advantages and limitations of these frameworks. We presented the view on data mining research as continuous and never-ending development process of an adaptive DM system towards the efficient utilization of available DM techniques for solving a current problem impacted by the dynamically changing environment. We discussed one of the traditional information systems frameworks and, drawing the analogy to this framework, we considered a data mining system as the special kind of adaptive information system. We adapted the information systems development framework for the context of data-mining systems development. Three basic groups of information systems research methods, applicable for data mining research were discussed, including theoretical, constructive, and experimental approaches. We demonstrated how these approaches could be applied iteratively for the development of a data-mining system. The theoretical backgrounds need to be exploited during the constructive work and the constructed artefact can be used for experimentation. The results of constructive and experimental work can be used to refine theory. Thus, all the research approaches are heavily connected to each other.

In this paper we considered the development process of a data mining system and the constructive research as main means that needs to be supported by the theory-testing research. Further analysis of IS framework briefly presented in subsection 3.2 can be beneficial in the context of the data mining artefact construction and use.

Beside traditional IS framework considered for data mining, adaptation of a knowledge management framework and knowledge engineering perspective towards data mining framework construction is the other direction of our further research.

ACKNOWLEDGMENTS

This research is partly supported by the COMAS Graduate School of the University of Jyväskylä, Finland and by the Science Foundation Ireland under Grant No. S.F.I.-02IN.1I111.

REFERENCES

Ankerst, M., 2002, Report on the SIGKDD-2002 panel the perfect data mining tool: Interactive or automated, *SIGKDD Explorations* **4**(2), 110–111.

Bensusan, H., 1999, Automatic bias learning: An inquiry into the inductive basis of induction, PhD thesis, School of Cognitive and Computing Sciences, University of Sussex.

Boulicaut, J., Klemettinen, M., and Mannila, H., 1998, Querying inductive databases: A case study on the MINE RULE operator, in: *Proceedings of 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*, France, pp. 194–202.

Boulicaut, J., Klemettinen, M., and Mannila, H., 1999, Modeling KDD processes within the inductive database framework, in: *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery*, Springer-Verlag, London, UK, pp. 293–302.

Brunk, C., Kelly, J., and Kohavi, R., 1997, MineSet: An integrated system for data mining, in: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, D. Heckerman, H. Mannila, and D. Pregibon, eds., AAAI Press, California, pp. 135–138.

Davis, G., 2000, Information systems conceptual foundations: Looking backward and forward, in: *Organizational and Social Perspectives on IT*, R. Baskeville, J. Stage, and J. DeGross, eds., Proceedings of the IFIP International Working Conference on the Social and Organizational Perspective on Research and Practice in Information Technology, Kluwer, Boston, pp. 61–82.

Dietterich, T., 1998, Approximate statistical tests for comparing supervised classification learning algorithms, *Neural Computation* **10**(7):1895–1923.

Fayyad, U. M., 1996, Data mining and knowledge discovery: Making sense out of data, *IEEE Expert* **11**(5):20–25.

Fayyad, U. M., and Uthurusamy, R., 2002, Evolving data into mining solutions for insights, *Communications of the ACM* **45**(8):28–31.

Hilario, M., and Kalousis, A., 1999, Characterizing learning models and algorithms for classification, CUI – Univercity of Geneva, TR UNIGE-AI-9-01.

Hilario, M., and Kalousis, A., 2000, Building algorithm profiles for prior model selection in knowledge discovery systems, *Engineering Intelligent Systems, Special Issue on Data Mining* **8**(2), IEEE Press.

Iivari, J., Hirscheim, R., and Klein, H., 1998, A paradigmatic analysis contrasting information systems development approaches and methodologies, *Information Systems Research* **9**(2):164–193.

Imielinski, T., and Mannila, H. 1996, A database perspective on knowledge discovery, *Communications of the ACM* **39**(11):58–64.

Ives, B., Hamilton, S., and Davis, G., 1980, A framework for research in computer-based management information systems, *Management Science* **26**(9):910–934.

Järvinen, P., 1999, On Research Methods, Tampere, Opinpaja; http://www.uta.fi/~pj/.

Kiang, M., 2003, A comparative assessment of classification methods, *Decision Support Systems* **35**:441–454.

Kleinberg, J., Papadimitriou, C., and Raghavan, P., 1998, A microeconomic view of data mining, *Data Mining and Knowledge Discovery* **2**(4):311–324.

Kohavi, R., and John, G., 1998, The wrapper approach, in: *Feature Selection for Knowledge Discovery and Data Mining*, H. Liu and H. Motoda eds., Kluwer Academic Publishers, pp. 33–50.

Lyytinen, K., 1987, Different perspectives on information systems: Problems and solutions, *ACM Computing Surveys* **19**(1):5–46.

Mannila, H., 2000, Theoretical framework for data mining, *SIGKDD Explorations* **1**(2).

Mehta, M., Rissanen, J., and Agrawal, R., 1995, MDL-based decision tree pruning, in: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD 1995)*, U. M. Fayyad and R. Uthurusamy, eds., AAAI Press, Montreal, Canada, pp. 216–221.

Michalski, R. S., and Wnek, J., 1993, Constructive induction an automated design of knowledge representation spaces for machine learning. Reports of the Machine Learning and Inference Laboratory, MLI 93–11, School of Information Technology and Engineering, George Mason University, Fairfax, VA, November.

Michalski, R. S., 1997, Seeking knowledge in the deluge of facts, *Fundamenta Informaticae* **30**:283–297.

Nunamaker, W., Chen, M., and Purdin, T., 1990–91, Systems development in information systems research, *Journal of Management Information Systems* **7**(3):89–106.

Quinlan, J. R., 1993, C4.5 Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA.

Piatetsky-Shapiro, G., 2000, Knowledge discovery in databases: 10 years after, *SIGKDD Explorations* **1**(2).

Tkach, D., 1998, Information mining with the IBM intelligent miner family, An IBM Software Solutions White Paper; www.acm.org/sigs/sigmod/disc/disc99/disc/ibm/whitefam3.pdf.

Wolpert, D. H., and MacReady, W. G., 1996, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* **1**(1):67–82.

FILLING THE KNOWLEDGE MANAGEMENT SANDWICH: AN EXPLORATORY STUDY OF A COMPLEX WORK ENVIRONMENT

Henry Linger and Jeremy Aarons*

1. INTRODUCTION

On 29 January 2004 the first of the Meteosat Second Generation (MSG) satellites MGS-1 (renamed Meteosat-8) commenced routine operations, beginning a new era in weather forecasting. The first of three geostationary satellites situated over Europe, it is the most sophisticated satellite currently operating, providing more precise and detailed data at more frequent intervals than any previous satellite. As such it will increase the power of forecasters to predict a range of severe weather events with previously unattainable speed and accuracy. In addition to these services for professional meteorologists, anyone with a satellite dish, a PC, and a relatively inexpensive plug-in decoder card can tap into the live data feed for free. As a result, virtually anybody with access to the right equipment can forecast the weather for themselves. One possibility that may emerge from this new availability of data is that individuals could to do the sorts of tasks that are presently a lucrative business for institutional weather forecasting. Given this ability for almost any individual to construct their own personalised forecast, one may be tempted to ask, as a recent article in *New Scientist* magazine does, "Who needs weather forecasters?" (Mullins, 2004)

The belief that new technology such as these satellites will make weather forecasters an endangered breed is based on a serious misconception. In particular, the idea that all it takes to forecast weather is access to more and detailed data is quite wrong. As our studies of professional weather forecasters have revealed, accurate and informative weather forecasting requires far more than just the input of detailed meteorological data and its interpretation. Having the raw data feeds can be informative, but more is required in order to be able to effectively develop a forecast. Forecasting is a complex and distributed cognitive task (Kelder and Turner, 2004), performed by a team of highly trained and experienced experts, in an organizational setting that facilitates learning, sharing, and interactive decision

* School of Information Management and Systems, Monash University, 26 Sir John Monash Drive, Caulfield East, Victoria 3145, Australia, henry.linger@sims.monash.edu.au, jeremy.aarons@sims.monash.edu.au.

making. Our studies reveal that the challenge of improving forecasting is as much about improving the support for this collaborative process in its organizational setting, as it is about improving the quality and complexity of the guidance materials (satellite, radar, ground observations, and numerical prediction data) that are the inputs to the forecast process.

Our aim in this paper is to use a case study of a complex work environment, exemplified by weather forecasting, to explore how such complex work practices can be understood and supported. Such knowledge work requires an understanding of the pragmatics of the activity system, the conceptualisation of the work and an understanding of the organisational context and the social, cultural and cognitive aspect of the task. In this paper we argue that task-based knowledge management (Burstein and Linger, 2003) can define the conceptualisation and pragmatics of the activity system while a bottom-up, disunified and localised investigation of the activity system defines the epistemological requirements (Aarons, 2004).

In the next section, we present the case study at the Victorian Regional Office of the Australian Bureau of Meteorology. The section examines the organizational setting, the forecasting and informational environment, the forecast process and how the forecast is actually produced. In the following section the case study is discussed in terms of insights about the nature of work practices in such a complex environment. The following section interprets these insights within a knowledge management framework. The conclusion brings together the empirical and theoretical understanding of how to support knowledge work in a complex environment.

2. THE CASE STUDY: WEATHER FORECASTING

2.1. Introduction to the Case Study

This discussion summarises the findings of studies conducted at the Victorian Regional Office of the Bureau of Meteorology in September 2003. The approach taken was to undertake an ethnographic study, involving observational studies of forecasters at work, delving into the messy details of the tasks performed by forecasters, along the lines of Schultze's (2000) ethnographic investigation into knowledge work. The aim of this exercise was to get an overview of how forecasting is currently being conducted, by investigating what goes on within the activity of the forecaster. This involved developing an understanding of all the factors that contribute to the construction of a weather forecast. This includes the all the processes involved in the forecasting *task*, and the relevant *artefacts* and *actors* that contribute to this task.

To achieve this we first familiarised ourselves with the operational environment within which forecasters work, including the *physical* environment, the *informational* environment as well as the *social* environment. The informational environment includes the particular IT systems (hardware and software) used by the forecasters, as well as the other technical systems and information sources that forecasters rely on for their forecasting. The social environment is the interactive social space that the forecaster shares with other forecasters, other staff, and outside parties.

The study was conducted by observing a meteorologist at work in their normal working environment, sitting with the forecaster for the duration of a shift, and when possible having the forecaster talk through the tasks being performed. The observational study was also supplemented by further discussions with the forecasters during breaks in their shift. These provided the forecaster with opportunities for expressing their understanding and presenting the way they think they are doing their work, as well as providing a forum for any clarification and explanation that we required.

Throughout our study we investigated the details of the various tasks and activities undertaken by the forecaster. In particular, we looked at how the forecaster's time was distributed across each particular activity undertaken as part of their shift routine. This involved looking at the details of forecast product preparation, including the use of guidance materials such as Numerical Weather Predictions (NWP) products and other decision support tools, as well as the various interactions with other forecasters, and communication with other parties (eg. media, outside requests for information).

Although this study specifically concerns the Victorian office of the Bureau of Meteorology, the findings have been supplemented by further observational studies undertaken in Tasmania in March 2004. The specific details of the findings here may vary from the situation in other Bureau offices. However the basic structure and forecast procedure is essentially the same across the offices (Bally, 2002). Our study is also informed by the results of two other studies of forecasters. Our initial understanding of forecasting was heavily informed by Bally (2002), which presented a series of detailed process models of forecasting work. We also referred to Kelder's (2003) study of forecasting at the Tasmanian Regional Office, which characterised forecasting from a Distributed Cognition perspective (Hutchins, 1995).

2.2. Background: The Forecasting Environment

The Australia Bureau of Meteorology* is a federally funded government organization that operates as an Executive Agency within the Environment and Heritage Portfolio. The Bureau's Head Office in located in Melbourne and serves as both an administrative and operational headquarters. Forecasting is conducted in each of the regional branches, located in each of the seven State capitals and in Darwin.

In each forecasting Regional Office (R.O.) there is always at least one forecaster on shift at any time of the day or night. For most forecasters a standard forecasting shift is 12 hours long – in Victoria the shifts are from 7 to 7. A forecaster typically works for two or three days in a row, and then has at least one day off. On every shift a shift supervisor is present. This person is a senior forecaster or *SPOC* (Senior Professional Officer Grade C). In addition up to three *PO2* (Professional Officer Level 2) forecasters may be on shift at any time, working under the supervision and guidance of the SPOC. At particularly busy times of the year additional forecasters may also be brought in, such as at the peak of the fire season.

