

**Christoph Bussler
Armin Haller et al. (Eds.)**

LNCS 3812

Business Process Management Workshops

**BPM 2005 International Workshops
BPI, BPD, ENEI, BPRM, WSCOBPM, BPS
Nancy, France, September 2005
Revised Selected Papers**

 **Springer**

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Christoph Bussler Armin Haller et al. (Eds.)

Business Process Management Workshops

BPM 2005 International Workshops

BPI, BPD, ENEI, BPRM, WSCOBPM, BPS

Nancy, France, September 5, 2005

Revised Selected Papers

Volume Editors

Christoph Bussler
Armin Haller
National University of Ireland, Galway
Digital Enterprise Research Institute (DERI)
IDA Business Park, Lower Dangan, Galway, Ireland
E-mail: {christoph.bussler,armin.haller}@deri.org

Library of Congress Control Number: 2006920907

CR Subject Classification (1998): H.3.5, H.4.1, H.5.3, K.4.3, K.4.4, K.6, J.1

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-32595-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-32595-6 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11678564 06/3142 5 4 3 2 1 0

Preface

Six parallel Business Process Management workshops were held on September 5th, 2005, in conjunction with the Third International Conference on Business Process Management (BPM 2005) in Nancy, France. This was the first time BPM had associated workshops, and the workshop program was a great success.

The topics of the workshops ranged from fundamental process modeling primitives to the recently emerged field of Web service choreography and orchestration, represented in the “Workshop on Business Processes and Services” and the “Workshop on Web Service Choreography and Orchestration;” a topic which intersects the research fields of business process management and Web services. Another strong focus was on business process design and business process intelligence, emerging areas that have gained increasing importance in supporting business process reengineering to derive superior process designs. These topics were covered in the respective workshops.

A widely discussed topic in several sessions was business process interoperability in the workshop on “Enterprise and Networked Enterprises Interoperability.” Finally, the last workshop in these proceedings called “Business Process Reference Models” aimed at discussing different views on reference models in order to come to a common understanding of the terms involved.

We would like to thank the workshop organizers for their efforts in the workshop preparation, the organization of the review process, the exciting workshop programs and their management onsite and for their cooperation in the post-publication process.

Further, we thank all the authors for their submissions to the workshops, the Program Committees for their hard work during a brief reviewing period, the invited speakers and presenters for their very interesting presentations and the audience for their interest, questions and discussions.

Christoph Bussler
Armin Haller

Organization

Workshop Organization Committee

Christoph Bussler

Workshop Chair

Digital Enterprise Research Institute (DERI), Ireland

Armin Haller

Publications Chair

Digital Enterprise Research Institute (DERI), Ireland

BPS 2005

Marlon Dumas, Queensland University of Technology, Australia

Schahram Dustdar, Vienna University of Technology, Austria

Frank Leymann, University of Stuttgart, Germany

WSCOBPM 2005

Christoph Bussler, Digital Enterprise Research Institute (DERI), National University of Ireland

Alistair Duke, Next Generation Web Research, British Telecommunications plc (BT)

Dumitru Roman, Digital Enterprise Research Institute (DERI), Leopold-Franzens Universität Innsbruck

Michael Stollberg, Digital Enterprise Research Institute (DERI), Leopold-Franzens Universität Innsbruck

BPI 2005

Malu Castellanos, Hewlett-Packard Labs, USA

Ton Weijters, U. of Eindhoven, The Netherlands

ENEI 2005

Boudjlida, Nacer, UHP Nancy 1, LORIA, France

Panetto, Hervé, UHP Nancy 1, CRAN, France

BPD 2005

Tom Davenport, School of Executive Education, Babson College

Hajo A. Reijers, Department of Technology Management, Eindhoven University
of Technology

Michael Rosemann, Faculty of Information Technology, Queensland University
of Technology

BPRM 2005

Ekkart Kindler, Paderborn, Germany

Markus Nüttgens, Hamburg, Germany

Table of Contents

Workshop on Business Processes and Services

Preface (BPS 2005)	1
Guided Interaction: A Language and Method for Incremental Revelation of Software Interfaces for Ad Hoc Interaction <i>Phillipa Oaks, Arthur H.M. ter Hofstede</i>	3
Towards P2P-Based Semantic Web Service Discovery with QoS Support <i>Le-Hung Vu, Manfred Hauswirth, Karl Aberer</i>	18
Global and Local QoS Guarantee in Web Service Selection <i>Danilo Ardagna, Barbara Pernici</i>	32
Web Service Discovery and Dynamic Invocation Based on UDDI/OWL-S <i>JianJun Yu, Gang Zhou</i>	47

Workshop on Web Service Choreography and Orchestration for Business Process Management

Preface (WSCO BPM 2005)	57
Standards for Web Service Choreography and Orchestration: Status and Perspectives <i>Alistair Barros, Marlon Dumas, Phillipa Oaks</i>	61
From Collaboration Models to BPEL Processes Through Service Models <i>Giorgio Bruno, Marcello La Rosa</i>	75
A Framework for Automated Negotiation of Service Level Agreements in Services Grids <i>André Ludwig, Peter Braun, Ryszard Kowalczyk, Bogdan Franczyk</i>	89
A Constrained Object Model for Configuration Based Workflow Composition <i>Patrick Albert, Laurent Henocque, Mathias Kleiner</i>	102

A High-Level Specification for Mediators (Virtual Providers)
Michael Altenhofen, Egon Börger, Jens Lemcke 116

WSMX Process Mediation Based on Choreographies
Emilia Cimpian, Adrian Mocan 130

An Abstract Machine Architecture for Web Service Based Business
Process Management
Roozbeh Farahbod, Uwe Glässer, Mona Vajihollahi 144

Workshop on Business Process Intelligence

Preface (BPI 2005) 159

Conformance Testing: Measuring the Fit and Appropriateness of Event
Logs and Process Models
A. Rozinat, W.M.P. van der Aalst 163

Mining Staff Assignment Rules from Event-Based Data
*Linh Thao Ly, Stefanie Rinderle, Peter Dadam,
Manfred Reichert* 177

Towards a Framework for the Agile Mining of Business Processes
*Barbara Weber, Manfred Reichert, Stefanie Rinderle,
Werner Wild* 191

Genetic Process Mining: A Basic Approach and Its Challenges
*A.K. Alves de Medeiros, A.J.M.M. Weijters,
W.M.P. van der Aalst* 203

On Web Services Workflow Mining
Robert Gombotz, Shahram Dustdar 216

**Workshop on Enterprise and Networked Enterprises
Interoperability**

Preface (ENEI 2005) 229

**Session: Enterprise Modelling Language for Enterprise
Interoperability**

Exchange of Business Process Models Using the POP* Meta-model
Reyes Grangel, Ricardo Chalmeta, Stefan Schuster, Iñaki Peña 233

UEML 1.0 and UEML 2.0: Benefits, Problems and Comparison <i>Giuseppe Berio</i>	245
Facilitating Interoperability: A Cross-Analysis of the Language UEML and the Standard ISO/DIS 19440 <i>Petia Wohed, Birger Andersson, Hervé Panetto</i>	257
Session: Networked Enterprises Interoperability	
Inter-agent Communications During the Virtual Enterprise Creation <i>Kamel Boukhelfa, Mahmoud Boufaïda</i>	269
Applying Enterprise Models to Design Cooperative Scientific Environments <i>Andrea Bosin, Nicoletta Dessì, Maria Grazia Fugini, Diego Liberati, Barbara Pes</i>	281
Architecture-Based Interoperability Evaluation in Evolutions of Networked Enterprises <i>Pin Chen</i>	293
Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM <i>Bernhard Selk, Sebastian Kloeckner, Antonia Albani</i>	305
Session: Interoperability Requirements and Models	
Interoperability in Service-Based Communities <i>Toni Ruokolainen, Lea Kutvonen</i>	317
CoSeRT: A Framework for Composing Service-Based Real-Time Applications <i>Marisol Garcia-Valls, Iria Estevez-Ayres, Pablo Basanta, Carlos Delgado-Kloos</i>	329
Supporting Business Experts in the Design of B2B Transactions Through Interactive Process Simulation <i>Michael Schmitt, Christophe Incoul, Eric Dubois</i>	342
Requirements to Improve the Synchronisation of Inter-enterprise Models <i>Cristina Campos, Reyes Grangel, Ricardo Chalmeta, Òscar Coltell</i>	353

On Automating Networked Enterprise Management
Ustun Yildiz, Olivier Perrin, Claude Godart 363

Session: Interoperability Applications and Case Studies

A Methodology for Process Evaluation and Activity Based Costing in Health Care Supply Chain
Michelle Chabrol, Julie Chauvet, Pierre Féniès, Michel Gourgard 375

Web Services Experiment Using TravelXML: Standard XML for Electronic Commerce in the Travel Industry
Michiko Oba, Norikazu Matsuyama, Kojiro Nakayama, Norihisa Komoda 385

Data Level Enterprise Applications Integration
Marko Vujasinovic, Zoran Marjanovic 390

Workshop on Business Process Design: Past, Present, Future

Preface (BPD 2005) 397

Design Processes for Sustainable Performances: A Model and a Method
Antonella Longo, Gianmario Motta 399

Designing Business Process Variants - Using the BAT Framework as a Pragmatic Lens
Mikael Lind, Göran Goldkuhl 408

BPR Implementation: A Decision-Making Strategy
Selma Limam Mansar, Hajo A. Reijers, Fouzia Ounnar 421

Applying Specialization to Petri Nets: Implications for Workflow Design
George M. Wyner, Jintae Lee 432

“Intelligent” Tools for Workflow Process Redesign: A Research Agenda
Mariska Netjes, Irene Vanderfeesten, Hajo A. Reijers 444

Risk Management in the BPM Lifecycle
Michael zur Muehlen, Danny Ting-Yi Ho 454

Workshop on Business Process Reference Models

Preface (BPRM 2005) 467

Business Process Reference Models: Survey and Classification <i>Peter Fettke, Peter Loos, Jörg Zwicker</i>	469
Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation <i>Oliver Thomas</i>	484
On the Syntax of Reference Model Configuration – Transforming the C-EPC into Lawful EPC Models <i>Jan Recker, Michael Rosemann, Wil van der Aalst, Jan Mendling</i>	497
Configurable Process Models as a Basis for Reference Modeling <i>W.M.P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, M.H. Jansen-Vullers</i>	512
Author Index	519

Preface (BPS 2005)

Service-oriented computing (SOC) is emerging as a promising paradigm for integrating software applications within and across organizational boundaries. In this paradigm, independently developed and operated applications are exposed as (Web) services which are then interconnected using a stack of Web-based standards including SOAP, WSDL, UDDI, WS-Security, etc. While the technology for developing basic services and interconnecting them on a point-to-point basis has attained a certain level of maturity and adoption, there are still many open challenges when it comes to managing interactions with complex services or managing interactions involving large numbers of services.

There exist strong links between business process management (BPM) and SOC. On the one hand, BPM may rely on SOC as a paradigm for managing resources (especially software ones), describing process steps, or capturing the interactions between a process and its environment. On the other hand, a service may serve as an entry point to an underlying business process, thereby inducing an inherent relation between the service model and the process model. Also, services may engage in interactions with other services in the context of collaborative business processes.

The First International Workshop on Business Processes and Services (BPS 2005) was organized with the aim of bringing together researchers and practitioners in the areas of BPM and SOC in order to further the fundamental understanding of the relations between business processes and services. The workshop's call for papers attracted nine submissions, of which the Program Committee selected four as full papers. In addition, two speakers presented their latest research and perspectives at the workshop: Wil van der Aalst on the topic of *interaction patterns* (organized jointly with the Workshop on Web Services Choreography and Orchestration for BPM), and Daniela Grigori on the topic of *service protocol adaptation*. Finally, a panel discussion on *intelligent processes and services for the adaptive enterprise* was held in conjunction with the Workshop on Business Process Intelligence. The panel was moderated by Malu Castellanos (HP Labs) and was composed of Wil van der Aalst (Eindhoven University of Technology), Boualem Benatallah (University of New South Wales), Frank Leymann (Stuttgart University), and Manfred Reichert (University of Twente).

The workshop was held during the preamble to the Third International Conference on Business Process Management (BPM 2005). We thank the officers and organizers of BPM 2005 for their support, as well as the members of the BPS 2005 Program Committee for their help in coming up with an exciting program.

September 2005

Marlon Dumas
Schahram Dustdar
Frank Leymann

Organization

Workshop Organizing Committee

Marlon Dumas, Queensland University of Technology, Australia
Schahram Dustdar, Vienna University of Technology, Austria
Frank Leymann, University of Stuttgart, Germany

Program Committee

Karim Baina, ENSIAS, Morocco
Alistair Barros, SAP Research, Australia
Malu Castellanos, HP Labs, USA
Sara Comai, Politecnico di Milano, Italy
David Edmond, Queensland University of Technology, Australia
Marie-Christine Fauvet, University of Grenoble, France
Jose Fiadeiro, University of Leicester, UK
Dimitrios Georgakopoulos, Telcordia, USA
Avidgor Gal, Technion, Israel
Volker Gruhn, University of Leipzig, Germany
Manfred Hauswirth, EPFL, Switzerland
Rania Khalaf, IBM Research, USA
Heiko Ludwig, IBM Research, USA
Jeff Nickerson, Stevens Institute of Technology, USA
Mike Papazoglou, Tilburg University, The Netherlands
Cesare Pautasso, ETH Zürich, Switzerland
Manfred Reichert, University of Twente, The Netherlands
Hajo Reijers, Eindhoven University of Technology, The Netherlands
Shazia Sadiq, University of Queensland, Australia
Jianwen Su, UCSB, USA
Samir Tata, INT Evry, France
Mathias Weske, University of Potsdam, Germany
Andreas Wombacher, University of Twente, The Netherlands
Jian Yang, Macquarie University, Australia

Guided Interaction: A Language and Method for Incremental Revelation of Software Interfaces for Ad Hoc Interaction

Phillipa Oaks and Arthur H.M. ter Hofstede

School of Information Systems - Faculty of Information Technology,
Queensland University of Technology,
GPO Box 2434, Brisbane, QLD 4001, Australia
{p.oaks, a.terhofstede}@qut.edu.au

Abstract. At present, most of the interest in web services is focussed on pre-planned B2B interaction. Clients interact with services using advance knowledge of the the data and sequence requirements of the service and pre-programmed calls to their interfaces. This type of interaction cannot be used for ad hoc interaction between services and their clients such as mobile devices moving in and around rich dynamic environments because they may not have the necessary knowledge in advance.

For unplanned ad hoc interaction an interaction mechanism is required that does not require clients to have advance knowledge of programmatic service interfaces and interaction sequences. The mechanism must ensure clients with different resources and diverse competencies can successfully interact with newly discovered services by providing assistance such as disambiguation of terminology, alternative types of inputs, and context sensitive error reporting when necessary.

This paper introduces a service interaction mechanism called *guided interaction*. Guided interaction is designed to enable clients without prior knowledge of programmatic interfaces to be assisted to a successful outcome. The mechanism is grounded in core computing primitives and based on a dialogue model. Guided interaction has two parts, the first part is a language for the exchange of information between services and their clients. The second part is a language for services to create interaction plans that allow them to gather the data they require from clients in a flexible way with the provision of assistance when necessary. An interpreter uses the plan to generate and interpret messages in the exchange language and to manage the path of the dialogue.

1 Introduction

Ad hoc is defined in Wordnet¹ as “unplanned” or “without apparent forethought or prompting or planning”. In an ad hoc interaction environment, a software client could find, using for example a discovery mechanism, a software service

¹ wordnet.princeton.edu/

that provides the capability [1] it (the client) requires. Depending on the mechanism used to find services, the client may have little or no knowledge about the inputs the service requires, the dependencies between the data inputs, or the order of invocation of its operations, or the type and formatting information associated with these inputs.

Automated ad hoc interaction between web-based applications is a desirable goal. Applications that can automatically locate and interact with software services without a priori knowledge of their interfaces will be able to achieve many tasks that are beyond human resources at present. The Internet and the world wide web have now made many different types of information accessible on demand. The numbers of providers and the types of information available mean that the current interaction mechanisms based on prior or discovered knowledge about software interfaces and pre-programmed one-on-one interactions with those interfaces will not scale up to provide the potential benefits of ubiquitous web accessibility. The increasingly large number of software services² means that their use needs to be automated to the largest possible extent in order to fully profit from the promise and potential of ubiquitous service access.

Ad hoc interaction is very different from the present situation where software applications interact in a planned manner via interfaces. A software interface provides a static view of the operation signatures provided by a software service. Operation signatures detail the names of the operations and their input and output data types.

A partial solution to enable ad hoc interaction is the idea of “standard interfaces”³. In this solution all providers of a particular function use the same interface⁴. The solution relies on a common agreement between heterogeneous service providers on the best interface for a particular operation in a particular domain.

The problem is that that standard interfaces limit ad hoc interaction in two ways. Firstly, it is the responsibility of the client application’s programmer to know or find out in advance how to call the operations supplied by the interface. Secondly, it locks providers into performing the task in a single way. If the same task can be performed with the same inputs but formatted in a different manner or with a different set of input data the provider must supply (non-standard) interface operations for each of these variants.

Another possible approach to ad hoc interaction is the use of techniques to obtain information about the operation signatures of software services at runtime such as dynamic CORBA and Java reflection. These techniques are used by client programs to gather information at runtime such as the names of oper-

² When accessible over Web-based standards, such software services are usually known as “Web services”.

³ www.learnxmlws.com/book/chapters/chapter11.htm

⁴ Many common software development environments help the developer in this task by reading the service interface and automatically generating the code necessary to interact with the service. However, these tools are only usable if the interface of the service is available when the client application is developed.

ations and the data types the operations expect as input and return as output. One of the difficulties of doing this on a large scale is that the information that is available at runtime via reflection is syntactic rather than semantic. Client programs would need to interpret this derived syntactic information and create semantically correct request objects or messages to send to the provider at runtime. This interpretation effort places a large computational burden on the client at runtime which is only possible because the developer has programmed the client software with the necessary discovery, interpretation and message generation logic.

In reality the number of service providers, the number of possible operations and the different contexts in which those operations will be performed means both of these approaches will not scale to solve the problems of ad hoc interaction in the heterogeneous web services environment. These approaches do not have the flexibility required for ad hoc interaction such as the ability to allow alternative or equivalent inputs, the ability to provide help in the form of disambiguation of terminology at runtime with context sensitive error reporting and some form of dialogue control for services and their clients.

The next section (2) introduces *Guided interaction* interaction as a scalable and flexible solution to the problems of ad hoc interaction between web services. Guided interaction allows the use of alternative inputs, the provision of help and context sensitive error reporting and dialogue control for both the services and their clients. Section 3 introduces a shared language for interaction. Section 4 introduces the means of generating and interpreting messages using the language and gives an illustrative example of how guided interaction can be used to direct ad hoc interactions between heterogeneous services. Section 5 reviews some related work and the paper concludes with section 6.

2 Guided Interaction

Guided interaction is an abstraction mechanism that hides the details of web service interfaces by providing level of indirection between clients and the web service itself. A *guide* is a type of mediator or facade which presents a “user friendly” interface to a back end service. Clients may be other services, software agents or people. A guide, representing a service provider can tell clients what the capabilities of the service are, and proactively seek the input data the service needs. Guided interaction is based on a shared language for the exchange of information and dialogue control.

The guide is responsible for asking client applications what they want the service to do and for requesting the input data the service needs. This is a reversal of the current paradigm which makes the client application responsible for knowing or finding out what the software service can do, either by interpreting static interfaces or using advanced techniques like reflection.

Guides could be implemented as an alternative means of accessing one or more of the capabilities delivered by a single provider. It is not necessary that these capabilities are implemented as web services described by Web Services

Description Language (WSDL)⁵ documents. For example, guided interaction could provide access to legacy applications without the overhead of conversion to WSDL. Guides could also be implemented by independent third party mediators or brokers to provide access to any publicly accessible interfaces.

Web services communicate by exchanging messages. The language provided by guided interaction introduced in the next section is based upon well understood computer interaction and data exchange mechanisms. However, an interaction language is not sufficient on its own, there must also be a way of interpreting, managing, and generating messages in the language [2]. A language to create plans that guide a dialogue manager to interpret and generate messages and to manage conversations is described in section 4.

3 Interaction Language

Guided interaction provides a means for two software entities to interact with one another without prior agreements in place regarding the exact syntax and semantics of the messages they can exchange with one another.

Clients do not have to know in advance a service's operation signatures or the order of operations before asking the provider to perform a capability. However, the client should have access to appropriate data before engaging with the service. For example, before engaging a bank service a client will have information relevant to banking such as account numbers and transaction amounts. Clients can request further information about specific items as they are guided through the data input process. Dynamic disambiguation of terminology is an important feature of guided interaction. A means of facilitating shared understanding of the syntax and semantics of the terms used by the service provider is essential for loosely coupled ad hoc interaction.

In free form natural language dialogues the interpretation of the type, purpose and content of messages is a complex process. There are several ways the complexity of the interpretation process can be reduced to mitigate the cognitive load on participants [3]. Each of these techniques is employed in guided interaction.

- Change from free form dialogue (where anyone can say anything) to directed dialogue where control is given to one of the participants. The controller determines the type and sequence of messages which can be sent. The other party cannot interrupt or give more information than requested.
- Explicitly state the intent of a message.
- Explicitly state the purpose or type of a message.
- Define the set of allowable responses for each message.

In addition to containing its *intent*, *performative* and application dependent *content* each message contains contextual information. The context dependent information includes conversation or process ids for correlation supplied by each

⁵ <http://www.w3.org/2002/ws/desc/>

party in the dialogue. Messages also contain a message ids and references to previous messages if appropriate. The identities of the sender and receiver of the message are also part of the contextual information.

A suggested message format is described in the message schema shown in figure 1. The actual structure of a message is flexible especially when implemented in XML, because in XML the elements can be accessed by name rather than position.