The SPOC acts in a coordinating role, setting the weather policy for the shift and reviewing forecasts before they are published. In addition the SPOC performs a number of other functions, including: warning policy and preparation; analysis and diagnosis; media

* For more details see http://www.bom.gov.au.

liaison (eg. radio interviews); emergency service liaison; phone calls from key clients; quality control; and general managing of the R.O.

The PO2 forecasters spend the bulk of their shift working according to a fixed routine, spending much of their time working on and writing the scheduled forecasts, which are saved into a work file for review by the shift supervisor before being published. This routine, however, is often disrupted by a number of other activities, including answering phone enquiries, updating weather warnings, and helping other forecasters with specialised tasks such as fire weather forecasting when required.

Although the allocation of tasks between the forecasters on shift is fairly clearly defined, in practice there is a lot of interaction and collaboration between the forecasters, and there is often sharing or even swapping of tasks. This collaborative environment is well supported by the physical layout of a typical forecasting office set up. The work spaces are located close together in an open office environment, so that the forecasters are in close proximity to each other. This means that each forecaster is always aware of what the other forecasters are doing, and can easily participate in another forecaster's task if their assistance is required (Kelder, 2003).

The informational environment is nested within the physical setup of the office space, and is focused around the forecaster workstation and the multiple screens accessed by the forecaster during forecasting. The key elements of the informational environment are: a dual OS environment (Windows and Linux); a forecast preparation system and integrated data viewer; multiple data viewing consoles; the large synoptic chart; NWP printouts; and other paper artefacts (shift handover forms, media summaries).

In this informational environment the forecasters have access to a range of numerical weather prediction systems, which provide crucial inputs to the forecast process. These NWP models include a number of global weather models, as well as numerous more specialised and more local scale models.* Although there are numerous different systems feeding into the forecasting process, many of these have been integrated into the Australian Integrated Forecast System (AIFS) (Kelly and Gigliotti, 1997; Kelly et al., 2004), a combination of packages for viewing data, managing databases and preparing and distributing messages. Within this system there are two main applications used by forecasters: the data viewer, nicknamed "Kenny", which integrates the viewing of many different forms of guidance data within a single system; and "Linkage", an integrated forecast preparation package, developed by the Bureau, that brings together all the forms and forecast templates in a single system for editing and publishing. These applications are accessed using the multiple screen setup of the forecaster workstation (Figure 1), with which the forecaster can view multiple forms of guidance data at once, while simultaneously viewing and editing the content of forecast products.

Forecasters also consult additional screens strategically located around the office, which display current radar data and the current ground observations obtained from automatic weather stations. This data can also be accessed from each forecaster's workstation. Additionally, each forecaster routinely consults a screen which lists the current alerts and warnings. This screen display a list of up-to-date weather events that are worthy of special

* A summary of the Bureau's NWP products can be found at http://www.bom.gov.au/nmoc/NWP.shtml More details can also be found at http://www.bom.gov.au/nmoc/anal_prog.shtml [May, 2004].

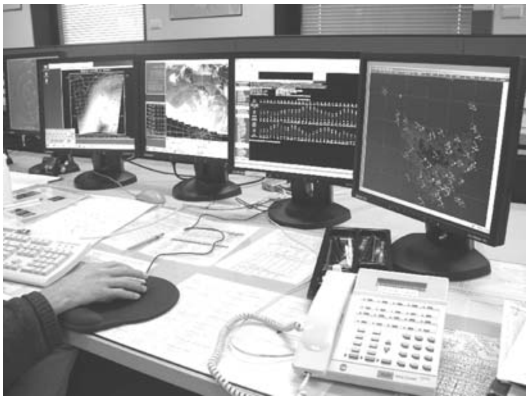


Figure 1. A typical 4-screen forecasting workstation.

notice, such as when wind gusts are over a certain magnitude at a particular location. These alerts are customisable according the individual forecaster's particular needs, and provide further support for the forecasting task.

2.3. The Activity in Focus: The Forecast Process

In brief, the forecast process involves the synthesis and analysis of a large amount of meteorological data, leading to the production of a number of different forecast products (Linger et al., 2001). Essentially there are three basic components to the forecast process:

1. data inputs – NWP guidance and other data inputs
2. the activity of the forecaster – applying the Weather Forecast Policy and writing the forecasts
3. the outputs – forecast products

What the forecaster does is integrate the data inputs to construct a mental model of the weather system, which then gets translated into the outputs for the weather products (Bally, 2002). This forecasting process relies significantly on the skill and experience of the forecaster, who must integrate the many diverse forms of guidance data with their own evolving mental picture of the weather system. This mental model is held in the forecaster's memory and is never made explicit, although it is used to develop representations at various stages in the forecast process. The most obvious place this occurs is when a new observations chart appears one of the forecasters on shift sketches out the isobars and fronts by hand in pencil on the large synoptic chart. The location of isobars and fronts are based on a number of factors: extrapolation from the previous synoptic chart; guidance from the satellite picture; NWP guidance; current readings from weather stations located around the state. This is done routinely every 3 hours, with the charts kept in a large stack before they are eventually filed away in archives.

During our observational sessions forecasters referenced this chart at fairly regular intervals. It seemed that the forecasters were using the explicit representation of the synoptic chart to help consolidate their own mental model of the weather system. When questioned about the relevance of the chart, the forecasters stated that the actual process of sketching the chart was an important part of the process of developing their own mental model of the weather. This emphasises the importance of the forecaster's mental model of the evolving weather system in the forecast process.

This mental model is often referred to as the "Weather Forecast Policy", and consists of past, present and future information about weather objects, areas, and points (Bally, et al., 2004). In a general sense the Policy is both a summary of the current analysis of available data, including observations and numerical data, and a guide for deciding which NWP models to rely on for that day's forecasting, and how to read the outputs of those models. As such, the Policy defines the framework within which the forecasters situate their forecasting work. Once the Policy has been set the issued forecasts must remain consistent with this Policy, unless there is a formal decision to change to Policy.

This conception of the Forecast Policy should be distinguished from another formal sense in which this term is used. Each day, near the beginning of the morning shift, the Shift Supervisor prepares a document that is also known as the "Forecast Policy". This formal document is accessible by all forecasters from within the forecasting system (AIFS), and essentially provides a summary of the more intangible Forecast Policy. However for most purposes the Policy is held in the memory of all forecasters on the shift, and forms the basis for their forecasting decisions. The key elements of the Policy are also used to automatically populate a number of forecast products within the AIFS system. (Kelder, 2003: 146) Although the Policy is set in the morning by the Shift Supervisor, it is not a fixed entity, and the other forecasters have continual input into the process of reviewing and updating the policy:

> "... the RFC Shift Supervisor maintains overall responsibility for forecast policy for the entire State. This forecast policy is developed on an ongoing basis throughout the day as new information becomes available and provides general guidance to all forecasting staff .... Group consultation at various times during the day contributes to the development of the forecast policy." (Bureau of Meteorology, 1999b)

During their routine forecasting work the forecasters are continually looking at incoming flow of data: the direct observations, satellite and radar images, and the new runs of NWP models. Through this process they are constantly updating their own mental picture of the evolving weather system, and constantly reassessing the details of the Forecast Policy. If it seems that the evolution of weather systems or data models conflicts with the current Policy, forecasters are able to discuss or query that Policy directly by talking to the senior forecaster. (Kelder, 2003:112)

2.4. Performing the Activity: Forecasting in the Context of Shift Work

2.4.1. The Routine and Disruptions to the Routine

Although forecasting is a continuous practice it is situated in the context of shift work. A forecaster's shift is structured around a routine series of forecasts that must be updated and published at regular intervals. Additionally, a number of other routine tasks are performed by the forecaster, such as the sketching of the synoptic chart and the updating of

NWP printouts. This routine is also regularly disrupted. The most common disruptions are the numerous telephone calls from clients requesting particular weather information. The routine is also disrupted when particular non-standard and specialist forecasts are urgently required at short notice. This most commonly occurs when particular weather warnings need to be issued and updated at frequent intervals, such as when thunderstorms or tropical cyclones develop rapidly. Another common form of non-standard forecast, especially in the summer months, is a spot fire forecast, where a local-scale forecast is issued covering the vicinity of a particular bush fire. This is a particularly challenging task, as it involves a detailed and fine grained knowledge of the relevant guidance data, as well as an understanding of the local geography of the particular region, and the practical requirements of those who will rely on the forecast (eg. fire fighters). Another example of an unexpected forecasting task is when a particular client requests a detailed short term wind nowcast. The case observed involved the operator of an oil platform in Bass Strait wanting to know if it would be safe to airlift people in by helicopter.

2.4.2. The Shift Changeover

A forecasting shift begins with a formal shift changeover period, in which the incoming forecasters are briefed. This is a 30–45 minute briefing, where the forecaster ending the shift passes on all the essential knowledge needed to continue forecasting into the new shift. This involves the outgoing forecaster summarising the evolving state of the weather system over the duration of the previous shift, discussing the status of currently issued warnings, as well as flagging any additional warnings that may need to be issued in the near future, and reviewing the guidance data with the new forecaster. The review of guidance involves explaining how the various guidance types have been used, and which ones were seen as significant. This stage also allows the previous forecaster to give an overview of the justification for the forecasting decisions made during the earlier shift (which is really the only stage in which justifications are made explicit). The shift handover thus provides the starting point for the development of the Forecast Policy by the oncoming forecaster as discussed previously.

2.4.3. Group Meeting for Chart Analysis, Attended by all RO Forecasters

After the Policy has been set the entire forecasting team come together for a meeting to summarise and discuss the Policy, review the current state of the weather system, and flag any issues that may be cause for further attention during the forthcoming shift. As the Bureau staff explained, this meeting is a leftover from past practices, before detailed displays and sophisticated forecast systems became available and accessible by all forecasters. However it still serves the useful function of bringing forecasters up to speed on current Policy and the issues and events that could arise within the next shift.

At this meeting the shift supervisor talks through the present state of the weather system by reviewing satellite data; the issued warnings (eg. wind, thunderstorm, fire, grazier); and looking at weather features that should be monitored closely, such as the progress of fronts and the evolution of thunderstorms and other severe storms. As a group, the forecasters also review the NWP predictions from the multiple models available to forecasters. They look at the forecast charts from the five major global models, displaying the outputs

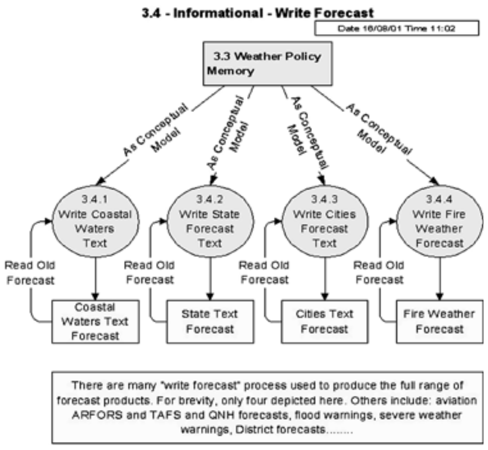

3.4 - Informational - Write Forecast
Date 16/08/01 Time 11:02
3.3 Weather Policy Memory
As Conceptual Model
As Conceptual Model
As Conceptual Model
As Conceptual Model
3.4.1 Write Coastal Waters Text
3.4.2 Write State Forecast Text
3.4.3 Write Cities Forecast Text
3.4.4 Write Fire Weather Forecast
Read Old Forecast
Read Old Forecast
Read Old Forecast
Read Old Forecast
Coastal Waters Text Forecast
State Text Forecast
Cities Text Forecast
Fire Weather Forecast
There are many "write forecast" process used to produce the full range of forecast products. For brevity, only four depicted here. Others include: aviation ARFORS and TAFS and QNH forecasts, flood warnings, severe weather warnings, District forecasts........

Figure 2. The product preparation process (adapted from Bally, 2002).

side-by-side on a projection screen. They also look at the outputs of local models which give a more detailed analysis of some specific features of the weather system such as wind vectors.