The other elements contained in the message, Intent, Performative and Content, are introduced in the next sections.

```

<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.guided.org"
  xmlns="http://www.guided.org">
  <xsd:annotation>
  <xsd:documentation>
    Message schema
  </xsd:documentation>
  </xsd:annotation>

  <xsd:simpleType name="ConversationId"><xsd:restriction base="xsd:anyURI"/></xsd:simpleType>
  <xsd:simpleType name="ProcessId"><xsd:restriction base="xsd:anyURI"/></xsd:simpleType>
  <xsd:simpleType name="MessageId"><xsd:restriction base="xsd:anyURI"/></xsd:simpleType>
  <xsd:simpleType name="ParticipantId"><xsd:restriction base="xsd:anyURI"/></xsd:simpleType>
  <xsd:simpleType name="Content"><xsd:restriction base="xsd:string"/></xsd:simpleType>
  <xsd:simpleType name="Intent"><xsd:restriction base="xsd:string">
    <xsd:enumeration value="Ask"/>
    <xsd:enumeration value="Tell"/></xsd:restriction></xsd:simpleType>
  <xsd:simpleType name="Performative"><xsd:restriction base="xsd:string">
    <xsd:enumeration value="Result"/>
    <xsd:enumeration value="Input"/>
    <xsd:enumeration value="Pick"/>
    <xsd:enumeration value="Select"/>
    <xsd:enumeration value="Help"/>
    <xsd:enumeration value="Error"/>
    <xsd:enumeration value="Refuse"/>
    <xsd:enumeration value="Status"/>
    <xsd:enumeration value="Pause"/>
    <xsd:enumeration value="Resume"/>
    <xsd:enumeration value="Restart"/>
    <xsd:enumeration value="Cancel"/></xsd:restriction></xsd:simpleType>

  <xsd:element name="Message">
  <xsd:complexType>
  <xsd:all>
    <xsd:element name="cid" type="ConversationId"/>
    <xsd:element name="mid" type="MessageId"/>
    <xsd:element name="mref" type="MessageId"/>
    <xsd:element name="sender" type="ParticipantId"/>
    <xsd:element name="receiver" type="ParticipantId"/>
    <xsd:element name="intent" type="Intent"/>
    <xsd:element name="perf" type="Performative"/>
    <xsd:element name="content" type="Content"/>
  </xsd:all>
  </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

Fig. 1. XML Schema for Messages

3.1 Intent

In computer interaction there are only two reasons for communicating: to ask questions and to tell answers. This is true for any software interface from command lines to windows and APIs. A client, be they human or software can only ask for information or actions, and tell information or the result of actions, and a provider (human or software) can only ask for information or actions and tell information or results. This is also true in linguistics where only three types of sentences are used in a technical context, interrogative, imperative and declarative. Interrogative and imperative sentences request either information or actions and declarative sentences tell information or the result of actions.

This leads to the definition of the *Intent* of a message, Ask or Tell. At this level it is not distinguished whether a request is for information or actions. A conversation is the exchange of Ask and Tell messages with two constraints:

- C 1 For each Ask message there is only one corresponding Tell message in response.
- C 2 A Tell message can only be sent in response to an Ask message.

An explicit statement of the Intent of a message allows both parties to understand their exact responsibilities in regard to the message. The use of only two intentions, mirrors the commonly used and well understood “Get” and “Set” operations of APIs and the “Get”, “Put” and “Post” operations in the REST architecture [4]. From an implementation perspective, both the client and provider require a very simple interface with only two operations, one for receiving Ask messages and the other for receiving Tell messages.

3.2 Performative

The purpose or type of the message is described by a *Performative*. The performative acts like a prompt, it indicates what kind of information is being sought or what kind of response is required. There are three well understood ways computer programs can get the information they require. They can ask the user (human or software) client to *input* one or more parameter values. They can ask the user to *pick* from a list of acceptable values or they can ask the user to *select* an operation from a menu, where each operation represents a capability the program can deliver.

A small set of performatives is defined based on software communication protocols and human computer dialogues. These performatives reflect core information gathering abstractions and dialogue management or control mechanisms.

- The **functionality** of a service is requested and delivered by the performatives *Result*, *Input*, *Pick*, *Select* and *Help*.
- The service **management** performative *Status*, provides information about the state of the dialogue at runtime.

- The **dialogue control** performatives are *Pause*, *Resume*, *Restart* and *Cancel*.
- The performatives *Error* and *Refuse* provide alternative responses.

The performative **Result** starts a conversation. It is sent by a client to the service to request the capability named in the content of the message. The intention behind the use of the word “Result” rather than “Do”, “Perform” or “Start” is, a dialogue is initiated by a client requesting the service to perform its advertised capability and to tell the result. So, a message containing *Ask*, *Result*, “*ConvertCurrency*” should be read as “Ask (for the) Result (of performing the) ConvertCurrency (capability)”. Although this may seem a little awkward it is necessary to ensure every performative has exactly the same name for the request and its response.

The performative **Input** requests input data for a single item (parameter) similar in function to a text box or form item. As it is unrealistic to assume two heterogeneous services will use exactly the same terminology and data structures, the focus is on describing what kind of content is required and the data type the service expects rather than the value of content (as done by VXML grammars).

Clients have to match the provider’s request for input with the data they hold. If the client does not understand the terms used in the request they can ask for **Help** from the guide. This means a service’s internal parameter definitions should include a set of alternative parameter names and data types which are *equivalent in this context*. This will allow the provider to offer alternative information to the client. A more sophisticated provider may offer the client pointers to other services to convert or reformat data into a usable form or it may make use of these services directly.

The performative **Pick** asks the client to select from a list of acceptable values such as (AUD, GBP, USD, ERD, NZD) for currency codes. Its function is similar to a list box in Windows.

Select offers a menu of choices representing the capabilities of the provider. Select can be used to offer more finely grained capabilities than those described in a service advertisement e.g. from “ConvertCurrency” to “ConvertUSDtoGBP” or “ConvertUSDtoAny”.

Select could also assist with the provision of a *generic* interface to a service provider. For example, if a client sends a message with *Ask Result “Menu”* to determine which capabilities a service can provide the provider could respond with an *Ask Select “...”* message containing a list of its capabilities.

Informally, the difference between Tell **Error** and **Refuse** is an error is returned if the service (or client) tried but *cannot* generate a correct response. Refuse means the service (or client) *will not* provide the required response such as in the case of privacy or security concerns. The distinction depends on the context of the participants.

The dialogue management performatives **Pause**, **Resume**, **Restart** and **Cancel** have the effect indicated by their names. Both participants (client and provider) can use these performatives. If, for example the provider sends an *Ask*

Pause message to the client, the client does whatever is necessary and sends a *Tell Pause* (or *Error* or *Refuse*) message in reply. When the provider is ready to resume, it sends an *Ask Resume* message to the client and the client responds with *Tell Resume* (or *Error* or *Refuse*).

The **Status** performative interrogates the state of the dialogue rather than the back end processing of the capability. Although this is an incomplete view of the service, it does enable reporting on whether the provider is waiting for input from the client, or processing the last response, or has passed the client input to the back end processor.

An important feature of this set of performatives is that they are *context* and *content* independent. This allows them to be used for the collection of input data for services in any domain.

There is one constraint that applies to all performatives except *Error* and *Refuse*. For each *Ask performative* message there are only three valid responses:

C 3 *Tell* (the same) *performative* or *Tell Error* or *Tell Refuse*.

The advantage of constraining the allowable responses is the dialogue is predictable and manageable.

There are three constraints on the types of messages that can be sent by providers or clients.

C 4 Service providers cannot *Ask* for a *Result* from clients, i.e. a service cannot ask its client to perform a capability while it is performing a capability for that client. It could however, in the context of a new conversation request a capability from another service, which may be its current client.

C 5 Providers and clients cannot *Ask* for an *Error* or *Refuse*. These performatives are reserved for *Telling* the unsuccessful results of *Ask* requests.

C 6 Clients cannot *Ask* for *Input*, *Pick* or *Select* from the service provider. If the client requires more information it can *Ask* for *Help*.

These six constraints combined with the specified values for *Intent* and *Performatives* give the complete semantics of the dialogue language.

3.3 Content

In so far as possible the dialogue language and interaction mechanism are independent of the actual content of messages. Thus guided interaction is a generic dialogue model that is not tied to specific types of services or domains. Implementations could use XML for the content of messages, with the structure described with an XML schema.

A simple structure comprising a parameter name and data type can be used for the input performative. Several of the other performatives lend themselves to further structuring. For example a help request could contain the parameter name and data type and the reason help is being sought e.g. not understood or not recognized.

4 Interaction Plans

Guided interaction does not specify the order of messages instead an *interaction plan*, created by the software service provider, details the inputs the software service requires to perform one of its capabilities. The guide uses the interaction plan to present a “user friendly” interface to the underlying software service.

The interaction plan allows the incremental revelation of the data elements a software service requires from its clients. These “incremental interfaces” are generated at runtime and they allow the software service provider to take responsibility for “asking” the client application for the information it needs in order for the software service to perform one of its capabilities.

Incremental interfaces ensure the path of the dialogue between the client and the provider is based on the availability of input information within the client. The provider can assist the client to match requests for input to the clients’ own internal data holdings by using parameter names (and alternative names if necessary) grounded in vocabularies represented for example as “ontologies”. In addition, the service provider can offer configuration, customization and navigation options to the client.

Inspired by existing “dialogue architectures”, the guide is structured around the performance of three functions: interpretation of user input, conversation management and generation of output for the user [5]. This section is concerned with the structures for the management of dialogues and the generation of messages.

An interaction plan represents a decomposition of the inputs required by the service and it is constructed using *instructions*. An interaction plan is an internal structure of the service provider, the client does not need, and does not have, access to the plan. An interaction plan is instantiated when the service provider receives and accepts an “Ask Result” message from a client asking for the provision of a capability. This message starts the conversation. Each instruction in an instantiated plan is identified by the clients conversation id and a local conversation id. This allows many concurrent instantiations of the same or different plans for the same or different client applications.

The primary purpose of an instruction is to describe the item a value should be collected for. In figure 2 this is shown as the Instruction *collects* an *Item* identified by an id. When an instruction is selected for processing, the dialogue manager instantiates the appropriate item and generates a request message to the client seeking an input value for the item. Processing of the instruction is paused until a response is received from the client.

On receipt of an input value for the item it is evaluated using the boolean *evaluate* function. If the evaluation returns true, the instruction identified by *next on success* is selected for processing and the input value is stored. If the evaluation fails the instruction identified by *next on failure* is selected for processing.

The intuition behind only two choices (of which instruction to select next) is that input from the client is either valid or invalid. In the case of valid input the plan can proceed to the next step. In the case of invalid input there are two alternatives, either the item is critical to the provision of the capability and invalid input means the capability cannot be performed so a fatal error must be

reported to the client. The other choice is to request input in a different way or for a different item.

There are two cases where alternative requests could be used. The first is that on failing to elicit a value for an item where the initial request used name and data type of the value required, the alternative request offers a list of acceptable values for the same item so the user can pick one of the values. The second alternative, is to switch to a different input set. For example, a service that can perform its capability with either a text file or a URL reference; if the client fails to return a text file, the alternative is to switch to a request for a URL.

In this way, the course of the conversation is driven by what the service needs to know, which in turn depends on the information the client has been able to supply for previous items. The conversation path is driven by the remaining data requirements rather than an external conversation protocol which defines which messages can be sent in which order.

Every message in a dialogue between two parties (the service and its client) could be recorded or logged. Logging allows participants to keep records of all messages sent and received during a guided interaction. The advantages of logging messages include non-repudiation and process improvement.

The use of instructions including three special instructions is detailed in the context of an example in the next section.

Example. To illustrate instructions and the evaluation and choice mechanisms a *ConvertCurrency* interaction plan is shown in figure 3. The figure shows a tree representation of the plan. In this representation instructions are shown as boxes with three layers. The first layer gives the instruction id. The second layer identifies either; the item for which a value is being collected, or the reason for an

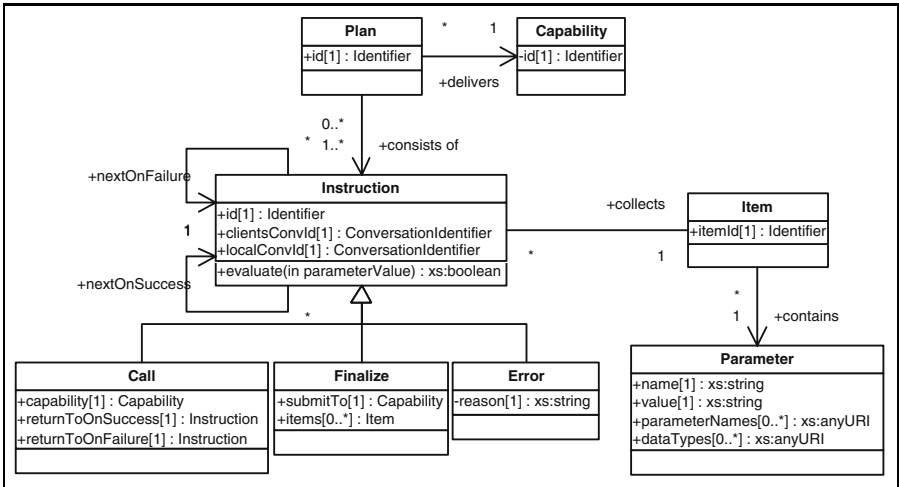


Fig. 2. Instruction schema

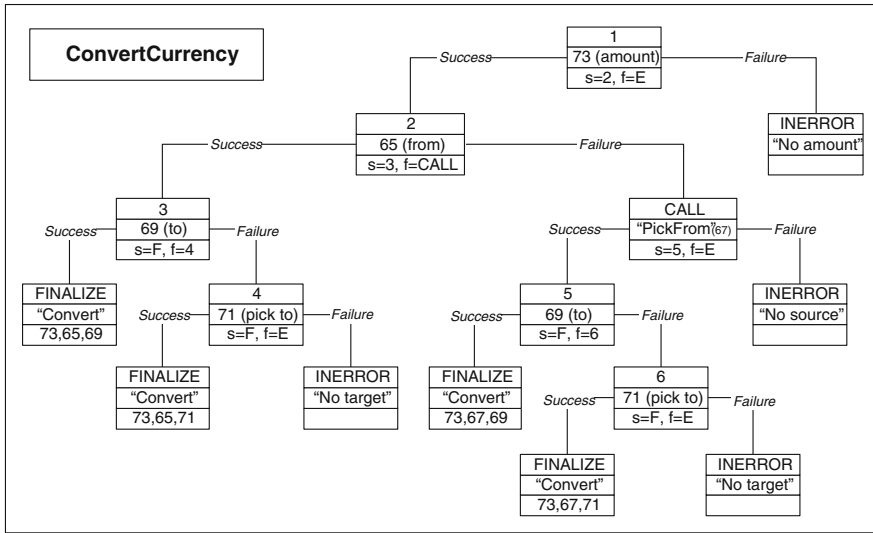


Fig. 3. A tree representation of a “ConvertCurrency” interaction plan

error, or the name of a sub-dialogue to call, or the name of the software service to submit the collected values to. The third layer describes the next instruction to use to when the evaluation of the user input is successful or when it fails.

The three special instructions: INERROR, FINALIZE and CALL are illustrated on the tree.

INERROR signals a fatal error from which the dialogue manager cannot recover. The text shown forms the basis of the error message sent to the client application before the conversation is terminated. There are several INERROR instructions shown on figure 3.

A FINALIZE instruction indicates that the collection of inputs is complete and the values collected for the listed items should be submitted to the specified software service. The successful instruction following instruction number 3 in the ConvertCurrency plan (figure 3) has a FINALIZE instruction where the values in item ids 73,65 and 69 are submitted to the “Convert” software service.

A CALL instruction indicates that a subdialogue (representing another capability) is to be initiated. The failure alternative after instruction 2 is a call to the “PickFrom” capability which collects item 67. The next instructions on success (5) or failure (INERROR) after the call returns are also specified with the CALL instruction.

The following paragraphs illustrate a complete path through the tree to the FINALIZE node at the bottom of the tree in figure 3 where the items 73, 67 and 71 are submitted to the “Convert” service.

The plan starts with a request for an amount to convert from the client (item 73). When this is successful instruction 2 is selected for processing. Instruction 2

asks for a source currency code from the client (item 65 - from). The client cannot supply a value so the dialogue manager calls a sub-dialogue to gather item (67 - pick from). The sub-dialogue uses item 67 to offer the client a list of currency codes to pick a value for the source currency. When the client picks a value it is evaluated and the called “pick from” capability terminates with a FINALIZE if the value is valid or INERROR if not.

The subdialogue returns successfully to instruction 5 (specified in the call). Instruction 5 requests the destination currency code (item 69 - to). As before if the client cannot tell a value the dialogue manager uses the alternative parameter (item 71 - pick to), which offers another pick list of currency codes to the client. A valid destination currency value returned from the client finishes the capability with the items 73, 67 and 71 submitted for processing.

Figure 3 shows that capability plans can be visualized as binary trees with every branch of the tree terminating at a leaf node with a FINALIZE or an INERROR instruction id. This means that sub-dialogues initiated with a CALL instruction also terminate with either a FINALIZE or INERROR instruction (rather than an explicit “return” instruction).

The reason for not using an explicit return instruction is that it permits the modeling of all dialogues in the same way, i.e. not differentiating between main and sub-dialogues. The advantage of this approach is that the provider can either call the “PickFrom” capability plan from within another plan (as a sub-dialogue) or expose the capability directly to the client.

To summarize, the items necessary for the performance of a capability are gathered according to the order of instructions specified in the capability plan. The plan has a binary tree structure with each leaf of the tree either a FINALIZE or INERROR instruction. The plan allows the definition of alternative input mechanisms for example request a single parameter value or offer a list for the client to pick a value. Alternative input sets can also be defined, for example the request could be for a flight number or the request could be for an airline and a destination. A plan can “call” another plan as a subdialogue. An extended example is given in the technical report at <http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2005/BPM-05-12.pdf>.

The client receiving a message from the service asking for input will check within its list of parameters for a matching name and data type. If the client can match the request with one of its parameter names, it will send a message containing the parameter’s value as the content. If it cannot match the request to a parameter name directly, it can send a request for help message to the provider, to get an alternative name (or data type). This process can be repeated until a match is found or the list of alternative names is exhausted. The failure to make a match means the client must return an error message.

The client does not have to know in advance the order of information required by the provider and it remains unaware of the internal processes used to select each data item or the use of sub-dialogues to collect information.

A message may ask the client to select from a list of capability ids representing the capabilities that the service can provide (i.e. a menu). In this case, it is

assumed the client has a list of *goals* that it needs to satisfy or a list of capabilities it requires. The client tries to make a match between the offered capabilities and its list of goals. There may be constraints or priorities associated with the goals which determine which one will be selected.

5 Related Work

Much of the work that has been done in the area of conversations for agents [6] and lately web services [7, 8] is directed at specifying the order of messages in a conversation. The argument for this is that if both sides are aware of the correct sequence of messages they can participate correctly in the conversation.

A problem with this approach is how to define, share and agree on which sequence or plan to use in an ad-hoc situation. Handling compliance checking, including digressions for help and errors handling has not been properly specified to date, and there is an added overhead of the effort required to agree on which protocol to use [9]. Furthermore, in an autonomous and heterogeneous environment it is not feasible to specify the behaviour of entities that are not within the ambit of your control.

Several of the large software companies have been looking at web service conversation mechanisms. IBM [10] describe conversations based on Conversation Policies (CP), which describe messages, sequencing and timing. They provide a good motivation for web service conversations including peer-to-peer, proactive, dynamic, loosely coupled interaction, that is not clearly realized by the specification. Later work has used CPs to manage perceived deficiencies in BPEL [11].

Web Services Conversation Language (WSCL) 1.0⁶ is a contribution from Hewlett Packard also described in [12]. WSCL models the conversation as the third party in an interaction. A conversation controller keeps state of conversation and changes state based on message types. There is a heavy reliance on message types being correctly identified and containing correct data, (or vice-versa). This means both parties must understand the message types and correct data before interacting, and missing or incorrect information will terminate the conversation. There is no mention of the problems introduced by alternative outputs, e.g. errors and the myriad of states these alternative paths can generate.

In [13] the idea of guiding clients using WSDL is proposed. In this proposal clients are told by the service which of its operations can be called next. The interactions are still performed in the context of the WSDL description, so no guidance as to the types of inputs the service requires can be given. The path of interaction is driven by the client deciding which operation to ask the service to perform. This means the client has to choose which capability it needs to elicit from the service at each stage.

Although SAP's Guided Procedures⁷ use a similar name they are not related to web service interaction. Guided procedures are pre-defined workflow templates

⁶ <http://www.w3.org/TR/wsc110/>

⁷ www.sapgenie.com/netweaver/CAF.htm

which can be extended with context dependent process modifications by users. These workflows are run in SAP's NetWeaver⁸ tool.

6 Conclusion

Guided interaction allows clients to request services (guides) to perform their advertised capability or function. It allows those services to request information from the client, such as single data items, a pick from a list of data items, or a selection from a menu of functions. Both clients and services can request further information from one another to assist disambiguation. In addition the interaction language provides the means for both parties to have a limited form of dialogue control, and insight into the state of the dialogue.

The service provider via the guide uses an internal plan to generate messages for the collection of input data. The plan is an internal structure and the client does not have access to the plan. The focus of guided interaction is not on specifying the order in which messages are sent, but on specifying the information that is (still) required before the service provider can execute a back end process. It is the nature and sequence of the data requirements specified in the plan and the ability of the client to provide the data that determine the path of a dialogue rather than a pre-defined conversation protocol.

There are several activities that guides can perform which are necessary in the context of ad hoc heterogeneous service interaction. They can provide help and disambiguation to mitigate data description mismatches. They can use alternative input sets for clients that operate with different types of information. They can use sub-dialogues to collect commonly recurring sets of data such as credit card details or perform common tasks such as logging in. They can provide context sensitive messages when fatal errors occur. In addition, guides can maintain multiple concurrent dialogues about the the same or different capabilities making it scalable in the web environment.

Ad hoc interaction cannot be both effective and efficient in the way that pre-programmed interactions with interfaces can. Guided interaction sacrifices some efficiencies to allow software clients' access to previously unknown services without the overhead of runtime computational complexity. Clients with prior knowledge of, or experience with, a service would not necessarily need to use guided interaction.

There are several other advantages to using this mechanism for software service to software application interaction. The first is that client applications do not have to know the service's interface in advance or interpret an interface such as a WSDL document on-the-fly. Another advantage is that clients can interact at runtime with *any* service that provides this mediation layer rather than being tied to specific service implementations via hard coded calls to interfaces. The advantage for the software service provider, is that they can offer flexible alternatives to clients and they are not constrained by static interface declarations.

⁸ www.sap.com/solutions/netweaver/index.epx

References

1. Oaks, P., ter Hofstede, A., Edmond, D.: Capabilities: Describing what services can do. In Orłowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J., eds.: First International Conference on Service Oriented Computing (ICSOC 2003), Proceedings, Trento, Italy, Springer-Verlag (2003) 1–16
2. Erbach, G.: Languages for the Annotation and Specification of Dialogues (2001) ESSL1 01 Presentation, available from: www.coli.uni-sb.de/~erbach/essl1i01/index.html, (12 June 2003).
3. Durfee, E.H.: Practically coordinating. *AI Magazine* (1999) 99–115
4. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, California (2000) Available from: www.ics.uci.edu/~fielding/pubs/dissertation/top.htm, (20 June 2004).
5. Blaylock, N., Allen, J., Ferguson, G.: Synchronization in an asynchronous agent-based architecture for dialogue systems. In: Proceedings of the 3rd SIGdial Workshop on Discourse and Dialog, Philadelphia, USA (2002)
6. Odell, J.J., Parunak, H.V.D., Bauer, B.: Representing Agent Interaction Protocols in UML. In Ciancarini, P., Wooldrige, M., eds.: *Agent-Oriented Software Engineering*. Springer-Verlag (2001) 121–140
7. Tari, Z., McKinlay, M., Malhotra, M.: An XML-based Conversation Protocol for Web Services. In: SAC’2003, Melbourne, USA, ACM (2003) 1179–1184
8. Yi, X., Kochut, K.J.: Process composition of web services with complex conversation protocols: a colored petri nets based approach. In: Design, Analysis, and Simulation of Distributed Systems, Proceedings, Arlington, USA (2004)
9. Paurobally, S., Cunningham, J.: Achieving common interaction protocols in open agent environments. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS02), Bologna, Italy (2002)
10. Hanson, J.E., Nandi, P., Kumaran, S.: Conversation support for Business Process Integration. In: Proceedings 6th IEEE International Enterprise Distributed Object Computing Conference (EDOC-2002), Lausanne, Switzerland, IEEE Press (2002) 65–74
11. Kumaran, S., Nandi, P.: Conversational Support for Web Services: The next stage of Web services abstraction (2002) Available from: www-106.ibm.com/developerworks/webservices/library/ws-conver/, (5 June 2003).
12. Frolund, S., Govindarajan, K.: cl: A Language for Formally Defining Web Service Interactions. Technical Report HPL-2003-208, HP Laboratories, Palo Alto, USA (2003)
13. Petrone, G.: Managing flexible interaction with web services (2003) Appeared in the Workshop on Web Services and Agent-based Engineering (WSABE 2003), held in conjunction with AAMAS-2003, Melbourne, Australia.

Towards P2P-Based Semantic Web Service Discovery with QoS Support*

Le-Hung Vu, Manfred Hauswirth, and Karl Aberer

School of Computer and Communication Sciences,
Ecole Polytechnique Fédérale de Lausanne (EPFL),
CH-1015 Lausanne, Switzerland
{lehung.vu, manfred.hauswirth, karl.aberer}@epfl.ch

Abstract. The growing number of web services advocates distributed discovery infrastructures which are semantics-enabled and support quality of service (QoS). In this paper, we introduce a novel approach for semantic discovery of web services in P2P-based registries taking into account QoS characteristics. We distribute (semantic) service advertisements among available registries such that it is possible to quickly identify the repositories containing the best probable matching services. Additionally, we represent the information relevant for the discovery process using Bloom filters and pre-computed matching information such that search efforts are minimized when querying for services with a certain functional/QoS profile. Query results can be ranked and users can provide feedbacks on the actual QoS provided by a service. To evaluate the credibility of these user reports when predicting service quality, we include a robust trust and reputation management mechanism.

1 Introduction

The increasing number of web services demands for an effective, scalable, and reliable solution to look up and select the most appropriate services for the requirements of the users. This is specifically complicated if numerous services from various providers exist, all claiming to fulfill users' needs. To solve these problems, a system basically has to provide expressive semantic means for describing web services including functional and non-functional properties such as quality of service (QoS), semantic search capabilities to search distributed registries for services with a certain functional and QoS profile, and mechanisms for allowing users to provide feedbacks on the perceived QoS of a service that can be evaluated by the system regarding their trustworthiness.

In this paper we present our approach to address these issues. It is based on requirements from a real-world case study of virtual Internet service providers

* The work presented in this paper was (partly) carried out in the framework of the EPFL Center for Global Computing and was supported by the Swiss National Funding Agency OFES as part of the European project DIP (Data, Information, and Process Integration with Semantic Web Services) No 507483. Le-Hung Vu is supported by a scholarship of the Swiss federal government for foreign students.

(VISP) in one of our projects¹. In a nutshell, the idea behind the VISP business model is that Internet Service Providers (ISPs) describe their services as semantic web services, including QoS such as availability, acceptable response time, throughput, etc., and a company interested in providing Internet access, i.e., becoming a VISP, can look for its desired combination of services taking into account its QoS and budgeting requirements, and combine them into a new (virtual) product which can then be sold on the market. At the moment this business model exists, but is done completely manually.

Since many ISPs can provide the basic services at different levels and with various pricing models, dishonest providers could claim arbitrary QoS properties to attract interested parties. The standard way to prevent this is to allow users of the service to evaluate a service and provide feedbacks. However, the feedback mechanism has to ensure that false ratings, for example, badmouthing about a competitor's service or pushing own rating level by fake reports or collusion with other malicious parties, can be detected and dealt with. Consequently, a good service discovery engine would have to take into account not only the functional suitability of the services but also their prospective quality offered to end-users regarding to the trustworthiness of both providers and consumer reports. According to several empirical studies [15, 11], this issue of evaluating the credibility of user reports is one of the essential problems to be solved in the e-Business application area.

To achieve the high scalability, in our work we focus on developing a decentralized discovery approach and for improved efficiency we use a structured overlay network as the decentralized service repository system. In the following we assume that web services are being described semantically including QoS properties, for example, using WSMO², service descriptions can be stored in distributed registries, and users can provide feedbacks on the experienced QoS. Based on these realistic assumptions we will devise a framework for P2P-based distributed service discovery with QoS support.

Regarding the semantic characterization of Web Services several properties can be considered, of which the most obvious are the structural properties of the service interface, i.e., the input and output parameters of a service. Another important aspect, in particular for distinguishing services with equivalent functional properties, relates to QoS characteristics. In our approach we intend to support both aspects. As described above, for QoS it is of interest to compare the announced with the actual service performance, for which we take a reputation-based trust management approach. Other characteristics of Web Services, in particular the process structure of the service invocation also have been considered, e.g., Emekci et al [14], but we consider these as less important, since they are difficult to use in queries and unlikely to be the primary selection condition in searches, and thus not critical in terms of indexing. However, we may expect that the service interface will be usually used as a search condition with good selectivity among a large number of web services. In order to support these queries

¹ <http://dip.semanticweb.org/>

² <http://www.wsmo.org/>

we have to index unordered key sets (corresponding to a service interface), where the keys are usually taken from a (shared) domain ontology. To the best of our knowledge, although the issue of indexing semantic data in structured overlay networks has already been mentioned somewhere, e.g., [6, 12, 29], none of them have taken into account the structural properties of web services while indexing semantic service descriptions for the benefits of service discovery.

The major contribution of this paper is the proposal of a new distributed service discovery framework which is expected to be scalable, efficient and reliable. With the use of structured peer-to-peer overlays as the service repository network, the system is highly scalable in terms of the number of registries and services. Our approach uses multiple unordered key sets as index terms for semantic web service descriptions, thus make it possible to quickly identify the registries containing most likely matched services according to user requests. The local semantic service matchmaking at a specific registry can also be performed efficiently thanks to the combination of the ontology numerical encoding scheme [8] with the pre-computation of the matching levels between service advertisements and possible user queries [28] to reduce the time-consuming reasoning steps. In addition, our search algorithm exploits the generalization hierarchy of the underlying ontology for approximate matching and will use QoS information to rank the search results according to preferences of users. Our QoS-based service selection and ranking algorithm also takes into account the issue of trust and reputation management sufficiently, thereby returning only the most accurate and relevant results w.r.t. user requirements.

2 Related Work

Our framework uses a novel ontology-based approach to distribute service advertisements appropriately among a P2P network of registries. This method is different from that of METEOR-S [31] and HyperCup [25] as we do not base it on a classification system expressed in service or registry ontologies. In these approaches, the choosing of a specific registry to store and search for a service advertisement depends on the type of the service, e.g., business registry is used for storing information of business-related services. In fact, these proposals is good in terms of organizing registries to benefit service management rather than for the service discovery itself. Although publishing and updating service description information based on their categories is relatively simple, it would be difficult for users to search for certain services without knowing details of this classification, and it would be hard to come up with such a common service or registry ontology. To some extent our approach is similar to WSPDS [17], but our methods are specifically targeted at *structured* P2P overlay networks in order to support more efficient service publishing and discovery. We use our P-Grid P2P system [1] as the underlying infrastructure, which at the time of this writing, is among the very few P2P systems which support maintenance and updating of stored data. [26] indexes service description files (WSDL files) by a set of keywords and uses a Hilbert-Space Filling Curve to map the n-dimensional

service representation space to an one-dimensional indexing space and hash it onto the underlying DHT-based storage system. However, the issue of characterizing a semantic service description as a multi-key query in order to support semantic discovery of services has not yet been mentioned in this work. As aforementioned, Emekci et al [14] suggest to search services based on their execution paths expressed as finite path automata which we consider less important since this is difficult to use as primary selection condition in queries as user would need to know and describe the execution flow of their required services.

Regarding QoS, although the traditional UDDI registry model³ does not refer to quality of web services, many proposals have been devised to extended the original model and describe web services' QoS capabilities, e.g., QML, WSLA and WSOL [13]. The issue of trust and reputation management in Internet-based applications as well as in P2P systems has also been a well-studied problem [11, 15]. However, current QoS provisioning models have not sufficiently considered the problem of evaluating the credibility of reporting users. The existing approaches either ignore this issue totally [3, 7, 16, 30] or employ simple methods which are not robust against various cheating behaviors [14, 18]. Consequently, the quality of ranking results of those systems will not be assured if there are dishonest users trying to boost the quality of their own services and badmouthing about the others. [10] suggests augmenting service clients with QoS monitoring, analysis and selection capabilities. This is a bit unrealistic as each service consumer would have to take the heavy processing role of both a discovery and a reputation system. Other solutions [20, 21, 23, 24] use mainly third-party service brokers or specialized monitoring agents to collect performance of all available services in registries, which would be expensive in reality.

An advanced feature of our architecture is that we perform the service discovery, selection and ranking based on the matching level of service advertisements to user queries both in terms of functionality and QoS as well as taking into account trust and reputation adequately. Our QoS provisioning model is developed from [7, 16, 18] using concepts of integrating QoS into service description by [24] and [30]. The trust and reputation management mechanism originally combines and extends ideas of [2, 9, 19] and is the first solution to address the most important issues sufficiently.

3 A Model for P2P-Based Web Service Discovery with QoS Support

Fig. 1 shows the conceptual model of our distributed service discovery framework.

Service advertisements with embedded QoS information are published in P2P-based registries by various providers (1), and users can query for services with certain functionalities and required QoS levels (2) using any registry peer as their access point. The P2P-based registries then take care of routing the request to

³ <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>

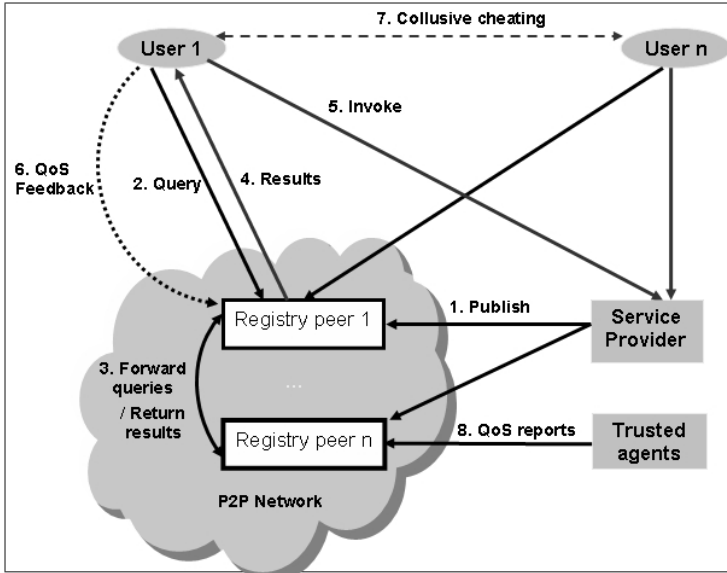


Fig. 1. Framework model

the peer(s) that can answer it (3). The results will be returned to the user (4) and this user may invoke one of the found services (5). Additionally, users can express feedbacks on the QoS they could obtain from a service to the registry peers managing that service (6).

The evaluation of QoS reports by the registry peers has to account for malicious reporting and collusive cheating of users (7) to get a correct view of the QoS properties of a service. Additionally, we also allow trusted agents in the model to provide QoS monitoring for certain services in the system (8). These well-known trusted agents always produce credible QoS reports and are used as trustworthy information sources to evaluate the behaviors of the other users. In reality, companies managing the service searching engines can deploy special applications themselves to obtain their own experience on QoS of some specific web services. Alternatively, they can also hire third party companies to do these QoS monitoring tasks for them. In contrast to other models [20, 21, 23, 24, 30] we do not deploy these agents to collect performance data of all available services in the registries. Instead, we only use a small number of them to monitor QoS of some selected services because such special agents are usually costly to setup and maintain.

Fig. 2 shows the internal architecture of a registry peer.

The communication module provides an information bus to connect the other internal components; interacts with external parties, i.e., users, trusted agents, and service providers, to get service advertisements, QoS data, and feedbacks; and provides this information to the internal components. Additionally, it is the registry peer's interface to other peers (query forwarding, exchange of service

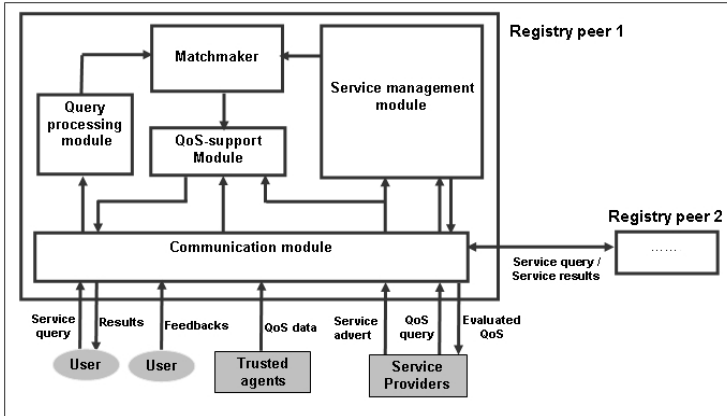


Fig. 2. Registry Peer Structure

registrations and QoS data) and for the user to submit queries and receive results. The query processing module analyzes a semantic web service query into user's required functionality and the corresponding QoS demand of the needed service and then forwards them to the matchmaker. The matchmaker compares the functional requirements specified in a query with the available advertisements from the service management module to select the best matching services in terms of functionality. The list of these services is then sent to the QoS support module, which performs the service selection and ranking, based on QoS information provided in the service advertisements and QoS feedback data reported by the users, so that the result contains the most relevant web services according to user request. Providers are also able to query the evaluated QoS of their own services and decide whether they should improve their services' performance or not.

4 Service Description, Registration, and Discovery

A semantic service description structure stored in a peer registry includes:

- a WSDL specification of the service.
- service functional semantics in terms of service inputs, outputs, pre-conditions, post-conditions and effects, which is described by WSMO ontology concepts using the techniques proposed by [27].
- optional QoS information with the promised QoS for the service.

During operation of the system this information will be matched against semantic queries which consist of:

- functional requirements of user in terms of service inputs, outputs, pre-conditions, post-conditions and effects, also expressed in WSMO concepts.

- optional user’s QoS requirements provided as a list of triples $\{q_i, n_i, v_i\}$, where q_i is the required QoS parameter, n_i is the order of importance of q_i in the query (as user preference) and v_i is the user’s minimal required value for this attribute.

Quality properties of web services are described by concepts from a QoS ontology and then embedded into the service description file using techniques suggested by Ran [24] and WS-QoS [30]. In our work, the value of a quality parameter of a web service is supposed to be *normalized* to a non-negative real-valued number regarding service-specific and call-specific context information where higher normalized values represent higher levels of service performance. For instance, a web service with a normalized QoS parameter value for reliability of 0.99 will be considered as more reliable to another one with a normalized reliability value of 0.90. In this case the normalized reliability is measured as its degree of being capable of maintaining the service and service quality over a time period T . For experimental evaluations, we have developed a QoS ontology for the VISP use-case using WSMO. This QoS ontology includes the most relevant quality parameters for many applications, i.e., availability, reliability, execution time, price, etc. We currently assume that users and providers share a common ontology to describe various QoS concepts. However, this could be relaxed with the help of many existing ontology mapping frameworks. The QoS provisioning model is described in details in [32].

4.1 A Closer Look at Semantic Service Descriptions

In our architecture, a semantic service description, i.e., a service advertisement or a service query, will be associated with a multi-key vector, which we call the *the characteristic vector* of the service. Based on this vector service advertisements are assigned to peer registries. Similarly, discovery of registries containing services relevant to a user query is also based on the characteristic vector of the query itself.

First, all ontological concepts representing *inputs* and *outputs* of a web service advertisement/service request will be categorized into different *Concept Groups* based on their semantic similarity. This similarity between two concepts is computed based on the distance between them in the ontology graph and their number of common properties as proposed by previous work, e.g., [5]. Each group has a root concept defined as the one with the highest level in the ontology graph, i.e., the most general concept, among all member concepts.

A semantic service description, i.e., a service advertisement or a service query, is then characterized by the concept groups to which the service’s inputs and outputs belong. According to [8], ontological concepts can be mapped into numerical key values in order to support semantic reasoning efficiently. Therefore, we can utilize keys to represent concepts and a group of similar concepts can be associated with a Bloom key built by applying k hash functions h_1, h_2, \dots, h_k to the key of each concept member, allowing us to quickly check the membership of any concept to that group [4]. For each input I_i (or output O_i) of a service,

we firstly find the concept group CG_i that it belongs to. As the order of inputs/outputs of a service generally has no sense in determining its functionality, we define a total ordering of various concept groups as in Definition 1 so that service queries/advertisements with similar interfaces would have the same characteristic vector regardless the differences in the order of their parameters. The *characteristic vector* of this service description is then represented by the list of corresponding Bloom keys of all CG_i s, sorted in the descending *order* of CG_i .

Definition 1. A concept group CG_x is considered as having *higher order* ($>$) than another group CG_y iff:

1. The level of CG_x in the ontology graph is higher than the level of CG_y or:
2. Both CG_x and CG_y have the same level and CG_x is in the left of CG_y in the ontology graph.

The partitioning of ontological concepts is illustrated in Fig. 3 where C_j is an ontological concept and CG_i is a concept group. The task of fragmenting the ontology graph is similar to that of relational and semi-structured database systems, which could be performed semi-automatically by the system with additional user support.

In Fig. 3, the root concepts of $CG_1, CG_2, CG_3, CG_4, CG_5$ and CG_6 are C_2, C_3, C_4, C_5, C_6 and C_9 , respectively. The total ordering of all concept groups is $CG_1 > CG_2 > CG_3 > CG_4 > CG_5 > CG_6$. As an example, let us assume that we have a service description S_1 with inputs C_7, C_{14}, C_{10} and outputs C_{12}, C_{16} which belong to concept groups CG_1, CG_6, CG_2 and CG_4, CG_3 ,

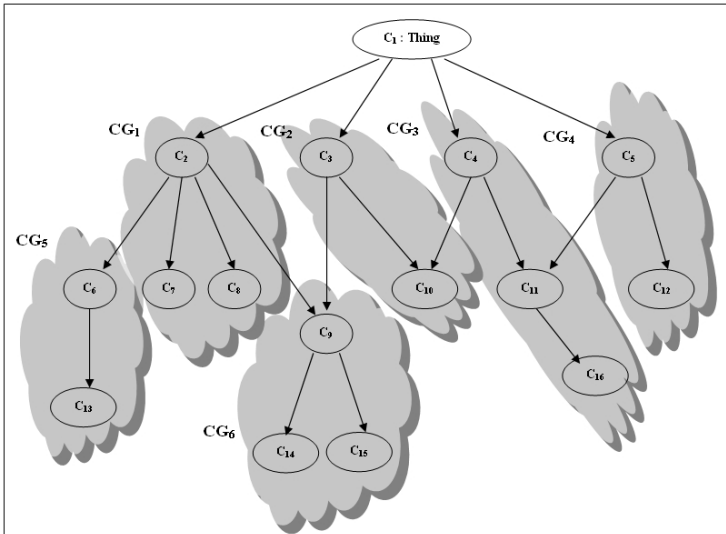


Fig. 3. Ontology graph partitioning

respectively. Regarding the above ordering relation, this service description is then represented by *the characteristic vector* $V = \{k_1, k_2, k_6, k_d, k_3, k_4\}$, where k_i is CG_i 's Bloom key and k_d is a dump value to separate S_1 's inputs and outputs.

Although we are using only inputs and outputs of a service in its multiple-key representation, we believe that the extension of this idea to other features in a semantic service description, e.g., pre-conditions, post-conditions, effects, could be done in a similar fashion. The strategy used for partitioning the ontological graph will not affect the correctness but mainly the efficiency of the discovery algorithm. For instance, although it is tempting to allow a concept to belong to more than one group while partitioning, this increases the discovery time because we need to contact different registries to search for all possibly matching services. Therefore, we prefer to have only one group for each concept. For simplicity, we currently assume that all registries agree on one ontology of concepts, but this restriction will be relaxed soon with our on-going work.

4.2 Mapping of Service Advertisements to Registries

Each registry peer is responsible for managing certain web services that operate on a certain set of concepts. The mechanism to assign these sets to peers works as follows:

1. Each vector $V_i = \{k_{i1}, k_{i2}, \dots, k_{in}\}$, where k_{ij} ($j = \overline{1..n}$) is a group's Bloom key or dump value k_d , is mapped to a combined key K_i using a special function H_c that includes all features of each individual member key k_{ij} .
2. Using the existing DHT-based searching mechanism of the underlying P-Grid network [1], we can easily find the identifier RP_i of the registry peer corresponding to the index key K_i .
3. The registry peer RP_i is responsible for storing the description of those services with the same characteristic vector V_i .

This assignment of services to registries during the publishing phase will help us to quickly identify the registry(-ies) most likely to contain the semantic web service descriptions matching with a service request during the discovery time. Using Bloom filters, the step of checking the membership of a concept in certain concept groups can be done fast and with very high accuracy level. Therefore, the computation of the characteristic vector of a service request can be done efficiently. Eventually, the question of searching for the registry(-ies) most likely to store a matched services becomes the problem of finding the peers capable of answering a multi-keyword query which corresponds to this characteristic vector in the P2P network. This problem can be solved by using one of the two following approaches. The first one is to simply concatenate all k_{ij} s together and then use this as the index/search key in the underlying P2P network. The second possibility is to deploy another type of peers in the network as *index peers* to keep identifiers of those registries that manage keywords related to various combination of k_{ij} s. Of course, there is another naive method in which we can

search for all peers storing each concept term and then intersect all partial matches to get the final results. However, we reason that this approach would be inefficient due to the following reason. As the semantics of the parameters in a service interface are generally different from each other, a registry containing service advertisements with only one satisfactory parameters does not necessarily store service descriptions with the full interface as user requires. This means it would be costly to forward the service query to many (distributed) registries and wait for all semantic matchmaking in these repositories to terminate and get the final results.

We have decided to use the first method because in this way, the keyword generating function H_c will generate similar keys K_i s for services with similar characteristic vectors $\{k_{i1}, k_{i2}, \dots, k_{in}\}$. Since P-Grid uses *prefix-based query routing* as its search strategy, services corresponding to similar K_i s, which are likely to offer comparable functionalities, will be assigned to registries adjacent to each other (P-Grid clusters related information). This is important as with the very high number of registries and published services, the query for services will only need to be forwarded to a small number of adjacent peers. Otherwise, we will have to wait for the results to be collected from a lots of widely distributed registries, making the searches become highly inefficient. Moreover, this is advantageous for the exchanges of QoS reports and user reputation information among neighboring registries during the QoS predicting process later.

Regarding Fig. 3, supposed that we have three services: S_1 operating on two concepts C_2, C_3 and producing C_4 , S_2 operating on two concepts C_2, C_9 and producing C_{14} , S_3 operating on two concepts C_2, C_9 and producing C_{15} . The characteristic vectors of S_1 will be $\{k_1, k_2, k_d, k_3\}$ whereas S_2, S_3 will have the same characteristic vector as $\{k_1, k_6, k_d, k_6\}$, with k_1, k_2, k_3, k_6 is the Bloom key of the concept groups CG_1, CG_2, CG_3, CG_6 and k_d is a dump key, respectively. According to our way of distributing service descriptions, S_1 will be assigned to one registry peer P_1 with index key $K_1 = k_1 || k_2 || k_3$ and S_2, S_3 will be assigned to another peer P_2 with another index entry $K_2 = k_1 || k_6 || k_d || k_6$.

4.3 Pre-computation of Service Matching Information to Support Semantic Service Discovery

Since the publishing task usually happens once and is not a computationally intensive process, we can devote more time in this stage to reduce later discovery time, as suggested by Srinivasan et al [28]. However, their proposed approach is not scalable since it requires to store the matching information of all services which match each concept c_i in the ontology, thus producing much redundant information. Hence, we improve their method by observing that if a concept c_i of a group CG_i , is similar to another concept c_j (also belonging to this group), then both of them should have approximately the same distance, i.e., the same level of semantic similarity, to the root concept of CG_i .

Accordingly, for each CG_i , we store a matching list containing semantic distances from each parameter of each service to CG_i 's root concept. For example, assuming that we have a registry peer responsible for managing those services

which operate on the list of concept groups CG_1, CG_2, \dots, CG_k . Then in the matching table of this registry, we store for each group $CG_i, i = \overline{1..k}$, a list $Ldst_i$ of records $\{[S_{i1}, d_1], [S_{i2}, d_2], \dots, [S_{in}, d_n]\}$, where S_{ij} represents a web service, $d_j \in [0, 1]$ is the semantic similarity between the concept represented by one parameter of S_{ij} with the root concept of $CG_i, j = \overline{1..n}$, n is the number of services in this registry.

A query for a service is first submitted to a registry peer. At this entry point the characteristic vector of the query is computed as in Section 4.1 and Section 4.2. Using the combined key of this characteristic vector as a search key, the query is then forwarded by P-Grid's routing strategy to a registry most possibly containing matching services. For each service query's parameter c_i belonging to group CG_i , the discovery algorithm at this registry computes its matching level d_i with CG_i 's root concept rc_i . Afterward, it finds the list L_i of those services S_{ijs} each of which has (at least) one parameter with an approximate matching level d_{ij} with rc_i , i.e., $d_{ij} \approx d_i$, by browsing the matching list $Ldst_i$ of each rc_i . We then intersect all L_i s to get the list L_c of possibly matching services. Note that if c_{i1} and c_{i2} have the same matching level d_i with CG_i 's root concept, we can only conclude that c_{i1} and c_{i2} are *possibly* similar. Consequently, simply intersecting all L_i s does not help us in finding the services which accurately match the query as in [28]. However, they do allow us to select the list of all possible matches and filter out non-related services, which really reduces the searching time in case the number of registered services is high. Finally, we utilize another service semantic matchmaking algorithm, e.g. [22], to further select from L_c the list L of most suitable services in terms of functionality.

For supporting queries with QoS requirements, we use another table to store the matching information for frequently accessed QoS attributes. With each QoS attribute q_j in this QoS matching table, we have a list $Lqos_j$ of records $\{S_{ij}, w_{ij}, predicted_{ij}\}$, in which S_{ij} identifies a service, w_{ij} is a weight depending on the semantic similarity between q_j and the QoS attribute q_{ij} supported by S_{ij} , and $predicted_{ij}$ is the value of q_{ij} predicted by our QoS-based service selection and ranking engine. Apparently, we only store in $Lqos_j$ information of those S_{ijs} with w_{ijs} greater than a specific threshold. The idea behind is that we will give higher ranks for services which offer the most accurate QoS concepts at the higher levels compared to the ones required by users. Note that although it is possible to use QoS properties as ranking criteria for service queries without explicit QoS requirements, we have not yet employed this in our current study. Therefore, the QoS-based service selection and ranking phase will be performed only if users provide their QoS requirements explicitly in corresponding queries.

Given the list L of services with similar functionalities, the discovery engine performs the QoS-based service selection and ranking as in Algorithm 1.

To facilitate the discovery of services with QoS information, we must evaluate how well a service can fulfill a user query by predicting its QoS from the service's past performance reported in QoS feedbacks. In our model, we apply time series forecasting techniques to predict the quality values from various information sources. Firstly, we use the QoS values promised by providers in their service ad-

Algorithm 1 QosSelectionRanking(ServiceList L, ServiceQuery Q)