This group analysis of the guidance data plays an important role in the weather forecasting process, as it allows for a shared understanding of the weather system and the status of NWP guidance, and facilitates group input into the current Forecast Policy.

2.4.4. Forecast Product Production

The production of forecasts products according to the fixed schedule follows the process illustrated in Figure 2, from Bally (2002). The stages represented in rectangles are *artefacts*, the ovals *activities*. The starting stage 3.3 Weather Policy is a conceptual and subjective artefact that is held in the minds of forecasters, not explicitly encoded within a system. The writing stages 3.4.x are cognitive conceptual activities that occur within the minds of forecasters and are not explicitly recorded. Only the final forecast outputs are explicitly and objectively encoded in a physical entity or system.

In writing a routine forecast the forecaster brings up the previously issued forecast using the Linkage system. This system automatically updates some aspects of the previous forecast, such as the issue time, and automatically populates the forecast form with some data derived from the official Forecast Policy. However the system does not automatically update the text description of the weather – this must be done manually by the forecaster.

This process of drafting the forecast text is a highly complex task. It involves analysing and synthesising the meteorological data, and summarising the essential elements of forecast within four lines of text (a strict requirement). As a result, the choice of expression and vocabulary is a particularly difficult and complex task, which requires a detailed knowledge of all aspects of the forecasting process, as well as an appreciation of how those forecasts will be read and applied by their clients.

3. DISCUSSION: CASE STUDY FINDINGS

Our study has shown that the way forecasts are constructed by forecasters has remained largely unchanged since the initial revolution in weather forecasting, made possible by profound improvements in accurate satellite imaging and powerful numerical modelling from the 1950s onwards (Fishman and Kalish, 1994). In particular:

- the evolving 'mental picture' of the weather system in the forecaster's head is of paramount importance to forecasting
- the interactions between forecasters is a significant factor in the forecast process
- forecast products continue to be time consuming activities as an enormous amount of time was spent manually editing products, even though this is now done online: it is a very slow process of retrieving the previous version of each product and typing the edits to form the updated product
- the forecast process is consistently disrupted, including external phone calls and the need to produce non-standard forecasts
- the forecaster's 'expert knowledge' of their domain, including their knowledge of local geography and topology, is a key element in forecast construction
- the forecaster's expertise is an expression of their training and extensive experience rather than the volume and diversity of data

Our study has thus shown that forecasting is not only dependent on data but is a complex socio-technical activity that is also knowledge intensive.

Our studies have also revealed that although some new tools have been built over recent years, these tools are essentially designed to facilitate the existing ways of forecasting rather than to improve the forecasting process. Although the quality and accuracy of guidance data has improved considerably, the NWP models, satellite imagery, and automatic weather station observations are used selectively for guidance and not to automate the forecast process. The AIFS system, with its integrated data viewer and forecast preparation system, provides the facility to incorporate and integrate a large number of data inputs, and brings together all the forms and forecast templates in a single system for editing and publishing. However it only streamlines certain aspects of the current forecast process, and is ill suited to meet some of the future challenges facing weather forecasting. These future challenges are a result of scientific and technological developments, as well as a renewed public (and legal) interest in accurate and informative weather forecasting.* These have created a challenge for weather forecasting to meet the contradictory demands inherent in the forecasting process in terms of:

- the increased work load from the diversity of weather products that need to be prepared within a fixed timeframe
- the volume of diverse material that needs to be referenced to prepare forecasts
- the management of that material to ensure that relevant material is available during the forecast process

* Much of the impetus for improving forecasting systems has arisen from two recent investigations into high profile cases of perceived forecasting problems: the 1998 Sydney to Hobart yacht race tragedy (Bureau of Meteorology, 1999a) and the 1999 Sydney hailstorm investigation (Bureau of Meteorology, 1999b).

- increased demands for justification of forecasts
- the increasing need for meteorologists to exercise judgement and to use their experience and knowledge to overcome the limitations of scientific knowledge

The tools available to forecasters at present are limited in their scope and can only meet some aspects of the demands of weather forecasting. The challenge therefore is to find a way to support the activity of the forecaster in terms of both the *pragmatics* of the task (the specifics of the products and the context of work) and the *epistemological requirements* of the task (to produce accurate forecasts). The following section presents a two pronged approach to frame forecasting in terms of knowledge management as a means to meet this challenge.

4. THEORETICAL REFLECTIONS: THE KM PERSPECTIVE

A complex work environment, as exemplified by weather forecasting, is a *socially* and *technologically* distributed activity that involves intellective work, learning, sharing, and interactive decision making. Such an environment can be interpreted through a knowledge management lens to view the nature of work as knowledge work and the activity as knowledge management.

From a knowledge management perspective, the activity system needs to be understood in terms of the structures and process that enables the construction of the required output (the pragmatics) and an explicit understanding of the knowledge that enables effective construction of the output (the epistemological requirements)

The pragmatics can be addressed through the Task-based Knowledge Management (TbKM) framework developed by Burstein and Linger (2003). The framework integrates two levels of understanding of the activity system, the task:

- the explicit models of the structure and process of a task, as conceptualised by the actor performing the task to document her understanding of the task
- models that enable the task to be efficiently performed

The TbKM framework provides tangible artefacts to support the intellective work as well as the practical actions required to produce organisationally defined outcomes. In this sense the framework supports knowledge work as defined by Iivari and Linger (1999, 2000) as it:

- allows the object of work to be defined (the inputs, outputs and performance)
- identifies the body of knowledge that underpins the work (the conceptual models)
- allows instantiation both in terms of the item of work
- supports the production of knowledge as an aspect of the outcome (learning)
- represents and inscribes the objects of work (the models) providing the artefacts with which the actor can perform her work

The TbKM approach has been used successfully in a number of domains including weather forecasting (Burstein and Linger, 2003). At the Bureau of Meteorology this approach has contributed to a strategic initiative, the "Forecast Streamlining and Enhancement Project" (FSEP) that aims to redevelop forecasting systems to conform with knowledge management principles (Ryan, 2003; Bell, 2003).

However the framework does not provide the means to understand and make sense of the organisational context and the social, cultural and cognitive factors that are involved in a complex work environment. Our approach to these epistemological requirements involves a bottom-up, disunified and localised investigation into the workings of an activity system (Aarons, 2004), in this case weather forecasting. The investigations are made by a thorough analysis of the components of the work environment, looking at how those components piece together, and the processes that flow between those components. This approach can be characterised as a disunified methodology (Aarons, 2001).

From a knowledge management perspective, the starting point of such an analysis is to determine precisely what aspects of knowledge are relevant to that particular case, and to give an account of the factors underlying these knowledge components. This involves assessing the relevant cognitive, social and pragmatic factors involved in that particular work activity. From such an analysis a complex picture can be developed, piecing together the details until they form a complex model of the activity system.

This approach maintains a connection with real-world processes and properties, and results in models that represent these real processes and properties. Importantly, this approach acknowledges the significant social dimension of knowledge work in such organizational and work settings, while retaining the idea that social processes are deeply connected to real properties and processes. The case study reinforces this theoretical perspective, highlighting the complexity of the work environment. Our observations show the forecasting task is dialectically grounded in the guidance material and the cognitive models of the weather and geography. But the conduct of the task is also influenced by social interactions, including disruptions, and the practical activity of writing or drawing as well as the physical structure of the work setting and the organisational imperatives regarding the products.

In a more general way, the disunified methodology contributes to building a theoretical framework for supporting knowledge work that is grounded in reality, but also incorporates the relevant social, practical, and pragmatic concerns that are central to the work tasks. This theoretical framework can be extended with the pragmatics that TbKM can contribute. Combining these two approaches, results in a theoretical framework that supports a theory of collaborative knowledge work within a realist and pluralist metaphysical framework (as outlined in Cartwright, 1999). Importantly, the framework, as a model of a complex work environment, provides the means to study and understand knowledge work and derive the design criteria for artefacts to support this work.

Figure 3 below illustrates this combination. The sandwich, referred to in the title is a metaphor to emphasis that each approach, on its own, does not provide the necessary tools to deal with the complex work environments even though each approach is necessary. Individually, each approach is the bread necessary to make a sandwich. However a sandwich is only satisfying if it has a filling. Combining the two approaches allows a rich theoretical and practical understanding of collaborative knowledge work. It is this theoretical framework that is the filling in the sandwich.

The TbKM approach, disunified methodology and the theoretical model of collaborative knowledge work all contribute to answering the general question: How do we study

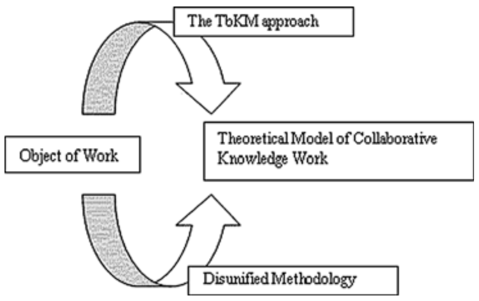


Figure 3. The "Sandwich Approach".

and support knowledge work in a complex work environment? While this is not a trivial problem in itself, the more challenging issue is to move from an understanding to an intervention.

5. CONCLUSION

In this paper we have explored a complex work environment in which highly trained and experienced experts engage in knowledge work using weather forecasting to illustrate such a setting. Our investigation reveals that as well as the pragmatics of the work task, forecasters also have epistemological requirements in order to construct accurate and informative forecasts. Based on this work, and our previous studies, we propose a theoretical framework that supports a theory of collaborative knowledge work that is derived from a task-based knowledge management framework and the disunified methodology exemplified in the case study. This framework is grounded in the reality of the physical and informational environment but also incorporates the relevant social, practical, and pragmatic concerns that are central to the work tasks.

The central issue presented in the paper is the importance of the disunified methodology to identify the epistemological requirements and the contribution of those requirements to understanding a complex work environment. Our future work will define and characterise the theoretical framework for collaborative work by integrating the two approaches identified in this paper.

REFERENCES

Aarons, J., 2001, *Thinking Locally: A Disunified Methodology of Science*, PhD manuscript, School of Philosophy and Bioethics, Monash University, (unpublished).

Aarons, J., 2004, The Disunity of Knowledge Work, *Proceedings of the Fifth European Conference on Organizational Knowledge, Learning and Capabilities OKLC2004*, Innsbruck, April 2004, (May 1, 2004); http://www.ofenhandwerk.com/oklc/pdf_files/I-5_aarons.pdf.

Bally, J., 2002, *Forecast Information Flows Analysis.* Bureau of Meteorology internal technical report.

Bally, J., Boneh, T., Nicholson, A., and Korb, K., 2004, Developing an Ontology for the Meteorological Forecasting Process, *Proceedings of the 2004 IFIP International Conference on Decision Support Systems (DSS2004)*, Prato, Italy, (In Press).

Bureau of Meteorology, 1999a, Preliminary report on meteorological aspects of the 1998 Sydney to Hobart yacht race, (February 20, 2004); http://www.bom.gov.au/inside/services_policy/marine/sydney_hobart/prelrept.html.

Bureau of Meteorology, 1999b, Report by the Director of Meteorology on the Bureau of Meteorology's Forecasting and Warning Performance for the Sydney Hailstorm of 14 April 1999, (February 20, 2004); http://www.bom.gov.au/inside/services_policy/storms/sydney_hail/hail_report.shtml.

Bell, I., 2003, *FSEP Principles*, Bureau of Meteorology internal technical report.

Burstein, F., and Linger, H., 2003, Supporting Post-Fordist work practices: a knowledge management framework for supporting knowledge work, *Information Technology & People*, Special Issue on Organizational Implications of Knowledge Management Systems.

Cartwright, N., 1999, *The Dappled World,* University of Chicago Press, Chicago.

Fishman, J., and Kalish, R., 1994, *The Weather revolution: Innovations and Imminent Breakthroughs in Accurate Forecasting*, Plenum Press, New York.

Hutchins, E., 1995, *Cognition in the Wild*, The MIT Press, Cambridge, Mass.

Iivari, J., and Linger, H., 1999, Knowledge work as collaborative work: a situated activity theory view, *Proceedings of the Hawaiian International Conference on Systems Science* (HICSS'32).

Iivari, J., and Linger, H., 2000, Characterizing Knowledge Work: A Theoretical Perspective, *Proceedings of the Americas Conference on Information Systems AMCIS'2000.*

Kelder, Jo-Anne, 2003, *Generating Insights for Meteorological Information Systems Design Without Burdening the Participants*, Bachelor of Information Systems Honours Dissertation, University of Tasmania.

Kelder, Jo-Anne and Turner, Paul, 2004, In the eye of the storm: the role of distributed cognition theory for informing the design of meteorological information system. Presented at ISOneWorld Conference – April 14–16 , 2004 Las Vegas, NV, USA.

Kelly, J., and Gigliotti, P., 1997, The Australian Integrated Forecast System (AIFS): Overview and current status, Preprints, 13th Int. Conf. on IIPS for Meteorology, Oceanography and Hydrology, pp. 141–144.

Kelly, J., Donaldson, A., Ryan, C. J., Bally, J., Wilson, J., and Potts, R. J., 2004, The Australian Bureau of Meteorology's next generation forecasting system, *Preprints, 20th International Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology*, (May, 2004); http://ams.confex.com/ams/pdfpapers/70519.pdf.

Linger, H., Burstein, F., Kelly, J., Ryan, C., and Gigliotti, P, 2000, Creating a learning community through knowledge management: the Mandala Project, *Proceedings of the IFIP WG 8.3 conference "Decision support through knowledge management"*, Sweden, pp. 122–36.

Linger, H., Burstein, F., Ryan, C., and Kelly, J., 2001, Implementing a Knowledge Management System: the Case of Meteorological Forecasting, in: *Knowledge Management for Information Communities*, Burstein and Linger, eds., Australian Scholarly Publishing, Melbourne, pp. 139–153.

Mullins, J., 2004, Outlook good for home forecasting, *New Scientist* **181**(2432):24.

Ryan, C., 2003, The Forecast Streamlining and Enhancement Project. Bureau of Meteorology – internal technical report.

Schultze, U., 2000, A confessional account of an ethnography about knowledge work, *MIS Quarterly* **24**(1):1.

DETERMINING AN APPROPRIATE APPROACH TO THE IMPLEMENTATION OF A WFMS

Mehmet N. Aydin*

1. INTRODUCTION

In Information Systems research, more than three decades the effective use of methods has been studied. In the 1980s and 1990s, the rationales behind structured, brand-named methods, the so-called conventional methods, were being questioned as being IT-oriented, complex, rigid, and inappropriate for today's organizations.

One of the problems, often cited in the Information Systems Development (ISD) literature, is that conventional ISD methods adopt 'one-size-fits-all' approach to system development. We know that every project has different characteristics, so projects are unique and there is no best way or approach that works for all projects. The truth is that 'one-size-fits-all' paradigm does not work in reality and practitioners need to tailor the methodology according to their projects' needs or characteristics. The central issue in this research stream and in our paper is how to determine an appropriate approach to the implementation of a workflow management system (WfMS) so that we can reduce the risk of failures of such an implementation.

This paper is concerned with the problem of 'one-size-fits-all' in the context of implementation of WfMSs. In other words, we want to illuminate and propose a way to tackle this problem. For illustration purposes we provide a sample project to explicate the use of our model. This paper is a kind of a positioning paper in the sense that it frames the aforementioned research issue and suggests further research topics.

The structure of this paper is as follows. The motivation behind the research has been outlined in this section. The remainder of the paper consists of three key sections: (i) a review of related research and the introduction of a model, (ii) elaborations of key constructs in the model and (iii) illustrations of the model with a case and conclusions of the research.

* University of Twente, Department of Business Information Systems, School of Business, Public Administration & Technology, P.O. Box 217 7500 AE Enschede, The Netherlands, m.n.aydin@utwente.nl.

2. REVIEW OF RELATED RESEARCH AND PROPOSED MODEL

Before presenting relevant literature, we should indicate that for many terms used in this paper, such as 'development method', 'approach', we adopt their definitions in (Iivari, Hirschheim, and Klein, 2001), for the terms, 'workflow', 'WfMSs', we refer to (Stohr and Zhao, 2001) and (van der Aalst and Hee, 2002).

An ISD approach simply means a high level description of the method including the goals and the guiding principles, and the beliefs, fundamental concepts, and principles of a system development process. Whereas an ISD method refers to an organized collection of concepts, beliefs, values, and normative principles supported by material resources.

As indicated in the previous section, IS research has been working on the issue of 'one-size-fits-all' for several years and one can find several theoretical models proposed to solve the problem of 'one-size-fits-all'. Most of the models use the idea of tailoring as a promising and appropriate way to tackle this problem. The idea simply asserts that a method should be tailored to the project situation at hand (Slooten and Hodes, 1996).

Different paradigms or mechanism have been used to operationalize the way to tailor a method (Aydin, 2004). In this paper, we do not want to delve into discussions of the existing models and paradigms behind the proposed models, but rather we stay at an abstract level and propose a model that can be used in the context of implementation of WfMSs. (Aydin, Harmsen, Sloten, and Stegwee, 2004) suggest using two complementary perspectives – the engineering and socio-organizational perspectives – to tailor a method to the project situation at hand. The former is of interest to the school of method engineering, emphasizes the structural aspects of the method, and usually employs contingency-based models for tailoring method. The latter appears to be concerned with a better understanding how a method and its components are invented on-the-fly. This perspective employs the body of knowledge contained in the socio-organizational literature (Baskerville and Stage, 2001).

With respect to the research related to development of WfMSs, it is surprising to note that little research addresses the 'one-size-fits-all' issue. In fact, our quick literature review indicates that both empirical and theoretical research concerning this issue is very limited. One of the reasons might be that implementation of WfMSs does not require special attention and we would use the same suggestions as proposed in implementations of typical IS solutions.

However, Stohr and Zhao point out the need for a study on WfMSs implementation as one of the research issues in the field of WfMSs (Stohr and Zhao, 2001). One can argue that WfMSs by nature is different from other systems. In fact, we need to answer first what really makes WfMSs different from other systems?

(Cardoso, Bostrom, and Sheth, 2003) discuss the difference between WfMSs and ERP in terms of three dimensions: domain scope, technology scope and system implementation. Van der Aalst and van Hee explain the need for a specific method for WFSs and state that 'Existing methods for the development of IS place a strong emphasis upon defining data structures and the way in which the application is presented to its users. Organisational change and the (re)design of processes receive limited attention in these methods. (...) A method for developing a workflow system should therefore focus upon the business process and embrace both the organisation and technology' (Van der Aalst and van Hee, 2002, p. 211–212).

(Weske, Goesmann, Holten, and Striemer, 2001) analyse a number of WfMSs projects in empirical settings. They presented problems, issues encountered during six projects and draw some lessons and propped development methodology for Workflow Application Development Method (WADM). Their empirical findings provide us a good starting point for our model.

Before introducing our model, we shall indicate the scope of tailoring and our focus in this paper. A WADM in principle aims at supporting project participants to achive project goals in an efficient and effective way. A method is more than just a collection of stages-activities, deliverables, and techniques. Further, it embeds a certain spirit, a paradigm describing the way of thinking enacted in the project. The functioning of a method can be described in terms of: the way of thinking (presuppositions, principles, strategies), the way of working (modelling tasks, decisions, stages-activities), the way of modelling (deliverables), the way of supporting (techniques, tools and job aids to be used) and the way of controlling (a managerial aspect of the project). So, it is not surprising to see that methods are limited to their ways of thinking, working, modelling, and controlling and are usually proposed for certain project situations, i.e. those projects having certain characteristics. But, project characteristics are not upfront in practice, they just emerge or are recognized in the course of a project. In case the method's preferred characteristics do not fit the actual project characteristics, then the method does not function properly in an actual project situation.

In other words, the existing methods do not cope with the uniqueness of each project situation, this is especially true for WfMSs implementation projects. So, one can expect the mismatches between the proposed method and the project situation at hand. Clearly, we need to tailor different aspects of method to better suit the project situation at hand. The model we propose for tailoring of a WADM is based on the notion of situation-specific modelling. The picture below illuminates the meaning of situation-specific modelling that is an essential part of our theoretical basis and shows key constructs underpinning this notion. The rationale behind these constructs can be justified theoretically in the literature of DSS (see, for instance, (Mintzberg, Raisinghani, and Theoret, 1976), but needs more space. So, in this section we will briefly explain these constructs and go into detail about them in the next section.

The agent is anyone who is involved or is in charge of approach determination. His/her task is to decide on an appropriate implementation approach and related fragments. He takes into account the contextual aspect of the situation and the method base. Contextual

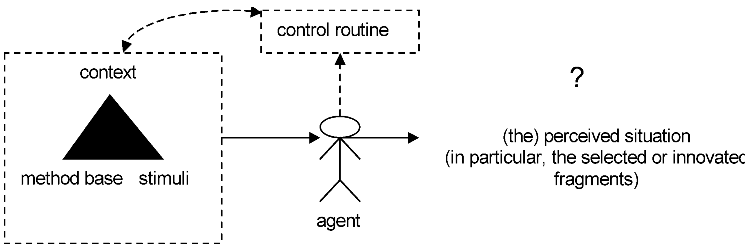

control routine
context
method base
stimuli
agent
?
(the) perceived situation
(in particular, the selected or innovatec
fragments)

Figure 1. A proposed model based on the idea of situation-specific modelling.

aspect can be operationalized using project characteristics, constraints, contingencies and other factors. A method base includes the formal or structured fragments (prescribed in method resources) and informal fragments (in some forms in our head). A stimulus is something that drives the decision (our way of thinking) leading to the selected fragments. It can be an intention, a success or risk factor, motivation, a constraint, etc. A control routine describes the interplay between these constructs and plans (routes) our decisions.

For the purpose of this paper, we focus on only the relationships between the context at hand and the method base, and the way to reach an appropriate implementation approach to WfMSs implementation. So, we shall first explain these two constructs.

A method base is a conceptual thing, a kind of repository embodies coherent parts of a method, called method fragment. Namely, the term method fragment refers to a description of a WADM, or any coherent part thereof. A fragment can be either part of prescribed method or be invented-on the fly. Fragments can be principles, fundamental concepts, products to be delivered, activities needing to be performed, job aids – techniques, tools, hints, tips - to be used, etc. Some of them are essential in a system development approach; others are primarily used in the execution of an ISD approach.

The term context refers to a collection of relevant conditions and surrounding influences that make a project situation unique and comprehensible (Hasher and Zacks, 1984). Different schools of thoughts use different contextual factors to understand the context and its implication on an implementation approach that suits the context. For instance, the method engineering perspective provides a lengthy list but it is apparent that social and organizational issues are not the focus of attention. The socio-organisational perspective, however, does put more emphasis on social and organisational elements of the context. Our aim is to propose those contextual factors that characterize WFMSs implementation projects. These characteristics will be used as determinants for the selection of appropriate strategic fragments of a WADM.

3. ELABORATIONS OF KEY CONSTRUCTS IN THE MODEL

3.1. The Way to Contextualize a WfMSs Project Situation

The characteristics to be mentioned here are a starting point, not an end point, for a better understanding of the project context at hand. Due to space limitation, in this paper we do not include those characteristics that are related to a project organization such as management commitment, knowledge and experience with tools, techniques, resistance and conflict, shortage of means, human resources. Rather we focus on those characteristics, which appear to be essential for WfMs.

(van der Aalst and Hee, 2002) distinguish unstructured, information centric approaches (Computer-Supported Cooperative Work – CSCW) and structured process centric ones (production workflows). (Georgakopoulos and Hornick, 1995) adopt the notion of task structure, complexity from (Korzeniowski, 1993) as a way to classify WfMSs. Later on, (Stohr and Zhao, 2001) use similar notions to discuss flexibility spectrum for WFMSs.

As indicated before, (Cardoso et al., 2003) used the three dimensions to characterize a WfMS and compare it with an ERP. The proposed characteristics are valuable, but appear to be too generic. For instance, a domain characteristic is mentioned as one of the distin-

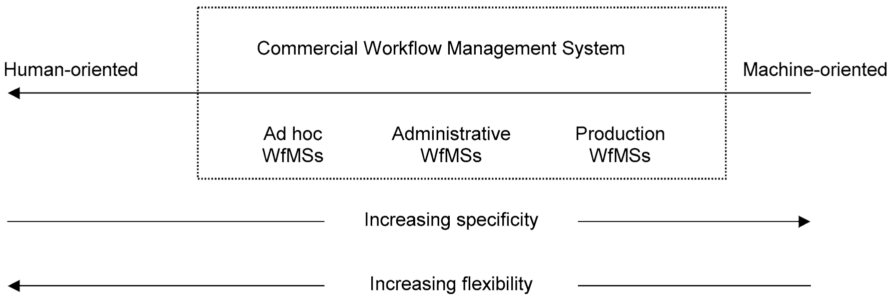


Figure 2. The flexibility spectrum for WfMSs (Stohr and Zhao, 2001 p. 287).

guishing factor for WFMSs compare to ERP. They claim that WfMSs are used in usually ad-hoc and dynamic domains whereas ERP system is more appropriate for static domains. The term domain refers to a specific industry or a business function such as administrative processes for school administration.