```

1: Derive the list of QoS requirements in Q:  $L_q = [q_1, n_1, v_1], \dots, [q_s, n_s, v_s]$ 
2: Initialize  $QosScore[S_i] = 0.0$  for all services in L;
3: for each quality concept  $q_j \in L_q$  do
4:   for each service  $S_i \in L$  do
5:     Search the list  $L_{qos}$  of  $q_j$  for  $S_i$ ;
6:     if  $S_i$  is found then
7:        $PartialQosScore = w_{ij} \frac{predicted_{ij} - v_j}{v_j}$ ;
8:        $QosScore[S_i] = QosScore[S_i] + \frac{n_j}{\sum n_j} PartialQosScore$ ;
9:     else
10:      Remove  $S_i$  from L;
11:     end if
12:   end for
13: end for
14: Return the list L sorted in descending order by  $QosScore[S_i]$  s;

```

vertisements. Secondly, we collect consumers' feedbacks on QoS of every service. Thirdly, we use reports produced by trusted QoS monitoring agents. In order to detect possible frauds in user feedbacks, we use reports of trusted agents as reference values to evaluate behaviors of other users by applying a trust-distrust propagation method and a clustering algorithm. Reports that are considered as incredible will not be used in the predicting process. Through various experiments, this proposed service selection and ranking algorithm is shown to yield very good results under various cheating behaviors of users, which is mainly due to the fact that the use of trusted third parties monitoring QoS of a relatively small fraction of services can greatly improve the detection of dishonest behavior even in extremely hostile environments. The detail of this QoS-based service selection and ranking phase as well as various experimental results are presented in [32].

5 Conclusions and Future Work

In this paper we proposed a new P2P-based semantic service discovery approach which uses a natural way of assigning service descriptions to registry peers. Also, we presented a service selection and ranking process based on both functional and QoS properties. In order to support flexible queries we index unordered key sets where the keys are taken from a shared domain ontology. This problem of indexing of web service descriptions in structured overlay networks to support service discovery has not been addressed so far in the literature. The QoS model includes a user feedback mechanism which is resilient against malicious behaviors through the application of a trust and reputation management technique that allows us to discover a variety of cheating attempts by providers and service users. As we use a P2P system as the underlying infrastructure, our system scales well in terms of number of registries, search efficiency, number of properties in service descriptions, and number of users.

We already implemented the QoS-based service selection and ranking algorithm with trust and reputation evaluation techniques as a QoS support module in our framework. Many experiments were also performed to prove the effectiveness of our trust and reputation approach under various situations. In the next stage, we will implement the matchmaker based on the work initiated by Paolucci et al [22] and the service management module based on the UDDI standard. The existing implementation of the P-Grid system, Gridella⁴, is used as the basis for the communication module. The next step would be to extend our model such that registry peers are able to manipulate with heterogeneous and distributed ontologies. Also, it would be beneficial to extend the indexing scheme to include service pre-conditions, post-conditions, effects, etc., in semantic service description structures. Moreover, further work should be done on the use of QoS properties as ranking criteria for service queries without explicit QoS requirements. In addition, we are studying the possibility of developing and utilizing a caching mechanism to exploit the locality and frequency of service usages in the system.

References

1. K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Puceva, and R. Schmidt: P-Grid: a self-organizing structured P2P system, *SIGMOD Rec.*, 32(3):29–33, 2003.
2. K. Aberer and Z. Despotovic: Managing trust in a peer-2-peer information system, Proceedings of CIKM'01, USA, 2001.
3. A. S. Bilgin and M. P. Singh: A DAML-based repository for QoS-aware semantic web service selection, Proceedings of ICWS'04, USA, 2004.
4. B. H. Bloom: Space/Time trade-offs in hash coding with allowable errors, *Commun. ACM*, 13(7):422–426, 1970.
5. S. Castano, A. Ferrara, S. Montanelli, and G. Racca: Matching techniques for resource discovery in distributed systems using heterogeneous ontology descriptions, Proceedings of ITCC'04, USA, 2004.
6. S. Castano, A. Ferrara, S. Montanelli, and D. Zucchelli: Helios: a general framework for ontology-based knowledge sharing and evolution in p2p systems, Proceedings of DEXA'03, USA, 2003.
7. Z. Chen, C. Liang-Tien, B. Silverajan, and L. Bu-Sung: UX - an architecture providing QoS-aware and federated support for UDDI, Proceedings of ICWS'03.
8. I. Constantinescu and B. Faltings: Efficient matchmaking and directory services, Proceedings of WI'03, USA, 2003.
9. F. Cornelli, E. Damiani, S. C. Vimercati, S. Paraboschi, and P. Samarati: Choosing reputable servants in a P2P network, Proceedings of WWW'02, USA, 2002.
10. J. Day and R. Deters: Selecting the best web service, Proceedings of CASCON'04, 2004.
11. Z. Despotovic and K. Aberer: Possibilities for managing trust in P2P networks, Technical Report IC200484, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland, Nov. 2004.

⁴ <http://www.p-grid.org/Software.html>

12. H. Ding, I. T. Solvberg, and Y. Lin: A vision on semantic retrieval in p2p network. Proceedings of AINA'04, USA, 2004.
13. G. Dobson: *Quality of Service in Service-Oriented Architectures*. <http://digs.sourceforge.net/papers/qos.html>, 2004.
14. F. Emekci, O. D. Sahin, D. Agrawal, and A. E. Abbadi: A peer-to-peer framework for web service discovery with ranking. Proceedings of ICWS'04, USA, 2004.
15. A. Jøsang, R. Ismail, and C. Boyd: A survey of trust and reputation systems for online service provision, *Decision Support Systems*, 2005 (to appear).
16. S. Kalepu, S. Krishnaswamy, and S. W. Loke: Reputation = f(user ranking, compliance, verity), Proceedings of ICWS '04, USA, 2004.
17. F. B. Kashani, C.-C. Chen, and C. Shahabi: WSPDS: Web services peer-to-peer discovery service. Procs. of *International Conference on Internet Computing*, 2004.
18. Y. Liu, A. Ngu, and L. Zheng: QoS computation and policing in dynamic web service selection. Procs of WWW Alt. Conf., USA, 2004.
19. E. Manavoglu, D. Pavlov, and C. L. Giles: Probabilistic user behavior models, Proceedings of ICDM'03.
20. E. M. Maximilien and M. P. Singh: Reputation and endorsement for web services, *SIGecom Exch.*, 3(1):24–31, 2002.
21. M. Ouzzani and A. Bouguettaya: Efficient access to web services, *IEEE Internet Computing*, p.p. 34–44, March/April 2004.
22. M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara: Semantic matching of web services capabilities, Proceeding of ISWC'02, UK, 2002.
23. C. Patel, K. Supekar, and Y. Lee: A QoS oriented framework for adaptive management of web service based workflows, Proceeding of Database and Expert Systems 2003 Conf., p.p. 826–835, 2003.
24. S. Ran: A model for web services discovery with QoS, *SIGecom Exch.*, 4(1):1–10, 2003.
25. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl: A scalable and ontology-based P2P infrastructure for semantic web services, Proceeding of P2P'02, USA, 2002.
26. C. Schmidt and M. Parashar: A peer-to-peer approach to web service discovery, Proceeding of WWW Conf., 2004.
27. K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller: Adding semantics to web services standards, Proceedings of ICWS'03.
28. N. Srinivasan, M. Paolucci, and K. P. Sycara: Adding OWL-S to UDDI, implementation and throughput, Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, USA, 2004.
29. C. Tang, Z. Xu, and S. Dwarkadas: Peer-to-peer information retrieval using self-organizing semantic overlay networks, Proceedings of ACM SIGCOMM'03, USA, 2003.
30. M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller: A concept for QoS integration in web services, Proceedings of WISEW'03, Italy, 2003.
31. K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller: METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of web services, *Inf. Tech. and Management*, 6(1):17–39, 2005.
32. L.-H. Vu, M. Hauswirth, and K. Aberer: QoS-based service selection and ranking with trust and reputation management, Proceedings of OTM'05, R. Meersman and Z. Tari (Eds.), LNCS 3760, p.p. 466–483, 2005.

Global and Local QoS Guarantee in Web Service Selection

Danilo Ardagna and Barbara Pernici

Politecnico di Milano

Abstract. In Service Oriented systems, complex applications can be composed from a variety of functionally equivalent Web services which may differ for quality parameters. Under this scenario, applications are defined as high level business processes and service composition can be implemented dynamically by identifying the best set of services available at run time. In this paper, we model the service composition problem as a mixed integer linear problem where local constraints, i.e., constraints for component Web services, and global constraints, i.e., constraints for the whole application, can be specified. Our approach proposes the formulation of the optimization problem as a global optimization, not optimizing separately each possible execution path as in other approaches. Experimental results demonstrate the effectiveness of our approach.

1 Introduction

With the development of the Service Oriented Architecture, complex applications can be composed as business processes invoking a variety of available Web services with different characteristics. The goal is to develop applications by specifying component web services in a process only through their required functional characteristics, and selecting web services during process execution from the ones included in a registry of available services. We assume that service descriptions are stored and retrieved from enhanced UDDI registries, which provide also information about quality parameters on the provider side, as proposed in [2, 18]. Usually, a set of functionally equivalent services can be selected, i.e., services which implement the same functionality but differ for the quality parameters. Under this scenario, applications are defined as business processes composed of abstract web-services and services selection can be performed dynamically by identifying the best set of services available at run time, taking into consideration also end-user preferences, process constraints and execution context.

Web service selection introduces an optimization problem. In the work presented in [18] two main approaches have been proposed: local and global optimization. The former selects at run time the best candidate service which supports the execution of the running high level activity. The latter identifies the set of candidate services which satisfy the end user preferences for the whole application. The two approaches allow specifying Quality of Service (QoS) constraints at local and global level, respectively. A local constraint allows selecting a Web service according to a desired characteristic. For example, a Web service

can be selected such that *its* price or *its* execution time are lower than a given threshold. Global constraints pose constraints over the *whole* composite service execution, e.g., constraints like "The overall execution time of the application has to be less than 3 seconds" or "the total price has to be less than 2\$."

Note that the end-user is mainly interested in global constraints. For example, she could be concerned in the total execution time of the application instead of the execution time of single activities. Furthermore, in Web service environments services composition could be hidden to the end user (i.e., she can not distinguish among basic and composed services).

In this paper we propose a global approach for Web services selection and optimization. The problem of Web Services composition (WSC) with QoS constraints can be modeled as a mixed integer linear programming problem. The problem is NP-hard, since it is equivalent to a Multiple Choice Multiple Dimension Knapsack problem. Our approach is implemented in the MAIS¹ architecture, a platform which supports the execution of Web services in multi-channel adaptive systems [14]. The remainder of this paper is organized as follows. The next section reviews other literature approaches. Section 3 introduces the set of quality dimensions considered in the optimization problem. The composition approach and optimization problem formulation are presented in Section 4. Experimental results in Section 5 demonstrate the effectiveness of our solutions. Conclusions are drawn in Section 6.

2 Related Work

Recently, QoS issues of Web services have obtained great interest in the research community. In [13] is presented a framework for composed services modeling and QoS evaluation. A composite service is modeled as a directed weighted graph where each node corresponds to a Web service and a weight represents the transition probability of two subsequent tasks. The author shows how to evaluate quality of service of a composed service from basic services characteristics and graph topology. The work in [18] presents a middleware platform for Web services selection and execution, where local and global optimization approaches are compared. Other literature proposals partially support global constraints. The work presented in [12] introduces an agent based framework where agents can migrate to invoke Web services locally. Anyway, network traffic and execution time are the only quality dimensions considered and constraints can be specified only locally. If only local constraints are considered, the service composition is very simple and can be performed at run time by a greedy approach which selects the best candidate service suitable for the execution.

Our approach starts from the work presented in [18]. For the sake of simplicity in the following we assume that a composite service is characterized by a single initial task and a single end task. Let us indicate with $ws_{i,j}$ the i -th Web service, candidate for the execution of task t_j . In the following we assume that cycles are

¹ MAIS (Multichannel Adaptive Information Systems) project web site: <https://www.mais-project.it>

unfolded according to the maximum number of iterations. We give the following definitions:

Execution Path. A set of tasks $\{t_1, t_2, \dots, t_n\}$ such that t_1 is the initial task, t_n is the final task and no t_i, t_j belong to alternative branches. Execution paths will be denoted by ep_l . Note that an execution path can include parallel sequences.

Sub path. A sub path of an execution path ep_l is a sequence of tasks $[t_1, t_2, \dots, t_n]$, $t_i \in ep_l \forall i$, from the begin to the end task which does not contain any parallel sequence. A sub path will be indexed by m and denoted by sp_m^l .

Critical Path. The critical path of an execution path ep_l is the sub path which corresponds to the highest execution time of the execution path.

Execution Plan. An execution plan of an execution path ep_l is a set of ordered couples $(t_j, ws_{i,j}^-)$, indicating that task t_j included in ep_l is executed by a given Web service $ws_{i,j}^-$. Execution plans will be indexed by k and denoted as ep_k^l .

Global Plan. The global plan is a set of ordered couples $(t_j, ws_{i,j}^-)$, which associates every task t_j to a given Web service $ws_{i,j}^-$ and satisfies local and global constraints for all execution paths.

Note that, the set of execution paths of an activity diagram identifies all the possible execution scenarios of the composite service. Authors in [18] separately optimize each execution path and obtain the global plan by composing separate solutions according to the frequency of execution. Their approach has several limitations. First, in the optimization of a single execution path the fulfillment of availability and response time constraints is guaranteed only for the critical path. Authors assume that if the execution of a task in another sub paths fails, then it can be restarted without adding any delay to the overall execution of the composite service (i.e., other sub paths execution time plus the time required to restart a service are always lower of the critical path execution time). Anyway this assumption is not verified. Furthermore, the global plan is obtained as the merge of separate execution plans. If a task belongs to multiple execution paths, then the task is executed by the service identified for the most frequently executed execution path. Note that, in this way the fulfillment of global constraints can not be guaranteed. Let us consider the example reported in Figure 2. There are two Execution Paths ep_1 and ep_2 which include respectively states t_1 and t_2 , and t_1 and t_3 . Execution frequencies are reported in parenthesis. The Figure indicates also the price and the execution time associated with candidate services (denoted by $p_{i,j}$ and $e_{i,j}$, respectively). Let us assume that the end user requires the price is less or equal to 6\$, while the overall execution time is less or equal to 7 sec. In [18] approach the optimum solution for execution path ep_1 selects $ws_{2,1}$ and $ws_{2,2}$, (which are the only feasible solution for the execution path). Two execution plans are feasible for the execution path ep_2 : ep_2^1 selects $ws_{1,1}$ and $ws_{1,3}$, (which imply 5\$ price and 6 sec execution time), ep_2^2 selects $ws_{2,1}$ and $ws_{2,3}$ (which imply 6\$ price and 7 sec execution time). ep_2^1 is the optimum solution since it dominates ep_2^2 by introducing lower price and execution time. The global plan identified by [18] approach, then selects services $ws_{1,1}$, $ws_{2,2}$ and $ws_{1,3}$. Note anyway that if at run time the execution path ep_1 is taken, then

the budget constraint is violated since the cost of execution of $ws_{1,1}$ and $ws_{2,2}$ is 8\$. Vice versa, the set of services $ws_{2,1}$, $ws_{2,2}$ and $ws_{2,3}$ does not introduce any global constraints violation.

The work presented in [3, 5, 4] proposes a genetic algorithm for the solution of the WSC problem. The work is based of the reduction formulas presented in [7] considers also global plan re-optimization but only sub-optimum solutions are identified since tasks specified in cycles are always assigned to the same service.

In this paper we present a solution to the above mentioned problems and propose an optimization problem formulation for the WSC which guarantees the fulfillment of global constraints.

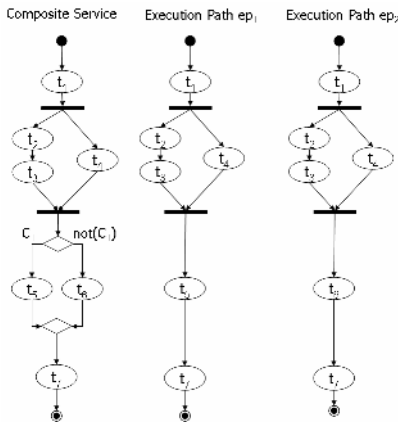


Fig. 1. Process Execution Paths

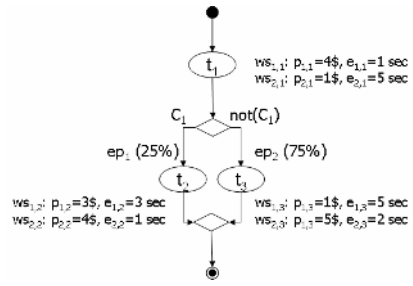


Fig. 2. A Constraint Violation Example

3 The Quality Model

The WSC problem is multi-objective since several quality criteria can be associated with Web services execution. In this paper we focus on the following set of quality dimensions, which have been the basis for QoS consideration also in other approaches [9, 18, 15]: (a) *execution time*, (b) *availability*, (c) *price*, (d) *reputation*. A comprehensive list of quality dimensions [6] has been defined in the MAIS project. The approach proposed for the classical dimensions of the literature and considered in this paper could be easily generalized to other dimensions according to specific application requirements considering all interested QoS dimension in the weighted sum in the objective function under the hypothesis that the aggregated value for quality dimensions can be evaluated as sum, average, product, min or max of the corresponding quality dimension of component services.

For the sake of simplicity, in the following we will assume that a Web service implements a single operation. For each Web service, the set of quality dimensions which defines the quality profile is stored in the MAIS registry [14]. Note

that, if the same service is accessible from the same provider, but with different quality characteristics (e.g. quality level), then multiple copies of the same service will be stored in the registry, each copy being characterized by its quality profile. The end user can express preferences among quality dimensions, e.g., she can be more focused on execution time instead of price.

4 Interleaving Web Services Selection and Execution in the MAIS Framework

The MAIS registry is an extension of a UDDI registry, where services are registered with associated keywords and their WSDL specification [2]. The registry implements also a domain ontology, where semantic information for service input/output is maintained, and a service ontology, where services and semantic relationships among them are described. An overview of the MAIS framework can be found in [14]. Applications are expressed as processes built from *abstract* components. Our goal is to discover at run time the optimum mapping between each component and a Web service which implements the corresponding abstract description, such that the overall QoS perceived by the end user for the application instance execution is maximized, while some global constraints are guaranteed. Enumerating all of the possible combination of Web services is prohibitive since, as it will be discussed in Section 4.3, the WSC problem is NP-hard. The optimization problem is solved by formulating a mixed integer linear programming (MILP) [17].

The quality values advertised by Service Providers are the parameters of the WSC problem; parameters are subject to variability and this is the main issue for the fulfillment of global constraints. The variability of quality values is due mainly to the high variability of the workload of Internet applications, which implies variability of services performance. In our approach, Web service selection and optimization and Web services execution are interleaved. A re-optimization should be performed if a service invocation fails or, from a theoretical point of view, after the execution of each task since new Web services with better characteristics could become available. On the other hand, re-optimization introduces a system overhead since it is time consuming and the MAIS registry has to be accessed in order to update the set of candidate services and their corresponding quality values. In the MAIS-P (MAIS service provider) platform the optimization is computed when the composite service execution starts. The re-optimization is then performed periodically. Furthermore, re-optimization is performed if the end user changes the service channel and if a Web service invocation fails. In a multi-channel information system the same service can be accessed through different channels. At a high level of abstraction, channel examples are a PC connected to the service provider through the Internet or a PDA connected through a wireless LAN. If the end user can switch to a different channel while he is accessing a service, then he need to express different preferences for different channels. In the example above, if the end user swaps from the PC to the PDA, then he could be more focused on the price of the

service instead of its performance since he can expect a higher delay from a wireless connection with restricted bandwidth.

In the latter case, if a service fails a different service will be invoked at runtime to substitute the faulty one and this may lead to a global constraint violation.

In the following section the composite web service specification and model will be introduced. Section 4.2 will discuss the QoS evaluation for composed services. Section 4.3 will formalize WSC as a MILP model. In Section 4.4 the re-optimization of a running application will be discussed. An overview of the optimization tool will be presented in Section 4.5.

4.1 Composed Web Service Specification and Process Model

In the MAIS framework a composite service is specified as a high-level business process in BPEL4WS language. Some annotations are added to the BPEL4WS specification in order to identify:

- the maximum number of iterations for cycles;
- the expected frequency of execution of conditional branches;
- global and local constraints on quality dimensions.

The maximum number of iterations and frequency of execution of conditional branches can be evaluated from past executions by inspecting system logs or can be specified by the composite service designer. If an upper bound for cycles execution can not be determined, then the optimization could not guarantee that global constraints are satisfied [18]. At compilation time, cycles are unfolded according to the maximum number of iterations. Note that, in BPEL4WS only structured cycles can be specified, i.e., cycles with only one entry and exit point are allowed [16]. Furthermore we consider only XLANG-style process specifications, hence cycles can be always unfolded and activity diagram can always be modeled by Directed Acyclic Graphs (DAGs).

Table 1. Notation

Symbol	Description
$e_{i,j}$	$ws_{i,j}$ execution time
$a_{i,j}$	$ws_{i,j}$ availability
$p_{i,j}$	price for $ws_{i,j}$ execution
$r_{i,j}$	$ws_{i,j}$ reputation
WS_j	set of indexes corresponding to candidate Web services of task t_j
\mathcal{A}	set of tasks included in the composite service specification
\mathcal{A}_l	set of tasks included in the execution path ep_l
$freq_l$	frequency of execution for the execution path ep_l
L	number of execution path arising from the composite service specification

As it will be discussed in Section 4.3, the optimization problem will consider all of the possible execution scenarios according to their probability of execution, which can be evaluated by the product of the frequency of execution of branch conditions included in execution paths and annotated in the BPEL4WS specification. Under these definitions, a local constraint can predicate only on properties of a single task. Vice versa, global constraints can predicate on quality attributes of an execution path or on a subset of tasks of the activity diagram, for example a set of subsequent tasks or a sub path. In the current implementation, exceptions are not considered in the optimization problem, global constraints will be guaranteed only for the nominal execution of the composite service. In the following, the notation reported in Table 1 will be adopted.

4.2 QoS in Composed Web Services

The quality criteria defined in Section 3 can be applied to basic services and can be evaluated for a composite service under the hypothesis that it is executed according to a given execution plan ep_l^k . Table 2 reports the aggregating functions which allow computing QoS attributes for an execution plan.

The price is given by the sum of prices of service invocations of all tasks included in the execution plan, while the reputation is obtained as the average reputation of Web services. The execution time of a sub path is given by the sum of execution time of all services included in the sub path, while the execution time of the execution plan is given by the maximum execution time of all sub paths sp_m^l included in it. Availability is evaluated under the assumption (usually adopted in the literature [18]) of stochastic independence of Web services. In this way, availability is computed as the product of availability of all services selected (i.e. a composite service can be executed only if all Web services invoked are available). An execution path has many potential execution plans. We will refer to the quality values of an execution path as the quality values of the selected execution plan, the corresponding quality dimensions will be denoted as indicated in Table 2, but omitting index k .

Table 2. Execution Plans QoS

Quality Dimension	Aggregation function
Price	$price_l^k = \sum_{t_j \in ep_l^k} p_{\bar{i},j}$ of services
Reputation	$rep_l^k = \frac{1}{ A_l } \sum_{t_j \in ep_l^k} r_{\bar{i},j}$
Execution Time	$exeTime_l^k = \max_{sp_m^l \in ep_l^k} \sum_{t_j \in sp_m^l} e_{\bar{i},j}$
Availability	$avail_l^k = \prod_{t_j \in ep_l^k} a_{\bar{i},j}$

4.3 Optimization Problem Formulation

The WSC problem is multi-objective and a Simple Additive Weighting (SAW) technique is used to evaluate the overall value of QoS from multiple quality dimensions [10]. The SAW method is one of the most widely used techniques to obtain a *score* from a list of dimensions having different units of measure.

The raw values for each quality dimension need to be normalized in order to produce a final scalar value for the overall QoS. Each quality dimension is associated with a weight and the value or score for an execution plan is calculated by multiplying the quality dimension's weight by its normalized value and then summing across the quality dimension.

Quality dimensions can be modeled by a matrix. Every execution path ep_l is associated with a matrix $\mathbf{Q}^l = [q_{k,j}^l]$ where, the number of rows equals the number of candidate plans K^l , the number of columns equals the number of quality dimensions J and $q_{k,j}^l$ expresses the value of the execution plan ep_k^l for the j -th quality dimension. In the normalization phase, positive criteria, i.e., quality attributes such that the higher the value the higher the quality, and negative criteria, i.e., quality attributes such that the higher the value the lower the quality, are scaled in different ways, as defined in equations (1) and (2) respectively:

$$v_{k,j}^l = \frac{q_{k,j}^l - \min Q_j^l}{\max Q_j^l - \min Q_j^l} \quad (1)$$

$$v_{k,j}^l = \frac{\max Q_j^l - q_{k,j}^l}{\max Q_j^l - \min Q_j^l} \quad (2)$$

where $\min Q_j^l = \min_{k \in K^l} q_{k,j}^l$ and $\max Q_j^l = \max_{k \in K^l} q_{k,j}^l$ are the minimum and maximum values respectively, for the quality dimension j . Note that, if $\max Q_j^l = \min Q_j^l$ then every execution plan is characterized by the same value for the quality dimension j , the quality dimension is not a differential characteristic for the execution path ep_l and $v_{k,j}^l$ is set to 1 for every k .

Availability and reputation are examples of positive criteria, price and execution time are vice versa negative criteria. The overall score associated with an execution plan ep_k^l is evaluated as:

$$score(ep_k^l) = \sum_{j=1}^J w_j v_{i,j}^l; \quad \sum_{j=1}^J w_j = 1$$

where $w_j \in [0, 1]$ represents the weight assigned by the end user to the j -th quality dimension. Note that, as discussed in [18] the evaluation of $v_{k,j}^l$ requires the assessment of the maximum and minimum value of each quality dimensions for each execution plan. This can be done without considering all possible execution plans, since, for example, the execution plan of maximum price can be obtained by assigning to each task the candidate service of maximum price. The normalization phase complexity is $O(\mathcal{A}_l)$.

The WSC problem can be formulated as a mixed integer linear programming problem. The decision variables of our model are the followings:

- $y_{i,j} :=$ equals 1 if the Web service $ws_{i,j}$ executes task t_j , 0 otherwise;
- $x_j :=$ start time of task t_j ;
- $e_j :=$ task t_j duration;

If we number the quality dimensions as $exeTime = 1$, $avail = 2$, $price = 3$, $rep = 4$, then the score function for the execution path ep_l can be defined as:

$$\begin{aligned} score(ep_l) = & w_1 \frac{\max Q_1^l - exeTime_l}{\max Q_1^l - \min Q_1^l} + w_2 \frac{avail_l - \min Q_2^l}{\max Q_2^l - \min Q_2^l} \\ & + w_3 \frac{\max Q_3^l - price_l}{\max Q_3^l - \min Q_3^l} + w_4 \frac{rep_l - \min Q_4^l}{\max Q_4^l - \min Q_4^l} \end{aligned} \quad (3)$$

Our goal is to maximize the weighted average of the score of execution paths where weights are given by execution path frequency of execution $freq_l$. The WSC problem can be formulated as:

$$P1) \quad \max \sum_{l=1}^L freq_l score(ep_l)$$

Problem P1) can be associated with *assignment constraints* which guarantee that a task is associated with a single Web service in the solution, *task duration constraints* which expresses the duration of tasks in term of the duration of the selected service and with *local* and *global constraints*. Let us denote with E the execution time bound for the composite service execution; let A be the availability bound for the composite service while B and R indicate the budget and the reputation bound respectively.

Assignment constraints. Each task t_j in the solution has to be assigned to exactly one Web service, that is for every j one variable $y_{i,j}$ has to be set to 1. This condition can be expressed by the following constraint family:

$$\sum_{i \in WS_j} y_{i,j} = 1; \forall j \in \mathcal{A} \quad (4)$$

Task duration constraints. The duration of each task can be expressed by the following constraints:

$$\sum_{i \in WS_j} e_{i,j} y_{i,j} = e_j; \forall j \in \mathcal{A} \quad (5)$$

that is, for constraint (4) only one candidate service is selected and hence the duration of a task is given by the selected Web service execution time. Furthermore, we have to include precedence constraints for subsequent tasks in the activity diagram. If $t_j \rightarrow t_k$ indicates that task t_j and t_k are connected by a link in the activity diagram, then we have:

$$x_k - (e_j + x_j) \geq 0; \forall t_j \rightarrow t_k \quad (6)$$

that is, if a task t_k is a direct successor of task t_j , then the execution of task t_k must start after task t_j termination.

Local constraints. Local constraints can predicate on properties of a single task and can be included in the model as follows. For example if the end-user requires that the price for task t_{j_1} has to be less or equal to 2\$ then the following constraint is introduced:

$$\sum_{i \in WS_{j_1}} p_{i,j_1} y_{i,j_1} \leq 2$$

Execution time constraint. As reported in Table 2, the duration of an execution path $exeTime_l$ is the maximum execution time over the set of sub paths of the execution path. The maximum value v_{max} of a set V is defined as the value in the set ($v_{max} \in V$) such that $v \leq v_{max}$, $\forall v \in V$. The execution time of the composite service can be expressed as:

$$\sum_{j \in sp_m^l} e_j \leq exeTime_l; \forall sp_m^l \in ep_l \quad (7)$$

and the execution time constraint can be expressed by the following set of equations:

$$exeTime_l \leq E; \forall l \quad (8)$$

Note that, in our formulation the execution time constraint is guaranteed for every sub paths and not only for critical paths as in [18].

Availability constraint. Under the hypotheses discussed in Section 4.2, the availability of an execution path is given by the product of availability of the selected services and can be expressed as:

$$avail_l = \prod_{j \in \mathcal{A}_l} \prod_{i \in WS_j} a_{i,j}^{y_{i,j}}; \quad (9)$$

and availability requirements for the composite service can be specified as:

$$avail_l \geq A; \forall l \quad (10)$$

Price and reputation constraints. From Table 2, price and reputation of the composite service can be expressed as:

$$price_l = \sum_{j \in \mathcal{A}_l} \sum_{i \in WS_j} p_{i,j} y_{i,j} \quad (11)$$

$$repl_l = \frac{1}{|\mathcal{A}_l|} \sum_{j \in \mathcal{A}_l} \sum_{i \in WS_j} r_{i,j} y_{i,j} \quad (12)$$

and corresponding constraints are expressed by the following set of equations:

$$price_l \leq B; \forall l \quad (13)$$

$$repl_l \geq R; \forall l \quad (14)$$

Optimization Problem Analysis. The Problem P1) has integer variables and a non-linear constraint family (the availability bounds expressed by equations (9) and (10)). Availability bounds can be linearized by applying the logarithm function. Equation (10) then becomes:

$$\sum_{j \in \mathcal{A}_l} \sum_{i \in WS_j} \ln(a_{i,j}) y_{i,j} \geq \ln(A) \quad \forall l \quad (15)$$

Note that A and $a_{i,j}$ are positive real numbers less or equal to 1, then their logarithm is negative. Hence the previous inequality can be written as a less or equal inequality with positive coefficients (the objective function is linearized in the same way). The constraints family (15) introduces capacity constraints [17] (less or equal inequality with positive coefficients). Also constraints (14) written as demand constraints (greater or equal inequality with positive coefficients) can be transformed into capacity constraints. The reputation r of a service is a real number in $[0, 1]$ and it is a positive criteria (the higher the value, the higher the reputation). We can consider the complementary reputation rc defined has $rc = 1 - r$ and write Equation (14) as follows:

$$\frac{1}{|\mathcal{A}_l|} \sum_{j \in \mathcal{A}} \sum_{i \in WS_j} rc_{i,j} y_{i,j} \leq RC \quad (16)$$

that is, intuitively, we can solve the WSC problem by introducing constraints on the reputation of candidate services or in the same way by expressing constraints on their untrustworthiness. Formally we can apply this transformation and constraints (14) and (16) are equivalent since we have the constraint (4), i.e., each task has to be associated with a Web service.

After transforming availability and reputation constraints P1) is equivalent to a Multiple choice Multiple dimension Knapsack Problem (MMKP) [1]. A MMKP is one kind of knapsack problem where the resources are multi-dimensional, i.e., there are multiple resource constrains for the knapsack (e.g., weight and volume) and items are classified in groups. Let there be n groups of items. Group j has n_j items. Each item of the group has a particular value and it requires resources. The objective of the MKKP is to pick exactly one item from each group for maximum total value of the collected items, subject to the resource constraints of the knapsack. In mathematical notation, let $c_{i,j}$ be the value of the i -th item in j -th group, $w_{i,j,p}$ be the amount of resource p required by the i item in the j group and W_p the amount of the p resource. Then the problem is:

$$\begin{aligned} \max & \sum_{j=1}^n \sum_{i=1}^{n_j} c_{i,j} y_{i,j} \\ \sum_{j=1}^n \sum_{i=1}^{n_j} w_{i,j,p} y_{i,j} & \leq W_p; \quad \forall p \\ \sum_{i=1}^{n_j} y_{i,j} & = 1; \quad \forall j \\ y_{i,j} & \in \{0, 1\}; \quad \forall i, j \end{aligned}$$

The WSC problem is a MMKP where each task corresponds to a group, each candidate service for a task is an item in a group and each capacity constraint of

the problem formulation corresponds to a resource p ($c_{i,j}$ and W_p can be related to web services quality dimensions and weights w_j by considering equations (3), (5), (8), (9), (11) and (12)). Every instance of the MMKP can be formulated as a WSC; hence WSC is NP-hard.

4.4 Re-optimization

In the MAIS platform the re-optimization is performed periodically, when a service invocation fails and for end-user channel switches. Re-optimization requires some information on the current state of the composite service execution. Re-optimization starts revising the process instance specification. The BPEL4WS engine provides the current running activity and the set of conditional branches which were verified during the process execution. In this way a portion of the composite service specification can be eliminated from the optimization problem: tasks which belong to conditional branches that were not verified during the composite Web service execution will never be executed and can be eliminated from the re-optimization problem. Vice versa, additional constraints can be introduced for tasks that have been completed.

In general the re-optimization process is less cumbersome than the optimization, since the DAG can be simplified, additional constraints can be added and the number of decision variables can be reduced.

4.5 QoS Optimization Tool Implementation

The Knapsack problem and its variants are very well known problems in the Operations Research literature which provides several methods to solve the MMKP. Problem instances of medium size can be solved anyway, by commercial integer linear programming solvers. The instances of the WCS problem we have considered have been solved through *CPLEX*, a state of the art integer linear programming solver. The solver identifies the optimum solution of the MKKP, i.e., for a given composite Web Service, the optimum global plan is identified.

The optimization/re-optimization tool has been implemented in Java. The input of the optimizer are the annotated BPEL4WS specification of the composite Web service and an XML file which specifies the set of candidate services, their quality values and the set of weights for quality dimensions specified by the end user (see Figure 3). First, cycles are unfolded and the BPEL4WS specification is translated into a graph internal representation. Execution paths are extracted from the DAG and a depth-first algorithm identifies the set of sub paths of every execution paths. These operations are implemented in the *Process Translator* module. The depth-first approach allows optimizing the sub path generation process. If for example two sub paths have a common path, then the second sub path can be obtained from the partial result computed from the first one.

Finally, MILP model constraints are generated and the optimization problem is solved by running *CPLEX*. Re-optimization is implemented in the *Process Tuner* module which has as input an XML file which specifies the state of the running process and the set of conditions verified during the process execution.

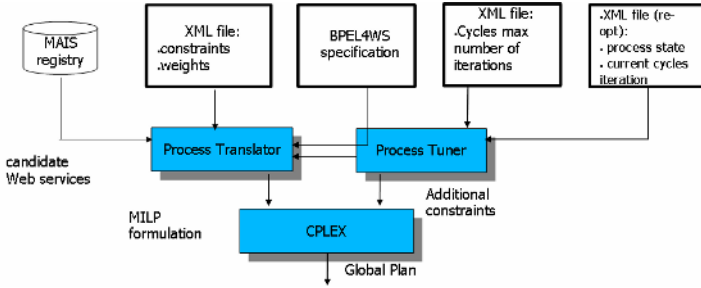


Fig. 3. Process Re-optimization

The Process Tuner module modifies the current instance process specification and generates additional constraints which assigns Web services for tasks already executed. The new specification is then sent to the Process Translator module. The problem formulation obtained from the Process Translator module and the additional constraints generated by the Process Tuner module are sent to *CPLEX* and a new solution is then obtained.

5 Experimental Results

The effectiveness of our approach has been tested on a wide set of randomly generated instances. The number of tasks has been varied between 7 and 70 by defining the set of tasks reported in Figure 1 in a loop and varying its maximum number of iterations. The number of candidate Web services per task has been varied between 10 and 50 with step 10. Quality of services values have been random generated according to the values reported in the literature. Availability was randomly generated assuming a uniform distribution in the interval 0.95 and 0.99999 (see [11]). Reputation was determined in the same way by considering the range [0.8, 0.99]. Web Service execution time is given by the sum of its service time (i.e., the time required for a single user request execution) and the service waiting time (the time a user has to wait in a queue while the service is accessed by other users). The service time was randomly generated assuming a uniform distribution in 0.5 sec and 8 sec as in [8]. The maximum queue length guaranteed by the SP was randomly generated assuming a uniform distribution in the range [1.5, 10] as in [19]. We assumed the price was proportional to service reputation and inversely proportional to the queue length (i.e., the higher the waiting time, the lower is the price). Analyses have been performed on a 3 GHz Intel Pentium IV Workstation, optimizations terminated in less than 2 sec. In order to evaluate the effectiveness of our approach we compared our solutions with the solutions provided by the local optimization approach proposed in [18]. For every test case, we first run the local optimization algorithm. Then, we perform our global optimization including as global constraints the value of the quality dimensions obtained by the local optimization. Results shows that the global optimization gives better results since bounds for quality dimensions can

be always guaranteed and the objective function is improved by 1-4%. Note that a limited improvement in the objective function corresponds to an improvement more significant in the value of the quality dimensions. As an example, Table 3 reports results of the solutions of some problem instances where the optimization function includes only service price and execution time and corresponding weights are set to 0.5. Results show that an improvement of few percent units in the objective function, corresponds to a greater improvement in quality dimensions. The small improvement in the objective function is due to the normalization introduced by the SAW technique since values are divided by terms $\max Q_j^l - \min Q_j^l$ which are usually large. On average global optimization allows improving local optimization results by 20-30%.

Table 3. Global and local optimization comparison

Global Optimization			Local optimization			Global vs. Local Optimization		
F. obj.	Price (\$)	Exec. Time (sec.)	F. obj.	Price (\$)	Exec. Time (sec.)	Δ F. obj	Δ Price	Δ Exec. Time
0.9455	5.90	736.84	0.9378	6.10	909.18	0.82%	3.47%	23.39%
0.9907	15.20	47.02	0.9903	16.89	51.72	0.05%	11.13%	10.00%
0.9900	12.80	48.41	0.9887	13.27	56.10	0.13%	3.69%	15.89%
0.9688	20.38	153.13	0.9415	24.29	256.84	2.90%	19.19%	67.72%
0.9671	20.33	127.26	0.9546	23.42	220.21	1.31%	15.23%	73.03%
0.9912	13.23	38.62	0.9885	14.40	54.01	0.27%	8.81%	39.86%
0.9800	18.91	105.72	0.9638	21.43	186.59	1.68%	13.32%	76.50%
0.9867	20.64	47.89	0.9799	22.45	91.51	0.69%	8.76%	91.09%
0.9859	6.65	33.53	0.9819	7.68	42.17	0.41%	15.43%	25.77%
0.9919	9.08	14.30	0.9905	10.50	19.24	0.15%	15.64%	34.56%

6 Conclusions

In this paper we have presented an optimization approach for the composition of Web services which allows specifying constraints on quality requirements for the end user both at local and global level. The optimization problem has been modeled as a mixed integer linear programming problem. We have shown that the problem is NP-hard, since it is equivalent to a MMKP. With respect to other literature approaches we identify the global solution instead of local optima [3] and we guarantee the fulfillment of global constraints [18].

Future work will consider re-optimization issues connected with cycles execution. In the current implementation, in order to satisfy global constraints, if the business process contains some cycles, they are unfolded and the optimization considers the worst case scenario, i.e. the maximum number of iteration of each cycle. Anyway, if a cycle is executed a lower number of times, then the Web service selection is sub-optimal. Future work will consider probabilistic execution of

cycles included in composite services specifications. Furthermore, the optimization of execution of multiple process instances will be considered. This is very critical since if a very large number of requestors are assigned to the same "best" service critical load conditions could be reached and the quality of service could degrade.

References

1. M. Akbar, E. Manning, G. Shoja, and S. Khan. Heuristic solution for the Multiple-Choice. In *Conference on Computational Science*, 2001.
2. D. Bianchini, V. D. Antonellis, B. Pernici, and P. Plebani. Ontology-based methodology for e-Service discovery. *Information Systems*, in press, 2005.
3. G. Canfora, M. Penta, R. Esposito, and M. L. Villani. A Lightweight Approach for QoS-Aware Service Composition. In *ICSOC 2004 forum paper*, 2004.
4. G. Canfora, M. Penta, R. Esposito, and M. L. Villani. An Approach for QoS-Aware Service Composition based on Genetic Algorithms. In *GECCO 2005 Proc.*, 2005.
5. G. Canfora, M. Penta, R. Esposito, and M. L. Villani. QoS-Aware Replanning of Composite Web Services. In *ICWS 2005 Proc.*, 2005.
6. C. Cappiello, P. Missier, B. Pernici, P. Plebani, and C. Batini. QoS in multichannel IS: The MAIS Approach. In *Workshop on Web Quality, Munich*, Jul. 2004.
7. J. Cardoso. Quality of Service and Semantic Composition of Workflows. PhD. Thesis, Univ. of Georgia, 2002.
8. G. Chafle, S. Chandra, V. Mann, and M. G. Nanda. Decentralized Orchestration of Composite Web Services. In *WWW 2004 Conference*, pages 134–143, 2004.
9. S. Chandrasekaran, J. A. Miller, G. Silver, I. B. Arpinar, and A. P. Sheth. Performance Analysis and Simulation of Composite Web Services. *Electronic Market: The Intl Journal of Electronic Commerce and Business Media*, 13(2):120–132, 2003.
10. H.C.-L and K. Yoon. *Multiple Criteria Decision Making*. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 1981.
11. A. Hiles. *E-Business SLA*. The Rothstein Catalog on Service Level Books, 2002.
12. Z. Maamar, Q. Z. Sheng, and B. Benatallah. Interleaving Web Services Composition and Execution Using Software Agents and Delegation. In *AAMAS*, 2003.
13. D. Menascé. QoS issues in Web Services. *IEEE Internet Comp.*, 6(6):72–75, 2002.
14. S. Modafferi, E. Mussi, A. Maurino, and B. Pernici. A Framework for Provisioning of Complex e-Services. In *IEEE Services Computing Conf.*, pages 81–90, 2004.
15. M. Ouzzani and A. Bouguettaya. Efficient Access to Web Services. *IEEE Internet Comp.*, 37(3):34–44, 2004.
16. P. Wohed, W. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. <http://tmitwww.tue.nl/research/patterns/download/bpel4er.pdf>, Feb. 2005.
17. L. Wolsey. *Integer Programming*. John Wiley & Sons, 1998.
18. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Trans. on Soft. Eng.*, May 2004.
19. L. Zhang and D. Ardagna. SLA Based Profit Optimization in Autonomic Computing Systems. In *ICSOC 2004 Proc.*, pages 173–182, 2004.

Web Service Discovery and Dynamic Invocation Based on UDDI/OWL-S*

JianJun Yu and Gang Zhou

National Laboratory of Software Development Environment,
Beihang University, Xueyuan Road No.37, Beijing, China
yujj@nlsde.buaa.edu.cn
gzhou@nlsde.buaa.edu.cn

Abstract. With the development of web technology and e-business, the Web Services paradigm has been widely accepted by industry and academic research. More and more web applications have been wrapped as web services, which triggers the change of research focus from finding and integrating them to maximizing reuse. Usually a service requestor retrieves (web) service advertisements out of a UDDI registry based on keywords, like in search engines, which inevitably brings the difficulty of discovering the most appropriate service from a massive number of existing services. This paper briefly introduces a new mechanism to discover and match services in a more fine-grained manner by taking advantage of UDDI and OWL-S. UDDI is used to discover approximate services syntactically by adding new elements and an API. OWL-S is used to match the exact service semantically using ontologies and inference. Finally this paper draws out a framework for discovering and invoking services dynamically.

1 Introduction

With the development of web technology and e-business, the web service has been widely accepted by industry and academic research as a distributed and loose-coupled component which is able to realize sharing and interchanging resources on web.

On the other hand, these services are universally distributed and of various kinds whilst with quite a diverse description. If there lacks a universal description, discovery and integration mechanism, certainly it's difficult to find and invoke the exact service. Therefore, how to find and integrate the exact service from massive services has become the current focus of research on service matching.

The service registry is an indispensable component in Service-Oriented Architecture. It provides description, discovery and integration of services through uniform API interface on the basis of UDDI protocol. Nevertheless, based on

* This research was supported by NSFC Grant 60403003 and FANEDD Grant 200241 of China.

syntactical structure, UDDI aims to make searches on keywords and classification. Thus, wide application of Web Services may be negatively affected for failing to well match the requirements from customers.

UDDI can hardly meet the requirement of automatic service discovery due to lack of semantic description of web service. While in OWL-S, ServiceProfile gives the semantic description of web service which helps to match the service precisely via ontology and inference mechanism. But the disadvantage is obvious with little supporting software.

By combining these two service discovery mechanisms mentioned above, this paper aims to use UDDI to make extensive inquiry and OWL-S to match service accurately in order to retrieve the exact service. This new mechanism provides a method to aggregate and classify massive loose-coupled and heterogenous services on Web. It enables the service provider to publish services more easily and the service requestor to discover, integrate and invoke services more quickly and succinctly.

This paper is organized as follows:

The second part introduces the mechanism of UDDI discovery as well as its advantages and disadvantages and then adds new elements and API to enhance service description and discovery. The third part introduces how ServiceProfile matches and infers the exact service. The fourth part brings in the framework of service discovery and dynamic invocation as well as its stereotype. The fifth part gives the related work. The final part is conclusion.

2 UDDI Discovery

UDDI [6] registry can be divided into two types: public UDDI registry and private UDDI registry. Corporations like IBM, Microsoft, SUN, SAP, etc. have established public UDDI registry for commercial purpose. Detailed information can be found in [7].

Private UDDI registry is generally applied to service integration within corporations which always built on open source package that enables service provider to have more control over service description, security and collaboration. UDDI registry discussed in this paper will be based on private UDDI registry.

UDDI defines four core data structures as service metadata, which are illustrated in Fig.1.

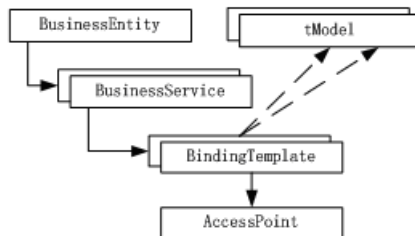


Fig. 1. UDDI core data structures

As is shown in Figure 1, BusinessEntity describes specific information about the service provider, including the name of corporation, contact information and taxonomy. One BusinessEntity can contain one or more BusinessService which advertises service capability with one or more BindingTemplate. BindingTemplate contains the service access point and refers to a series of tModel. tModel depicts the criteria and classification of a service. Services with the same function refer to the same tModel. Service requestor uses inquiry API to search tModel or BusinessEntity and obtain relevant registry information. There are three patterns for discovering web services: the browse pattern, the drill-down pattern and the invocation pattern. The browse pattern characteristically performs finding general information by keyword searching, classification and taxonomy. The drill-down pattern retrieves more specific information of the service using the UUID identifier. The invocation pattern involves getting an instance of the retrieved service.

Service provider advertises service capability with save_xxxx API. When creating a UDDI entry, service provider needs to refer existing tModel or create new tModel to classify the functionality of the service. Service requestor retrieves service description with find_xxxx and get_xxxxDetail APIs.

The core data structures and APIs give a universal method for service publishing and inquiry. We can use IBM uddi4j to interact with different UDDI registries.

Although UDDI gives a universal API to publish and inquire service information, its disadvantages are obvious:

- 1) Similar to a search engine, UDDI uses syntactic structures that lacks relevancy and semantic classification of services. However, in many cases, services advertised at UDDI are not exactly the same as their actual functionality or they are identified by abbreviation and synonymous words. Therefore, searching based on syntax cannot discover the right service easily.

- 2) In UDDI, there lacks sufficient metadata for service description to support automatic service discovery. As for a program, it will be easier to understand the capability and performance of the services by retrieving both the functionality and non-functionality information from UDDI registry. Apparently, current structures of UDDI need improved and added more description information to support further service discovery.

- 3) More APIs are needed in UDDI to get the specified information. Now, service requestor needs to call find_tModel, get_tModel, find_bindingTemplate and get_bindingTemplate APIs to get the service access point. This process is not necessarily and reduces the search efficiency. So, new APIs should be defined to get the specified information directly. For example, service requestor can get the access point from tModel through find_wsdl_with_tModel API.

- 4) UDDI registry accepts service information passively, which means that if a service changes and so does access point without updating service description in the registry, service requestor will probably use false information and fail to invoke the service.

Therefore current UDDI cannot entirely meet the requirement of exact web service discovery and dynamic service invocation. Service requestor hopes that they can retrieve the right service and satisfy reliable and high-efficient invocation by merely describing their needs and requirements. When one service is down, service requestor can choose another service to guarantee uninterrupted service invocation.

This paper improves core data structures and adds one API to enhance the veracity and efficiency of UDDI inquiry:

1) Add ServiceProperty in BusinessService as non-functional description and Add ServiceInterface in tModel as functional description. ServiceProperty mainly includes the name, type and value of the property. The data structure is as follows:

```
<ServiceProperty PropertyName=""
  PropertyType="" PropertyValue="" />
```

BusinessService can have one or more ServiceProperty which are used to describe the non-functionality of the service, such as the response time of the service, the rate of CPU occupancy and the maximal process count of the service.

ServiceInterface is used to describe the functionality of a service. The data structure is as follows:

```
<ServiceInterface>
  <ServiceMethod name="">
    <Parameters>
      <Param name="" type="" />
    </Parameters>
    <return name="" type="" />
  </ServiceMethod>
</ServiceInterface>
```

ServiceInterface has one or more child nodes named ServiceMethod which is used to describe the name of method, the name and type of input and output.

2) Add find_wsdl_with_tModel API. Retrieve the access point of the service by inquiring the name of tModel or matching the ServiceInterface structure. The data structure is as follows:

```
<find_wsdl_with_tModel>
  [<tModelName/>[<tModelName />]...]
  [ServiceInterface]
</find_wsdl_with_tModel>
```

3) Add a UDDI Monitor used to monitor the service state at real-time in uddi4j. It can route to the most appropriate service according to the service state.

3 OWL-S Matching

Semantic Web Service mainly use OWL-S [8] as its protocol which enables web services to be machine understandable, including service dynamical discovery, invocation, composition and monitoring. OWL defines three upper ontologies, including ServiceProfile, ServiceModel and ServiceGrounding. ServiceProfile defines what the service does; that is, it gives service description needed by a service requestor to determine whether the service meets his requirement. ServiceModel defines how the service works; that is, it describes the workflow and possible execution paths of the service. ServiceGrounding specifies the details of how to access a service. Typically it will specify a communication protocol and service-specific details such as port numbers used in contacting the service.

ServiceProfile describes three types of service information: service provider information, functionality and non-functionality. The service provider information consists of contact information about the service. The functionality of the service is expressed in terms of inputs, outputs, preconditions and effects. These features assist when reasoning about several services with similar capabilities. Non-functionality of the service means service QoS and service state, including service response time, service process ability. [9] gives detailed information about service non-functionality.

In OWL-S, service matching is based on ServiceProfile using ontologies. Service provider and service requestor refer to OWL ontologies, and then the match-

Table 1. Translation between UDDI and ServiceProfile

ServiceProfile Element	UDDI Element
Actor:name	Contacts:PersonName
Actor:phone	Contacts:phone
Actor:fax	/
Actor:email	Contacts:email
Actor:physicalAddress	Contacts:address
Actor:WebURL	discoveryURLs
serviceName	BusinessService:name
intendedPurpose	ServiceProperty
textDescription	BusinessService:description
requestedBy	Contacts
providedBy	Contacts
serviceType	ServiceProperty
serviceCategory	CategoryBag
communicationThru	ServiceProperty
qualityGuarantee	ServiceProperty
qualityRating	ServiceProperty
Input	ServiceInterface or WSDL
Output	ServiceInterface or WSDL
Precondition	/
Effect	/

ing engine can perform inferences to recognize semantic token despite different syntax and model between service provider and service requestor.

[1] gives matching algorithm mainly based on input and output of the service.

Just like UDDI, ServiceProfile also provides the metadata for services. However, since OWL-S language is not mature and ontology definition is not authoritative, there is inevitably few supporting software and practical application.

This paper uses UDDI2SWSPro(based on OWL-S/UDDI MatchMaker [11]) to create ServiceProfile for the subsequent service matching based on current UDDI. If ServiceInterface is published in UDDI, this structure will be used to create functionality in ServiceProfile. If not registered, WSDL is used to create functionality in ServiceProfile. The contact information in ServiceProfile is created by Contacts in UDDI. The non-functionality in ServiceProfile is created by ServiceProperty. To be more specific, corresponding relation between ServiceProfile and UDDI is as Table 1.

4 Framework and Prototype

The framework mainly uses improved mechanism mentioned above to realize service discovery and matching, and then puts forward a prototype to support service dynamic invocation based on the result services founded using UDDI and OWL-S.

User's requirements are classified into three types: 1) The name of the service; 2) The functionality of the service; 3) The non- functionality and QoS of the service. This paper uses service name to achieve service discovery, input and output class in ServiceProfile to achieve service functional matching, and ServiceProperty to achieve service non-functional matching.

To get the exact service, the following steps should be adhered to:

1) Use improved UDDI to inquire about services extensively, get their UDDI entries and store them in UDDIEntry DB, retrieve their WSDL documents and store them in WSDL DB. Private UDDI receives the registration of services and retrieves entries from public UDDI registries at the same time using subscription API. The functionality of the service is described by ServiceInterface in tModel, and non-functionality of the service is described by ServiceProperty in BusinessService. The service name is used to inquire tModel and BusinessService. Therefore the inquiry will be mapped to two API inquiries: `find_service` and `find_wSDL_with_tModel`.

2) Use UDDI2SWSPro to create contact information, non-functionality and functionality in ServiceProfile from UDDIEntry DB and WSDL DB.

3) Use matching engine to match functionality and non-functionality of the services, and then rank the founded services according to the matching degree.

The system architecture is described as Fig.2.

The resulting web services are stored in UDDI Monitor. UDDI Monitor is used as a service subscriber and service dispatching center. ServiceSubscription receives all updated web services information from UDDI registry immediately which may have already disabled or cannot satisfy actual needs in function. All

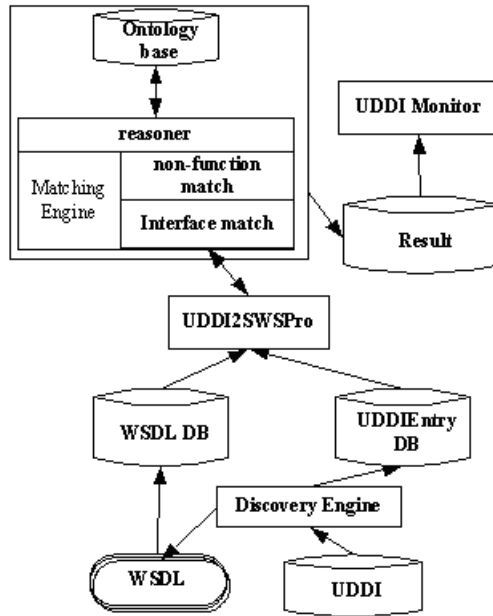


Fig. 2. Discovery and Matching Framework

services received from ServiceSubscription will be put into a service pool, which is classified into active service pool and inactive service pool. Active service pool caches all the services in operation; inactive service pool stores disabled services or services that should be activated. The ServiceListener manages the service pools by polling service state information periodically. If the service is disabled, it will be put into inactive service pool. If the current state of the service cannot satisfy the user's requirement, for example, the load is too heavy, then another available service in active pool will be released to respond the client request. The priority of the current service will decline and the service will be put to the end of the active queue. If a disabled service becomes activated, it will be released from inactive queue and put to the end of the active queue.

When a service is invoked, it will be changed the previous access point by ServiceProxy and routed to the ServiceSelector. ServiceSelector decides whether the first service in the current active service pool satisfies the requirement. If it satisfies the requirement, it will be taken out and invoked; if not, the second service will be taken out instead till the requirement of the user is satisfied.

For example, service provider transfers the service to another server, and changes its service access point in UDDI registry. ServiceSubscription receives the notification immediately, and then changes the access point of the exact service in service pool. Then the next request will be routed to the new service address dynamically. During the process of invocation, if the current web service breaks down, the ServiceListener will mark this service as an inactive service, and then put it to the end of the inactive queue. It will not be marked as an

active service until it recovers to normal state. If the capability of current service is poor, the ServiceSelector will choose another suitable service to invoke.

The framework described as Fig.3.

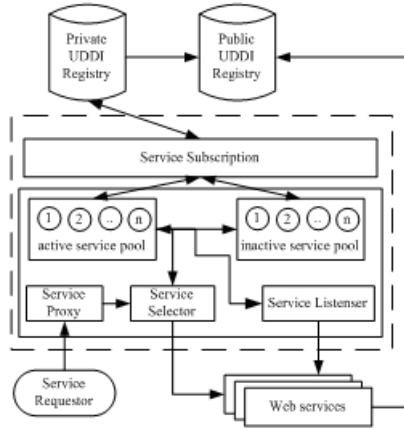


Fig. 3. UDDI Monitor framework

5 Related Work

Currently most of work distributed to service finding uses UDDI or OWL-S. But the literature on Web Services and Semantic Web is not abundant for a specified area, that is, UDDI basically uses syntactic service discovery and OWL-S uses semantic service discovery. They should be merged together to meet the user's requirement using enhanced descriptions.

[4] gives an enhanced UDDI to record user's defined properties. [5] gives an active UDDI allowing the extension of UDDI API to enable fault-tolerant and dynamic service invocation. They all should define more metadata and APIs to support UDDI translated to ServiceProfile and make service discovery more accurate.

[1] gives semantic matching of web services capability which shows how service capabilities are presented in the ServiceProfile and how a semantic matching between advertisements and requests is performed. [2] presents basic requirements for service discovery and evaluates current web service technology (WSDL, UDDI, DAML-S) that is used to address these requirements. [3] gives semantic API matching for automatic service composition. In this paper, these works are analyzed and summarized to give a universal solution for service matching.

6 Conclusion

Web service discovery and matching is an important aspect in Web Services oriented technology. Current available industry standards such as UDDI and

WSDL can only support syntactic service discovery which will not be sufficient for exact service matching. OWL-S gives a semantic service matching which can find the service exactly. However, OWL-S is still in its infancy and a lot of work has to be done in order to overcome its limitations and problems.

This paper combines UDDI with OWL-S to give a solution for service discovery and matching, which uses UDDI to discovery extensively and OWL-S to match service accurately. When the result services are invoked, this paper gives a framework and prototype extended from previous work in [10] to assure the services uninterrupted invocation at runtime.

References

1. Paolucci, M., Kawamura, T., Payne T.: Sematic Matching of Web Services Capability. ICWS 2002
2. Pilioural, T., Tsalgatidou, A.: Using WSDL/UDDI and DAML-S in Web Service Discovery. ESSW 2003
3. Caragea, D., Syeda-Mahmood, T.: Semantic API Matching for Automatic Service Composition. WWW 2004. **2** 436
4. ShaikhAli, A., Rana, O., Al-Ali, R.: UDDIe: An Extended Registry for Web Services. SAINT-2003 IEEE Computer Society Press
5. Jeckle, M., Zengler, B.: Active UDDI - An Extension to UDDI for Dynamic and Fault-Tolerant Service Invocation. In 2nd Annual International Workshop of the Working Group on Web and Databases of the German Informatics Society (GI). October 2002
6. UDDI: UDDI Version 3.0. <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>. 2002
7. UDDI Business Registry Node URLs. <http://www.uddi.org/find.html>.
8. OWL-S 1.1. <http://www.daml.org/services/owl-s/1.1/>.
9. O'Sullivan, J., Edmond, D., and Hofstede, A.: What's in a service? Towards accurate description of non-functional service properties. Distributed and Parallel Databases Journal - Special Issue on E-Services, **12** (2-3) 117-133. 2002
10. Yu, J., Zhou, G.: Web Service Dynamic Invocation Based on UDDI. CEC'04 EAST
11. Srinivasan, N., Paolucci, M., and Sycara, K.: An Efficient Algorithm for OWL-S based Semantic Search in UDDI. SWSWPC 2004. San Diego, California, USA, 2004

Preface

(WSCOBPM 2005)

This workshop addresses research around methods, concepts, models, languages and technologies for choreography and orchestration of Web services with special focus on Web service technologies and solutions for business process management.

Despite different focal points, the general objective of business process management (BPM) technologies is to provide support for business experts to model, monitor, and optimize processes on the business level, whereby the underlying technological details should be hidden from the business experts and adapted automatically according to the actions performed on the business level. Current BPM solutions lack support for automated integration and composition of functionalities, and most technologies only provide partial support for BPM. The manual effort to map high-level business process models to actual software implementations is still rather high, thus existing BPM technologies cannot be considered as mature to support BPM.

In order to overcome the deficiencies of current BPM technologies, Web services and service-oriented architectures (SOA) have been identified as the basic technical building block for the next generation of Web-based business solutions. A Web service offers a modular functionality, and has a seamless usage interface that hides technical details from a user client. Web services technologies allow automated discovery, composition, contracting, and execution of Web services, thereby providing a new technology for information systems. The current Web service technology stack allows exchange of messages between Web services (SOAP), describing the technical interface for consuming a Web service (WSDL), and advertising Web services in a registry (UDDI). However, these technologies do not explicitly describe all aspects of a Web service's functionality; neither do they provide support for the Semantic Web, i.e., descriptions on the meaning of the information to be interchanged between a client and a Web service. Consequently, the emerging concept of "Semantic Web Services" aims at providing more sophisticated support for automated discovery, composition, execution, and monitoring/management of Web services.

Choreography is concerned with interaction and conversation of Web services, wherein languages, communication technologies, formal models along with techniques for operations like service compatibility determination or validity checking of conversation protocols are of interest. Orchestration is concerned with arrangement of several services to more complex functionalities, wherein mainly service composition is of interest. Choreography and orchestration with Web services are considered as the enabling technologies of Web service-based process management.

The set-up of the workshop intersected the research fields of BPM and Web services. The workshop addressed researchers, professionals, and industrial practitioners, aiming at establishing a starting point for closer collaboration and exchange in future work.

Michael Stollberg
Dumitru Roman
Alistair Duke
Christoph Bussler

Organization

Organization Committee

Christoph Bussler

Digital Enterprise Research Institute (DERI)
National University of Ireland

Alistair Duke

Next Generation Web Research
British Telecommunications plc (BT)

Dumitru Roman

Digital Enterprise Research Institute (DERI)
Leopold-Franzens Universität Innsbruck

Michael Stollberg

Digital Enterprise Research Institute (DERI)
Leopold-Franzens Universität Innsbruck

Program Committee

Wil van der Aalst (Eindhoven University of Technology, The Netherlands)

Michael Altenhofen (SAP, Germany)

Boualem Benatallah (University of New South Wales, Australia)

Liliana Cabral (Open University, UK)

Jessica Chen-Burger (University of Edinburgh, UK)

John Davies (British Telecom, UK)

Marin Dimitrov (Ontotext, Bulgaria)

John Domingue (Open University, UK)

Manfred Hauswirth (EPFL, Switzerland)

Martin Hepp (DERI Innsbruck, Austria)

Dimka Karastoyanova (TU Darmstadt / University of Stuttgart, Germany)

Akhil Kumar (Penn State University, USA)

Olivier Perrin (University of Nancy 2, France)

Marco Pistore (University of Trento, Italy)

Axel Polleres (DERI Innsbruck, Austria)

Chris Priest (HP, UK)

Steve Ross-Talbot (Enigmatec Corporation, UK)

Ioan Toma (DERI Innsbruck, Austria)

Laurentiu Vasiliu (DERI, Ireland)

Alexander Wahler (Niwa Web Solutions, Austria)

Mathias Weske (HPI, Germany)

Michal Zaremba (DERI, Ireland)

Standards for Web Service Choreography and Orchestration: Status and Perspectives

Alistair Barros¹, Marlon Dumas², and Phillipa Oaks²

¹ SAP Research Centre, Brisbane, Australia
alistair.barros@sap.com

² Queensland University of Technology, Australia
{m.dumas, p.oaks}@qut.edu.au

Abstract. Web service composition has been the subject of a number of standardisation initiatives. These initiatives have met various difficulties and had mixed degrees of success, and none of them has yet attained both de facto and de jure status. This paper reviews two of these initiatives with respect to a framework wherein service composition is approached from multiple interrelated perspectives. One conclusion is that standardisation initiatives in this area have not been built on top of an explicitly defined overarching conceptual foundation. The paper outlines a research agenda aimed at identifying requirements and concepts that should be addressed by and incorporated into these standards.

Keywords: web service, web service composition, standard.

1 Introduction

There is an increasing acceptance of Service-Oriented Architectures (SOA) as a paradigm for integrating software applications within and across organisational boundaries. In this paradigm, independently developed and operated applications are exposed as (Web) services which are then interconnected using a stack of standards including SOAP, WSDL, UDDI, WS-Security, etc.

Standardisation is a key aspect of the uptake of the Web services paradigm. Web services standardisation initiatives such as SOAP and WSDL, as well as the family of WS-* specifications (e.g. WS-Policy, WS-Security, WS-Coordination) aim at ensuring interoperability between services developed using competing platforms. Standards in this area can be divided into various groups including:

- Transport: based mainly on HTTP(S) and SMTP.
- Formatting: based mainly on XML and XML Schema.
- Messaging: based on SOAP and various WS-* specifications (e.g. WS-Addressing, WS-Security and WS-Reliable-Messaging).
- Coordination and context: including yet-to-be standardised specifications such as WS-Coordination, WS-Atomic-Transaction, etc.
- Structure and policy description: based on WSDL and WS-Policy (which acts as a placeholder for elements defined in other WS-* specifications).
- Process-based service composition.

The standards in the latter category deal with the interplay between services and business processes. A number of discontinued standardisation proposals in this category have been put forward (e.g. WSFL, XLang, BPML, WSCL, and WSCI), leading to two ongoing standardisation initiatives: the Business Process Execution Language for Web Services (BPEL4WS or BPEL for short) [2] and the Web Services Choreography Description Language (WS-CDL) [9]. The attention raised by this category of standards hints at the fundamental links that exist between business process management and SOA. On the other hand, it is striking that despite all the efforts put into them, none of these initiatives has attained both *de jure* and *de facto* adoption. For example, in BPEL, for which several more or less complete implementations exist, it is challenging to build non-trivial compositions that can be interchanged between different implementations.

Service composition covers three distinct but overlapping viewpoints:

- *Behavioural interface* (also called *abstract process* in BPEL and *collaboration protocol profile* in ebXML): This viewpoint captures the behavioural dependencies between the interactions in which a given individual service can engage or is expected to engage. We distinguish two types of behavioural interfaces: *provided* (i.e. “as-is”) behavioural interfaces capturing “what a service actually provides” and *expected* (i.e. “to-be”) behavioural interface capturing what a service is expected to provide in a given setting.
- *Choreography* (also called *global model* in WSCI and *multiparty collaboration* in ebXML¹): This viewpoint captures collaborative processes involving multiple services and especially their interactions seen from a global perspective.
- *Orchestration* (also called *executable process* in BPEL): This viewpoint deals with the description of the interactions in which a given service can engage with other services as well as the internal steps between these interactions.

The next section provides more precise definitions and examples for these viewpoints. Section 3 discusses some of the issues that remain unresolved in relation to the standards for choreography and orchestration. Finally, we outline some directions for further research and development in Section 4.

2 Viewpoints in Service Composition

In this section, we present several viewpoints from which behavioural models for service composition can be captured and the relations between these viewpoints.

2.1 Choreography

A *choreography model* describes a collaboration between a collection of services to achieve a common goal. It captures the interactions in which the participating services engage to achieve this goal and the dependencies between these interactions, including: causal and/or control-flow dependencies (i.e.. that a given

¹ <http://www.ebxml.org>

interaction must occur before another one, or that an interaction causes another one), exclusion dependencies (that a given interaction excludes or replaces another one), data-flow dependencies, interaction correlation, time constraints, transactional dependencies, etc.

A choreography does not describe any internal action of a participating service that does not directly result in an externally visible effect, such as an internal computation or data transformation. A choreography captures interactions from a global perspective meaning that all participating services are treated equally. In other words, a choreography encompasses all interactions between the participating services that are relevant with respect to the choreography's goal.

A choreography of a well-understood service interaction scenario is shown in the form of an UML activity diagram² in Figure 1. Three services are involved in this choreography: one representing a “customer”, another one a “supplier” and a third one a “warehouse”. The elementary actions in the diagram represent business activities that result in messages being sent or received. For example, the action “order goods” undertaken by the customer results in a message being sent to the supplier (this is described as a textual note below the name of the action). Of course, every message sending action has a corresponding message receipt action but to avoid cluttering the diagram, only the sending or the receipt action (not both) are shown for each message exchange. For example, the action “send RFQ to Supplier” in activity “Request Quote” implies that there is a corresponding action “receive RFQ from Customer” on the Supplier's side, but this latter action is not shown in the diagram.

Note that Figure 1 does not include the activities and alternative paths required to deal with errors and exceptions that one could realistically expect in the scenario in question. Including this information would add considerably to the complexity of the model.

2.2 Behavioural Interface

A *Behavioural interface* captures the behavioural aspects of the interactions in which one particular service can engage to achieve a goal. It complements structural interface descriptions such as those supported by WSDL, which capture the elementary interactions in which a service can engage, and the types of messages and the policies under which these messages are exchanged.

A behavioural interface captures dependencies between elementary interactions such as control-flow dependencies (e.g. that a given interaction must precede another one), data-flow dependencies, time constraints, message correlations, and transactional dependencies, etc. It focuses on the perspective of one single party. As a result, a behavioural interface does not capture “complete interactions” since interactions necessarily involve two parties. Instead, a behavioural interface captures interactions from the perspective of one of the participants and can therefore be seen as consisting of communication actions performed by that participant. Also, behavioural interfaces do not describe internal tasks such as internal data transformations.

² <http://www.uml.org>

Figures 2, 3 and 4 show examples of behavioural interfaces corresponding to the supplier, warehouse and customer roles in the choreography of Figure 1.

Note that a role defined in a choreography may be associated with multiple behaviours and multiple WSDL interfaces. Moreover, for a given role in a choreography, an arbitrary number of behavioural interfaces may be defined that would provide the same functionality but not necessarily using the same interactions or the same order of interactions. For example, in Figure 2 the shipping order is sent to the warehouse in a parallel thread to the one where the payment details are received from the customer. An alternative would be that payment is received from the customer before the shipping order is sent out.

Depending on whether an interface captures an “as is” or a “to be” situation, a distinction can be made between *provided* and *expected* (or *required*) interfaces. A provided (behavioural) interface is an abstraction of the way a given service interacts with the external world. On the other hand, an expected (behavioural) interface captures an expectation of how a service should behave in order to play a given role in a choreography. Thus, an expected interface corresponds to a contract that a given party needs to fulfill to successfully collaborate with other parties. Ideally, the provided and expected interfaces of a service coincide. In practice however, it may happen that the interface provided by a service is different from the interface that it is expected to provide in a given scenario. In this case, the provider of the service is responsible for mediating between the interface that it is expected to provide, and the one that it actually implements. This mediation (or *adaptation*) process has been the subject of several research efforts [17,4].

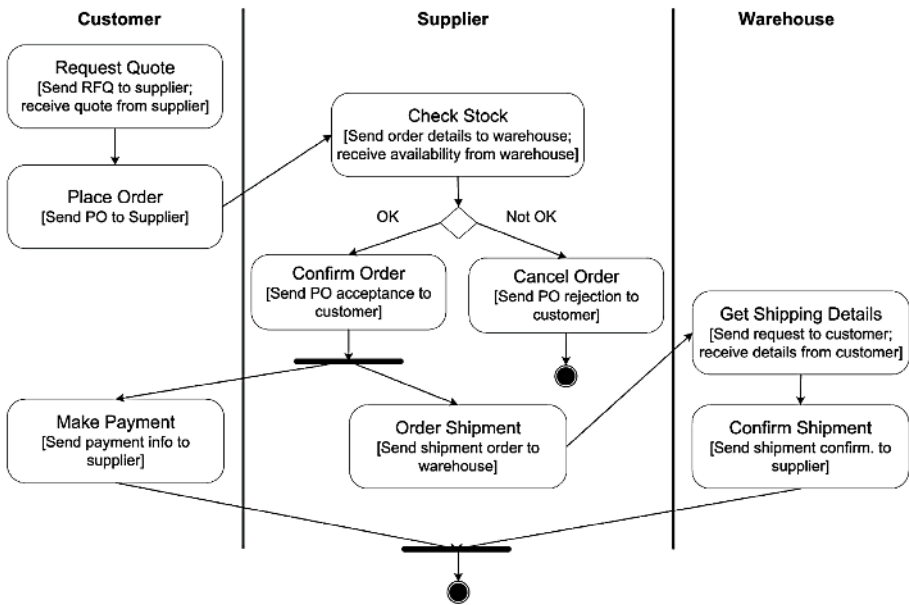


Fig. 1. Order processing scenario. Partly inspired from an example in [1].

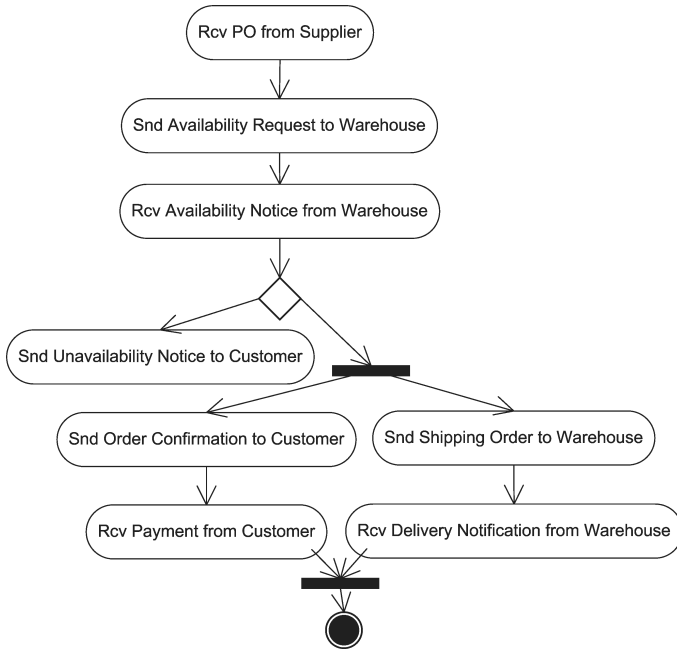


Fig. 2. Supplier behavioural interface

Another way to understand the distinction between provided and expected interfaces is to think of the provided interface as being linked to a service and possibly derived from the service’s orchestration (e.g. the orchestration shown in Section 2.3), while an expected interface is linked to a role of a choreography and possibly derived from this choreography (e.g. the interface in Figure 2 which can be seen as derived from the choreography in Figure 1).

The distinction between provided and expected interfaces is not present in existing Web services standards. In fact, some may argue that this distinction falls outside the scope of these standards. Indeed, the same language (e.g. the abstract process part of BPEL) can be used for describing both provided and expected interfaces. Nonetheless, from a methodological point of view it is important to keep this distinction in mind.

2.3 Orchestration

An *orchestration model* describes both the communication actions and the internal actions in which a service engages. Internal actions include data transformations and invocations to internal software modules. An orchestration may also contain communication actions or dependencies between communication actions that do not appear in any of the service’s behavioural interface(s). This is because behavioural interfaces may be made available to external parties and thus, they only need to show information that actually needs to be visible to

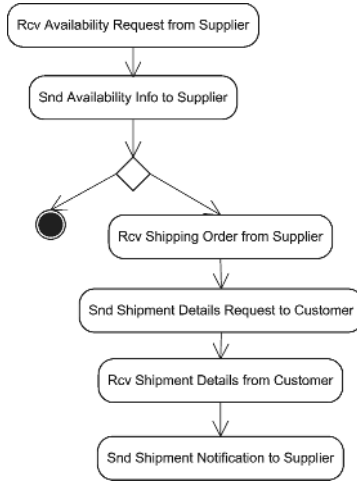


Fig. 3. Warehouse behavioural interface

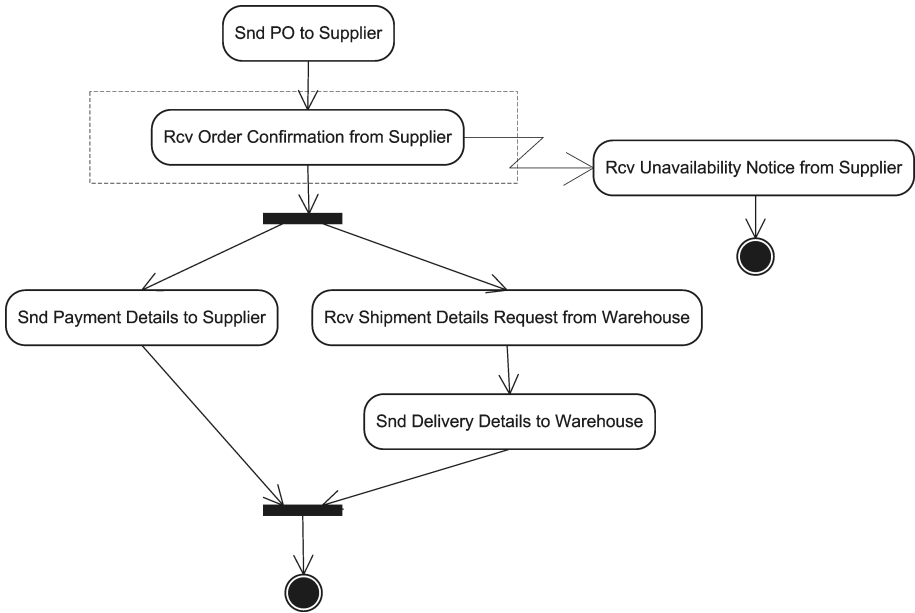


Fig. 4. Customer behavioural interface

these parties. Orchestrations are also called “executable processes” since they are intended to be executed by an *orchestration engine*.

Figure 5 shows an orchestration of a supplier service. This orchestration includes an internal action for validating the payment, shown in dotted lines in the diagram. This may correspond for example to an interaction with a service

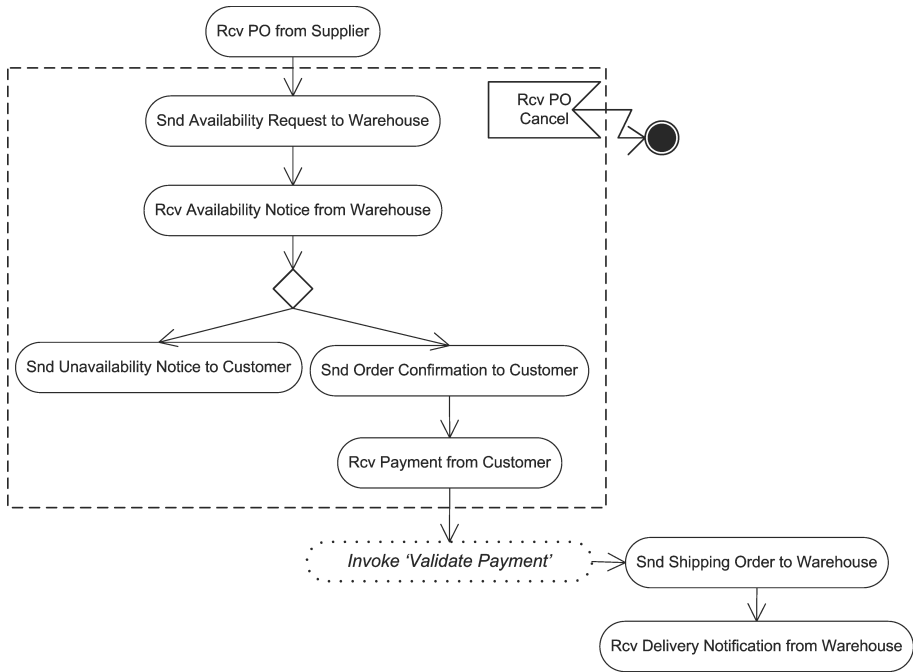


Fig. 5. Supplier service orchestration

that is not exposed to the outside world. Other internal actions may be included in this orchestration. The orchestration of Figure 5 also supports the possibility of an order cancellation request being received from the customer anytime before the payment, leading to termination of the process.

2.4 Relations Between Viewpoints

The viewpoints presented above overlap. This overlap can be exploited within service composition methodologies to perform consistency checks between viewpoints or to generate code. For example, an expected behavioural interface can be used as a starting point to generate an “orchestration” skeleton that can then be filled up with details regarding internal tasks and refined into a full orchestration. This has the advantage that once the orchestration is fully refined, the provided behavioural interface of the service will coincide with the expected behavioural interface. On the other hand, an existing orchestration can be used to generate the provided behavioural interface of a service by appropriately hiding actions not to be exposed. The resulting provided behavioural interface can then be checked for consistency against an expected behavioural interface. In this way, it is possible, for example, to detect situations where a given service does not send messages in the order in which these are expected by other services with which it is required to collaborate. These mismatches can then be resolved either

by altering the orchestration or by building a wrapper that mediates between the provided and the expected behavioural interfaces.

Similarly, a choreography model can be used for the following purposes:

- To generate the expected behavioural interface for each service intending to participate in the collaboration. This expected behavioural interface can then be used during the development of the services in question as outlined above. For example, given the choreography of Figure 1, it would be possible to derive the behavioural interface expected from the supplier service (and same for the customer or the warehouse).
- To check (at design time) whether the behavioural interface of an existing service conforms to a choreography and thus, whether the service in question would be able to play a given role in that choreography.
- To generate the skeleton of an orchestration model for each participant, with the internal actions to be added as necessary by the relevant role.

Formal definitions of the above service modelling viewpoints and their relations can be found in [6]. Informal definitions are given in [15] and [1].³

3 Status and Issues with Current Standards

At present, the set of web services standards that are able to support the representation of design-time information include WS-CDL which is intended to cover the choreography viewpoint, BPEL which is intended to cover both the orchestration and the behavioural interface viewpoints, and WSDL (used in conjunction with XML Schema) which is targeted at describing the structural aspects of interfaces. Another specification, namely WS-Policy, serves as a placeholder for capturing interface-level information not covered by WSDL and BPEL, like for example reliability, security, and transactional capabilities of a service.

However, in order to enable the vision of a standardised approach to service-oriented design, it is necessary to address a number of issues, most of which are related to the inter-connection between the choreography and the interface viewpoint. Below, we summarise some of these issues.

Formal grounding. One of the core requirements for WS-CDL as defined in its charter⁴ is to provide a means for tools to validate conformance to choreography descriptions in order to ensure interoperability between collaborating web services. Such static conformance checking would be facilitated if WS-CDL was based on, or related to, a formal language for which validation techniques are already in place. Unfortunately, although WS-CDL appears to borrow terminology from pi-Calculus [11] there is no comprehensive mapping from WS-CDL to pi-calculus or any other formalism. Even if a formalisation of WS-CDL was undertaken in the future, it would be an *a posteriori* exercise rather than an *a*

³ In [1], a choreography is called a *coordination protocol* and a *provided interface* is called a *role-specific view of a coordination protocol*.

⁴ <http://www.w3.org/TR/2004/WD-ws-chor-reqs-20040311>

priori effort to ensure the coherence and consistency of the language. This has been recognized in ongoing initiatives such as WSMO [16] which adopts a more formal approach to choreography modelling using Abstract State Machines.

In the case of BPEL, providing the means for enabling conformance validation or other semantic verification, is not within the standardisation initiative's scope. Nonetheless, a number of efforts outside the standardisation initiative itself have aimed at providing formal semantics to various subsets of BPEL in terms of finite state machines [8], process algebras [10], abstract state machines [7], and Petri nets [12,13]. Some of these formalisations can be used to statically check semantic properties of orchestrations or to check that a given behavioural interface (defined as a BPEL abstract process) conforms to a BPEL orchestration. WofBPEL [13] for example uses Petri net analysis techniques to statically detect dead actions (i.e. actions that will never be executed) or actions that may compete for the same message, in a given BPEL orchestration.

Lack of explicit meta-model. Both BPEL and WS-CDL fail to define an abstract syntax separately from their concrete (XML) syntax. An abstract syntax in this setting can take the form of a service behaviour meta-model, that is, a model whose instances correspond to service behaviour models formulated from either the choreography, interface or orchestration viewpoint. We advocate that a service behaviour meta-model should be developed independently of a particular interchange format. An explicitly defined meta-model sets the stage not only for the definition of an interchange format but also for the definition of corresponding modelling notation(s) as well as model transformations. This is especially important in the case of choreography modelling, since choreographies are more a design than an implementation artefact and thus a visual modelling notation for service choreographies is likely to be more useful than an XML syntax (although the latter may be useful for interchange purposes).

Multi-party interactions. Close inspection of WS-CDL's and BPEL's expressive power suggests that they were developed with basic assumptions of process orchestration expressiveness, and therefore a basic level of messaging supported by this functionality. Interactions occur between pairs of roles or across partner links that, at a given point in time, link one party to another. In other words only binary interactions are supported. Missing is the explicit support for multi-party interactions and more complicated messaging constraints which these bring.

Some key requirements to consider are those which emerge in multi-party scenarios. One is multiple instances of interactions which can arise for the same interaction types at the same time. For example, the processing of a purchase order can involve several competing suppliers (known only at runtime due to the specific content of the purchase). Responses might be time-critical and all suppliers might be required to receive the request and respond within a specified duration. The preparation of requests, the sending of requests and the receipt of responses might need to be done in parallel. There might be a constraint over the number of suppliers that are required to successfully receive the request in order for the overall issue of the request to go-ahead. Moreover, when the number of suppliers is large, assumptions about the number of responses need

to be relaxed. A minimum number might be required, before further steps in the process are taken, while any remaining responses might be ignored.

Such a scenario is not unusual in real-scale B2B applications involving large numbers of parties, which require sophisticated orchestration support. In particular, this type of scenarios require support for multi-party and multi-instance interactions, competing interactions, atomicity constraints on interactions, and partial synchronisation of responses. In fact, one such scenario was discussed in the collection of use cases during the Web Services Choreography group's requirements gathering⁵. As it stands it is unclear how WS-CDL can conveniently support these sorts of multi-party interactions without serialising the interaction and/or using low-level book-keeping mechanisms based on arrays and counters. Workflow languages, capable of supporting multi-party instances, would be constricted by the single instance, binary interactions supported in WS-CDL.