We believe that the essence of characterization for WfMSs can be related to the following factors: stability, formality, complexity of workflow to be studied. These characteristics can be measured in terms of high, medium and low. Stability refers to an extent to which the goals, procedures, roles will not change over time enabling a stable specification of the requirements to develop a WfMSs. By formality we mean that to what extent the existing or proposed rules, procedures are clear, agreed and documented by project participants, especially by the users, modelers and system developers. Complexity refers to a level of complexity in terms of number of activities to be included in the model, a number of different parties involved in a workflow, a degree of dependency with other systems.

3.2. Determining an Appropriate Implementation Approach Using Method Fragments

We identify four fragment types and relate them to the aspects of a method. These types are as follows: strategic fragments referring the way of thinking, product fragments referring the way of modelling, process fragments referring the way of working, and technical fragments referring the way of controlling and supporting in the implementation of a WfMS. Two levels are considered for method tailoring (see Figure 3): the scenario level at which essential aspects of an approach are formulated and the route map level at which tactical and operational fragments are configured.

The focus on this paper is on tailoring strategic fragments and related methodical aspects, including the way of thinking. For other fragments, especially process and product fragments, van der Aalst and van Hee proposed a method, named IPSD (Interactive, Process-oriented system development). Their proposed method uses some features or aspects of typical Business Process Reengineering methods and agile ISD methods. These features are, for instance, user involvement, prototyping, incremental and evolutionary development, a cyclical process. With this method one can find a number phases, corresponding deliverables, activities etc. It partially includes a number of techniques, but lacks management controlling related methodical aspects.

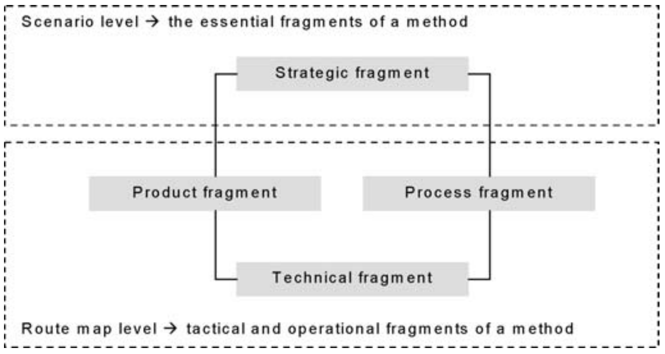

Scenario level → the essential fragments of a method
Strategic fragment
Product fragment
Process fragment
Technical fragment
Route map level → tactical and operational fragments of a method

Figure 3. The fragments as building blocks for a method.

Another WADM is proposed by (Weske et al., 2001). It is similar to IPSD, but the method is proposed especially in order to counteract the problems identified in real projects. These two studies will provide a basis for an initial method base and its fragments specific to WADM. Besides these studies, we use our insights gained in a large-scale IT organisation in which several packaged WfMS have been implemented. We conducted in-depth interviews with a project manager who was responsible for the use of an agile method for the implementation of well-known WfMS in one of the leading financial institutes in Europe. Consequently, we were investigating which parts of method were perceived as challenging or difficult to use such a method in the implementation of WfMSs. Recently, (van der Aalst, Weske, and Wirtz, 2003) and his colleagues studied strengths and weaknesses of modelling approaches by taking into three kinds of factors: with respect to coverage of perspectives, support for flexibility and analysis issue. The modelling fragments are the models developed using well-known techniques including UML, WF-nets, Workflow Evolution.

We consider the principles and the essential techniques of a method as essential fragments to determine an appropriate implementation approach. In this case, several decisions need to be made concerning with the suitability of each principles and essential techniques. So, we propose the following fragments and corresponding decision variables as a starting point to determine an appropriateness of the method fragments to be enacted in a WfMS implementation. We further classify them into several aspects such as modelling aspect, design-development aspect and user engagement aspect. Consequently, we have the following:

Strategic fragments which are related to modelling aspects:

- Modelling scope (the boundary of target system and dimensions): the extent to which the approach considers tracing of several perspectives such as functional, information, process, organisational and operation (see (Curtis, Kellner, and Over, 1992) and (Jablonski and Bussler, 1996) as mentioned in (van der Aalst et al., 2003 p. 2) and (Stohr and Zhao, 2001 p. 285)).
- Approach orientation (the orientation of problem-solving system): (problem or solution orientedness) and social aspect (technical-administrative or social-organisational) (see (Offenbeek and Koopman, 1996)).

- The analysis starting point (knowledge acquisition strategy): current situation or future situation (direct acceptance of user requirements, actual system is a starting point, possibly from the point of view of the old system, determining information requirements from scratch (starting from perspective of the object system).
- Re-use (design) strategy: using a reference (architecture) model, or a new architecture or combination of both.

Strategic fragments which are related to design-development aspects:

- Dividing strategy: increment strategy (how to partition the business process into increments containing workflows).
- Realization strategy: the way to realize a number of increments: at once (no subsystem), concurrently (parallel), overlapping, consecutively (subsystems are developed one after another, incremental).
- Development strategy: conversion strategy: linear, overlapping, throw-away, keep-it prototyping, evolutionary or reverse engineering.
- Delivery strategy: big bang (at-once), incremental, evolutionary.

Strategic fragments which are related to 'stakeholder engagement' aspects:

- Validation strategy: (immediate acceptance, definition of norms and test cases, by means of which assessment takes place whether the chosen solutions meet the requirements; prototyping; validation by usage).
- Engagement strategy: apply interaction model of van Offenbeek and in particular apply for the user engagement (degree of user involvement and responsibility).

The mechanism that relates the dominant characteristics of the context at hand to the choice of an appropriate option for each strategic fragment is the heart of the tailoring. Such a mechanism requires a tailoring rationale that reflects the relationships between the dominant characteristics and the fragments. For the sake of simplicity we shall pinpoint how the incremental-iterative strategy can be determined for a WfMS with an illustrative case.

4. ILLUSTRATION OF THE MODEL USING A CASE PROJECT

The sample case was extracted from an interpretive field study, described in detail elsewhere (Aydin et al., 2004). The sample project was about the implementation of a well-known WfMS for the fulfilment process in one of the leading financial institutes in the world. The fulfilment process is triggered by a request from the teller via different channels, including customer contact centre through phone, e-mail or face-to-face contact (for a visual representation of the system see Figure 4).

The process covers many business transactions, which are shared by different organizational units and include specific business services such as payment, money transfer. These services are partly automated and integrated with back-end systems, which were typical mainframe systems, for instance Mainframe/ OS390. This process was subject to improvements in terms of attaining better-streamlined business transactions. In fact, the organization had first used CSCW (Lotus Notes) for this process, but it turned out that due

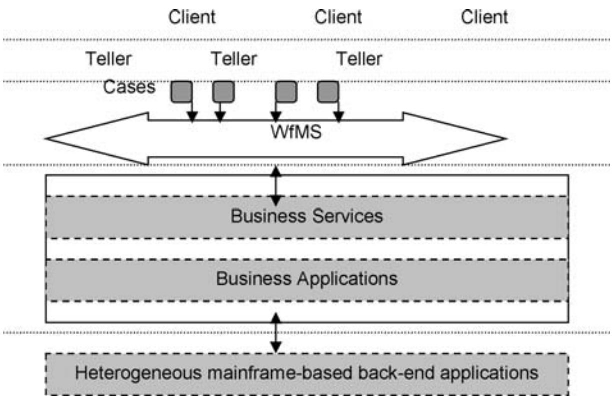


Figure 4. A workflow management system that integrates the front- and back- end systems of the company.

to the nature of the process they needed a WfMS. So, a well-known WfMS was chosen before the actual start of the project.

The (the development and target) infrastructure, modelling tools (typical object-oriented modelling tool used for all IT projects), were new to the project participants, including the project manager. In addition, the existing (business, process, information and technology) architecture was either incomplete or not specific to that process.

The project manager was supposed to use an agile method for this project. The project manager was not very knowledgeable about the method, and this type of projects. The organization assigned an expert to help this project manager in order to ensure method adherence in this project. Namely, the expert worked with the project manager to determine an appropriate approach for the project at hand. The idea was that if they determine an appropriate implementation approach and formulate some countermeasures against possible problems encountered in the project then the risk of failures would be reduced.

Our investigation in this project situation was limited to early stages of project and we conducted in-depth interviews with the project manager, the expert (a method engineer). First, we realized that they used a kind of checklist to be aware of possible implementation problems in case an agile method is to be used. After the thorough analysis of the project situation at hand, the following characteristics were identified as dominant:

- Introducing new technology.
- High dependency on external party; the supplier.
- A large number of different parties to involve and manage.
- Focus on connecting to current architecture instead of focus on new architecture.
- The nature of workflow partly is subject to change and partly formally defined and very complex.
- Focus on user guidance instead of focus on developer guidance.

Then the following issues were addressed in their project analysis:

- Communication with large number of different parties.
- Finding the right representatives of parties involved.

- To set up decision making and escalation path.
- Focus too much on technical implementation instead of organizational implementation.
- Substantial customization needed to meet specific requirements, losing out-of-the box characteristics.

They also specified challenging principles of the method. They were as follows:

Table 1. Perceived problems concerning the realization of principles of an agile method in the implementation of a WfMS

Agile method's principles	Always problem and difficult to manage	Sometimes and manageable	No problem
High user involvement	X		
Empowered team	X		
Focus on delivery of the product		X	
Fitness for business purpose		X	
Iterative and incremental development	X		
All changes are reversible			X
Requirements at a high-level			X
Testing throughout the lifecycle	X		
Co-operative approach		X	

For the focus of this paper, we shall show how the principle of iterative and incremental development was modified and realized in this project. The method basically recommends that 'many-increments with iterations' is an ideal development strategy for an agile system development'. This strategy suggests that a solution can be split into components that are based on prioritized requirements (Slooten and Hodes, 1996). More formally, an increment is a part of the WFMS that is delivered to, and used by, a user before the total system is operational. Having iterations means that some stages and corresponding activities need to be repeated through incorporating continuous feedback from the user. Such an iterative aspect of a development strategy contributes to the achievement of fitness for business purpose, which is another principle of the method.

The expert noted that some workflows were partially stable, formal and clear enough to develop the system in a linear way. But for some workflows iterations were needed. So, they decided to use the hybrid development as an appropriate approach for the implementation of the WfMS. The term hybrid underscores the mixture of typical agile development strategy (iterative and incremental systems development) and a linear development strategy in such a project context. Now, we shall discuss the implication of this case and conclude the paper with some research issues.

5. CONCLUSIONS

In the previous section, we have just explicated how the dominant characteristics were influential on the development strategy, which is one of the examples of strategic fragments, which correspond a particular type of method fragments. Of course, for a complete approach determination one should take into account all relevant fragments. This paper does neither claim the completeness of fragments that constitute implementation approach and nor the completeness of dominant characteristics for the context in which a WFMS to be implemented. We intend to provide an initial list of dominant factors and relevant methodical aspects for WfMS. The proposed model and its key constructs are derived from the literature of decision support system and are considered as fundamentals of the notion of situation-specific modelling. This notion is rooted in the discipline of the social-psychology which serve as a promising theoretical ground for advancing in DSS.

The case in the previous section is used to illustrate how the model can be applied in real implementation of a WfMS along with an agile method. One of the research opportunities in this respect is to test the model in other WfMSs implementations with other agile methods. Namely, one should investigate the use of other agile methods in different WFMS implementation settings to further discern the role of the key constructs described in the model. For instance, one can find new dominant characteristics and/or new aspects of implementation approach. Another research opportunity is to formulate the relationships between dominant project characteristics and fragments of implementation approach. These relationships can be formulated in terms of heuristics and shared with other practitioners in a large-scale IT organization. The elicitation, formulization of heuristic knowledge is another research topic.