Relationships between standards. Positioned over the web services composition layer of the Web Services stack, WS-CDL and BPEL are required to interoperate with a number of web service standards, notably WSDL and WSDL-MEPs for static service binding, and WS-Reliable-Messaging for lower level quality of messaging. Yet the mapping remains open, and conceptual sufficiency in aligning WS-CDL, in particular, with these standards is arguably limited. Consequently, the mapping of WS-CDL and BPEL to the eight WSDL 2.0 Message Exchange Patterns (MEPs) is yet to be precisely determined.

In terms of messaging quality of service, WS-CDL relies on WS-Reliable-Messaging principally among other standards from the Web Services stack (e.g. others might be WS-Addressing). The extent of quality of service messaging on which WS-CDL depends is not fully established, and the mapping for reliable messaging at the very least remains open. In general, no a priori configurability of WS-CDL specifications for different quality of messaging service is in place. This in our view limits the layering and exploitation of choreography for lower level services from current and oncoming messaging standards.

Also, it remains open how WS-CDL's "Workunit" construct can be mapped in WS-BPEL. Here we refer to the "blocked wait" feature of this construct, that occurs when the "block" condition associated to a WorkUnit evaluates to true. In this case, an activity is allowed to proceed once an interaction or variable assignment action, which may occur in a completely different part of the choreography, supplies the required data. It is not clear how this would be mapped in terms of WS-BPEL's Pick and Switch constructs or what the complexity of the mapping would be.

More generally, the relationships between choreography and behavioural interface (i.e. "abstract process" in WS-BPEL) may be non-trivial, and there are currently no precise notions of conformance between WS-CDL choreographies and WS-BPEL abstract processes. Understanding these relations is crucial if these two specifications are to be used together in practice. It is worth noting that the definition of such relationships, as well as the mapping from WS-CDL to

⁵ <http://lists.w3.org/Archives/Public/public-ws-chor/2003Aug/att-0016/mpi-use-case-ab.html>

BPEL would be much simpler if WS-CDL had a similar set of control-flow constructs as BPEL (i.e. Sequence, Flow/Parallel, While, Switch, Pick, and possibly also control-links). Ultimately, the fundamental difference between the concept of choreography on the one hand, and the concept of behavioural interface (i.e. BPEL executable process) on the other, is that a choreography focuses on interactions seen from a global viewpoint, while behavioural interfaces focus on communication actions seen from the viewpoint of one of the participants. This has nothing to do with control flow, and arguably WS-CDL and BPEL could very well share the same set of control-flow constructs.

Service semantics. The existing association between WS-CDL, BPEL and WSDL is arguably too restrictive. A choreography or orchestration “wired” to specific WSDL interfaces (either indirectly through references to operations or more directly through an association between roles and their behaviours specified by reference to WSDL interfaces) cannot utilise functionally equivalent services with different WSDL interfaces. In other words, the choreography or orchestration is statically bound to specific operation names and types, which may hinder the reusability of choreography or orchestration descriptions.

Cast more generally, choreography or orchestration descriptions which abstractly describe behaviour at a higher level, in terms of capability, would allow runtime selection of participants able to fulfil that capability, rather than restricting participation in the choreography or orchestration to participants based on their implementation of a specific WSDL interface or WSDL operations.

Semantic descriptions of web service functionality would assist in overcoming the problem of lock-in in to specific WSDL interfaces. Although work on semantic web services such as OWL-S⁶ has introduced the notion of semantic service descriptions, the OWL-S ontology in particular does not allow the explicit description of service capability. OWL-S semantic service descriptions are limited to describing the inputs and outputs or results of a service rather than what functionality the service actually performs. This has been acknowledged in the research community, and efforts are underway that focus on describing service capabilities rather than operations [14].

Design vs. execution. Finally, it is worth noting that both BPEL and WS-CDL are XML-based. The development of a graphical language is not within the charters of these standardisation initiatives. In the case of WS-CDL, which is aimed at specifying design artifacts (as opposed to executable code) placing emphasis on an XML representation seems a distraction from its intention. Indeed, any exploitation of a choreography language is likely to be based on graphical languages in order to achieve user convenience in capturing specifications.

4 Conclusion and Future Research Directions

The issues discussed above suggest that the WS-CDL standardisation effort came too early in the evolution of SOAs. Indeed, WS-CDL has attempted at the same

⁶ <http://www.daml.org/services/owl-s/>

time to be ground-breaking and to create a consensus. In this respect, it is insightful to compare the development of WS-CDL with that of BPEL. BPEL stemmed from two sources, WSFL and XLang, that derived themselves from languages supported by existing tools (namely MQSeries Workflow and BizTalk). Furthermore, at the same time and soon after the first versions of the BPEL specification, and before the BPEL specification went into a formal standardisation process, prototype and commercial implementations started to appear. In contrast, WS-CDL was developed without any prior implementation and does not derive from any language supported by an implementation. Recently, a first partial WS-CDL implementation has been announced,⁷ but it may take time before this implementation attains maturity and other implementations start to appear. Thus, these efforts may come too late to provide much necessary feedback on the WS-CDL specification.

Whether or not WS-CDL becomes a de jure standard and is adopted by a wide user base, its development would have been instrumental in promoting the notion of service choreography as a basis for service-oriented development. Still, many issues remain to be resolved before the emergence and adoption of SOA infrastructures that integrate the notion of service choreography. To advance this vision, we propose a research agenda structured around three major tasks:

- *Identify and document a library of service interaction patterns.* Generally speaking, patterns document known solutions to recurrent problems that occur in a given software development context. A pattern captures the essence of a problem, provides examples, and proposes solutions. The value of patterns lies in their independence from specific languages or techniques and the fact that they capture common situations, abstracting away from specific scenarios or cases. In particular, a library of patterns of service interactions would provide a foundation to analyse and improve existing languages and techniques for choreography and behavioural interface modelling, and/or to design new ones. A first attempt at collecting such library of patterns and using them to analyse the scope and limitations of BPEL is reported in [3].
- *Define a service interaction meta-model.* The insights gained from the service interaction patterns and from the analysis of existing approaches to service choreography and service behaviour definition in terms of these patterns, could serve as the basis for identifying a set of fundamental concepts directly relevant to the service behaviour modelling viewpoints defined in this paper. This would provide a kernel *service interaction meta-model* that could then be enriched with concepts found in existing service choreography and behaviour definition languages such as WS-CDL, ebXML BPSS [5], and BPEL (especially its “abstract process” component). Importantly, the meta-model should be formalised, for example by defining a type system or a mapping into a well-established formalism.
- *Define concrete syntaxes for service interactions definition.* Once a service interactions meta-model has been defined, a design-level (possibly visual)

⁷ <http://www.pi4tech.com>

notation and an interchange format can be specified. Effectively, the meta-model would serve as an *abstract syntax* for service interactions definition while the design-level notation and the interchange format would be seen as concrete syntaxes. The design-level notation could be based upon existing languages rather than developed from scratch. Visual process modelling notations such as BPMN or UML activity and sequence diagrams could be used as the basis for defining a high-level notation for service interactions modelling. The interchange format on the other hand could be defined in terms of XML schema. Importantly, the elements in these concrete syntaxes would map directly to the concepts of the service interaction meta-model.

The patterns, meta-model, design-level notation and interchange format, would together provide the basis for a model-driven service development infrastructure. In particular, model transformations could be defined from the service interaction meta-model into the meta-models of implementation languages, and these transformations could serve as the basis for code generation. Also, model-transformations could be defined for switching between the various viewpoints (e.g., splitting a choreography into several expected behavioural interfaces or merging several interrelated expected behavioural interfaces into a choreography). Finally, this infrastructure could also support behaviour mediation, and specifically, for defining mappings between expected and provided interfaces.

Acknowledgment. The second author is funded by a Queensland Government “Smart State” Fellowship co-sponsored by SAP.

References

1. G. Alonso, F. Casati, H. Kuno and V. Machiraju, *Web Services: Concepts, Architectures and Applications* (Springer Verlag, 2003).
2. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. *Business Process Execution Language for Web Services, version 1.1*, May 2003. Available at: <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel>
3. A. Barros, M. Dumas, and A. H.M. ter Hofstede. Service Interactions Patterns In *Proceedings of the 3rd International Conference on Business Process Management (BPM)*, Nancy, France, September 2005. Springer Verlag, pp. 302-218. Extended version available as Technical Report FIT-TR-2005-02, Faculty of IT, Queensland University of Technology, <http://sky.fit.qut.edu.au/~dumas/ServiceInteractionPatterns.pdf>
4. B. Benatallah, F. Casati, D. Grigori, H. Motahari-Nezhad, and F. Toumani. Developing Adapters for Web Services Integration. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, Porto, Portugal, June 2005. Springer Verlag.
5. J. Clark, C. Casanave, K. Kanaskie, B. Harvey, J. Clark, N. Smith, J. Yunker, K. Riemer, (Eds) *ebXML Business Process Specification Schema Version 1.01*, UN/CEFACT and OASIS Specification, May 2001. Available at: <http://www.ebxml.org/specs/ebBPSS.pdf>.

6. R. Dijkman and M. Dumas. Service-oriented Design: A Multi-viewpoint Approach. *International Journal of Cooperative Information Systems* 13(4), December 2004. Earlier version available as a technical report at: <http://www.uwutwente.nl/webdocs/ctit/1/000000ed.pdf>
7. R. Farahbod, U. Glässer, and M. Vajihollahi. Specification and validation of the business process execution language for web services. *Proceedings of the 11th International Workshop on Abstract State Machines (ASM)*, pages 79–94, Lutherstadt Wittenberg, Germany, May 2004. Springer-Verlag.
8. X. Fu, T. Bultan, and J. Su. Analysis of Interacting BPEL Web Services. In *Proceedings of the International Conference on the World Wide Web Conference (WWW)*, pages 621–630, New York, NY, USA, 2004. ACM Press.
9. N. Kavantzias, D. Burdett, G. Ritzinger, and Y. Lafon. *Web Services Choreography Description Language Version 1.0*, W3C Working Draft, October 2004. Available at: <http://www.w3.org/TR/ws-cdl-10>.
10. M. Koshkina and F. van Breugel. *Verification of business processes for Web services* Technical report CS-2003-11, York University, October 2003. Available from: <http://www.cs.yorku.ca/techreports/2003>.
11. R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.
12. A. Martens. Analyzing Web Service Based Business Processes. In *Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE)*, pages 19–33, Barcelona, Spain, April 2004, Springer-Verlag.
13. C. Ouyang, W. M.P. van der Aalst, S. Breutel, M. Dumas, A. H.M. ter Hofstede, and H.M. Verbeek. *Formal semantics and analysis of control flow in WS-BPEL* Technical report BPM-05-13, BPMCenter.org, June 2005. Available from <http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2005/BPM-05-13.pdf>.
14. P. Oaks, A. H.M. ter Hofstede and D. Edmond. Capabilities: Describing What Services Can Do. In *Proceedings First International Conference on Service Oriented Computing (ICSOC 2003)*, Trento, Italy, December 2003.
15. C. Peltz, Web services orchestration and choreography, *IEEE Computer* **36**(8) (2003) 46–52.
16. D. Roman, E. Cimpian, U. Keller, M. Stollberg, and D. Fensel, *Choreography in WSMO*. WSMO Working Draft 12 November 2004. Available at: <http://www.wsmo.org/2004/d14/v0.1/20041112>
17. H.W. Schmidt and R.H. Reussner. Generating Adapters for Concurrent Component Protocol Synchronisation. In *Proceedings of the Fifth IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, Enschede, The Netherlands, March 2002. Kluwer Academic Publishers.

From Collaboration Models to BPEL Processes Through Service Models

Giorgio Bruno and Marcello La Rosa

Dip. Automatica e Informatica,
Politecnico di Torino, Torino, Italy
giorgio.bruno@polito.it, marcello.larosa@fastwebnet.it

Abstract. This paper proposes a model-based lifecycle for the development of web services, which is based on two kinds of models, collaboration models and service ones. After agreeing upon a collaboration model, which is a public specification, each party can work out a service model and then can turn it into a process written in an orchestration language such as BPEL. As the conceptual gap between a service model and its BPEL implementation is relevant, this paper is concerned with the automatic mapping of service models to BPEL processes, in line with model-based development. Moreover it discusses how to validate services with respect to collaboration models both at-design time and at run-time, and presents the bProgress software environment, which is made up of a number tools developed during this research.

1 Introduction

The technology of web services is gaining growing consensus as the platform of choice for carrying out collaborations within and across enterprise boundaries.

In its simplest form a collaboration takes place through a request-response interaction between two services, a requester and a provider: the requester sends a request, the provider reacts to the request by performing an action and then replies with a response.

Real cases are more complicated as a collaboration may require a number of interactions between the parties: for this reason the parties have to agree in advance on the message flow by working out a common model, called a collaboration model.

However in order to properly support a given collaboration, a service has to arrange its activities (receiving, sending and processing ones) within a control structure, hence it turns out to be a process. Therefore the emerging technology of orchestration languages and processes is a good choice for implementing such services.

On the other hand moving directly from a collaboration model, which is a rather neutral specification of the interactions between two services, to an orchestration process is too long a jump to be afforded in real applications and it is like skipping the design phase in software development.

As a matter of fact developing a collaboration is a process and, as such, it entails the usual phases of specification, design, implementation and operation.

At specification-time a collaboration is a model specifying the messages to be exchanged between the parties as well as the ordering and the timing constraints of those messages. A collaboration model is a public specification which the parties will use to develop and test their own services.

At design-time each party works out a service model, i.e. a more detailed model which, in addition to the activities concerned with sending/receiving the messages established in the collaboration model, includes the activities necessary for producing and processing such messages.

At implementation-time a service model is turned into a working solution based on an orchestration language, such as BPEL [1].

The contribution of this paper basically consists in defining strong connections between the specification phase and the design one and between the design phase and the implementation one.

Verifying the conformity of a service model to a collaboration model is a key issue of the first of the above-mentioned connections; such a verification is based on the relationships existing between services and collaborations (with respect to a given collaboration a service can act as a provider or as a requester for one or more instances) and will be discussed in two major cases, i.e. when the service provides a single collaboration instance or requires multiple instances.

As the conceptual gap between a service model and its BPEL implementation is relevant, the second of the above-mentioned connections is concerned with the automatic mapping of service models to BPEL processes, in line with model-based development [2].

This paper is organized as follows. Section 2 presents collaboration models and introduces the example of a selling collaboration, which will be used throughout this paper. Section 3 illustrates service models and discusses the assumptions the automatic mapping to BPEL processes rely on. Sections 4 and 5 describe how WSDL documents and BPEL processes are automatically generated. Section 6 discusses how service models can be validated with respect to a given collaboration model. Section 7 gives a short account of the bProgress environment, which is made up of a number tools developed during this research. A comparison with related work is the subject of section 8, while section 9 presents the conclusion.

2 Collaboration Models

A well-known example of collaboration is the purchasing of goods or services, whose description is as follows: the requester sends a request for quote (rfQ) to the provider, which may respond with a quote; if the requester accepts the quote, it will then send an order to the provider. That collaboration will be used throughout this paper and will be referred to as the selling collaboration, according to the provider perspective (the requester would call it a purchasing collaboration).

Basically a collaboration model in bProgress consists of messages placed within a control structure providing for sequential, alternative, repetitive and timeout-related paths. The model of the selling collaboration is shown in Fig. 1.

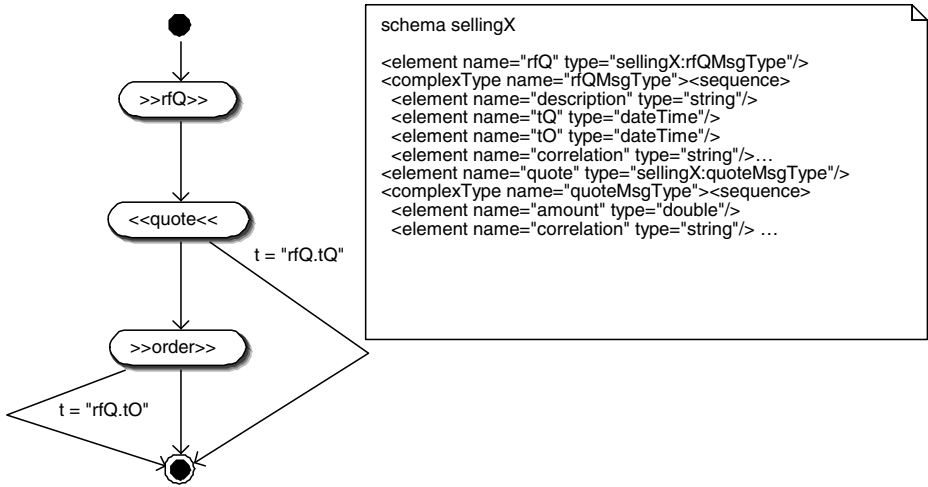


Fig. 1. The selling collaboration model (sellingC)

A message has a left-to-right direction (e.g. rfQ and order), or a right-to-left direction (quote), a left-to-right message being sent from the requester to the provider and a right-to-left one being sent from the provider to the requester.

Unlabelled links represent precedence constraints.

As a collaboration is assumed to be started by the requester, the first message is a left-to-right one, like rfQ, and is called the initial message. In some cases a collaboration could be started with two or more alternative messages, then the first element in the model would not be a left-to-right message but a branch leading to the various initial messages.

Each message (except the initial ones) must have a deadline. The meaning of a deadline is as follows: the receiver is bound to wait for a message until its deadline expires and no longer; it is useless for the sender to send a message if its deadline has expired. When a deadline expires, a timeout will occur; timeout links (i.e. those labelled with keyword “t”) establish the effects of timeouts. In the model shown in Fig.1 there are two deadlines, tQ and tO, and both are attributes of message rfQ: tQ is the time-limit for sending/receiving a quote, tO is the time-limit for sending/receiving an order.

The selling collaboration model is to be interpreted as follows. The provider can send a quote only after receiving an rfQ; the requester will wait for the quote until time-limit tQ and no longer, therefore if the quote is not received by that time, the collaboration will be ended. After receiving a quote, the requester can send an order; after sending the quote, the provider will wait for the order until time-limit tO and no longer, hence if the order is not received by that time, the collaboration will be ended. Other kinds of purchasing/selling collaborations can be found in [3].

Each message has a name and a type. By convention the name of the type is that of the message followed by suffix “MsgType”: if so, the type is not shown in the model (therefore “>>rfQ>>” is equivalent to “>>rfQ, rfQMsgType>>”).

Types are defined in an XML schema associated with the collaboration model. An excerpt from *sellingX*, i.e. the schema related to the selling collaboration, is shown in Fig.1. Correlation attributes are explained in the next section.

3 Service Models

A service model in *bProgress* is an abstract graphical representation of a service that is to be automatically mapped to a BPEL process. A service model is not a graphical representation of a BPEL process as it is based on higher-level abstractions. However in order to be automatically translated into BPEL processes, *bProgress* service models adopt the same conventions as BPEL as regards the generation of process instances and the correlation of messages to process instances.

In general a BPEL process begins by receiving the initial message (the case of multiple initial messages is left apart). When the BPEL run-time system receives the initial message for a given process, it generates an instance of that process and delivers it the message. As to the collaboration started by the initial message, the newly generated process instance is said to be its provider, while the process instance that sent that message is said to be its requester. Likewise, by extension, for a process and a service model.

At the very heart of a collaboration there is the possibility for the same pair of process instances to exchange messages over a period of time. In fact a message is directed to a process (more precisely to its endpoint) and, if it is not an initial message, it is also assumed to contain the information about the process instance it is to be delivered to. BPEL correlates a message to a process instance on the basis of the value of one or more attributes (called properties) of the message. This solution, relying on the payload of messages, is completely transparent, i.e. free from implementation details.

Correlations are automatically inserted by the *bProgress* code generator, provided that each message includes an attribute named *correlation* (this is the reason why messages types in Fig.1 include that attribute), whose value is able to identify the proper process instance on both sides of the collaboration. That value is set by the requester. As far as the *selling* collaboration is concerned, the correlation value is made up of the URL of the requester endpoint and of the id of the *rfQ* (i.e. the primary key of the corresponding data object managed by the information system on the requester side).

When the requester sends an *rfQ*, the BPEL sending activity reads the correlation value from the output message and associates a tag having that value with the sending process instance. Such tags are called correlation sets in BPEL and have to be declared in the process, as shown in the next section. When the *rfQ* is received, the BPEL run-time system generates an instance of the receiving process, reads the correlation value from the input message and associates a tag having that value with the newly generated instance. The quote is sent back to the requester with the same correlation value as the *rfQ*, hence it will be delivered to the requester process instance that previously sent the *rfQ*. When an order is sent, since it has the same correlation set as the quote and the *rfQ*, it will be delivered to the provider process instance that previously sent the winning quote.

An example of a selling service model is shown in Fig. 2. It provides a single selling collaboration, as shown in the “provides” clause. A new instance is started when an rfQ is received, hence the state of the process instance coincides with the state of the collaboration.

The selling service model is basically an extension of the collaboration model, in which left-to-right messages have been turned into receiving activities and right-to-left messages have been turned into sending activities, and some links have been expanded into processing activities.

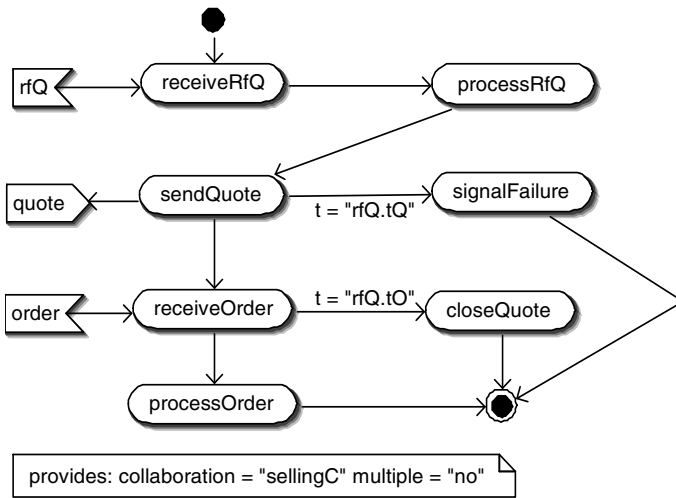


Fig. 2. The selling service model (sellingS)

In bProgress a service model is an extended UML activity diagram, including communication (i.e. sending and receiving) activities and processing ones. A communication activity is completely defined by the model, while a processing activity is to be supplemented with a BPEL content (as a textual addendum). The tasks of the processing activities are as follows: processRfQ writes the rfQ in the information system (on the provider side) and makes it generate the quote to be sent, processOrder writes the order in the information system, closeQuote and signalFailure report to the information system that the quote has been unsuccessful or has not been prepared in due time, respectively.

An example of a purchasing service model is shown in Fig. 3. Its task is basically to select the best supplier for a given request for quote on behalf of the information system (on the requester side). In fact it begins by receiving message purchasingInfo (from the information system), which contains the request for quote along with a list of suppliers to be involved. Then it sends the request for quote to each supplier, receives all the quotes (until all the quotes expected have been received or the time-limit established in attribute purchasingInfo.t has been reached), selects the best one, sends an order to the winning supplier and finally reports the result back to the

information system. The quote selected is the most convenient one, provided that it does not exceed a predefined amount (maxAmount) contained in purchasingInfo.

The purchasing service provides a single “purchasingC” collaboration (to the information system) as shown in the “provide” clause. PurchasingC, which is an inner collaboration on the requester side, is shown in Fig. 4 along with the corresponding message types. Those types are defined in schema purchasingX, which depends on schema sellingX shown in Fig. 1 (“wsa” denotes the WS-Addressing [4] schema).

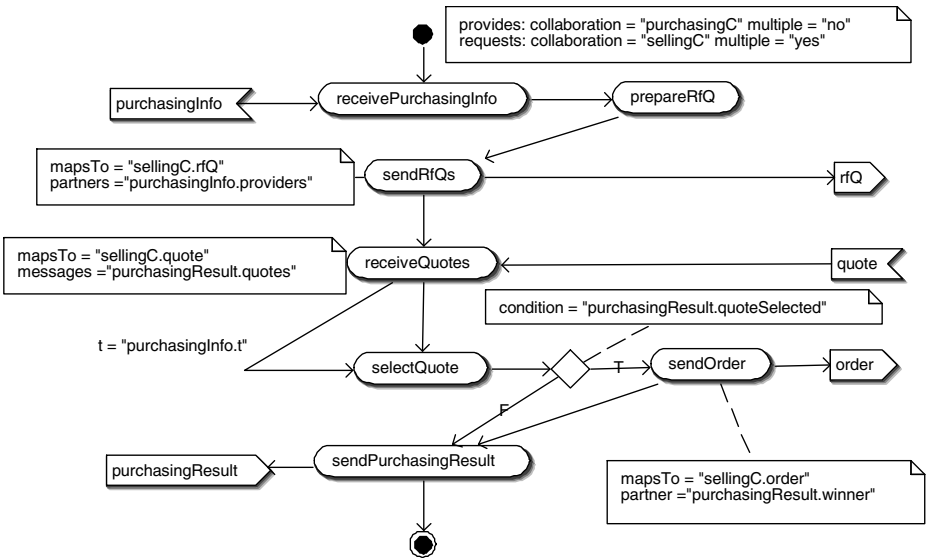


Fig. 3. The purchasing service model (purchasingS)

The purchasing service requests multiple instances of the selling collaboration, as shown in the “requests” clause. For this reason its model cannot be a mere extension of the selling collaboration model (as in the case of the selling service model); however some of its activities - sendRfQs, receiveQuotes and sendOrder - are meant to be mapped to operations on such collaborations, as follows.

Activity sendRfQs is a multiple-sending activity, as indicated by its parameters: mapsTo = sellingC.rfQ and partners = purchasingInfo.providers. Its meaning is that a copy of the rfQ is to be sent to each provider found in list purchasingInfo.providers. The details of BPEL code are illustrated in the next section.

Activity receiveQuotes is a multiple-receiving activity, as indicated by its parameters: mapsTo = sellingC.quote, messages = purchasingResult.quotes. Its meaning is that each message received is to be added to list purchasingResult.quotes.

SendOrder is a single-sending activity; its parameters, mapsTo = sellingC.order and partner = purchasingResult.winner, indicate that it is to send an order to the provider found in purchasingResult.winner (such a provider having been determined by activity selectQuote). In fact selectQuote looks for the most convenient quote and,

if there is one, it sets attribute `purchasingResult.quoteSelected` to “true” and attribute `purchasingResult.winner` to the endpoint of the winner (after retrieving it from list `purchasingInfo.providers`) and also prepares the order to be sent.

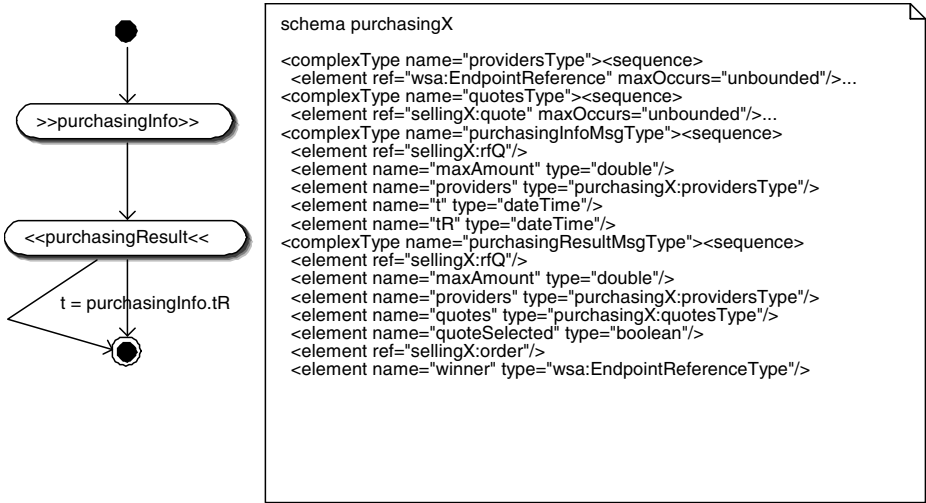


Fig. 4. The purchasing collaboration model (purchasingC)

4 Generating a WSDL Document from a Collaboration Model

This section explains how the bProgress code generator maps a collaboration model, such as `sellingC`, to a WSDL document.

A collaboration model implies the existence of two services, one on the provider side, the other on the requester side. The interfaces of such services are defined in the same WSDL document; the behavior of each service, instead, is defined in a distinct BPEL process, which relies on that WSDL document.

In general, a WSDL document can provide abstract information (messages, operations and portTypes) as well as deployment information (bindings and network addresses) for a number of web services. However, since a BPEL process is intended to be a reusable definition, which can be deployed in different scenarios, it is directly concerned only with the abstract information part of the WSDL documents it relies on.

The current version of the bProgress code generator assumes that all interactions are asynchronous, hence they correspond to WSDL one-way operations. Therefore an abstract web service turns out to correspond to a WSDL portType grouping a number of one-way operations.

An excerpt from the WSDL document generated from `sellingC` follows (“tns”, “sellingX”, “plnk” and “bpws” denote the current document, the schema shown in Fig. 1, the document defining partner link types and the BPEL schema, respectively).

```

<definitions name="sellingC"
  targetNamespace="http://www.polito.it/bProgress/sellingC" ...
<message name="rfQMsg"><part name="payload" element="sellingX:rfQ"/>
<portType name="providerPT">
  <operation name="rfQ"><input message="tns:rfQMsg"/>...
  <operation name="order"><input message="tns:orderMsg"/>...
<portType name="requesterPT">
  <operation name="quote"><input message="tns:quoteMsg"/>...
<plnk:partnerLinkType name="sellingCPLT">
  <plnk:role name="provider"><plnk:portType name="tns:providerPT"/>...
  <plnk:role name="requester"><plnk:portType name="tns:requesterPT"/>...
<bpws:property name="correlation" type="xsd:string"/>...
<bpws:propertyAlias propertyName="tns:correlationId" messageType=
"tns:rfQMsg" part="payload"
query="/sellingX:rfQ/sellingX:correlationId"/>

```

There are two portTypes called providerPT and requesterPT: the former groups all the input messages of the service model, the latter all the output messages.

BPEL requires the portTypes involved in a collaboration to be included in a partnerLinkType construct together with their corresponding role.

When generating a partnerLinkType, bProgress uses two roles, provider and requester, and assigns them to the portType grouping the input messages and to the one grouping the output messages, respectively.

5 Generating a BPEL Process from a Service Model

A BPEL process is basically a hierarchical structure of sending (invoke), receiving (receive) and processing (assign) activities. The structure of a process is determined by compound activities, such as sequence, switch, while, flow, and pick.

An excerpt from the BPEL selling process follows.

```

<process name="sellingP"
  xmlns:sellingC="http://www.polito.it/bProgress/sellingC" ...
<partnerLinks>
  <partnerLink name="sellingCPL" partnerLinkType="sellingC:sellingCPLT"
    myRole="provider" partnerRole="requester"/>...
<variables>
  <variable name="rfQ" messageType="sellingC:rfQMsg"/>
  <variable name="quote" messageType="sellingC:quoteMsg"/>
  <variable name="order" messageType="sellingC:orderMsg"/>
  <variable name="deadlineNotExpired" type="xsd:boolean"/>...
<correlationSets>
  <correlationSet name="sellingCCS" properties="sellingC:correlation"/>.
<sequence>
  <receive name="receiveRfQ" partnerLink="sellingCPL"
    portType="sellingC:providerPT" operation="rfQ" variable="rfQ"
    createInstance="yes">
    <correlations><correlation initiate="yes" set="sellingCCS"/>...
  <assign name="processRfQ"> ... prepares the quote
  <assign name="sendQuote_d"> ... sets inner variable deadlineNotExpired
    to true if the deadline of sendQuote has not expired
  <switch name="sendQuote_s">
    <case condition="bpws:getVariableData('deadlineNotExpired') ">
      <invoke name="sendQuote" partnerLink="sellingCPL"
        portType="sellingC:requesterPT" operation="quote"

```

```

inputVariable="quote">
  <correlations>
    <correlation initiate="no" set="sellingCCS" pattern="out"/>...
  <otherwise><sequence><empty name="signalFailure"/><terminate/>...
</pick name="receiveOrder" createInstance="no">
  <onMessage partnerLink="sellingCPL" portType="sellingC:providerPT"
    operation="order" variable="order">
    <correlations><correlation initiate="no" set="sellingCCS"/>...
    <sequence><empty name="processOrder"/><terminate/>...
  <onAlarm until="bpws:getVariableData('rfQ','payload',
    '/sellingX:rfQ/sellingX:t0')">
    <sequence><empty name="closeQuote"/><terminate/>...

```

A BPEL process sends and receives messages only through channels which are called `partnerLinks`. `PartnerLinks` are declared at the beginning of the process. A `partnerLink` refers to a `partnerLinkType` (taken from one of the WSDL documents referred to by the process) and establishes the role (the value of attribute `myRole`) played by the process with respect to the `partnerLinkType`.

For each collaboration model provided or requested by the service model there is a partner link in the BPEL process. In the selling process there is only one `partnerLink`, `sellingCPL` (named after the collaboration it provides), hence the process can receive an `rfQ`, or send a quote, through that channel. When a message is received, it is copied into a variable; when it is sent, it is read from a variable. Variables in the process correspond to the messages in the service model.

The behavior of the selling process is basically a sequence of five major activities, as shown in the model.

The first activity, `receiveRfQ`, copies the initial message into variable `rfQ` and tags the instance with the value read from attribute `correlation` of the message received. Such tags are called `correlation sets` in BPEL and have to be declared in the process, e.g. `sellingCCS` in the code shown above. When the requester sends an `rfQ`, as will be shown later on, the sending activity reads the correlation value from the output message and initializes the correlation set associated with the sending process instance, using that value. A quote must not be sent if it is late. In order to comply with such a constraint, the BPEL code generator produces two activities: the first one, `sendQuote_d`, sets an inner variable, `deadlineNotExpired`, to “true”, if the deadline of the quote has not expired, to “false” otherwise; the second activity is a two-branch switch structure, the first branch being taken if `deadlineNotExpired` is “true”.

The collaboration will be closed, if the order is not received within a given time-limit. Therefore activity `receiveOrder` is mapped to a `pick` structure whose purpose is to wait for the order to arrive or for the corresponding timeout alarm to go off: when one of those triggers occurs, the associated activity is carried out and the `pick` completes. Since activities `closeQuote`, `processOrder` and `signalFailure` have been left undefined in the model, they are mapped to BPEL empty activities.

The purchasing process

For lack of space only the BPEL code corresponding to multiple-sending activity `sendRfQs` is illustrated in detail. Given its parameters, that activity is turned into a

loop that iterates over the list of providers contained in attribute `providers` of variable `purchasingInfo` and sends the `rfQ` to each provider. Providers are simply denoted by their endpoint references as shown in Fig. 4.

The same `rfQ` is sent to each provider through partner link `sellingCPL`, which is a variable partner link as its partner endpoint reference has to be set before each sending operation. In fact, if the WSDL documents contain no deployment information, as is the case of the WSDL documents generated by `bProgress`, partnerLinks lack the information needed to reach the intended web services. For this reason, before using a partnerLink to send an initial message (like `rfQ`), the BPEL process has to set the partner endpoint reference within the partnerLink. An endpoint reference is a structure (based on WS-Addressing [4]) including the network address of the web service to be called along with other deployment information.

An excerpt from the code of `sendRfQs` follows.

```
<scope name="sendRfQs_s"><sequence>
  <assign>
    <copy><from expression="ora:countNodes('purchasingInfo', 'payload',
      '/purchasingX:purchasingInfo/purchasingX:providers/
        wsa:EndpointReference')"/><to variable="sendRfQs_c"/>...
    <copy><from expression="0"/><to variable="i"/>...
    <while name="sendRfqs_w" condition="bpws:getVariableData('i')
      &lt; bpws:getVariableData('sendRfQs_c')"><sequence>
      <assign>
        <copy><from variable="purchasingInfo" part="payload"
          query="/purchasingX:purchasingInfo/purchasingX:
            providers/wsa:EndpointReference[bpws:getVariableData('i')+1]"/>
          <to partnerLink="sellingCPL"/>...
        <copy><from expression="bpws:getVariableData('i')+1"/>
          <to variable="i"/></copy>...
      <invoke name="sendRfQs" partnerLink="sellingCPL"
        inputVariable="rfQ" portType="sellingC:providerPT"
        operation="rfQ">
      <correlations>
        <correlation initiate="yes" set="sellingCCS" pattern="out"/>...
```

The outer scope, `sendRfQs_s`, encompasses two sequential BPEL activities. The first one (the assign activity) determines the number of iterations (which corresponds to the number of endpoint references contained in attribute `purchasingInfo.providers`) and sets inner variable `sendRfQs_c` to that value; then it initializes the index of the loop (i.e. inner variable `i`) to 0. The second activity is a “while” whose body is performed as long as `i < sendRfQs_c`. At each iteration the endpoint reference of the current provider is copied into partner link `sellingCPL` and the index is incremented, then the `rfQ` is sent to the current provider.

The need for multiple-sending activities as well as multiple-receiving ones in BPEL has also been pointed out in [5], where the introduction of two new primitives, `broadcast` and `collect`, is proposed. The multiple-sending and multiple-receiving activities presented in this section can be used to implement well-known patterns, such as those dealing with multiple instances [6] and those related to asynchronous communication (`publish/subscribe`, `broadcast`) [7].

6 Validating Service Models

Validating a service model means making sure that it conforms with the collaborations it provides or requests.

If a service model is concerned (as a provider or a requester) with a single instance of a given collaboration, then it can be viewed as an extension of the model of that collaboration, just as the selling service shown in Fig. 2 is an extension of the selling collaboration shown in Fig. 1. In this case it is simply a matter of proving that the service model can be transformed, by means of suitable reduction rules, into a model which is equivalent to the collaboration one.

In fact in the selling service shown in Fig. 2 activity processRfQ can be viewed as a refinement of the precedence link connecting receiveRfQ to sendQuote, so it can be replaced with a simple link from receiveRfQ to sendQuote; likewise for activity processOrder. Moreover activity closeQuote can be viewed as a refinement of the timeout link connecting receiveOrder to the final state and hence it can be replaced with a simple link; likewise for activity signalFailure. At this point the resulting model turns out to be equivalent to the selling collaboration model. Such a reduction rule is the inverse of the second inheritance-preserving transformation rule (i.e. rule PJS) presented in [8].

Validating a service model that requests multiple instances of a given collaboration, such as the purchasing service shown in Fig. 3, in general cannot take place at design-time, since the states of the various collaboration instances can differ during the execution of the service. In fact if activity selectQuote worked badly, the order could be mistakenly sent to a supplier that did not provide any quote. In this case run-time checks are needed in order to prevent a service from sending or receiving a message in wrong order (or not complying with timing constraints). Such checks can be performed only if the states of ongoing collaborations are available and can be updated when needed.

In a previous paper [9] a solution based on collaboration objects was presented: their purpose is to separate validation logic from process logic as well as to provide high-level sending and receiving operations to workflow processes.

This paper presents a different solution in which for each collaboration a descriptor is maintained in the requesting BPEL process: the reason is to take advantage of the persistency features provided by the BPEL run-time system. However the handling of such descriptors (i.e. checking and updating actions) takes place by means of external stateless Java objects.

A collaboration descriptor basically contains the endpoint reference of the partner service, the state of the collaboration (i.e. the name of the next message to be sent or received) and the current deadline.

The bProgress code generator adds run-time checks, in terms of Java-based activities, to each sending or receiving operation. Therefore a BPEL process performs a pre-sending check before each sending activity in order to make sure that the message name is included in the state of the descriptor of the corresponding collaboration instance and the deadline has not expired. If that check succeeds, the state and the current deadline are updated according to the collaboration model; otherwise an exception is thrown. The service model must include fault paths leading to the proper exception-handling activities. Moreover a BPEL process performs a

post-receiving check after each receiving activity (except for initial messages); since the partner link involved contains the endpoint reference of the partner, the corresponding collaboration descriptor can be retrieved and then suitable checks and updates can be performed.

7 The bProgress Environment

The bProgress environment is a set of tools based on the Eclipse platform, intended to support new approaches for business processes and services.

Collaboration models and service ones are produced with a UML 2.0 visual tool. We have defined a set of profiles (i.e. <<receive>>, <<send>> and <<timeout>>) so as to specialize the graphical elements. The models are first exported to XMI 1.1 documents and then transformed into a simpler XML representation by means of an XSL transformation.

The bProgress code generator produces WSDL documents and BPEL processes from such internal XML representations, according to the rules presented in section 5; if required, it can add run-time checks.

The BPEL processes presented in this paper have been tested using Oracle BPEL Process Manager 10.1.2 Beta-3; the tests have been carried out on one purchasing process requesting collaborations of a number of different selling processes (providing the same selling collaboration).

8 Comparison with Related Work

Orchestration languages, such as BPEL, are a good implementation platform, however more abstract representations, i.e. service models, are needed to help designers focus on process logic and get rid of technical details. On the other hand this is an area suitable for applying model-based development [2] with the purpose of automatically deriving orchestration processes from service models.

Mapping models to BPEL processes has been addressed in several papers from different starting points: extended state machines are used in [10], while UML activity diagrams are adopted in [11]; however their focus is on control aspects rather than on collaboration issues.

Web service composition is a fundamental issue in service-oriented computing: it basically refers to the possibility of building a new service on top of some existing services. A survey of existing proposals is presented in [12] together with a comparative analysis with respect to some key requirements including composition correctness. As to the correctness it is shown in [13] that a composite service, made up of component services (modelled as Petri nets having one input place and one output place) and of standard composition operators, can be checked for deadlock and incorrect termination.

This paper addresses web service composition in terms of collaboration requests: the purchasing service, shown in Fig. 3, is in effect a composition of selling collaborations. The reason is to offer a broader perspective: in fact a component service can be involved in several interactions with the composite service (as is the

case of the selling service with respect to the purchasing service), not only in an initial request and in a final reply; moreover it may happen that a multiple composition of similar component services is needed (such as the multiple selling collaborations requested by the purchasing service). In such cases, it is hard to prove composition correctness; therefore section 6 has presented an approach based on run-time checks, which are meant to prevent a service from sending or receiving a message in wrong order (or not complying with timing constraints).

Providing a standard solution to web service composition entails a number of practical issues, such as the handling of endpoint references and the correlation of messages to process instances, to be taken into account at the same time. This paper has presented multiple-sending and multiple-receiving activities, which work under a number of assumptions: each message has a correlation attribute, there is a variable containing the endpoint references of the services to be involved in a multiple-sending activity, there is a variable where the messages received with a multiple-receiving activity can be stored. The combination of all such aspects can be thought of as a realization of well-known patterns, such as those dealing with multiple instances [6] and those related to asynchronous communication (publish/subscribe, broadcast) [7].

There are many similarities between collaboration models, as presented in this paper, and choreography description languages, such as WS-CDL [14]. The purpose of our research is different as it mainly consists in providing a model-based approach to the development of a collaboration (or choreography). For this reason it is essential to mediate between the capabilities of current technology (BPEL in this case) and conceptual requirements, such as the need for multiple-sending activities (and multiple-receiving ones).

9 Conclusion

This paper has shown that the notions of collaboration and service are strongly related, a collaboration being an abstract representation of the interactions between two services and a service being a process that can be involved in one or more collaborations as a provider or a requester.

According to the principles of model-based development this paper has illustrated how service models can be automatically mapped to BPEL orchestration processes, and has discussed the major issues to be taken into account.

Current work proceeds in several directions, including: the extension from binary collaborations to multi-party ones, the introduction of transactional support [15], and the integration with other notations, such as BPMN [16].

References

1. Andrews, T. et al.: Business Process Execution Language for Web Services Version 1.1. BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems (2003). <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>
2. Mellor, S., Clark, A. N., Futagami, T.: Special Issue on Model-Driven Development. IEEE Software, Vol. 20 (5). IEEE Computer Society (2003)

3. Dijk, A. V.: Contracting workflows and protocol patterns. In: van der Aalst, W.M.P., Hofstede, A.H.M. ter, Weske, M. (eds.): BPM 2003. Lecture Notes in Computer Science, Vol. 2678. Springer (2003) 152-167
4. Bosworth, A. et al.: Web Services Addressing (WS-Addressing). BEA, IBM, Microsoft (2003). <http://msdn.microsoft.com/ws/2003/03/ws-addressing/>
5. Mendling, J., Strembeck, M., Neumann, G.: Extending BPEL4WS for multiple instantiation. In: Dadam, P., Reichert, M. (eds.): INFORMATIK 2004. Lecture Notes in Informatics (LNI), Vol. 51. German Computer Science Association (2004) 524-529
6. van der Aalst, W. M. P., Hofstede, A. H. M. ter, Kiepuszewski, B., Barros, A. P.: Workflow patterns. Distributed and Parallel Databases, Vol. 14(1). Springer (2003) 5-51
7. Wohed, P., van der Aalst, W. M. P., Dumas, M., Hofstede, A. H. M. ter: Analysis of web service composition languages: the case of BPEL4WS. In: Song, I.Y., Liddle, S. W., Ling, T. W., Scheuermann, P. (eds.): 22nd Int. Conf. ER 2003. Lecture Notes in Computer Science, Vol. 2813. Springer (2003) 200-215
8. van der Aalst, W.M.P., Weske, M.: The P2P approach to interorganizational workflows. In: Dittrich, K. R., Geppert, A., Norrie, M.C. (eds.): 13th Int. Conf. CAiSE 2001. Lecture Notes in Computer Science, Vol. 2068. Springer (2001) 140-156
9. Bruno, G.: Modeling and using business collaborations. In: Pre-proceedings of the 1st Int. Conf. on Interoperability of enterprise software and applications, Geneva (2005) 114-125
10. Baina, K., Benatallah, B., Casati, F., Toumani, F.: Model-driven web service development. In: Persson, A., Stirna, J. (eds): 16th Int. Conf. CAiSE 2004. Lecture Notes in Computer Science, Vol. 3084. Springer (2004) 290-306
11. Mantell, K.: From UML to BPEL, IBM (2003). <http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/>
12. Milanovic, N., Malek, M.: Current solutions for web service composition. IEEE Internet Computing, Vol. 8 (6). IEEE Computer Society (2004) 51-59
13. Hamadi, R., Benatallah, B.: A Petri-net-based model for web service composition. In: Proceedings of the 14th Australasian Database Conference. Australian Computer Society (2003) 191-200
14. Kavantzias, N. et al. (eds): Web Services Choreography Description Language Version 1.0. W3C (2004). <http://www.w3.org/TR/ws-cdl-10/>
15. Dalal, S.; Temel, S.; Little, M.; Potts, M.; Webber, J.: Coordinating business transactions on the web. IEEE Internet Computing, Vol. 7 (1). IEEE Computer Society (2003) 30-39
16. White, S. A.: Introduction to BPMN, IBM (2004). <http://www.bpmn.org>

A Framework for Automated Negotiation of Service Level Agreements in Services Grids

André Ludwig¹, Peter Braun², Ryszard Kowalczyk², and Bogdan Franczyk¹

¹ University of Leipzig, Faculty of Economics and Management,
Information Systems Institute, 04109 Leipzig, Germany
{Ludwig, Franczyk}@wifa.uni-leipzig.de

² Swinburne University of Technology, Faculty of Information and Communication
Technologies, Hawthorn, Victoria 3122, Australia
{PBraun, RKowalczyk}@it.swin.edu.au

Abstract. An important aspect of managing service-oriented grid environments is negotiation of service level agreements. In this paper we propose a framework in which we adopt the three-layer architecture of agent-based negotiation to the problem of service level agreement negotiation in services grids. We report on the first experience with an implementation of the framework in the context of the WS-Agreement specification provided by the Global Grid Forum and present lessons learnt when using this framework in a simple practical scenario.

1 Introduction

Grid computing has emerged as a new paradigm for next-generation distributed computing. It supports a notion of virtual organizations that can share resources for solving large problems in science, engineering, and business.

Service-orientation in grid computing focuses on virtualization of grid resources such as computational resources, storage resources, networks, programs, databases and so forth, and representing them by means of an extensible set of services that may be accessed, shared and composed in various ways [12]. The Open Grid Services Architecture (OGSA) [13] has taken up this approach and introduced the concept of grid services. At the same time integration and management of distributed applications by means of services is the objective of Web Services [36]. In an attempt to take advantage of progress in these two areas, the Globus Alliance [16] in conjunction with industry support has further developed the existing Web Service standards and the OGSA specification, and proposed the WS-Resource Framework (WSRF) [3]. WSRF supports creation, addressing, inspection, and lifetime management of resources as stateful services. It defines the semantics of WS-Resources and summarizes how interoperability between components from different sources can be enhanced using a service-oriented resource view [5]. Rather than shared usage of computer resources in computational grid infrastructures, services grids use grid paradigms in the context of services providing service-oriented applications on demand.

One of the most important aspects of service-oriented computing environments is that their administration and management is driven by individual organizational goals and application requirements. In order to support cross-enterprise dynamic composition and enactment of services, a number of fundamental issues regarding management of service quality and regulation of service behavior must be addressed. Some of these issues are: (a) How can the behavior of services be adjusted dynamically and who does that? (b) If services are created dynamically based on requirements of the consumer, how do participants find a mutually acceptable configuration? (c) How can these agreed service configurations be stored?

The key concept in addressing these issues is service level agreement (SLA). Similarly to commercial situations where “best effort” service guarantees are not sufficient, the agreement documents that specify what the user receives from the offered resources and its relevant performance guarantees are required in the form of SLA. SLAs capture the mutual responsibilities of the provider of a service and its client with respect to functional and non-functional parameters. For example, an agreement may define bounds on service response time and availability, or other service level objectives that describe the required quality of a service. Hence the main motivation for creating SLAs between providers and consumers is to get a reasonable certainty of the provided service behavior.

In a distributed cross-enterprise services grid numerous services interact with each other simultaneously, taking the roles of a provider and a consumer at the same time. The conditions of each of these relationships need to be represented in a SLA document. Keeping track of creating such SLAs, monitoring and evaluating service performance against them, and triggering appropriate actions in cases of SLA violation and exceptions are tasks of overwhelming importance. They include analysis of which part of SLA is violated and which party is responsible for it, what consequences arise from the violation for the overall system, and what the monetary and legal impacts are for the participants. Currently these tasks are performed by humans and require substantial manual effort, hindering broader adoption of services grids across enterprises as manual connection and contract negotiation are too costly on a large scale. Therefore, automation support for these tasks, especially for negotiating SLAs, is required. This automation must include automated creation of SLAs (e.g. as the result of negotiation), and other tasks during SLA lifecycle including their fulfillment and termination. In this context a flexible and precise SLA language, appropriate SLA templates, and a standardized SLA terminology are needed.

In this paper we propose a framework for automated negotiation of service level agreements in services grids, with the focus on the agreement creation phase. The framework adopts the three layer architecture of agent-based negotiation [21] to grid service agreements, involving decomposition of the negotiation into the negotiation objects, negotiation protocols and decision making models that are represented as different services. In addition to presentation of the theoretical framework, we also demonstrate its adaptability in a practical scenario and report on our first experiences in implementing it in the context of the Web Service Agreement specification (WS-Agreement) [1].

The paper is structured as follows. In the next section we briefly summarize related work concerning service level agreements in service-oriented environments. In Section 3 the concept of service-based representation of agreements is introduced.

Section 4 discusses techniques for negotiating service level agreements and the proposed framework for automated negotiation of SLAs in services grids is described in Section 5. In section 6 we provide more details on a prototypical framework implementation. Finally, the conclusions are presented in section 7.

2 Related Work

Research in grid service management has resulted in various approaches for grid resource reservation [6, 8, 38, 18] and quality of service delivery at the resource level [17]. An important part of these approaches is dedicated to the question of how to manage a grid resource in relation to an agreement document defining the resource consumption and provision. A standard concept of arranging and coordinating the services on the Grid are SLAs [22]. Accordingly, different specifications for describing and managing SLAs in XML-based representations are proposed in the Web Service Level Agreement (WSLA) [26], by SLAng [24] and in HP reports [31].

In order to realize an agreement represented by a SLA, several approaches define general frameworks for Grid resource reservation, acquisition, task submission, and binding [38, 18]. In contrast to these advanced reservation and balancing techniques, the Service Negotiation and Acquisition Protocol (SNAP) introduces a protocol for managing the process of negotiating an access to the resources and their use in a distributed system [4]. To represent these SLAs for every grid service running on behalf of a client, the corresponding SLA service can be instantiated. It contains and validates the SLA according to the WS-Agreement specification [1].

Although most of the above work recognizes SLA negotiation as a key aspect of SLA management, they usually provide little guidance of how negotiation (especially automated negotiation) can be realized. In a more general context, automated negotiation has been an important part of agent research (e.g. [21, 33]). They propose negotiation mechanisms including different interaction protocols and decision making models for negotiation in multi-agent systems. In this paper we adapt the agent-based negotiation approach for dynamic automated negotiation of SLAs in service grids, and provide a practical framework where different negotiation protocols and decision making strategies can be realized by service-based SLAs.

3 Service-Based Representation of Agreements

This section gives a concise overview of how relationships between Grid participants can be modeled and managed in a standardized way. As stated before, the relationships between service providers and clients are represented in SLAs to express agreement about the behavior of a provided grid service. In service-oriented grid environments every element is represented as a service, e.g. a WS-Resource service. Following this notion of service orientation and virtualization of resources, SLAs can also be represented by a WS-Resource service. Such an approach is proposed by the WS-Agreement specification published by the Global Grid Forum (GGF) [14]. The fundamental idea of WS-Agreement is the representation of a SLA as WS-Resource service in an agreement service. It describes an XML-based language for specifying

an agreement between a service provider and a consumer, and a protocol for creation of an agreement using agreement templates. In this way each WSRF compliant agreement service represents an SLA and provides interfaces through which the provider and customer service management applications interact with each other. As described previously each WSRF service is capable of supporting lifecycle mechanisms.

The WS-Agreement specification consists of two basic layers (figure 1):

- the agreement layer: which provides a Web service-based interface that represents SLAs,
- the service layer which represents the application-specific layer of the provided business service.

The agreement layer represents a manageable interface for contacting and interacting with a service provider. It publishes information like acceptable agreement terms and enables the creation of agreement service instances in a factory service. The agreement service facilitates the representation of an agreement, captures the agreement terms, manages service lifetime and provides agreement composition capabilities.

The WS-Agreement model covers all periods of the SLA lifecycle. It contributes an abstract but substantial interface description for SLAs between providers and consumers and encourages the approach of service-orientation in grid computing environments. However, it does not specify how the service provider and consumer can come to an agreement and how the agreement process can be supported.

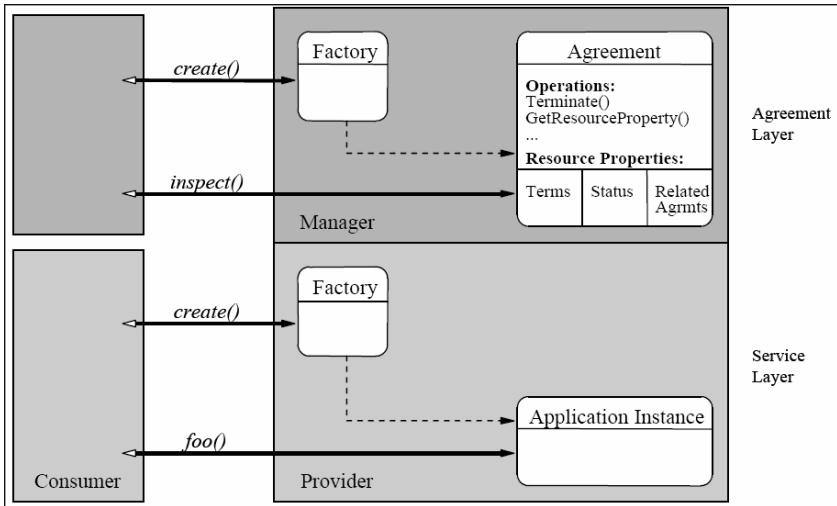


Fig. 1. WS-Agreement service model [1]

4 Negotiating Service Level Agreements

While all phases of the agreement lifecycle involve complex processes and require extensive investigation, the discussion in this work is restricted to the first period, i.e.

the creation of SLAs. The fundamental question of creating an agreement is: how do participants successfully create agreements? Humans, when faced with the need to reach an agreement on a variety of issues make use of negotiation, a process by which a joint decision is made by two or more parties. Typically, the parties first verbalize contradictory demands and then move towards an agreement by a process of concession making or search for new alternatives [29]. However, the scalable deployment and open architecture of WSRF environments enable a multiplicity of services with an unlimited number of service characteristics. Different organizational goals, service requirements and oppositional objectives require policies and technologies to manage the heterogeneity of a grid and make service negotiation a complex process. Currently operations like SLA creation and negotiation are subject to manual and human influence and call for additional support, e.g. for automated negotiation. In automated negotiations a broad range of issues have to be analyzed. That includes issues about the necessary negotiation interactions, characteristics of the negotiated services, and rules what decisions have to be made at what time [25].

A commonly recognized approach to automated negotiation is based on structuring its mechanism into: negotiation objects, negotiation protocols, and decision making models [25].

Negotiation protocols define a set of rules that prescribe the circumstances under which the interaction between agents takes place, called the rules of encounter [25]. They cover the permissible types of participants, the negotiation states, the events that cause negotiation states to change and the valid actions of the participants in particular states. While negotiation protocols are quite different for different categories of negotiation, they have one thing in common: interaction protocols expand the scope from the exchange of single messages to complete multi-step transactions (also called conversations or dialogues).

Negotiation objects are described by the range of issues over which agreement must be reached. The object can contain multiple attributes. These attributes can be classified as:

- service-specific attributes, such as quality of service (QoS), service level or other technical specifications,
- transaction attributes that are generic for the service, such as price, timings, penalties and so on.

Moreover, attributes may be:

- non-negotiable (i.e. having a fixed value),
- negotiable (i.e. having multiple possible values).

Decision making models provide a computational apparatus for making negotiation decisions according to the participants' negotiation strategies. The negotiation strategy governs the participant's general behavior and best course of actions and policies to achieve a goal. The sophistication of the model and the decisions that have to be made are influenced by the negotiation protocol in place, by the nature of the negotiation object, and by the range of operations that can be performed on it [23]. Examples of decision making models used for automated negotiation are game theory based models [27, 32], heuristic approaches [29, 30] and argumentation-based approaches [28, 28, 33].

5 SLA Negotiation Framework

The application of the WS-Agreement model enables the representation of SLA relationships by agreement services and provides standardized interfaces for agreement negotiations. However, the WS-Agreement specification gives no recommendations for how to come to these agreements and how to integrate the actual negotiation of these agreements into one context. This work proposes a negotiation framework for service level agreements (Fig. 2).

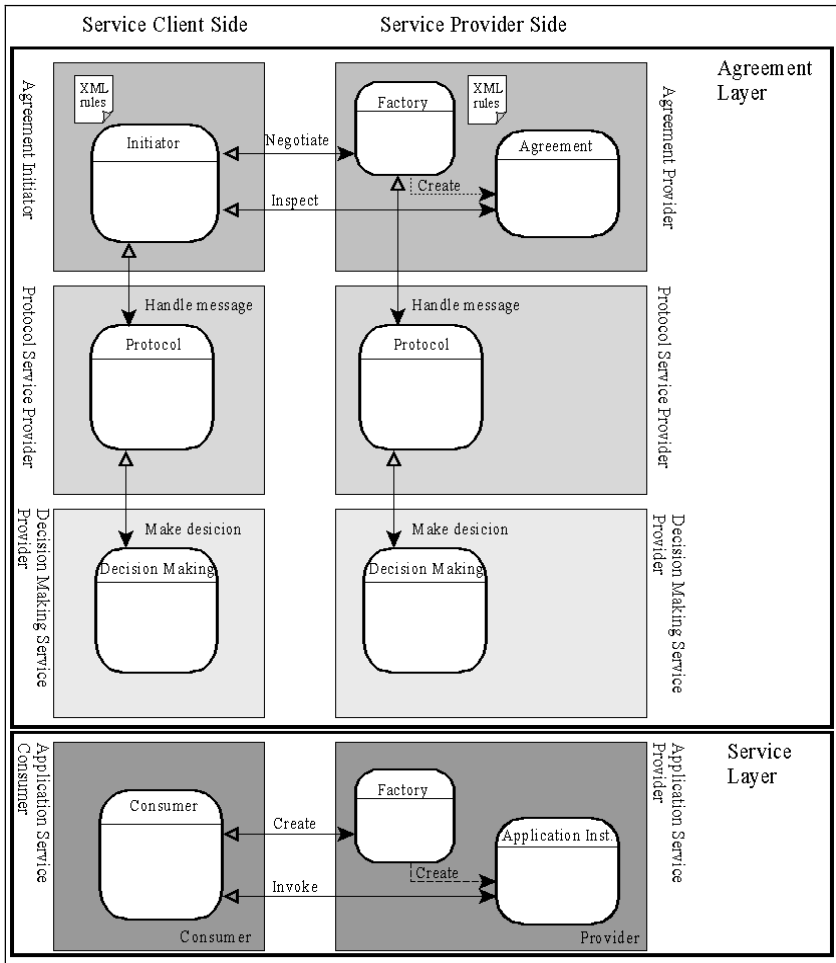


Fig. 2. SLA negotiation framework architecture

The underlying principle of the framework is decomposition of the negotiation mechanism into its basic elements: negotiation object, negotiation protocol, and decision making model in the context of service level agreements and on the basis of

WSRF grid mechanisms. By modularizing and structuring the agreement negotiation into its fundamental components it allows for dynamic adjustment of agreement policies and integrates interaction mechanisms, decision making management and dynamic control of service behavior. The framework is horizontally separated into the service client and service provider sides and vertically divided into the agreement and service layers, adopting the conceptual layered WS-Agreement service model. While the service layer is adopted from the specification without changes, the agreement layer extends the WS-Agreement service model. It consists of additional stand-alone components which fulfill well-defined, autonomous tasks during SLA negotiation. These components are:

- Agreement Provider and Initiator respectively,
- Protocol Service Provider,
- Decision Making Service Provider.

All components are encapsulated in their own services and can be offered by different parties following the service-orientation approach. The tasks of each component are described below.

Agreement Provider. The agreement provider represents a service provider in contractual matters. It provides interfaces that are necessary for interacting with a provider during service negotiation. It is responsible for describing the negotiation object (i.e. an application service) and its attributes (e.g. functional and non-functional properties). Beside that the agreement provider creates SLA documents in the form of agreement instances. The Agreement Provider incorporates the WS-Agreement model and has WSRF-compliant agreement factory and agreement port types.

The agreement factory service provides a manageability interface for negotiating with an agreement provider and is responsible for the interaction with an agreement initiator. It includes receiving and sending messages and advertising supported agreement templates. It facilitates the creation of agreement service instances and SLA lifetime management.

The agreement service represents the result of a successful negotiation in the form of a stable service level agreement between a service client and a service provider. It embodies a well understood service description and captures a mutual understanding of the expected application service behavior.

Neither the agreement factory nor the agreement service implement any negotiation logic itself – they provide only negotiation interaction interfaces and the SLA documents. Once a negotiation opponent (i.e. the agreement initiator) sends a message to the agreement factory service it forwards this message to a protocol service provider.

Protocol Service Provider. In order to make the agreement factory service independent from negotiation protocol-specific processing, it uses external protocol services. The protocol service decides who can do what and when, and how to react to events during negotiation. It enforces a coordinated behavior during a negotiation following the normative rules of the employed protocol. This includes rules about the types of participants, the negotiation states, events and actions that are taken on them. The protocol service offers interfaces that are appropriate for handling the received messages for the Agreement Provider.

Since the negotiation protocols can be different for different categories of negotiation, it is essential that the negotiating parties have a common understanding of meaning and order of the messages and their consequences. A convenient way to ensure mutually coordinated negotiation behavior is to use the negotiation protocols specified by standardization institutions such as FIPA [9]. Examples of FIPA protocols commonly used for automated negotiation between agents are FIPA's contract net protocol [10] and the iterated contract net protocol [11]. In the proposed framework a protocol service provider may offer various negotiation protocols that an agreement factory service can choose from. It also allows for multiple protocol service providers to offer numerous negotiation protocols that can be used as needed.

Nevertheless, the protocol service does not make decisions in response to the received messages, such as proposal assessments, evaluations or counter-offer generations. These operations are handled by an external decision making service.

Decision Making Service Provider. The decision making model of a negotiation is encapsulated in a decision making service. Similar to the protocol services several decision making service providers may offer various decision making models with different levels of sophistication encapsulated in numerous services.

In this context an important question is: how does the decision making service know the preferences and business rules (e.g. SLA parameter acceptance thresholds) of the actual application service provider? First of all, the service provider needs to formally define these preferences and business rules to make them available for processing by individual decision making services in a standardized way. For that purpose this framework incorporates the syntax and semantics of the Policy-driven Automated Negotiation Decision-Making Approach (PANDA framework) [15], which facilitates automated decision making during negotiation.

The PANDA framework defines so called decision strategy rules in a structured XML syntax. The basic building block of a strategy is a single rule, consisting of a condition part and an action to be performed if the condition is satisfied. The conditions are Boolean expressions and an action is a series of data sources. Each negotiation object that influences the decision is represented by a rule and only if the condition of the object's rule is fulfilled the action will be executed. The condition includes a Boolean operator, the minimal utility acceptance threshold and the relative utility weight. Additionally each rule encapsulates the parameters that describe the utility function for a certain object of negotiation.

6 Implementation Details

The proposed negotiation framework has prototypically been implemented and demonstrated with a simple business scenario in a Grid service environment. All components involved in the framework are implemented as Web services and hosted by the WSRF.NET 2.0 platform [36, 37], an implementation of the WSRF specification running on Microsoft's Internet Information Server. The presented services are developed using C# programming language in Microsoft's Visual Studio .NET 2003 environment.

The negotiation scenario presents a business model where a financial service provider offers financial services on the basis of Web services to several clients. One of these services that is implemented in our scenario simulates the evaluation of a person's credit history and anticipates the credit worthiness on a given taxpayer number. The service can be provided with different configurations described by several attributes. These are the service level, i.e. gold, silver, bronze describing the comprehensiveness of the calculated score, a quality of service index, an abstract value that includes availability of the service and the response time dependent on a requested level of throughput, the price per invocation and a minimal number of invocation requests. All of these attributes - service level, QoS value, price, and minimal invocation requests - are open for negotiation and form a multi-dimensional negotiation space.

After the service provider and its client allocate their extensible decision rules to the agreement initiator service and the agreement provider service, respectively, negotiation of the financial services can be initiated and executed. The demonstration integrates protocol services based on FIPA's contract net and iterated contract net protocols, and simple decision making services based on heuristic negotiation strategies. Figure 4 shows a screenshot of the user interface.

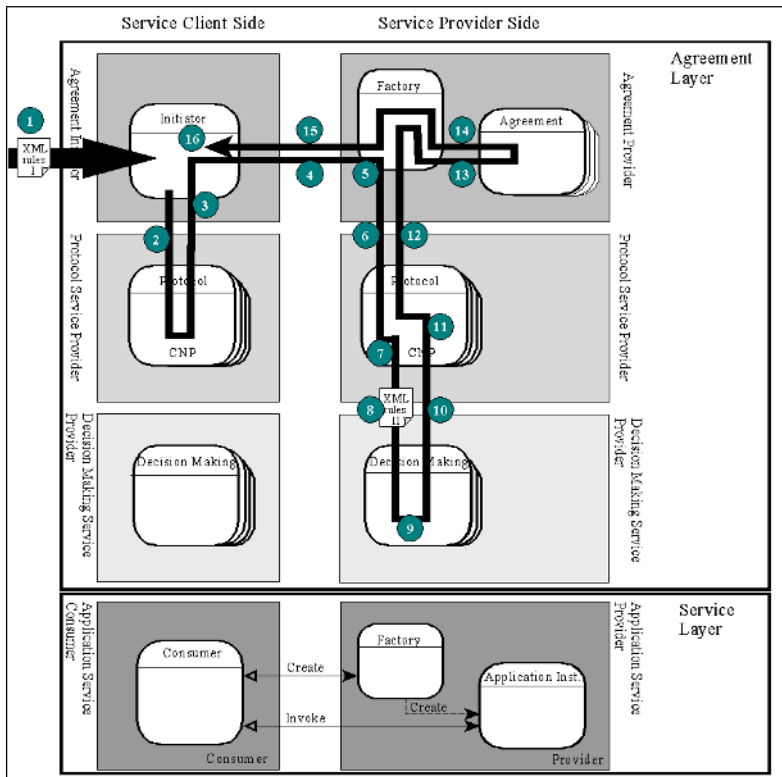


Fig. 3. Message flow during a call for proposal

The steps occurring during a call for proposal and proposal making executed one after another are illustrated by an arrow in Figure 3 and described below:

1. The whole negotiation process is initiated by an external enactment, i.e. manually by the service client. During this step the client's decision rules (XML rules I) are provided to the agreement initiator service. Also the choice of a negotiation protocol can be pre-defined here. However, the agreement initiator service can also choose a suitable negotiation protocol itself, i.e. if the agreement factory service insists on a particular protocol. As an example the FIPA contract net protocol (CNP) mentioned above is used.
2. The agreement initiator service sends a message to a suitable CNP protocol service to start negotiation. As the agreement negotiator maintains no negotiation logic, it just invokes the negotiation process. Together with this request it assigns the extensible decision rules of the client to the protocol service.
3. The CNP initiates negotiations with a 'call for proposal' message (CFP) that is sent to the service provider according to a pre-defined syntax (ACL). The protocol service creates such a CFP message and returns the ready-for-sending message to the agreement initiator.
4. The agreement initiator service contacts the agreement factory service and sends the CFP message.
5. The agreement factory service, as the manageable interface for contracting with a service provider, receives the message. As the decision rules of the contracting parties are usually contrary and kept private, the factory service stores another set of decision rules (XML rules II) for the service provider. This service provides only an interface and does not implement any operations itself. It analyses the value for the protocol suggested by the agreement initiator and assigns the message together with the decision rules to a suitable CNP protocol service.
6. The message together with the XML rules II are send to a CNP protocol service.
7. The protocol service analyses the message and decides on the consequences when receiving a CFP. In this example it decides to make a proposal. However, generation of a proposal is part of the negotiation strategy and is therefore encapsulated in an external decision making service.
8. The CNP protocol service sends a call for generating a proposal to the decision making service. It attaches the XML rules II.
9. The decision making service creates a proposal on the base of the decision rules.
10. The generated proposal is returned to the protocol service.
11. The protocol service creates a proposal message compliant with the ACL syntax and embeds the values of the created offer.
12. Afterwards it returns the proposal message to the agreement factory service.
13. The agreement factory service creates a new instance of an agreement service and writes the values of the received proposal message into the agreement instance.
14. The end-point reference of the agreement service instance is returned to the agreement factory service.
15. The agreement factory service sends the proposal message together with the end-point reference locator of the created agreement service instance back to the agreement initiator.

16. The agreement initiator triggers further actions following the sequence flow of the CNP. In particular it is able to request the agreement service instance, i.e. for evaluating the proposal by the client’s decision making service.

Figure 4 shows a screenshot of the messages recorded at the demonstration GUI during a call for proposal.

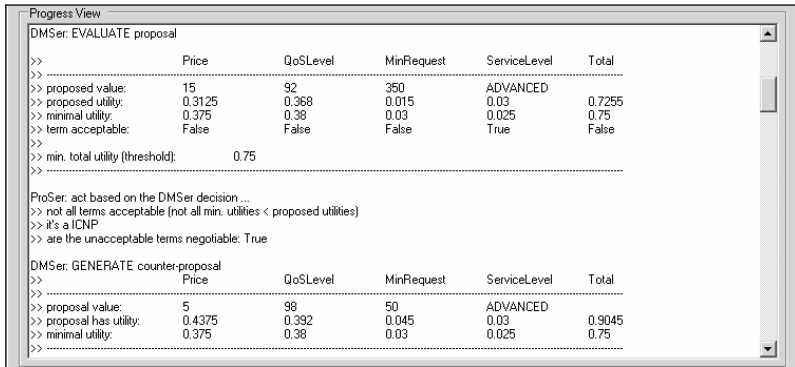


Fig. 4. Recorded messages on the SC GUI

7 Conclusions

This paper proposed a framework for dynamic creation of service level agreements based on automated negotiations between service providers and consumers. It provides several advantages over the existing approaches. The separation of the agreement and service layers as adopted from the WS-Agreement specification allows for a distinct encapsulation of the negotiation and application logics. It enables flexible relationships between the application and agreement service providers, and scenarios in which a number of application service providers can use various agreement providers to negotiate contracts with their clients. As described above, for that reason the service providers can leave their individual negotiation pre-configurations (e.g. supported contract types, acceptable agreements and negotiation constraints) encapsulated in the decision rules at the agreement provider side. Accordingly, the service clients can leave their decision rules at the agreement initiator component.

The second advantage comes from the modularization of the agreement layer. The participants – the agreement factory service in particular – have the flexibility to choose a suitable protocol depending on parameters such as the characteristics of the negotiation object or certain negotiation requirements of the client, e.g. if the client insists on negotiating on the basis of a particular protocol. In the same way it supports a flexible and dynamic choice of decision making models. It is possible to choose different levels of sophistication when making a decision and even changing the decision making model during a conversation is feasible.

The third advantage concerns the scalability and extensibility of the framework and its used components. Due to its modular architecture, additional protocol services or decision making models can easily be integrated to change the negotiation behavior of

the participant. We expect that this can significantly reduce the time necessary to reach an agreement and that it can allow making a large number of transactions within a small amount of time automatically.

Acknowledgments. This work has been partly supported by the EU FP6 Integrated Project on Adaptive Services Grid (EU-IST-004617) and the Adaptive Service Agreement and Process Management (ASAPM) in Services Grid project (AU-DEST-CG060081). The ASAPM project is proudly supported by the Innovation Access Programme - International Science and Technology established under the Australian Government's innovation statement, Backing Australia's Ability.

References

- [1] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J. Rofrano, J., Tuecke, S., Xu, M.: Web Service Agreement Specification (WS-Agreement) 1.1 (2004)
- [2] Chinnici, R., M. Gudgin, J.-J. Moreau, et al.: Web Services Description Language (WSDL) Version 1.2 (2003)
- [3] Czajkowski, K., Ferguson, D.F., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W.: The WS-Resource Framework (2004)
- [4] Czajkowski, K., Foster, I., Kesselman, C., Sander, V. and Tuecke, S.: SNAP: A Protocol for Negotiation of Service Level Agreements and Coordinated Resource Management in Distributed Systems. Job Scheduling Strategies for Parallel Processing: 8th International Workshop (JSSPP 2002), Edinburgh (2002)
- [5] Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Snelling, D., Tuecke, S., From Open Grid Services Infrastructure to Web Services Resource Framework: Refactoring and Evolution (2004)
- [6] Degermark, M., Kohler, T., Pink, S. and Schelen, O.: Advance reservations for predictive service in the internet. ACM/Springer Verlag, Journal on Multimedia Systems, 5(3) (1997)
- [7] Emorpha: FIPA-OS; Available at: <http://www.emorpha.com/research>
- [8] Ferrari, D., Gupta, A. and Ventre, G.: Distributed advance reservation of real-time connections. ACM/Springer Verlag, Journal on Multimedia Systems, 5(3) (1997)
- [9] FIPA: Foundation for Intelligent Physical Agents (FIPA); <http://www.fipa.org>
- [10] FIPA-CNP: FIPA Contract Net Interaction Protocol Specification; <http://www.fipa.org/specs/fipa00029/SC00029H.html>
- [11] FIPA-ICNP: Iterated Contract Net Interaction Protocol Specification; <http://www.fipa.org/specs/fipa00030/SC00030H.html>
- [12] Foster, I., Kesselman, C. and Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: International Journal of High Performance Computing Applications. 15 (2001) 3, pp. 200-222.
- [13] Foster, I., Kesselman, C., Nick, J. and Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In: Open Grid Service Infrastructure WG, Global Grid Forum (2002), pp. 5-25.
- [14] GGF: Global Grid Forum; <http://www.ggf.org>
- [15] Gimpel, H., Ludwig, H., Dan, A. and Kearney, B. D.: PANDA: Specifying Policies for Automated Negotiations of Service Contracts. In: IBM Research Report RC22844 (2003)
- [16] Globus: The Globus Alliance; <http://www.globus.org>

- [17] Guerin, R. and Schulzrinne, H.: Network quality of service. In Foster, I. and Kesselman, C. (editors). *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers (1999), pp. 479-503
- [18] Hafid, A., Bochmann, G. and Dssouli, R.: A quality of service negotiation approach with future reservations (nafur): A detailed study. *Computer Networks and ISDN Systems*, 30(8) (1998)
- [19] IBM, BEA, and Microsoft: *WS-Addressing* (2004)
- [20] Jade: Java Agent Development Framework; <http://sharon.cselt.it/projects/jade/index.html>
- [21] Jennings, N. R., Parsons, S., Sierra, C. and Faratin, P.: Automated Negotiation. Proc. 5th International Conference on Practical Application of Intelligent Agents and Multi-Agent Systems (PAAM-2000), Manchester, UK (2000) pp. 23-30
- [22] Keller, A., Kar, G., Ludwig, H., Dan, A. and Hellerstein, J. L.: Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture. Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS 2002) (2002) pp. 513-528.
- [23] Laasri, B., Laasri, H., Lander, S. and Lesser, V.: A Generic Model for Negotiating Agents. In: *International Journal on Intelligent and Cooperative Information Systems*. 1 (1992) 2, pp. 291-317.
- [24] Lamanna, D., Skene, J., Emmerich, W.: SLAng: A Language for Defining Service Level Agreements In Proc. of The International Workshop on Future Trends of Distributed Computing Systems (FTDCS'2003), San Juan, IEEE Computer Society Press (2003)
- [25] Lomuscio, A. R., Wooldridge, M. and Jennings, N. R.: A classification scheme for negotiation in electronic commerce (eds. F. Dignum and C. Sierra). In: *Lecture Notes in Computer Science*. 1991 (2001), pp. 19-33.
- [26] Ludwig, H., Keller, A., Dan, A., and King, R.: A Service Level Agreement Language for Dynamic Electronic Services. Proceedings of the 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002) (2002), pp. 25-32.
- [27] Neumann, J. V. and Morgenstern, O.: *The Theory of Games and Economic Behaviour*. Princeton University Press, Princeton, US (1944)
- [28] Parsons, S., Sierra, C. and Jennings, N.: Agents that reason and negotiate by arguing. In: *Journal of Logic and Computation*. 8 (1998) 3, pp. 261-292.
- [29] Pruitt, D. G.: *Negotiation Behaviour*. Academic Press, New York (1981)
- [30] Raiffa, H.: *The Art and Science of Negotiation*. Harvard University Press, Harvard, US (1982)
- [31] Sahai, A., Durante, A., Machiraju V.: Towards Automated SLA Management for Web Services, HPL-2001-130 (2002)
- [32] Sandholm, T. W.: Distributed Rational Decision Making. In: G. Weiss (ed.) *Multiagent Systems*, MIT Press, Cambridge, US (1999) pp. 201-258.
- [33] Sierra, C., Jennings, N. R., Noriega, P. and Parsons, S.: A Framework for Argumentation-Based Negotiation. Proc. 4th International Workshop on Agent Theories, Architectures and Languages, Rode Island, USA, pp. 177-192
- [34] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., Snelling, D. and Vanderbilt, P.: *Open Grid Services Infrastructure (OGSI) 1.0*. (2003), pp. 7-12.
- [35] University of Virginia, Grid Computing Group: *WSRF.net* (2004)
- [36] W3C: *Web Services* (2004). <http://www.w3.org/2002/ws/>
- [37] Wasson, G.: *WSRF.NET Programmer's Reference* (2005)
- [38] Wolf, L. and Steinmetz, R.: Concepts for reservation in advance. *Kluwer Journal on Multimedia Tools and Applications*, 4(3) (1997)

A Constrained Object Model for Configuration Based Workflow Composition

Patrick Albert¹, Laurent Henocque², and Mathias Kleiner^{1,2}

¹ ILOG, Gentilly, France
{palbert, mkleiner}@ilog.fr
² LSIS, Marseille, France
laurent.henocque@lsis.org

Abstract. Automatic or assisted workflow composition is a field of intense research for applications to the world wide web or to business process modeling. Workflow composition is traditionally addressed in various ways, generally via theorem proving techniques. Recent research [1] observed that building a composite workflow bears strong relationships with finite model search, and that some workflow languages can be defined as constrained object metamodels [2,3]. This lead to consider the viability of applying configuration techniques to this problem, which was proven feasible. Constrained based configuration expects a constrained object model as input. The purpose of this document is to formally specify the constrained object model involved in ongoing experiments and research using the Z specification language.

1 Introduction

We place ourselves in the scope of automatic or computer aided workflow composition, with immediate applications to Business Process Modeling or the Semantic Web. The basic assumptions for composing workflows is that there exists a form of directory listing of elementary workflows that are potential candidates for composition, as well as a directory listing of transformations that are usable to mediate between workflows having “bitwise” incompatible message type requirements. How and when a proper list of elementary workflows and transformations can be obtained is beyond the scope of this research, and is treated as if it was available to the program from the start.

We also assume that the composition process is goal oriented: a user may list the message types he can possibly input to the system (e.g. credit card number, expiry date, budget, yes/no answer etc...), and the same user may formulate the precise (set of) message(s) that must be output by the system (e.g. a plane ticket reservation electronic confirmation: the “goal”).

A previous work [1] proved the feasibility of using a configurator program to solve this problem, and presented a constrained object model adequate for this purpose, using the semi formal language UML/OCL. Although it was shown in [4] that such a use of UML/OCL is viable, the language is also known as having limitations, notably concerning relational operators. The original contribution

of this work is to propose a formal specification of the same constrained object model using the Z specification language. Z was shown suitable for such a usage in [5], via a framework for the Z specification of constrained object models¹. This research heads towards the complete formal specification of a constrained object model for workflow composition.

The plan of the article is as follows. The current section 1 is introductory, and briefly presents the context retained for dealing with workflow composition configuration in Subsection 1.1, and related work in 1.2. Section 2 briefly introduces Z and the predefined class construct used in the specification. Section 3 presents the specification of a constrained object model combining a metamodel for activities and data type ontologies. Section 4 concludes.

1.1 Context for Workflow Composition

We consider workflows defined using a variant of extended workflow nets, as are UML2 activity diagrams [2] or the YAWL language [3]. The underlying model is that of colored Petri nets, where messages (tokens) have types. The full context, similar to that in [6,7,8] is presented in greater detail in [1] and needs not be recalled here. We illustrate our central assumptions by considering the rather complex Producer/Shipper composition problem from [8], interesting because the participating workflows must be interleaved, and results must be aggregated.

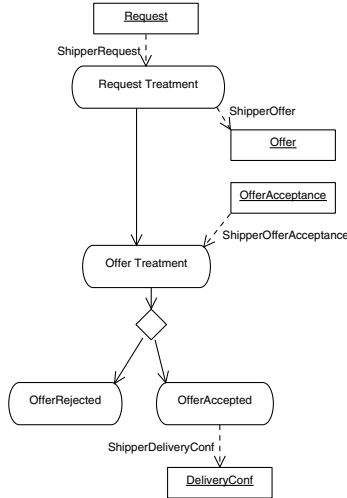


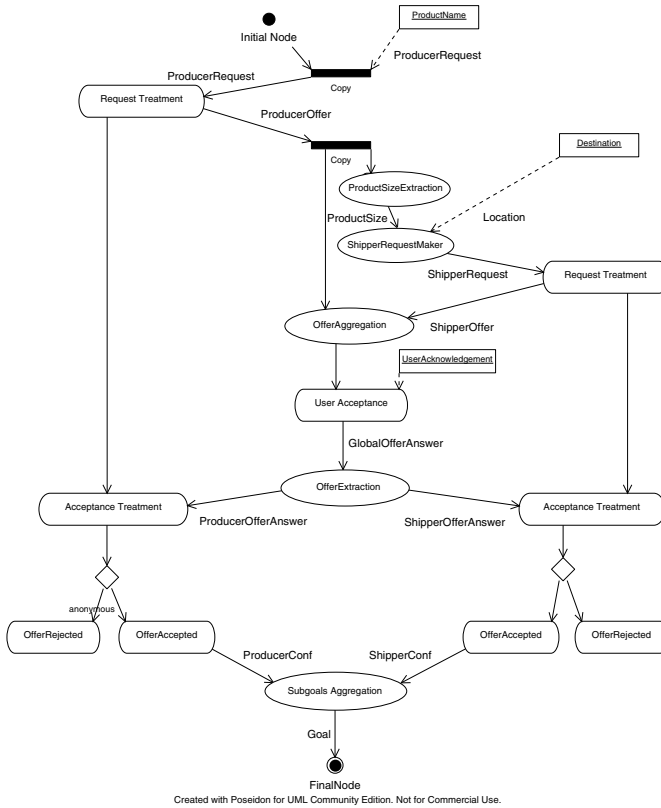
Fig. 1. The shipper provided workflow

Figure 1 illustrates the shipper’s partial workflow, as defined before search begins. The producer’s workflow is similar, modulo the message type ontologies. The composition result is shown in Figure 2. This composed workflow involves

¹ Also called Object Oriented Constraint Programs.

synchronization, interleaving, transformations (we used oval boxes to denote transformations) and it should be noted that some execution paths are discarded: indeed, under user rejection, the goal cannot be fulfilled. This illustrates why we later will need a Boolean attribute for active paths in the metamodel.

Basically, the external user is present via the message it inputs to the global workflow. In Figure 2, all the workflow elements that are not visible in the shipper workflow above or its producer counterpart must be introduced automatically in order to obtain a valid composition.



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Fig. 2. The shipper and producer composed workflow

1.2 Related Work

The underlying approach to Semantic Web Service composition using configuration techniques is presented in [1], as well as an OCL based problem specification. Automated workflow composition is a field of intense activity, with applications to at least two wide areas: Business Process Modeling and the (Semantic) Web Services. Tentative techniques to address this problem are experimented using

many formalisms and techniques, among which Situation calculus [9], Logic programming [10], Type matching: [11], Coloured Petri nets: [6,7], Linear logic: [12], Process solving methods [13,14,15], AI Planning [16], Hierarchical Task Network (HTN) planning [17,18], Markov decision processes [19].

2 Introducing Z

For space reasons, it is impossible to make this paper self contained, since this would suppose a thorough presentation of both the UML notation [2], and the Z specification language [20]. The reader, if novice in these domains, is kindly expected to make his way through the documentation, which is electronically available. We now provide a brief description of several useful Z constructs.

2.1 Data Types as Named Sets

Z data types are possibly infinite sets, either uninterpreted ($[DATE]$), or axiomatically defined as finite sets ($dom : \mathbb{F}\mathbb{N}$), or declared as free types ($colors ::= red \mid green \mid blue$). Other relation types are built as cross products of other sets.

2.2 Axiomatic Definitions

Axiomatic definitions allow to define global symbols having plain or relation types. For instance, a finite group is declared as

$$\begin{array}{|l}
 zero : dom \\
 inverse : dom \rightarrow dom \\
 sum : (dom \times dom) \rightarrow dom \\
 \hline
 \forall x : dom \bullet sum(x, inverse(x)) = zero \\
 \forall x : dom \bullet sum(x, zero) = x \\
 \forall x, y : dom \bullet sum(x, y) = sum(y, x) \\
 \forall x, y, z : dom \bullet sum(x, sum(y, z)) = sum(sum(x, y), z)
 \end{array}$$

The previous axiomatic definition illustrates cross products and function definitions as means of typing Z elements. Now axioms or theorems are expressed in classical math style, involving previously defined sets. For instance, we may formulate that the inverse function above is bijective (this is a theorem) in several equivalent ways as e.g. ' $inverse \in dom \rightsquigarrow dom$ ' (\rightsquigarrow defines a bijection), or explicitly using an appropriate axiom: ' $\forall y : dom \bullet \exists x : dom \bullet inverse(x) = y$ '.

2.3 Schemas

The most important Z construct, *schemas*, occur in the specification in the form of named axiomatic definitions. A schema $[D \mid P]$ combines one or several variable declarations (in the declaration part D) together with a predicate P stating validity conditions (or constraints) that apply to the declared variables. The reader is directed to the Z Reference Manual[20] for details.

<i>SchemaOne</i>
$a : \mathbb{N}$
$b : 1 \dots 10$
$b < a$

The schema name hides the inner declarations, which are not global. A schema name (*SchemaOne*) is a shortcut for its variable and predicate declarations that can be universally or existentially quantified at will. Schemas do not define object identity hence are not suitable as such to describe object oriented semantics.

2.4 Shortcut Notation for Class Specifications

Z being non object oriented in any way, the specification of an object system is verbose. In [5] are proposed the following shortcut definition for classes and types, which makes use of the keywords *class*, *abstract*, *discriminator*, *inherit*. We illustrate the general framework using a simple three class example. Assume that A,B,C are the classes in a constrained object system where B and C inherit A. The Z “extension” from [5] allows for the following simple declarations:

<i>class – A : abstract</i>
<i>–discriminators : default</i>
$a : \mathbb{N}$
$a < 10;$

<i>class – B : concrete</i>
<i>–inherit : A – default</i>
$b : \mathbb{N}_1$

<i>class – C : concrete</i>
<i>–inherit : A</i>
$a \geq 5;$

These class declarations are a shortcut for the declaration in the Z specification of diverse sets and axiomatic definitions, of the schemas : *ObjectDef*, *ClassDefA*, *ClassSpecA*, *ClassDefB*, ..., and of the sets *instances(ClassA)*, *A*, *instances(ClassB)*, ..., with:

[*ObjectReference*]
ReferenceSet == \mathbb{F} *ObjectReference*

Object references are central to the object system, since they allow for specifying object identity. We have three class names:

CLASSNAME ::= *ClassA* | *ClassB* | *ClassC*

The function *instances* maps class names to sets of object references:

$$\mid \text{instances} : \text{CLASSNAME} \rightarrow \text{ReferenceSet}$$

The *ObjectDef* schema introduces a part common to all object representations:

$$\boxed{\begin{array}{l} \text{ObjectDef} \\ \text{ref} : \text{ObjectReference} \\ \text{class} : \text{CLASSNAME} \end{array}}$$

Now, the *ClassDef'X'* schemas introduce the part specific to each class, plus inheritance using schema inclusion

$$\boxed{\begin{array}{l} \text{ClassDefA} \\ a : 1 \dots 10 \end{array}}$$

$$\boxed{\begin{array}{l} \text{ClassDefB} \\ \text{ClassDefA} \\ b : \mathbb{N}_1 \end{array}}$$

$$\boxed{\begin{array}{l} \text{ClassDefC} \\ \text{ClassDefA} \\ a \geq 5 \end{array}}$$

Class specifications introduce the common *ObjectDef* part and constrain the *class* attribute to its proper value:

$$\begin{aligned} \text{ClassSpecA} &\hat{=} \text{ClassDefA} \wedge [\text{ObjectDef} \mid \text{class} = \text{ClassA}] \\ \text{ClassSpecB} &\hat{=} \text{ClassDefB} \wedge [\text{ObjectDef} \mid \text{class} = \text{ClassB}] \\ \text{ClassSpecC} &\hat{=} \text{ClassDefC} \wedge [\text{ObjectDef} \mid \text{class} = \text{ClassC}] \end{aligned}$$

Then finally the object system can be modelled, by introducing the sets *A*, *B*, *C* of references that correspond to the usual understanding of object “types”, again accounting for inheritance:

$$\boxed{\begin{array}{l} A, B, C : \text{ReferenceSet} \\ A = \text{instances}(\text{ClassA}) \cup B \cup C \\ B = \text{instances}(\text{ClassB}) \\ C = \text{instances}(\text{ClassC}) \\ \text{instances}(\text{ClassA}) = \{o : \text{ClassSpecA} \mid o.\text{class} = \text{ClassA} \bullet o.i\} \\ \text{instances}(\text{ClassB}) = \{o : \text{ClassSpecB} \mid o.\text{class} = \text{ClassB} \bullet o.i\} \\ \text{instances}(\text{ClassC}) = \{o : \text{ClassSpecC} \mid o.\text{class} = \text{ClassC} \bullet o.i\} \\ \forall i : \text{instances}(\text{ClassA}) \bullet (\exists_1 x : \text{ClassSpecA} \bullet x.\text{ref} = i) \\ \forall i : \text{instances}(\text{ClassB}) \bullet (\exists_1 x : \text{ClassSpecB} \bullet x.\text{ref} = i) \\ \forall i : \text{instances}(\text{ClassC}) \bullet (\exists_1 x : \text{ClassSpecC} \bullet x.\text{ref} = i) \end{array}}$$

The sequel of the presentation makes use of the “class” shortcuts introduced above, and of some of the role dereferencing operators \rightarrow , \dashv , \cdot , \rightsquigarrow :

$[X]$
$_ \rightarrow _ : \mathbb{F} \textit{ObjectReference} \times (\textit{ObjectReference} \rightarrow X) \rightarrow \textit{bag } X$
$_ \dashv _ : \mathbb{F} \textit{ObjectReference} \times (\textit{ObjectReference} \rightarrow X) \rightarrow X$
$_ \cdot _ : \mathbb{F} \textit{ObjectReference} \times (\mathbb{F} \textit{ObjectReference} \rightarrow \mathbb{F} X) \rightarrow \mathbb{F} X$
$_ \rightsquigarrow _ : \textit{ObjectReference} \times (\mathbb{F} \textit{ObjectReference} \rightarrow \mathbb{F} X) \rightarrow \mathbb{F} X$
$\forall s : \mathbb{F} \textit{ObjectReference}; r : \textit{ObjectReference} \rightarrow X \bullet s \rightarrow r = \textit{bagOf}(r)(s)$
$\forall s : \mathbb{F} \textit{ObjectReference}; r : \textit{ObjectReference} \rightarrow X \bullet$ $s \dashv r = (\mu t : \textit{bagOf}(r)(s) \bullet \textit{first } t)$
$\forall s : \mathbb{F} \textit{ObjectReference}; r : \mathbb{F} \textit{ObjectReference} \rightarrow \mathbb{F} X \bullet s \cdot r = r(s)$
$\forall o : \textit{ObjectReference}; r : \mathbb{F} \textit{ObjectReference} \rightarrow \mathbb{F} X \bullet o \rightsquigarrow r = r(\{o\})$

where the function *bagOf* maps every function from *ObjectReference* to *X* to a function from sets of *ObjectReference* to bags of *X*, assuming the existence of a function *pickFirst* applied to any set of (totally ordered) object references:

$[X]$
$\textit{bagOf} : (\textit{ObjectReference} \rightarrow X) \rightarrow (\mathbb{F} \textit{ObjectReference} \rightarrow \textit{bag } X)$
$\forall f : \textit{ObjectReference} \rightarrow X \bullet \textit{bagOf}(f)(\emptyset) = \square$
$\forall f : \textit{ObjectReference} \rightarrow X \bullet$ $\forall d : \mathbb{F}_1(\text{dom } f) \bullet (\text{let } x == \textit{pickFirst}(d) \bullet$ $\textit{bagOf}(f)(d) = (\textit{bagOf}(f)(d \setminus \{x\}) \uplus (\{f(x) \mapsto 1\}))$)

3 A Metamodel for Workflow Composition

Workflow reasoning requires a workflow language with enough generality to be practically viable. Furthermore in our case, since we expect to treat workflow composition as a configuration task, it is of particular importance that the language is modular wrt. most if not all the workflow patterns referenced in [21]. The simplest such language is the extended workflow net YAWL language [3], now very close from a subset of UML2 [2] activity diagrams.

We present our constrained object model according to the standard model driven architecture recommendations, except for the use of Z as a formal specification language. The next subsection introduces the classes, their associations and attributes using class diagrams. Then follows a detailed presentation of the relevant constraints, in the form of Z axiomatic definitions.

3.1 Metamodel Specification

Actions. Actions are the core constituents of a workflow. As defined in UML2, actions may have a certain number of messages as their inputs and outputs. Those inputs/outputs have defined types, taken from existing ontologies of data

types. All actions have an “owner” (the original workflow they belong to: for instance, an action may belong to the Producer workflow). In UML2 the term workflow is synonym to that of Activity. All workflow parts that are dynamically added by the configurator in order to create the composed workflow belong to a newly introduced owner called the *Composition Workflow*. There are different types of actions, illustrated in the metamodel in Figure 3:

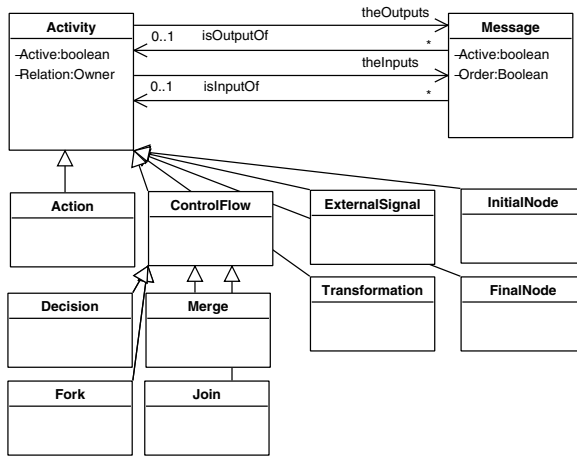


Fig. 3. Meta-model for workflows activities

- Initial nodes: the starting point of the workflow. Initial nodes don’t take any inputs. They are graphically represented using a black circle.
- Final nodes: a possible end of the workflow. Final nodes don’t produce any outputs and are represented using a white circle and a black dot in the center. There may be several final nodes in a workflow.
- Control nodes: joins, forks, merges, decisions. A fork initiates concurrency by duplicating its input token to all outputs. Join is the corresponding synchronization construct. Decision (also known as “split”) and merge are the standard if/else conditional branching constructs.
- Actions: operations executed locally by the workflow owner.
- Transformations: activities for transforming message data types with no further side effect. Transformations are called data mediators in the context of web service composition. Available transformations can be chosen by the composition designer, or they can be discovered (as e.g. in the context of Semantical Web Services). The distinction between actions in general and transformations in particular is widely acknowledged in the workflow/process/WS communities. Called “data mediators” in the context of WS, UML2 “transformations” have the sole effect of reformatting data, and do not enter in further interactions with other parts of the workflow.
- External Signals: An activity that outputs external messages, typically user provided messages.

Class specifications. We now introduce the Z specification of the constrained object model used for the configuration of workflow compositions. The class specifications listed below straightforwardly follow from Figure 3. We use the notational shortcuts introduced in [5] for class declarations. These shortcuts allow for straightforward class definitions, and introduce several (hidden) auxiliary data types and sets. We also take the freedom of introducing the type “Boolean” in the language, for the sake of simplicity ($Boolean ::= true \mid false$).

$class - Activity : abstract$ $active : Boolean$

and similarly for *Action*, *ControlFlow*, *ExternalSignal*, *InitialNode*, *FinalNode*, *Transformation*.

$class - Decision : concrete$ $-inherit : ControlFlow$

and similarly for *Merge*, *Split*, *Join*.

$class - Message : concrete$ $active : Boolean$ $order : \mathbb{N}$
$order \geq 0$

Relation specifications. According to Figure 3 there exists a relation between workflows and abstract actions: each action has a single owner workflow. This can be modeled using an injection ‘ $owner : Activity \rightarrow Workflow$ ’. The relations listed in Figure 3 between the Activity and Message classes can be specified as:

$outputs : Activity \leftrightarrow Message$ $inputs : Activity \leftrightarrow Message$ $isOutputOf : Message \rightarrow Activity$ $isInputOf : Message \rightarrow Activity$
$isOutputOf = outputs \sim$ $isInputOf = inputs \sim$

where $inputs \sim$ denotes the relational inverse of the relation $inputs$, and \rightarrow denotes a partial injection. Partial injections are useful to specify situations modeled using 0, 1 cardinalities as in Figure 3.

3.2 Ontology of Message Types

Each message has a related data type, from a workflow specific ontology. We use predefined ontologies for user interaction schemes, and import the ones required by the selected web services. User interaction schemes, as implemented by the

composition activity *OfferAcceptance*, constrain the types of their I/O messages. From an abstract standpoint, they output an *OfferAnswer* if both an *Offer* and an *UserAcknowledgement* are provided. However the precise *Offer/ OfferAnswer* type match is constrained: they must share the same owner workflow. For example, a *ShipperOfferAnswer* can be output only if a *ShipperOffer* is input to the user interaction. Figure 4 illustrates the fact that such answers belong to both the hierarchy of standard datatypes and of imported service ontologies.

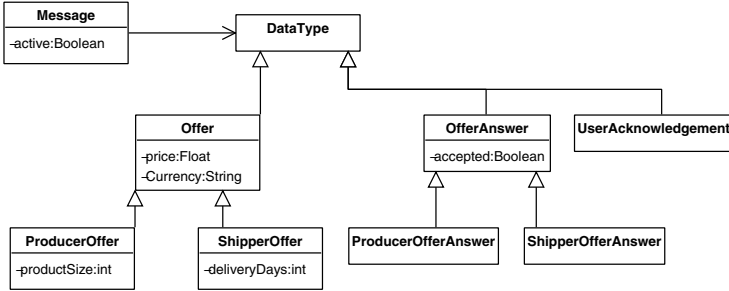


Fig. 4. Abstract model for workflow data types ontologies

Class specifications. The class specifications straightforwardly follow from Figure 4. We assume the free type '*Currency ::= Euro | Dollar | Yen ...*'