ACKNOWLEDGEMENT

I am grateful for the comments by Wil van der Aalst, Frank Harmsen, Kees van Slooten, and Robert Stegwee in this paper.

REFERENCES

Van der Aalst, W. M. P., Weske, M., and Wirtz, G., 2003, Advanced topics in workflow managements: Issues, requirements, and solutions, *Journal of Integrated Design and Process Science* **7**(3).

Van der Aalst, W. M. P., and Hee, K. v., 2002, *Worklfow management: Models, methods, and systems*, Cambridge, MIT press.

Aydin, M. N., Harmsen, F., van Slooten, K., Stegwee, R., 2004, Appropriate Delivery of Advice and Guidance on Method Adaptation, in: *AMCIS 2004, Proceedings of the Americas Conference on Information Systems*, New York, USA.

Aydin, M. N., 2004, Evolving Support Practices for Method Adaptation, in: *Proceedings of the IFIPDSS 2004*, Prato, Italia.

Baskerville, R., and Stage, J., 2001, Accommodating emergent work practices: Ethnographic choice of method fragments, in: *In realigning research and practice: The social and organizational perspectives*, B. Fitzgerald, N. Russo, and J. I. DeGross, eds., Kluwer Academic Publishers, Boston, pp. 11–27.

Cardoso, J., Bostrom, R. P., and Sheth, A., 2003, Workflow management systems vs, ERP systems: Differences, commonalities, and applications (September 10, 2003); http://citeseer.nj.nec.com/ 531507.html.

Curtis, B., Kellner, M. I., and Over, J., 1992, Process Modelling, *Communications of the ACM* **35**(9):75–90.

Georgakopoulos, D., and Hornick, M., 1995, An overview of workflow management: From process modelling to workflow aautomation infrastructure, *Distributed and Paralled Databases* **3**:119–153.

Iivari, J., Hirschheim, R., and Klein, H. K., 2001, A dynamic framework for classifying information systems development methodologies and approaches, *Journal of Management Information Systems* **17**(3):179–218.

Jablonski, S., and Bussler, C., 1996, *Workflow Management: Modelling Concepts, Architecture and Implementation*, International Thomson Computer Press.

Korzeniowski, P., 1993, Workflow software automates processes, *Software Magazin*.

Mintzberg, H., Raisinghani, D., and Theoret, A., 1976, The structure of "unstructured" decision processes, *Administrative Science Quarterly* **21**:246–275.

Offenbeek, M. A. G. v., and Koopman, P. L., 1996, Scenarios for system development: Matching context and strategy, *Behaviour & Information Technology* **15**(4):250–265.

Slooten, K. v., and Hodes, B., 1996, Characterizing IS development projects, in: *Method engineering: Principles of method construction and tool support*, S. Brinkkemper, K. Lyytinen and R. J. Welke, eds., Chapman & Hall, Atlanta, pp. 29–44.

Stohr, E. A., and Zhao, J. L., 2001, Workflow automation: Overview and research issues, *Information Systems Frontiers*, **3**(3):281–296.

Weske, M., Goesmann, T., Holten, R., and Striemer, R., 2001, Analysing, modelling and improving workflow application development processes, *Software Process Improvement and Practice* **6**:35–46.

ADDRESSING TACIT KNOWLEDGE IN ISD METHODOLOGIES

Fiona M. Murphy and Larry Stapleton*

1. INTRODUCTION

This paper identifies a gap in ISD methodologies regarding the exclusion of tacit user requirements in the development of information systems (IS). It recognises that this will lead to IS failure, since given that tacit requirements are not considered or incorporated, these systems will not address these types of requirements. In the mid 90's Clegg et al., (1997) argued that 80–90% of IT investments do not adhere to the performance objectives of the user. They identified a number reason for systems failure, one of them being the poor articulation of user requirements. Tacit knowledge is inarticulable (Wong and Radcliffe, 2000) and subjective (Baumard, 1999). Therefore requirements that result from tacit knowledge use are omitted from consideration in current ISD processes. This paper identifies three characteristics and five acquisition dimensions of tacit knowledge that have a significant impact upon the ISD process. Four well-known ISD methodologies are then critiqued in relation to these. This leads to a revised perspective on current ISD methodologies, which challenges the traditional view regarding the development of systems.

2. CHARACTERISTICS AND ACQUISITION DIMENSIONS OF TACIT KNOWLEDGE AND THEIR IMPACT ON ISD

This section defines tacit knowledge, and identifies how it can be transferred throughout the organisation. The major characteristics and acquisition dimensions of tacit knowledge and the impact they have on the development of IS are identified. Tacit knowledge is non-codifiable intelligence that is acquired through the informal take-up of learned behaviours and procedures (Howells, 1996). It is defined as "knowing more than we can tell", meaning that we know how to execute a certain task, but we cannot explain to another person (s) how to successfully perform that task (Polanyi, 1961, p. 93). Tacit knowledge

* ISOL Research Centre, Waterford Institute of Technology, Cork Road, Waterford, Republic of Ireland, lstapleton@wit.ie or fmmurphy@wit.ie.

is completely embodied in the individual; it is inherent in their practice and expertise. It can only be a transmitted through proficient execution and through a learning cycle that involves demonstrating and imitating (Fleck, 1997).

Stapleton (2001) states that many methodologies employed in ISD continue to focus on the creation of an information-processing *machine*. Social aspects and how they impact upon ISD are often ignored. Consequently, certain areas of knowledge have generally been omitted from consideration in current ISD methodologies. Tacit knowledge plays a major role in organisational information processing and decision-making that computer-based IS are designed to specifically support. Consequently, it follows that ISD must address tacit knowledge if it is to comprehensively support organisational information processing. However, tacit knowledge does not fit into the structured, rationalistic, and mechanistic viewpoints, which dominate many approaches to ISD. In the literature a number of important attributes for tacit knowledge were identified and a tacit knowledge framework was developed. These traits can be divided into three characteristics and five acquisition dimensions of tacit knowledge for this research. The three characteristics are implicitness, personal and non-measurability, while the dimensions for acquiring tacit knowledge are experiencing, interacting, showing-how and contextual learning which is sub-divided into social and cultural learning. Each of these are important aspects of organisational knowledge, which impact upon various aspects of ISD. This paper will now review each of these in turn, focusing upon their impact upon ISD.

2.1. Implicitness and its Impact on ISD

Throughout the literature tacit knowledge has been defined as a nebulous process, intuitive (Howells, 1996). It is highly idiosyncratic (Roberts, 2000), subsidiary awareness (Polanyi, 1961), is inarticulable and non-analytical (Wong and Radcliffe, 2000). It is deeply rooted in the ideals, values or emotions of individuals and is typically learned and transferred through experience (Nonaka and Konno, 1998). When developing an IS the analyst must take this type of know-how into consideration. It is from this knowledge that the user will 'see' the outcome of his actions before they are implemented.

2.2. Personal and its Impact on ISD

This has been defined as person-embodied (Howells, 1996; Wong and Radcliffe 2000), second nature, and intuitive (Wong and Radcliffe, 2000). Tacit knowledge is subjective and also includes good feeling [s] (Grant and Gregory, 1997). This form of knowledge is inbuilt into the individual's subconscious and may or may not be based on past experiences. The personal trait of tacit knowledge can be linked to Senge's 'Mental Models' (Polanyi, 1966). Senge describes mental models as "intuitions and 'gut instincts' that are difficult to communicate and share" (Senge and Fulmer, 1993, p. 22). Tacit knowledge is embodied in individuals within the organisation. It is this knowledge that the individual uses to make his decision. Senge (1990) states it is the individual's mental models that allow him to make sense of the world and also to determine what action should be taken. IS support the decision-making process, therefore the personal knowledge residing in each individual within the organisation who will be using the new IS needs to be considered. An emphasis upon the individual's personal knowledge is therefore critical if we are to address tacit knowledge in the ISD process.

2.3. Non-Measurability and its Impact on ISD

Tacit knowledge is difficult to quantify (Howells, 1996), is un-codifiable, and dynamic (Howells, 1996; Grant and Gregory, 1997). Also, it escapes observation and measurement, as it is elusive and indeterminate (Baumard, 1999). This poses a problem for the analyst conducting the requirements phase of ISD. Requirements capture is traditionally carried out at the start of new systems development. However, since tacit knowledge is elusive and does not remain static, therefore this paper suggests that the requirements phase be conducted throughout ISD so as to include tacit knowledge.

Implicitness, personal and non-measurable are the three characteristics of tacit knowledge and the impact they have upon the ISD process. The following are the five acquisition dimensions of tacit knowledge identified from the literature and the impact each has on ISD.

2.4. Interacting and its Impact on ISD

This acquisition dimension is described as an 'in the corridor' style of learning that is codified in local practices and communities (Wong and Radcliffe, 2000). It is culture bound; the knowledge is developed interactively through socialisation between individual co-workers (Grant and Gregory, 1997; Wong and Radcliffe, 2000). User interaction can take the form of gossip and norms i.e. this is the way we do things here. User interaction allows for knowledge to be shared within the enterprise on an informal basis. Schein (2001) has stated that it is the basic assumptions of the organisation that define what the worker pays attention to, the meanings of different things, emotional reactions to situations, and the actions to take in various types of situations. IS supports the decision-making needs of the organisation. According to Turban et al., (1999) the purpose of an IS is to provide solutions to problems within the organisation. However, according to Krackhardt and Hanson (1993), it is the informal networks of the organisation that are taken advantage of when unexpected problems arise within the organisation. Consequently to develop information systems without taking this form of interaction into account will undermine the development of the new IS. It is through interaction that people get a better understanding of how things are performed within the organisation.

2.5. Experiencing and its Impact on ISD

This acquisition dimension of tacit knowledge is identified as accumulative knowledge, derived from experience (Roberts, 2000; Howells, 1996). This tacit know-how is gained from experiences through trial and error (Howells, 1996; Roberts, 2000) and therefore it cannot be learned from books. It is only through experience that the user will be able to decide on the best course of action to pursue. Polanyi states "[experiential knowledge] guides integration of clues to discoveries" (Polanyi, 1966, p. 2). In ISD the user and analyst try to identify the best course of action to follow based on successful past experiences or failures. Experiential knowledge is highly skilled and cannot be learned from reading user manuals or books (Baumard, 1999). It is from past experiences that the user will be able to identify the right choice of action to pursue in relation to the decision-making process that IS support. Therefore when investigating requirements for ISD the analyst must include this area of knowledge into the methodology being used for developing the information system.

2.6. Contextual Learning and its Impact on ISD

Contextual learning is the knowledge that resides in individuals about how they perceive themselves in their society or organisational culture. It allows us to make sense of the world (Polanyi, 1962). It is transferred through informal local practices amongst co-workers and does not reside individually amongst workers but at an organisational level (Howells, 1996) – it is specific only to that particular organisation (Cohen and Levinthal, 1990). Contextual learning can be further divided into social and cultural learning.

In the literature **social learning** has been defined as an informal way of learning through direct contact with co-workers (Roberts, 2000; Howells, 1996). It is through a connection with society that humans develop common interests, traditions, and beliefs etc that govern their role and place in their environment. Social learning governs the assumptions users' make about themselves, others and their environment. It is from this that user's modify their theories-in-use, i.e. the theory that governs their actions. In the literature **cultural learning** has been defined as an informal learning of behaviours within the organisation through socialisation with workmates (Roberts, 2000; Baumard, 1999). This tacit knowledge form has been described as being critical knowledge that is firm specific (Cohen and Levinthal, 1990). It includes the language that is used, the customs and traditions that evolve, and the rituals the workers employ in a wide variety of situations (Schein, 1992).

Failure to identify the contextual learning dimension of tacit knowledge would clearly weaken IS Development, for ISD to be successful, how the workers view themselves, make sense of their society (Polanyi, 1962), and how they learn this knowledge must be identified within, for example the requirements phase of ISD (Checkland, 1999).