```

class - DataType : abstract
-----
class - Offer : abstract
- inherit : DataType
price : ℕ
currency : Currency
-----
    
```

and similarly according to Figure 4 for *OfferAnswer*, *UserAcknowledgement*, *ProducerOffer*, *ProducerOfferAnswer*, *ShipperOffer*, *ShipperOfferAnswer*.

Relations. There is a relation between the Message and DataType class, whereby each Message binds to at most one DataType object. This is specified using a partial function. The same DataType might be shared across several Messages, hence *dataType* is not injective: '*dataType : Message →+ DataType*'.

3.3 Semantics

The previous object models are not enough to describe valid compositions, and require a number of constraints governing the possible combinations of partial workflows and additional elements that form acceptable compositions. We specify here a limited number of these constraints that are representative enough to grasp the general idea.

Composition specific constraints. From the simplified and slightly adapted subset of the UML2 activity diagram metamodel in Figure 3, we observe that both the *Activity* and *Message* classes implement a Boolean attribute called “active”. This Boolean helps ensuring that a workflow can be composed from sub-workflows if and only if at least one valid path yields the expected goal. This allows our tool to produce composite workflows under the additional constraint that control flow constructs must match the following constraints applying to activities and messages:

- if an action is active, then all its input messages are active,
- if an active message is output of a join, all its inputs must be active,
- if an active message is output of a decision or a fork, then the input of this activity must be active,
- if an active message is output of a merge, at least one of this merge’s inputs must be active.

According with these, the program builds solutions such that at least one path leads from the initial node to a final node reached via a message having the correct goal type. In the case a valid path traverses a fork or a join, all other incoming/outgoing paths must be valid too. If a user wants a robust solution (meaning that all branches are valid), this can be obtained by forcing all parts of the workflow to be active.

Activation related constraints. These constraints are not problem-specific and therefore apply to any composition. The Boolean “active” denotes which part of a workflow indeed participate in the solution. The rationale for this is as follows: a workflow argument to a composition may involve decision/merge paths that are ignored because either the conditions for their activation are known as impossible (e.g. because from the connected message, we know that a test will always fail) or because an exterior message required for their successful execution is known as missing (e.g. a user message giving a credit card number in case the user has none).

“If an action is active, then all of its inputs must be active messages”:

$$\forall a : \text{Action} \bullet a.\text{active} \Rightarrow \forall m \in \text{inputs}(a) \bullet m.\text{active}$$

“If an active message is output of a join, then all its inputs must be active”:

$$\begin{aligned} \forall m : \text{Message} \mid m.\text{active} \wedge m \rightsquigarrow \text{isOutputOf} \in \text{Join} \bullet \\ \forall m' : \text{Message} \mid m' \in m \rightsquigarrow \text{isOutputOf} \rightsquigarrow \text{inputs} \bullet m'.\text{active} \end{aligned}$$

“If an active message is output of a decision or a fork, all its inputs must be active”:

$$\begin{aligned} \forall m : \text{Message} \mid m.\text{active} \wedge m \rightsquigarrow \text{isOutputOf} \in \text{Decision} \cup \text{Fork} \bullet \\ \forall m' : \text{Message} \mid m' \in m \rightsquigarrow \text{isOutputOf} \rightsquigarrow \text{inputs} \bullet m'.\text{active} \end{aligned}$$

“If an active message is output of a merge, one of its inputs must be active”:

$$\begin{aligned} \forall m : \text{Message} \mid m.\text{active} \wedge m \rightsquigarrow \text{isOutputOf} \in \text{Merge} \bullet \\ \exists m' : \text{Message} \mid m' \in m \rightsquigarrow \text{isOutputOf} \rightsquigarrow \text{inputs} \bullet m'.\text{active} \end{aligned}$$

“If a merge is active, then at least one of its inputs must be an active message”:

$$\forall a : \text{Merge} \bullet a.\text{active} \Rightarrow \exists m \in \text{inputs}(a) \bullet m.\text{active}$$

Composition related constraints. We assume the existence of a specific workflow instance called “Composition” ($\text{Composition} : \text{Workflow}$). “All messages input of an external workflow are output of the composition workflow”:

$$\begin{aligned} \forall m : \text{Message} \bullet m \rightsquigarrow \text{isInputOf} \rightsquigarrow \text{owner} \neq \text{Composition} \Rightarrow \\ m \rightsquigarrow \text{isOutputOf} \rightsquigarrow \text{owner} = \text{Composition} \end{aligned}$$

and conversely “All messages output of an external workflow are input of the composition workflow”:

$$\begin{aligned} \forall m : \text{Message} \bullet m \rightsquigarrow \text{isOutputOf} \rightsquigarrow \text{owner} \neq \text{Composition} \Rightarrow \\ m \rightsquigarrow \text{isInputOf} \rightsquigarrow \text{owner} = \text{Composition} \end{aligned}$$

Message ordering related constraints. Our model implements an integer “order” parameter in the Message class that is used to prevent building interlocking or looping constructs.

“the order of an action’s input message is lower than the action’s output messages orders (this “ordering” constraint allows to prevent looping situations in the composite workflow):

$$\begin{aligned} \forall m : \text{Message} \bullet \forall m' : \text{Message} \mid \\ m' \in m \rightsquigarrow \text{isInputOf} \rightsquigarrow \text{outputs} \bullet m.\text{order} < m'.\text{order} \end{aligned}$$

Pre-defined composition constraints. An OfferAcceptance action expects as input an Offer plus a UserAcknowledgement, and produces an OfferAnswer as a result. Both the Offer and the OfferAnswer point to actions having the same workflow owner, which is not the Composition workflow.

$$\begin{aligned} \forall o : \text{Offer}, a : \text{OfferAnswer} \mid o \rightsquigarrow \text{isInputOf} = a \rightsquigarrow \text{outputOf} \bullet \\ o \rightsquigarrow \text{isOutputOf} \rightsquigarrow \text{owner} = a \rightsquigarrow \text{isInputOf} \rightsquigarrow \text{owner} \end{aligned}$$

Also, Fork nodes have the same type for their inputs and outputs. Formulating such a constraint is possible, under the assumptions in [5]

$$\begin{aligned} \forall f : \text{Fork}, i, o : \text{Message} \mid i \rightsquigarrow \text{isInputOf} = o \rightsquigarrow \text{isOutputOf} = f \bullet \\ i \rightsquigarrow \text{getClass} = o \rightsquigarrow \text{getClass} \end{aligned}$$

Problem specific constraints. User provided message types fall into a few categories, as e.g. credit card information, age, or budget... Assuming the user input message type classes UserInput1 , UserInput2 , ...

$$\begin{aligned} \forall m : \text{Message} \mid m \rightsquigarrow \text{isOutputOf} \in \text{ExternalSignal} \bullet \\ m \rightsquigarrow \text{getClass} \in \text{UserInput1}, \text{UserInput2}, \dots \end{aligned}$$

Also, a concrete composition instance must list the available transformation types. In the context of (semantic) web service discovery, such transformations may be the result of a query to a repository of ontology mediators.

Finally, a precise workflow composition problem instance may involve policy related constraints: constraints that are required to filter out valid yet unwanted compositions, for instance in a way such that offer's prices fall below a given maximum value.

4 Conclusion

This work proposes a formal specification using the Z language of a constrained object model involved in automatic workflow composition. Constrained object models can be exploited straightforwardly by configurators to achieve automatic or assisted workflow composition. Our formalization abstracts from the technology used to assess the validity of such an approach, so that different configuration techniques can be tested or compared on the same problem.

Configuration expects a constrained object model to operate, hence puts the application design in a field familiar to many engineers. An essential part of the object model, the metamodel for activity diagrams, already exists as a (subset of) part of the UML2 specification relative to activity diagrams. Our Z specification is compatible with all UML2 class diagram features (including multiple inheritance and inheritance discriminators), thus allowing for the straightforward translation of class diagrams. The advantages of Z wrt. UML/OCL are in the statement of constraints. For instance UML dramatically lacks relational constructs and a cross product operator. Since in configuration problems, relations abound with extra semantics (injectivity etc.), object model constraints can be freely stated in Z, also taking advantage of using the richness of Z relational operators, and the ability to define additional operators.

Acknowledgments

The authors would like to thank the anonymous referees, and the European DIP integrated project and the ILOG company for their financial support in carrying out this research. Parts of this research were carried out while Laurent Henocque was invited as a researcher at GRIMAAG (University of Antilles Guyane).

References

1. Albert, P., Henocque, L., Kleiner, M.: Configuration based workflow composition. In: proceedings of International Conference on Web Services ICWS'05, Orlando, Florida, USA (2005) to appear
2. OMG: UML v. 2.0 specification. Object Management Group (2003)
3. van der Aalst, W., Aldred, L., Dumas, M.: Design and implementation of the yawl system. qut technical report, fit-tr-2003-07. Technical report, Queensland University of Technology, Brisbane (2003)
4. Felfernig, A., Friedrich, G., Jannach, D., Zanker, M.: Configuration knowledge representation using uml/ocl. In: Proceedings of the 5th International Conference on The Unified Modeling Language, Springer-Verlag (2002) 49–62

5. Henocque, L.: Modeling object oriented constraint programs in Z. RACSAM (Revista de la Real Academia De Ciencias serie A Mathematicas), special issue about Artificial Intelligence and Symbolic Computing (2004) 127–152
6. Yi, X., Kochut, K.: A cp-nets-based design and verification framework for web services composition. In: proceedings of 2004 IEEE International Conference on Web Services, July 2004, San Diego, California, USA (2004)
7. Dijkman, R., Dumas, M.: Service-oriented design: A multi-viewpoint approach, ctit technical report series no. 04-09. Technical report, Centre for Telematics and Information Technology, University of Twente, The Netherlands (2004)
8. Pistore, M., Barbon, F., Bertoli, P., Shaparau, D., Traverso, P.: Planning and monitoring web service composition. In: proceedings of the Workshop on Planning and Scheduling for Web and Grid Services held in conjunction with ICAPS 2004, Whistler, British Columbia, Canada (2004)
9. McIlraith, S., Son, T.: Adapting golog for composition of semantic web services. In: proceedings of Conference on Knowledge Representation and Reasoning. (2002)
10. Sirin, E., Hendler, J., Parsia, B.: Semi automatic composition of web services using semantic descriptions. In: proceedings of the ICEIS-2003 Workshop on Web Services: Modeling, Architecture and Infrastructure, Angers, France (2003)
11. Constantinescu, I., Faltings, B., Binder, W.: Large scale, type-compatible service composition. In: proceedings of IEEE International Conference on Web Services (ICWS 2004), San Diego, USA (2004)
12. Rao, J., Kungas, P., Matskin, M.: Logic-based web service composition: from service description to process model. In: proceedings of the 2004 IEEE International Conference on Web Services, ICWS 2004, San Diego, California, USA (2004)
13. Benjamins, V., Fensel, D. Special Issue on Problem-Solving Methods. International Journal of Human-Computer Studies (IJHCS) **49(4)** (1998) 305–313
14. Gómez-Pérez, A., González-Cabero, R., Lamaa, M.: A framework for design and composition of semantic web services. In: Semantic Web Services, 2004 AAAI Spring Symposium Series. (2004)
15. Thakkar, S., Knoblock, C., Ambite, J., Shahabi, C.: Dynamically composing web services from on-line sources. In: proceedings of AAAI-02 Workshop on Intelligent Service Integration, Edmondson, Canada (2002)
16. Carman, M., Serafini, L., Traverso, P.: Web service composition as planning. In: proceedings of ICAPS03 International Conference on Automated Planning and Scheduling, Trento, Italy (2003)
17. Sirin, E., Parsia, B., Wu, D., Hendler, J., Nau, D.: HTN planning for web service composition using SHOP2. Journal of Web Semantics **1** (2004) 377–396
18. Vukovic, M., Robinson, P.: Adaptive, planning based, web service composition for context awareness. In: proceedings of the Second International Conference on Pervasive Computing, Vienna, Austria (2004) to appear
19. Doshi, P., Goodwin, R., Akkiraju, R., Verma, K.: Dynamic workflow composition using markov decision processes. International Journal of Web Services Research **2(1)** (2005) 1–17
20. Spivey, J.M.: The Z Notation: a reference manual. Prentice Hall originally, now J.M. Spivey (2001)
21. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases (2003) 5–51

A High-Level Specification for Mediators (Virtual Providers)*

Michael Altenhofen¹, Egon Börger^{2,**}, and Jens Lemcke¹

¹ SAP Research, Karlsruhe, Germany

{michael.altenhofen, jens.lemcke}@sap.com

² Università di Pisa, Dipartimento di Informatica, I-56125 Pisa, Italy

boerger@di.unipi.it,

egon.boerger@sap.com

Abstract. We define a high-level model to mathematically capture the semantical meaning of abstract Virtual Providers (VP), their instantiation and their composition into rich mediator structures.

1 Introduction

For the configuration [1] and composition [2,3] of Web services in interaction protocols, a central role is played by process mediation (see [4], MIBIA [5], WSMF [6], WebTransact [7]). We propose here an abstract model for mediators (Sects. 2, 3), viewed as Virtual Providers (VP). The model supports provably correct mediator composition and the definition of appropriate equivalence concepts (Sect. 4), which underlay algorithms for the discovery and run-time selection of services satisfying given requests. In Sect. 5 we illustrate our definitions by a Virtual Internet Service Provider (VISP) case study.

We start with a simple interaction model where each single request receives a single answer from the VP, with no need to relate multiple requests. However, to process single requests the VP has a hierarchical structure at its disposition: Each request arriving at VP is viewed as root of a so-called *seq/par tree* of further requests, which are forwarded to other providers. The children of a request node represent subrequests which are elaborated in sequence. Each subrequest node may have in turn children representing multiple subsubrequests, which are elaborated independently of each other. Nestings of such alternating *seq/par trees* and other more sophisticated hierarchical subrequest structures can be obtained by appropriate compositions of VPs as defined in Sect. 4.1.

The compositionality of our mediator model stems from an explicit separation of its tree processing component from its communication interfaces for sending and receiving requests and answers. This separation, defined in Sect. 2 on the basis of an abstract message passing system, supports a flexible definition of the service behaviour of VPs and of their behavioural equivalence (Sect. 4), which also allows one to clearly identify the place of data mediation during the discovery

* Work and research on this paper were partly funded by the EU-project DIP.

** On sabbatical leave at SAP Research, Karlsruhe, Germany.

and runtime selection of providers able to satisfy given requests. Furthermore, the separation of communication from proper request processing supports a smooth integration of a variety of workflow and interaction patterns [8,9].

In Sect. 2.3 the single-request oriented model is refined by a notion of internal state, so that the relevant information about previous requests, which may be related to an incoming request, can be extracted from the internal state — in practical Web applications typically by a wrapping session handling module. This refinement step is only a tiny illustration of much more one can do to turn our abstract VP model in a faithful way into fully developed mediator code.

As modelling framework we use Abstract State Machines (ASM), a form of pseudo-code working on arbitrary structures. ASMs provide the necessary flexible and precise mechanism we need to capture dynamic behavior over abstract states.¹ An introduction into the ASM method for high-level system design and analysis is available in textbook form in [10], but most of what we use here is self-explanatory. The various refinements used are instances of the general ASM refinement concept defined in [11].

2 The Communication Interface of VIRTUALPROVIDERS

We see a Virtual Provider as an interface (technically speaking as an ASM module VIRTUALPROVIDER) providing the following five methods (read: ASMs):

- RECEIVEREQ for receiving request messages (elements of a set *InReqMssg* of legal incoming request messages) from clients,²
- SENDANSW for sending answer messages (elements of a set *OutAnswMssg*) back to clients,
- PROCESS to handle request objects, elements of a set *ReqObj* of internal representations of *ReceivedRequests*, typically by sending to providers a series of subrequests to service the currently handled request *currReqObj*,³
- SENDREQ for sending request messages (elements of a set *OutReqMssg*) to providers (possibly other VPs, see the VP composition in Sect. 4.1),
- RECEIVEANSW for receiving incoming answer messages (elements of a set *InAnswMssg*) from providers.

This module view of VIRTUALPROVIDER — as a collection of defined and callable machines, without a main ASM defining the execution flow — separates the specification of the functionality of VP components from that of their schedulers. The underlying architecture is illustrated in Fig. 1.

¹ This is not the place for a systematic comparison of different methods. Since we start modelling from scratch here, no other related work is used besides what is cited.

² Since instances of VIRTUALPROVIDER can be composed (see Sect. 4.1), such a client can be another VP' asking for servicing a subrequest of a received request.

³ Since the underlying message passing system is abstract, VIRTUALPROVIDER can be instantiated in such a way that also PROCESS itself can be a provider and thus service a subrequest 'internally'. This reflects that the mediation role for a request is different from the role of actually servicing it.

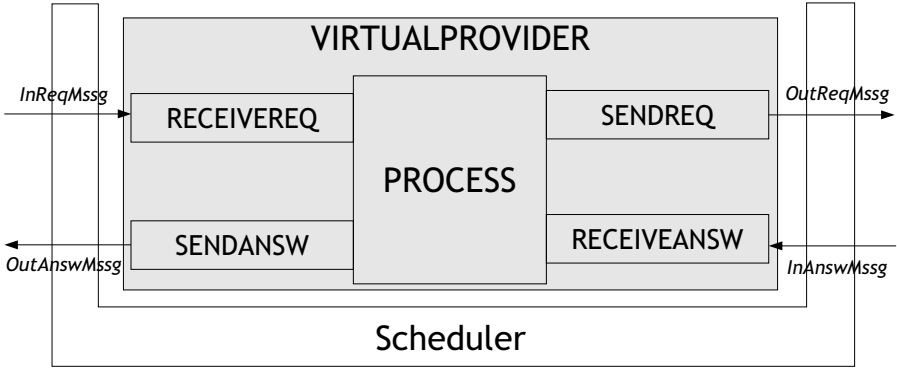


Fig. 1. Architecture