2.7. Showing-How and its Impact on ISD

Showing-how has been described as learning by watching, learning by doing and learning by using (Grant and Gregory, 1997; Howells, 1996). This acquisition dimension allows tacit knowledge to be transferred through demonstration, imitation and practice (Roberts, 2000; Polanyi, 1961). Schön (1987; 1983) suggests that through observations and skilful execution of performances and the individual's reflection on what has taken place, a description of tacit knowledge required to carry out the task can be made. This then allows for the tacit knowledge to be transferred from one person to the next. But these descriptions or constructions are attempts to make explicit the "intelligence that begins by being tacit and spontaneous. . . descriptions are conjectures that need to be tested against observation of their originals. . . " (Schön, 1987, p. 25). By comparing their description with an original the individual will learn how to perform the task at hand. Therefore this area of tacit knowledge allows know-how that is inexpressible to be passed between workers through guided imitation of an action (Polanyi, 1961). It is through illustration, replication and practice that the show-how feature of tacit knowledge is made available to others. Therefore show-how is an important acquisition dimension of tacit knowledge that needs to be identified within ISD, by enabling people to learn how to use the new systems in innovative ways, perform old tasks effectively and so on. At present knowledge that can only be transmitted through this medium is being omitted from the development of IS.

In conclusion there are three characteristics and five acquisition dimensions of tacit knowledge that have been identified within the literature. It is apparent that these facets of tacit knowledge are not detached from each other. It is also visible that each impacts upon ISD. Failure to include these tacit knowledge forms when developing an IS would result in a system that neglects to meet all the knowledge support requirements of the organisation. Furthermore not to meet all these needs suggests that ISD does not (or cannot in its present form) deliver satisfactory effective systems. However, this framework needs more empirical work in order to establish and refine it as a unified model of tacit knowledge for ISD, this is for future work. The next section gives a brief account of four current ISD methodologies and the extent to which each addresses the tacit knowledge characteristics and acquisition dimensions identified above.

3. OVERVIEW OF ISD METHODOLOGIES AND THE EXTENT TO WHICH EACH ISD METHOD ADDRESSES TACIT KNOWLEDGE

This section provides a brief overview of the different systems analysis and design methodologies that were critiqued in relation to the characteristics and acquisition dimensions of tacit knowledge that were discussed earlier. All of these issues impact upon each methodology in different ways, resulting in information systems that do not adhere to all the users requirements. As mentioned previously, tacit knowledge plays a major role in the information processing and decision-making needs of the organisation. Therefore since computer-based IS are designed to support these organisational needs, tacit knowledge must be included in the ISD process. Each methodology was selected as representative of a particular type of ISD approach.

3.1. UML / UP

Larman (1998) states, object-orientated (OO) analysis and design emphasizes a problem domain and logical solution from the object's perspective. The OO modelling approach that this paper investigates is the Unified Modelling Language / Unified Process (UML / UP) approach. Although barely a methodology, UML / UP was chosen for review as it is widely used in industry for systems development. UP is a software development process that is component based and uses the UML modelling standard. UML is a language for specifying, visualizing and constructing the artefacts of software systems. Larman (1998) states, the requirements phase of an OO methodology defines requirements that are unambiguous. Tacit knowledge is ambiguous and elusive; therefore to capture systems requirements with this approach is to exclude the tacit knowledge inherent to the user. Also requirements are identified through various electronic and paper documents. These are all explicit forms of knowledge, where the systems analyst can readily identify unambiguous requirements. Therefore the *implicitness* of tacit knowledge is largely ignored. Avison and Fitzgerald (1995) state with this approach there is a problem of users not really knowing what they require the new system to do. The system functions of the OO method describe what the system is supposed to do (Larman, 1998). These functions once identified should be listed in logical interrelated groupings. However the soft knowledge that is user-embedded is not taken into account, and the p*ersonal* knowledge of the user is omitted within this approach. With most ISD methodologies developers do not initially understand the current system and its environment. To overcome this, the system's users assist in the requirements capture for the new system. Bruegge and Dutoit (2000), state since the environment is dynamic, developers using this approach should encapsulate all assumptions they make about the environment at this stage. However, tacit knowledge is difficult to define and quantify; all assumptions about the environment cannot be described with use cases. Therefore the *non-measurability* of tacit knowledge is not taken into consideration.

Bruegge and Dutoit (2000) state the requirements phase describes the system and its interaction with the environment, including users, work processes and other systems. However, the literature (Paech, 1998; Bruegge and Dutoit, 2000) listed four levels of description for requirements elicitation and none identify the importance of the interaction between users in transferring knowledge. The *interactin*g dimension of tacit knowledge is not considered within this ISD approach. The system model is developed from the user's perception of reality (Bruegge and Dutoit, 2000). UML / UP does not take into account the experiential knowledge of the user in any explicit way. It identifies users, scenarios, use cases, relationships between use cases and non-functional requirements, nowhere does it incorporate techniques for exploring the experiences gained from prior successes or failures in working with IT. Explicit requirements are incorporated without a deep understanding of the experiences behind these requirements. Therefore *experiential* know-how is not identified in the requirements phase of this approach. This method identifies how systems developers view users, and from this develop detailed and concrete models that the future system will support. The developer thus may get a deeper understanding of how the user interacts with the current system, but neither social or cultural learning, which impacts upon both the ISD process and the effectiveness of the system after implementation is considered. The users' *contextual* learning, whether cultural or social, is omitted from this approach. The scenario in which the user operates is identified within the analysis and design stage of ISD, and these scenarios are concrete, focused, descriptions of a system feature. Showing-how involves the transfer of certain knowledge types through observation, imitation and practice. This formally inexpressible knowledge type cannot be described as a system scenario. Therefore UML / UP does not allow for the tacit knowledge dimension *showing-how* to be included within ISD. In summary, UML / UP fails to address in any coherent or comprehensive way, any of the major characteristics or dimensions of tacit knowledge as set out earlier in this paper.

3.2. SSADM

SSADM stands for Structured Systems Analysis and Design Methodology and is a data-driven methodology that places great emphasis on data modelling and the database. This methodology was chosen for review here, as it is one of the most widely used structured systems methodology. Avison and Fitzgerald (1995) identify the following major steps in SSADM: Feasibility Study, Requirements Analysis, Requirements Specification, Logical System Specification, and Physical design. SSADM is a highly structured methodology and it provides very detailed rules and guidelines for project development staff. SSADM investigates requirements through a number of different fact finding techniques (Avison and Shah, 1997; Goodland and Slater, 1995) that include interviewing people, studying existing documentation and problems, and analysing questionnaire responses. However, although the requirements may be expressed in narrative form, the analyst using SSADM expresses these in an independent implementation form, which will create a 'logical' requirements statement (Goodland and Slater, 1995). From this it can be deduced that only explicit data is given thought to with this ISD approach, since tacit knowledge cannot be easily expressed, therefore SSADM does not take into account the *implicitness* of tacit knowledge. The *personal* characteristic of tacit knowledge consists of intuition and hunches, but SSADM only considers the facts and information that can be articulated by the current users of the system. This explicit knowledge is gathered through the performing of interviews, by analysing system documentation, and responses to questionnaires (Avison and Shah, 1997). This tacit knowledge dimension is not considered by SSADM. SSADM defines requirements that can be expressed in a "quantifiable and measurable way so that they can be tested" when the system is delivered to the user (Goodland and Slater, 1995, p. 77). With this approach the analyst uses questionnaires to identify user requirements (Avison and Shah, 1997). However, tacit knowledge is non-quantitative and resists codification and therefore this type of knowledge cannot be gathered by statistical measures. *Non-measurable* knowledge is not considered within SSADM.

In SSADM the majority of requirements are identified through one-to-one discussions between the users and the analyst. Avison and Shah (1997) state interviews allow for a detailed description of specific aspects of the situation under investigation, which provides an in-depth understanding of any business area under review. SSADM identifies requirements that can be measured and tested before implementation (Goodland and Slater, 1995). How the user interacts with his co-worker and fellow users to allow knowledge to be transferred is not considered. Therefore the *interacting* acquisition dimension of tacit knowledge is omitted. SSADM investigates current processing and data in the requirements analysis phase but fails to incorporate the knowledge gleaned from past experiences. Therefore the *experiential* know-how of the user is another dimension of tacit knowledge that is not considered in this methodology. The analyst is concerned with the *facts* about the organisation or the organisational section that is of concern within the ISD project (Avison and Shah, 1997). This includes when the information is required, and by whom, what it consists of, its purpose, and the activities that create and produce the data. The knowledge that the users have in relation to the social and cultural *context* in which they operate is not considered. Within the SSADM methodology, the analyst examines existing literature and documentation that might relate to the area under investigation. Only explicit data is considered, so the *showing-how* acquisition dimension of tacit knowledge is overlooked. In summary, like UML/UP, SSADM largely ignores tacit knowledge.

3.3. Soft Systems Methodology (SSM)

Soft Systems Methodology (SSM) was developed as an approach to tackling the messy problems managers have to deal with (Checkland and Holwell, 1993). The aim of this methodology is to create a learning cycle, which would result in an improvement in social concern within the organisation (von Bulow, 1989). In 1990 Checkland revised the SSM 7 Stage Model and presented the four-activities model of SSM. The four activities model begins with defining the problem including culturally and politically and ends with taking action in the situation to bring about improvement. Checkland (1981) states that unstructured problems should not be put into an explicit format but that they should be handled without a firm description; unstructured problems should be alleviated rather than solved. The 'comparison stage' of SSM allows problems that are ambiguous and cannot be clearly articulated be debated about and explored until a solution reveals itself. Implicit problems may be alleviated within this methodology, but the tacit requirements of the user are not identified. Therefore the *implicitness* of tacit knowledge is not fully taken into account within the SSM methodology. SSM does allow for debate and discussion between the system's stakeholders, but it does not explicitly identify the personal knowledge held by the current systems users. Therefore the *personal* characteristic of tacit knowledge is not considered comprehensively within SSM. Checkland (1981) states the systems approach is a part of the scientific tradition and it takes the assumption that the world contains structured wholes, which can maintain their identity under different conditions. Tacit knowledge, does not maintain its structure, as it is dynamic and ambiguous. Consequently, the *non-measurability* characteristic of tacit knowledge is not taken into account within SSM.

The *interacting* dimension is included to some extent within the SSM methodology, through the approaches it recommends for involving users. It is from this that the analyst can view the interaction between users that allows for the transfer of tacit knowledge. Avison and Fitzgerald (1995) maintain, within SSM organisational change results in a learning process when theory and practice meet and affect each other. This is important for the transferral of tacit requirements, as tacit knowledge is included in the expertise and performance of the user. This analysis of past experiences is carried out in the comparison of root definitions of the relevant system with the conceptual model. This stage provides the systems developer with the means to reconstruct the past sequence of events and compare them with what had happened post-implementation, which is carried out through debate and discussion. From this it can be deduced that the *experiential* dimension of tacit knowledge is considered within SSM. An observed social system consists of logical congregations of linked activities and relationships, such as those of a community (Checkland, 1981). However, with the four-activities model (Mode 2) the cultural stream of analysis is implied in stage 1 of the model, i.e. defining the problem situation (Checkland, 1999). From this it can be concluded that the *contextual learning* dimension of tacit knowledge is considered implicitly to some degree within SSM. The analyst conducting the requirements investigation becomes a participant in the relevant user group; within SSM the roles of the subject and the researcher can be switched (Checkland, 1981). This allows the researcher to become the practitioner and to understand the process of change. This is relevant to finding certain tacit requirements of the user (s), especially tacit knowledge forms that are expressed and conveyed through show-how. However, these requirements that are needed to develop the system are not built upon, they are only identified so as to allow the users to become researchers, so that they themselves may get a better understanding of the processes in place. Therefore, s*howing-how* is considered to some degree within SSM. This methodology considers the experiencing dimension of the user; and it also takes into account to varying degrees the other seven issues. However, tacit knowledge is not explicitly looked at within this method. In conclusion, Checkland's SSM Methodology considers all of the characteristics and dimensions of tacit knowledge to some degree.

3.4. ETHICS

From her work at the Tavistock Institute, Mumford developed a socio-technical approach called ETHICS, which is an acronym for Effective Technical and Human Implementation of Computer-based Systems. ETHICS consists of 6 stages, which start with a diagnosis of needs and conclude with reports for the company, which describe the theory and practice of the research undertaken (Mumford, 2000a). Mumford also developed QUICKethics as a method for requirements analysis for managers. QUICKethics provides a method for identifying the implicitness of tacit knowledge at top-level management – information requirements of managers are identified through face-to-face interviews and discussions, followed by group meetings. "Cross-validation of different archives can lead to the first inkling that there is something non-expressed [tacit] in an organisation's history" (Baumard, 1999, p. 90). It is through user debate and discussion that the *implicit* dimension of tacit knowledge can be identified and explored with this approach. According to Mumford (2000a) personal information is collected through individual face-to-face interviews and discussions with managers, followed by group meetings to identify priority management information needs. QUICKethics identifies management needs through 4 steps, which allows managers to self-reflect on their needs, provides an opportunity for self-identification and encourages group discussions and decisions. Therefore the *personal* aspect of tacit knowledge is an explicit requirement of the ETHICS methodology. This approach identifies system requirements through debate and discussion, and participation between the user and analyst. However tacit knowledge cannot be recorded or measured and escapes observation (Baumard, 1999). Therefore *non-measurability* is not considered in the ETHICS method.

According to Mumford (2001), people at any level in an organisation can play a major role in the development of successful work systems. Mumford (1993) states that instead of just being the system 'designer', the system analyst also takes on the tasks of being teacher, advisor and learner. By participating with the users in the design of the new system the analyst can identify the user interaction that enables tacit knowledge to be transferred. Therefore the *interaction* dimension of tacit knowledge is central within ETHICS. Mumford states that system users can design the system properly but will require training and help to do so. User involvement is essential to the ETHICS methodology as it is the users that have the skills of knowing about their own work and system, and have a stake in the design of the new system. Within ETHICS, Mumford recommends task variety, which involves giving a user one or more tasks to perform or by rotating several different people around a number of different tasks. This recycling of knowledge-workers may lead to the removal of tacit competencies for a particular process, although over time a level of cross-skills will develop. This approach, rather than focussing on the importance of deep *experiential* knowledge could be based on developing people who are 'jacks of all trades' but 'master of none'. ETHICS is a socio-technical approach in that it gives equal credence to social and technical issues. Within this methodology the context is identified through user participation, which is considered an important feature of the design process (Mumford, 2000a). The *contextual learning* dimension of the user is considered to some degree within this approach. As stated previously the designer's role is not the traditional role. The analyst together with the user develops the system. This participation on all sides allows for the relationship between users and the system to be highlighted. However nowhere in this approach is the *showing-how* dimension of tacit knowledge explicitly taken into account. This methodology considers the interaction between users. The other dimensions of the tacit knowledge framework are included to some degree, however not in great detail. Summarising, ETHICS addresses, to a greater or lesser degree, various characteristics and dimensions of tacit knowledge.

This section gave a broad overview of some of the current ISD methodologies that are covered by the literature and critiqued these methodologies in relation to the dimensions of tacit knowledge that were developed earlier. The next section summarises these findings in order to assess the extent to which major ISD approaches considers tacit forms of knowledge based upon a similar approach taken by Galliers (1992).

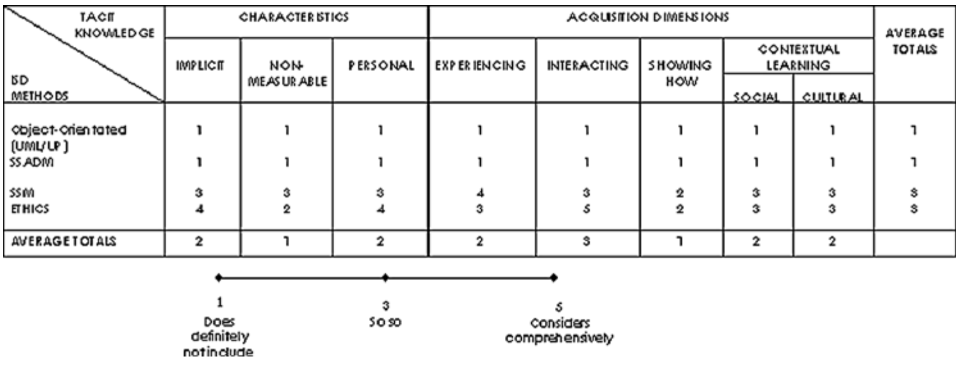
4. SUMMARY OF ANALYSIS

Figure 1 summarises the extent to which each methodology addresses various aspects of ISD. From Figure 1, the average totals of the hard methodologies averages at 1 meaning that these ISD approaches do not consider to any extent any form of tacit knowledge. The average totals for the soft approach to system design is 3, meaning that they are so-so in their identification of the characteristics and acquisition dimensions of tacit knowledge. The soft approach may not explicitly state tacit knowledge inclusion but tacit knowledge is indirectly included in the development of IS. Each tacit knowledge characteristic and acquisition dimension is given an overall average total in relation to the ISD methods analysed. The implicit and personal characteristics, and the contextual learning acquisition dimensions rate as 2, meaning that they are not considered within ISD. The experiencing and interacting dimensions of tacit knowledge are rated the highest with them averaging at 3, these acquisition dimensions are so-so in relation to ISD approaches – they are indirectly considered to some degree. The showing-how dimension and the non-measurable characteristic are the lowest ranking at 1, neither of these two forms of tacit knowledge are considered at all within ISD.

Overall there is an apparent gap between ISD methodologies being used and tacit knowledge. Tacit knowledge is important for the successful development of an information system, as it identifies the rationality behind the decision being made by the user. Therefore since information systems support the process of decision-making, this area of knowledge needs to be included in ISD.

5. CONCLUSION

This paper set out to explore the extent to which current ISD approaches address tacit knowledge. From the literature, three characteristics and five acquisition dimensions of



TACIT KNOWLEDGE / ISD METHODS	CHARACTERISTICS			ACQUISITION DIMENSIONS					AVERAGE TOTALS
	IMPLICIT	NON-MEASURABLE	PERSONAL	EXPERIENCING	INTERACTING	SHOWING HOW	CONTEXTUAL LEARNING		
							SOCIAL	CULTURAL	
Object-Orientated (UML/UP)	1	1	1	1	1	1	1	1	1
SSADM	1	1	1	1	1	1	1	1	1
SSM	3	3	3	4	3	2	3	3	3
ETHICS	4	2	4	3	5	2	3	3	3
AVERAGE TOTALS	2	1	2	2	3	1	2	2	

1
Does definitely not include
3
So so
5
Considers comprehensively

Figure 1. The ISD methods analysed in relation to the dimensions of tacit knowledge.

tacit knowledge were identified and the impact each has on the ISD process was highlighted. Next five current ISD approaches were critiqued in relation to the extent each addressed tacit knowledge. The result of this evaluation is that there are significant gaps in all ISD methodologies assessed, with soft systems approaches scoring significantly higher than hard systems approaches. ETHICS has the highest total for tacit knowledge inclusion. From the summary of analysis (Figure 1), it is evident that none of the ISD methodologies evaluated considers tacit knowledge important for IS development. The total averages for each characteristic and dimension individually is low, the lowest being the showing-how dimension and non-measurable characteristic, with the experiencing and interacting dimensions of tacit knowledge being the highest. From this it can be deduced that current IS technologies fail to take into account the tacit knowledge of the users. It is the user's tacit knowledge that enables an organisation to maintain its competitive advantage by enabling the user to generate better decisions. At present certain areas of knowledge have been omitted from consideration in current ISD approaches. The traditional view of ISD is that it creates information systems that support the decision-making and information processing needs of users. Tacit knowledge plays a large part in generating the decisions to be considered. This knowledge form is critical for ISD and should be taken into account for successful IS development and implementation. Therefore the tacit knowledge forms of the user needs to taken into account throughout the entire ISD process.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the comments and advice of the reviewers.

REFERENCES

Avison, D. E., and Fitzgerald, G., 1995, 2nd Ed., *Info Systems Development: Methodologies, Techniques & Tools*, McGraw-Hill, London.

Avison, D. E., and Shah, H. U., 1997, *The Info Systems Development Life Cycle*, McGraw-Hill, London.

Baumard, P., 1999, *Tacit Knowledge in Organisations*, Sage Publications, London.

Bruegge, B., and Dutoit, A. H., 2000, *Object-Orientated Software Engineering: Conquering Complex and Changing Systems*, Prentice Hall, New Jersey.

Checkland, P., and Holwell, S., 1993, Info mgt & organisational processes: an approach through soft systems methodology, *Journal of Info Systems* **3**:3.

Checkland, P., 1981; 1999, *Systems Thinking, Systems Practice*, Wiley, England.

Clegg, C., Axtell, C., Damodaran, L., Farbey, B., Hull, R., Lloyd-Jones, R., Nicholls, J., Sell, R., and Tomlinson, C., 1997, Info tech: a study of performance & the role of human & organisational factors, *Ergonomics* **40**:851.

Clegg, C. W., 2000, Socio-technical principles for system design, *Applied Ergonomics* **31**:463.

Cohen, W. M., and Levinthal, D. A., 1990, Absorptive capacity: a new perspective on learning & innovation, *Admin Science Quarterly* **35**:128.

Fleck, J., 1997, Contingent knowledge & tech development, *Tech Analysis & Strategic Mgt* **9**:383.

Galliers, R. D., 1992, Choosing information systems research approaches, in: *Info Systems Research*, R. D. Galliers, ed., Blackwell Scientific, Oxford.

Goodland, M., and Slater, C., 1995, *SSADM Version 4: A Practical Approach*, McGraw Hill, London.

Grant, E. B., and Gregory, M. J., 1997, Tacit knowledge, the life cycle & international manufacturing transfer, *Tech Analysis & Strategic Mgt* **9**:49.

Howells, J., 1996, Tacit knowledge, innovation & tech transfer, *Tech Analysis & Strategic Mgt* **8**:91.

Krackhardt, D., and Hanson, J. R., 1993, Informal networks: the company behind the chart, *Harvard Bus Review*, p. 104.

Larman, C., 1998, *Applying UML & Patterns: An Intro to Object-Orientated Analysis and Design*, Prentice Hall, New Jersey.

Mumford, E., 1993, *Designing Human Systems for Health Care: The ETHICS Method*, Eight Associations, Cheshire.

Mumford, E., 2000, A socio-technical approach to systems design, *Requirements Engineering* **5**:125.

Mumford, E., 2001, Advice for an action researcher, *Info Tech & People* **14**:12.

Nonaka, I., and Konno, N., 1998, The concept of "ba": building a foundation for knowledge creation, *California Mgt Review* **40**:40.

Paech, B., 1998, The four levels of use case description, *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations for Software Quality*, Pisa, Italy.

Polanyi, M., 1961, Knowing and being, *Mind N. S.* **70**:458.

Polanyi, M., 1962, Tacit knowing: it's bearing on some problems of philosophy, *Review of Modern Physics* **34**:601.

Polanyi, M., 1966, The logic of tacit inference, *Philosophy* **41**:1.

Roberts, J., 2000, From know-how to show-how? questioning the role of info & communication tech in knowledge transfer, *Tech Analysis & Strategic Mgt* **12**:429.

Schön, D. A., 1983, *The Reflective Practitioner*, Basic Books, New York.

Schön, D. A., 1987, *Educating the Reflective Practitioner*, Jossey-Bass, California.

Senge, P. M., 1990, *The Fifth Discipline: The Art & Practice of the Learning Organisation*, Century Business, Great Britain.

Senge, P. M., and Fulmer, R. M., 1993, Simulations, systems thinking & anticipatory learning, *The Journal of Mgt Development* **12**:21.

Schein, E. H., 1992, 2nd Ed., *Organisational Culture & Leadership*, Jossey-Bass, California.

Schein, E. H., 2001, Uncovering the levels of culture, in: *The Organisational Behaviour Reader*, J. S. Osland, D. A. Kolb, and I. M. Rubin, eds., Prentice Hall, New Jersey.

Stapleton, L., 2001, *Info Systems Development: An Empirical Study in Irish Manufacturing Companies*, Dissertation Submitted for the Degree of Doctor of Philosophy of the National University of Ireland, Dublin.

Turban, E., McLean, E., and Wetherbe, J., 1999, 2nd Ed., *Info Tech for Management*, Wiley, New York.

Von Bulow, I., 1989, The bounding of a problem situation & the concept of a system's boundary in soft systems methodology, *Journal of Applied Systems Analysis* **6**:90.

Wong, W. L. P., and Radcliffe, D. F., 2000, The tacit nature of design knowledge, *Tech Analysis & Strategic Mgt* **10**:247.

AUTHOR INDEX

INDEX