```

MODULE VIRTUALPROVIDER =
    RECIIVEREQ SENDANSW PROCESS SENDREQ RECEIVEANSW
    
```

2.1 Abstract Message Passing

For sending and receiving request and answer messages we abstract from a concrete message passing system by using abstract communication interfaces (predicates) for mail boxes of incoming and outgoing messages.

- *ReceivedReq* in RECIIVEREQ expresses that an incoming request message has been received from some client (supposed to be encoded into the message).
- *ReceivedAnsw* in RECEIVEANSW expresses that an answer message (to a previously sent supposed to be retrievable request message) has been received.
- An abstract machine SEND is used a) by SENDANSW for sending out answer messages to requests back to the clients where the requests originated, b) by SENDREQ for sending out requests to providers. We assume the addressees to be encoded into messages.

We separate the internal preparation of outgoing messages in PROCESS from their actual sending in SEND by using the following abstract predicates for mail boxes of outgoing mail:

- *SentAnswToMailer* expresses that an outgoing answer message (elaborated from a PROCESS internal representation of an answer) was passed to SEND.
- *SentReqToMailer* expresses that an outgoing request message (corresponding to an internal representation of a request) has been passed to SEND.

2.2 The SEND and RECEIVE Submachines

The interaction between a client and a VIRTUALPROVIDER, which is triggered by the arrival of a client’s request message so that *ReceivedReq(inReqMssg)* becomes true, is characterized by creating a request object (a request ID, say element *r* of

a set $ReqObj$ of currently alive request objects), which is appropriately initialized by recording in an internal representation the relevant data, which are encoded in the received request message. This includes decorating that object by an appropriate *status*, say $status(r) := started$, to signal to (the scheduler for) PROCESS its readiness for being processed.

This requirement for the machine RECEIVERREQ is captured by the following definition, which is parameterized by the incoming request message $inReqMsg$ and by the set $ReqObj$ of current request objects of the VP. For simplicity of exposition we assume a preemptive $ReceivedReq$ predicate.⁴

$$\begin{aligned} \text{RECEIVERREQ}(inReqMsg, ReqObj) = & \text{if } ReceivedReq(inReqMsg) \text{ then} \\ & \text{CREATENEWREQOBJ}(inReqMsg, ReqObj) \\ \text{where } \text{CREATENEWREQOBJ}(m, R) = & \\ & \text{let } r = new(R)^5 \text{ in INITIALIZE}(r, m) \end{aligned}$$

The inverse interaction between a VP and a client, which consists in sending back a message providing an answer to a previous request of the client, is characterized by the underlying request object having reached, through further PROCESSING, a *status* where a call to SENDANSW with corresponding parameter $outAnswMsg$ has been internally prepared by PROCESS — namely by setting the answer-mailbox predicate $SentAnswToMailer$ for this argument to *true*. Thus one can specify SENDANSW, and symmetrically SENDREQ with the request-mailbox predicate $SentReqToMailer$, as follows:

$$\begin{aligned} \text{SENDANSW}(outAnswMsg, SentAnswToMailer) = & \\ \text{if } SentAnswToMailer(outAnswMsg) \text{ then } \text{SEND}(outAnswMsg) \end{aligned}$$

$$\begin{aligned} \text{SENDREQ}(outReqMsg, SentReqToMailer) = & \\ \text{if } SentReqToMailer(outReqMsg) \text{ then } \text{SEND}(outReqMsg) \end{aligned}$$

For the definition of RECEIVEANSW we use as parameter the $AnswerSet$ function which provides for every *requester* r , which may have triggered sending some subrequests to subproviders, the $AnswerSet(r)$, where to insert (the internal representation of) each *answer* contained in the incoming answer message.⁶

$$\begin{aligned} \text{RECEIVEANSW}(inAnswMsg, AnswerSet)^7 = & \\ \text{if } ReceivedAnsw(inAnswMsg) \text{ then} & \\ \text{insert } answer(inAnswMsg) \text{ into } AnswerSet(requester(inAnswMsg)) \end{aligned}$$

⁴ Otherwise a DELETE($inReqMsg$) has to be added, so that the execution of RECEIVERREQ($inReqMsg, ReqObj$) switches $ReceivedReq(inReqMsg)$ to *false*.

⁵ new is assumed to provide at each application a sufficiently fresh element.

⁶ The function $requester(inAnswMsg)$ is defined below to denote the value of $seqSubReq$ in the state when the request message $outReq2Msg(s)$ for the parallel subrequest s was sent out to which the $inAnswMsg$ is received now.

⁷ Without loss of generality we assume this machine to be preemptive (i.e. $ReceivedAnsw(inAnswMsg)$ gets false by firing RECEIVEANSW for $inAnswMsg$).

Behavioural interface types. Through the definitions below, we link calls of `RECEIVEREQ` and `SENDANSW` by the *status* function value for a *currReqObj*. Thus the considered communication interface is of the “provided behavioural interface” type, discussed in [12]: The `RECEIVEREQ` action corresponds to receive an incoming request, through which a new *reqObj* is created, and occurs before the corresponding `SENDANSW` action, which happens after the outgoing answer message in question has been *SentAnswToMailer* when *reqObj* was reaching the *status deliver*. The pair of machines `SENDREQ` and `RECEIVEANSW` in `PROCESS` realizes the symmetric “required behavioural interface” communication interface type, where the `SEND` actions correspond to outgoing requests and thus occur before the corresponding `RECEIVEANSW` actions of the incoming answers to those requests.

2.3 Refinement by a “State” Component

It is easy to extend `RECEIVEREQ` to equip `VIRTUALPROVIDERS` with some state for recording information on previously received requests, to be recognized when for such a request at a later stage some additional service is requested. The changes on the side of `PROCESS` defined below concern the inner structure of that machine and its refined notion of state and state actions. We concentrate our attention here on the refinement of the `RECEIVEREQ` machine. This refinement is a simple case of the general ASM refinement concept in [11].

The first addition needed for `RECEIVEREQ` is a predicate *NewRequest* to check, when an *inReqMsg* is received, whether that message contains a new request, or whether it is about an already previously received request. In the first case, `CREATENEWREQOBJ` as defined above is called. In the second case, instead of creating a new request object, the already previously created request object corresponding to the incoming request message has to be retrieved, using some function *prevReqObj(inReqMsg)*, to `REFRESHREQOBJ` by the additional information on the newly arriving further service request. In particular, a decision has to be taken upon how to update the *status(prevReqObj(inReqMsg))*, which depends on how one wants the processing *status* of the original request to be influenced by the additional request or information presented through *inReqMsg*. Since we want to keep the scheme general, we assume that an external scheduling function *refreshStatus* is used in an update *status(r) := refreshStatus(r, inReqMsg)*.⁸ This leads to the following refinement of `RECEIVEREQ` (we skip the parameters *ReqObj*, *prevReqObj*):

```

RECEIVEREQ(inReqMsg) = if ReceivedReq(inReqMsg) then
  if NewRequest(inReqMsg) then
    CREATENEWREQOBJ(inReqMsg, ReqObj)
  else let r = prevReqObj(inReqMsg) in REFRESHREQOBJ(r, inReqMsg)

```

⁸ What if *status(prevReqObj(inReqMsg))* is simultaneously updated by the refined `RECEIVEREQ` and by `PROCESS` as defined below? In case of a conflicting update attempt the ASM framework stops the computation; At runtime such an inconsistency is notified by ASM execution engines. Implementations will have to solve this problem in the scheduler of VP.

3 The PROCESSING Submachine of VIRTUALPROVIDERS

In this section we define the signature and the transition rules of the ASM PROCESS for the processing kernel of a VIRTUALPROVIDER. The definition provides a schema, which is to be instantiated for each particular PROCESSING kernel of a concrete VP by giving concrete definitions for the abstract functions and machines we are going to introduce. For an example see Sect. 5.

Since we want to abstract from the scheduler, which calls PROCESS for particular current request objects $currReqObj$, we describe the machine as parametrized by a global instance variable $currReqObj \in ReqObj$. The definition is given in Fig. 2 in terms of control state ASMs, using the standard graphical representation of finite automata or flowcharts as graphs with circles (for the internal states, here to be interpreted as current value of $status(currReqObj)$), rhombuses (for test predicates) and rectangles (for actions).

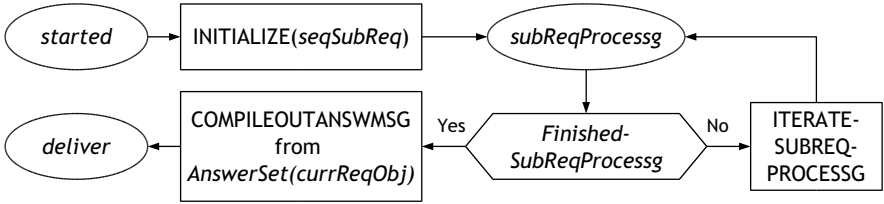


Fig. 2. PROCESSING($currReqObj$)

Figure 2 expresses that each PROCESSING call for a *started* request object $currReqObj$ triggers to INITIALIZE an iterative sequential subrequest processing, namely of the immediate subrequests of this $currReqObj$, in the order defined by an iterator over a set $SeqSubReq(currReqObj)$. This reflects the first part of the hierarchical VP request processing view, namely that each incoming (top level) request object $currReqObj$ triggers the sequential elaboration of a finite number of immediate subrequests, members of a set $SeqSubReq(currReqObj)$, called sequential subrequests. As explained below, each sequential subrequest may trigger a finite number of further subsubrequests, which are sent to external providers where they are elaborated independently of each other, so that we call them parallel subrequests of the sequential subrequest.

PROCESS uses for the elaboration of the sequential subrequests of $currReqObj$ a submachine ITERATESUBREQPROCESSG specified below. Once PROCESS has *FinishedSubReqProcessg*, it compiles from $currReqObj$ (which allows to access $AnswerSet(currReqObj)$) an answer, say $outAnswer(currReqObj)$, and transforms the internal answer information a into an element of $OutAnswMssg$ using an abstract function $outAnsw2Msg(a)$. We guard this answer compilation by a check whether $AnswToBeSent$ for the $currReqObj$ evaluates to true.

For the sake of illustration we also provide here the textual definition of the machine defined in Fig. 2. For this purpose we use a function $initStatus$ to yield

for a control state ASM its initial control status, which is hidden in the graphical representation. The function $seqSubReq(currReqObj)$ denotes the current item of the iterator submachine ITERATESUBREQPROCESSG defined below.

```

PROCESS(currReqObj) =
  if status(currReqObj) = started then
    INITIALIZE(seqSubReq(currReqObj))
    status(currReqObj) := subReqProcessg
  if status(currReqObj) = subReqProcessg then
    if FinishedSubReqProcessg then
      COMPILEOUTANSWMSG from currReqObj
      status(currReqObj) := deliver
    else
      StartNextRound(ITERATESUBREQPROCESSG)
where
  COMPILEOUTANSWMSG from o = if AnswToBeSent(o) then
    SentAnswToMailer(outAnsw2Msg(outAnswer(o))) := true
  StartNextRound(M) = (status(currReqObj) := initState(M))

```

The submachine to ITERATESUBREQPROCESSG is an iterator machine defined in Fig. 3. For every current item $seqSubReq$, it starts to FEEDSENDREQ with a request message to be sent out for every immediate subsubrequest s of the current $seqSubReq$, namely by setting $SentReqToMailer(outReq2Msg(s))$ to true. Here $outReq2Msg(s)$ transforms the outgoing request into the format for an outgoing request message, which has to be an element of $OutReqMssg$. Since those immediate subsubrequests, elements of a set $ParSubReq(seqSubReq)$, are assumed to be processable by other providers independently of each other, FEEDSENDREQ elaborates simultaneously for each s an $outReqMsg(s)$.

Simultaneously ITERATESUBREQPROCESSG also INITIALIZES the to be computed $AnswerSet(seqSubReq)$ before assuming *status* value *waitingForAnswers*, where it remains until *AllAnswersReceived*. When *AllAnswersReceived*, the machine ITERATESUBREQPROCESSG will PROCEEDTONEXTSUBREQ.

As long as during *waitingForAnswers*, *AllAnswersReceived* is not yet true, RECEIVEANSW inserts for every $ReceivedAnsw(inAnswMsg)$ the retrieved in-

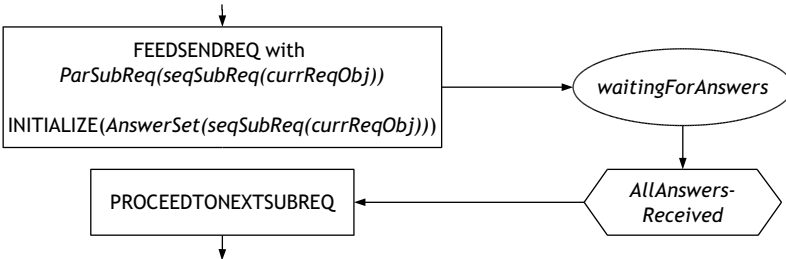


Fig. 3. ITERATESUBREQPROCESSG

ternal $answer(inAnswMsg)$ representation into $AnswerSet(seqSubReq)$ of the currently processed sequential subrequest $seqSubReq$, which is supposed to be retrievable as $requester$ of the incoming answer message.

```

ITERATESUBREQPROCESSG =
  if  $status(currReqObj) = initState(ITERATESUBREQPROCESSG)$  then
    FEEDSENDREQ with  $ParSubReq(seqSubReq(currReqObj))$ 
    INITIALIZE( $AnswerSet(seqSubReq(currReqObj))$ )
     $status(currReqObj) := waitingForAnswers$ 
  if  $status(currReqObj) = waitingForAnswers$  then
    if  $AllAnswersReceived$  then
      PROCEEDTONEXTSUBREQ
       $status(currReqObj) := subReqProcessg$ 
  where FEEDSENDREQ with  $ParSubReq(seqSubReq) =$ 
    forall  $s \in ParSubReq(seqSubReq)$ 
       $SentReqToMailer(outReq2Msg(s)) := true$ 

```

For the sake of completeness we now define the remaining macros used in Fig. 3, though their intended meaning should be clear from the chosen names. The **Iterator Pattern** on $SeqSubReq$ is defined by the following items:

- $seqSubReq$, denoting the current item in the underlying set $SeqSubReq \cup \{ Done(SeqSubReq(currReqObj)) \}$,
- The functions $FstSubReq$ and $NxtSubReq$ operating on the set $SeqSubReq$ and $NxtSubReq$ also on $AnswerSet(currReqObj)$,
- The stop element $Done(SeqSubReq(currReqObj))$, constrained by not being an element of any set $SeqSubReq$.

```

INITIALIZE( $seqSubReq$ ) = let  $r = FstSubReq(SeqSubReq(currReqObj))$  in
   $seqSubReq := r$ 
   $ParSubReq(r) := FstParReq(r, currReqObj)$ 

```

```

FinishedSubReqProcessg =
   $seqSubReq(currReqObj) = Done(SeqSubReq(currReqObj))$ 

```

```

PROCEEDTONEXTSUBREQ =
  let  $o = currReqObj$ 
   $s = NxtSubReq(SeqSubReq(o), seqSubReq(o), AnswerSet(o))$  in
     $seqSubReq(o) := s$ 
     $ParSubReq(s) := NxtParReq(s, o, AnswerSet(o))$ 

```

This iterator pattern foresees that $NxtSubReq$ and $NxtParReq$ may be determined in terms of the answers accumulated so far for the overall request object, i. e. taking into account the answers obtained for preceding subrequests.

```

INITIALIZE( $AnswerSet(seqSubReq)$ ) = ( $AnswerSet(seqSubReq) := \emptyset$ )

AllAnswersReceived = let  $seqSubReq = seqSubReq(currReqObj)$  in
  for each  $req \in ToBeAnswered(ParSubReq(seqSubReq))$ 
    there is some  $answ \in AnswerSet(seqSubReq)$ 

```

The definition foresees the possibility that some of the parallel subrequest messages, which are sent out to providers, may not necessitate an answer for the VP: A function *ToBeAnswered* filters them out from the condition *waitingForAnswers* to leave the current iteration round.

The answer set of any main request object can be defined as a derived function of the answer sets of its sequential subrequests:

$$AnswerSet(reqObj) = Combine(\{AnswerSet(s) \mid s \in SeqSubReq(reqObj)\})$$

4 Mediator Composition and Equivalence Notions

We show how to combine VIRTUALPROVIDERS and how to define their service behaviour, which allows one to define rigorous equivalence notions for VPs one can use a) to formulate algorithms for the discovery and runtime selection of providers suitable to satisfy given requests, and b) to prove VP runtime properties of interest.

4.1 Composing VIRTUALPROVIDERS

Instances VP_1, \dots, VP_n of VIRTUALPROVIDER can be configured into a sequence with a first VIRTUALPROVIDER VP_1 involving a subprovider VP_2 , which involves a subprovider VP_3 , etc. For such a composition it suffices to connect the communication interfaces in the appropriate way (see Fig. 1):

- SENDREQ of VP_i with the RECEIVERREQ of VP_{i+1} , which implies that in the message passing environment, the types of the sets *OutReqMssg* of VP_i and *InReqMssg* of VP_{i+1} match (via some data mediation).
- SENDANSW of VP_{i+1} with the RECEIVEANSW of VP_i , which implies that in the message passing environment, the types of the sets *OutAnswMssg* of VP_{i+1} and *InAnswMssg* of VP_i match (via some data mediation).

Such a sequential composition allows one to configure mediator schemes (see Fig. 4) where each element seq_1 of a sequential subrequest set $SeqSubReq_1$ of an initial request can trigger a set $ParSubReq(seq_1)$ of parallel subrequests par_1 , each of which can trigger a set $SeqSubReq_2$ of further sequential subrequests seq_2 of par_1 , each of which again can trigger a set $ParSubReq(seq_2)$ of further parallel subrequests, etc. This provides the possibility of unfolding arbitrary alternating seq/par trees. More complex composition schemes can be defined similarly.

4.2 Defining Equivalence Notions for VIRTUALPROVIDERS

To be able to speak about the relation between incoming requests and outgoing answers, one has to relate the elements of the corresponding sets *InReqMssg* and *OutAnswMssg* on the provider side (the left hand side in Fig. 1) or *OutReqMssg* and *InAnswMssg* on the requester side of a VIRTUALPROVIDER (the right hand side in Fig. 1). In the first case this comes up to unfold the function *originator*, which for an *outAnswMssg* yields the *inReqMssg* to which

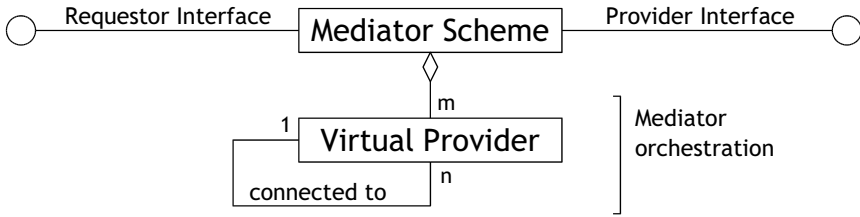


Fig. 4. Mediator Scheme

outAnswMsg represents the answer. In fact this information is retrievable by `COMPILEOUTANSWMSG` from the *currReqObj*, if it was recorded there by `CREATENEWREQOBJ(inReqMsg, ReqObj)` as part of `INITIALIZE`.

One can then define the *ServiceBehaviour*(*VP*) of a Virtual Provider *VP* = `VIRTUALPROVIDER` as (based upon) the correspondence between any *inReqMsg* and the *outAnswMsg* related to it by the *originator* function:

$$\text{originator}(\text{outAnswMsg}) = \text{inReqMsg}$$

Two `VIRTUALPROVIDERS` *VP*, *VP'* can be considered equivalent if an equivalence relation $\text{ServiceBehaviour}(VP) \equiv \text{ServiceBehaviour}(VP')$ holds between their service behaviours. To concretely define such an equivalence involves detailing of the meaning of service ‘requests’ and provided ‘answers’, which comes up to providing further detail of the abstract VP model in such a way that the intended ‘service’ features and how they are ‘provided’ by VP become visible in concrete locations.

On the basis of such definitions one can then formally define different VPs to be alternatives for a Strategy pattern [13, p. 315] for providing requested services. For the run-time selection of mediators, any suitable provider interface can be viewed as one of the implementations (“mediator orchestration”) of a Strategy pattern assigned to a requester interface. This provides the basis for investigating questions like: How can one assure that a provider interface matches the Strategy pattern of the requester? How and starting from which information can one build automatically the Strategy pattern implementations? Similar definitions can be used to characterize mismatches between requester and provider interfaces as well as *ServiceBehaviours*.

5 Illustration: Virtual Internet Service Provider

One of the use cases in the DIP project (see <http://dip.semanticweb.org>) deals with a *Virtual Internet Service Provider* (VISP). A VISP resells products that are bundled from offerings of different providers. A typical example for such a product bundle is an *Internet presence* including a personal Web server and a personal e-mail address, both bound to a dedicated, user-specific domain name, e.g. `michael-altenhofen.de`. Such an Internet presence would require this domain name to be registered (at a central registry, e.g. DENIC).

Ideally, the VISP wants to handle domain name registrations in a unified manner using a fixed interface. We assume now that this interface contains only one request message *RegisterDomain*, requiring four input parameters:

- **DomainName**, the name of the new domain that should be registered
- **DomainHolderName**, the name of the domain owner
- **AdministrativeContactName** the name of the domain administrator
- **TechnicalContactName**, the name of the technical contact

On successful registration, the answer will contain four so-called RIPE-Handles,⁹ uniquely identifying in the RIPE database the four names provided in the request message. We skip the obvious instantiation of VIRTUALPROVIDER to formalize this VISP.

5.1 A Possible VIRTUALPROVIDER Refinement for *RegisterDomain*

We now consider the case that the VISP is extending it's business into a new country whose domain name registry authority implements a different interface for registering new domain names, say consisting of four request messages:

- *RegisterDH* (DomainHolderName),
- *RegisterAC* (AdministrativeContactName),
- *RegisterTC* (TechnicalContactName),
- *RegisterDN* (DomainName, DO-RIPE-Handle, AC-RIPE-Handle, TC-RIPE-Handle).

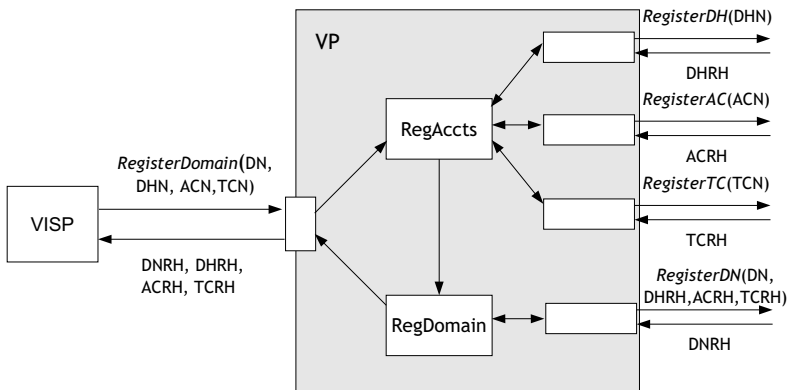


Fig. 5. VIRTUALPROVIDER Instance

A VP instance for that scenario is depicted in Fig. 5.¹⁰ Within this VP, the incoming request *RegisterDomain* is split into a sequence of two subrequests. The first subrequest is further divided into three parallel subrequests, each registering

⁹ RIPE stands for “Réseaux IP Européens”, see <http://ripe.net>.

¹⁰ We use mnemonic abbreviations for the request message and parameter names.

one of the contacts. Once all answers for these parallel subrequests have been received, the second sequential subrequest can be performed, whose outgoing request message is constructed from the answers of the previous subrequest and the `DomainName` parameter from the incoming request.

Using the notational convention of appending *Obj* when referring to the internal representations of the different requests, we formalize this VP instance by the following stipulations. We start with refining the INITIALIZE ASM:

```

INITIALIZE(RegisterDomainObj, RegisterDomain(DN, DHN, ACN, TCN)) =
  params(RegisterDomainObj) := {DN, DHN, ACN, TCN}
  SeqSubReq(RegisterDomainObj) := {RegAccnts, RegDomain}
  FstSubReq({RegAccnts, RegDomain}) := RegAccnts
  NrtSubReq({RegAccnts, RegDomain}, RegAccnts, _) := RegDomain
  NrtSubReq({RegAccnts, RegDomain}, RegDomain, _) := nil
  FstParReq(RegAccnts, RegisterDomainObj) :=
    {RegisterDH(DHN), RegisterAC(ACN),
     RegisterTC(TCN)}
  NrtParReq(RegDomain, RegisterDomainObj, AS) :=
    {RegisterDN(DN, handle(DHRHObj),
     handle(ACRHObj), handle(TCRHObj))}
  AnsToBeSent(RegisterDomainObj) := true
  ToBeAnswered({RegisterDH, RegisterAC, RegisterTC}) :=
    {RegisterDH, RegisterAC, RegisterTC}
  ToBeAnswered({RegisterDN}) := {RegisterDN}
  status(RegisterDomainObj) := started
    
```

where

$$\begin{aligned}
 AS &= \{DHRHObj, ACRHObj, TCRHObj\} \\
 handle(X) &= \begin{cases} DHRH & \text{if } X = DHRHObj \\ DNRH & \text{if } X = DNRHObj \\ ACRH & \text{if } X = ACRHObj \\ TCRH & \text{if } X = TCRHObj \end{cases}
 \end{aligned}$$

The derived function *Combine* computes the union of the two answer sets:

$$\begin{aligned}
 Combine(*RegisterDomainObj*) &= \\
 AnswerSet(*RegAccnts*) \cup AnswerSet(*RegDomain*)
 \end{aligned}$$

Function *answer* maps an incoming message to its internal representation:

$$answer(inAnswMsg) = \begin{cases} DHRHObj & \text{if } inAnswMsg = DHRH \\ DNRHObj & \text{if } inAnswMsg = DNRH \\ ACRHObj & \text{if } inAnswMsg = ACRH \\ TCRHObj & \text{if } inAnswMsg = TCRH \end{cases}$$

The abstract function *Formatted* is used to transform the parameters into the format expected by the requester, in our case the VISP:

$$\begin{aligned}
 outAnsw2Msg(\{DHRHObj, DNRHObj, ACRHObj, TCRHObj\}) &= \\
 Formatted(\{DNRH, DHRH, ACRH, TCRH\})
 \end{aligned}$$

In [14] we give five other simple examples for refinements of VP to capture the execution semantics of some workflow patterns discussed in [15].

6 Conclusions and Future Work

This paper provides a formal, high-level model of process mediation. By the use of ASMs as a modeling paradigm, the presented Abstract State Machine builds a base for “*communicating and documenting design ideas*” and supports “*an accurate and checkable overall understanding*” of the controversially discussed topic of process mediation. Furthermore, the ASM method allows to “*isolate the hard part of a system*” [10, p.14-15] and thus to concentrate on the essential parts for refinement.

Process mediation can be seen as one part of the whole Semantic Web services (SWS) usage process [6]. This area of current research also is under strong motion and encounters heterogeneous usage of terms. Different approaches are published as frameworks each presenting a consistent view on “their” SWS usage process (cmp. [5,6,7]). However, different frameworks are not necessarily using terms in the same way as competing frameworks do. ASMs could help providing a means of *explicit, exact and formal specification* and delimitation of terms used in different frameworks, and combine them towards a consistent view of the general SWS usage process.

Specifying a fixed, abstract SWS usage framework, moreover yields the possibility of bridging controversial approaches like dynamic composition vs. static composition through explicitly showing their differences by their individual refinements of the abstract framework. The same is true for the presented VP. As shown in Sect. 5, different specializations constrain the VP’s behaviour to specific patterns. So far, only very narrow refinements have been presented. We could also imagine more generous specializations framing the later refinement possibilities, e. g. especially for specific types of interaction patterns.

Another direction of research concerns replacing the simple communication patterns used by VP by more complex ones. RECEIVERREQ and SENDANSW are identified in [16] as basic bilateral service interaction patterns, namely as mono-agent ASM modules RECEIVE and SEND; The FEEDSENDREQ sub-machine together with SENDREQ in PROCESS realize an instance of the basic multilateral mono-agent service interaction pattern called ONETOMANYSEND in [16], whereas the execution of RECEIVEANSW in ITERATESUBREQPROCESSG until *AllAnswersReceived* is an instance of the basic multilateral mono-agent ONEFROMMANYRECEIVE pattern from [16]. One can refine VP to concrete business process applications by enriching the communication flow structure built from basic service interaction patterns as analysed in [16].

References

1. Stumptner, M.: Configuring web services. In: Proceedings of the Configuration Workshop at the 16th European Conference on Artificial Intelligence (ECAI). (2004) 10-1/10-6

2. Pistore, M., Barbon, F., Bertoli, P., Shaparau, D., Traverso, P.: Planning and monitoring web service composition. In: AIMS. (2004) 106–115
3. Lee, Y., Patel, C., Chun, S.A., Geller, J.: Compositional knowledge management for medical services on semantic web. In: WWW (Alternate Track Papers & Posters). (2004) 498–499
4. Wiederhold, G.: Mediators in the architecture of future information systems. IEEE Computer **25** (1992) 38–49
5. Bornhövd, C., Buchmann, A.: Semantically meaningful data exchange in loosely coupled environments. In: Proceedings of the International Conference on Information Systems Analysis and Synthesis (ISAS). (2000)
6. Fensel, D., Bussler, C.: The web service modeling framework wsmf. Electronic Commerce Research and Applications **1** (2002) 113–137
7. Pires, P.F., Benevides, M.R.F., Mattoso, M.: Building reliable web services compositions. In: Web, Web-Services, and Database Systems. (2002) 59–72
8. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases **14** (2003) 5–51
9. Barros, A., Dumas, M., ter Hofstede, A.: Service interaction patterns: Towards a reference framework for service-based business process interconnection. Technical report, Faculty of IT, Queensland University of Technology (2005)
10. Börger, E., Stärk, R.F.: Abstract State Machines. A Method for High-Level System Design and Analysis. Springer (2003)
11. Börger, E.: The ASM refinement method. Formal Aspects of Computing **15** (2003) 237–257
12. Barros, A., Dumas, M., Oaks, P.: A critical overview of the web services choreography description language (WS-CDL). White paper (2005)
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
14. Altenhofen, M., Börger, E., Lemcke, J.: An execution semantics for mediation patterns. In: Proc. of 2nd WSMO Implementation Workshop WIW'2005, Innsbruck, Austria, CEUR Workshop Proceedings (2005) ISSN 1613-0073, online CEUR-WS.org/Vol-134/lemcke-wiw05.pdf.
15. Cimpian, E., Mocan, A.: D13.7 v0.1 Process mediation in WSMX – WSMX working draft (2005) <http://www.wsmo.org/TR/d13/d13.7/v0.1/>.
16. Barros, A., Börger, E.: A compositional framework for service interaction patterns and communication flows. In: Proc. 7th International Conference on Formal Engineering Methods (ICFEM). LNCS, Springer (2005)

WSMX Process Mediation Based on Choreographies

Emilia Cimpian and Adrian Mocan

DERI, National University of Ireland, Galway, Ireland
emilia.cimpian@deri.org, adrian.mocan@deri.org

Abstract. One of the most difficult obstacles Web Services have to overcome in the attempt to exploit the true potential of the World Wide Web is heterogeneity. Caused by the nature of the Web itself, heterogeneity problems occur both at data level as well as at behavioral level of business logics, message exchange protocol and Web Service invocation.

Process mediation is one of the crucial points on the road towards establishing new, ad-hoc cooperation on the web between various business partners. If semantic enhanced data enables dynamic solutions for coping with data heterogeneity, semantically enhanced Web Services can do the same for behavioral heterogeneity. Based on Web Service Modeling Ontology (WSMO) specifications that offers support in semantically describing Web Services, we propose a solution that acts on these semantic descriptions and offers the means for defining of what we call a Process Mediator. Such a mediator acts on the public processes (represented as WSMO choreographies) of the parties involved in a communication and adjust the bi-directional flow of messages to suit the requested/expected behavior of each party.

1 Introduction

The advantages offered by the huge amount of information available and by the higher and higher number of services deployed on the Web are overcome by the inherent heterogeneity issues existing between all these resources. The information is represented using different languages and different conceptualizations of the same domain; similarly Web Services describe their functionalities in different ways and expect the clients to align with various interaction patterns in order to consume their functionalities.

Numerous approaches are proposing various solutions to cope with data heterogeneity by adding semantic meaning to data, and make it machine understandable. Even if this area is well explored and many semi-automatic solutions are available for this problem there is one more gap to fill: how the semantic enriched data is interchanged by machines. That is, even if the machine can understand the data they receive, they have to understand the communication process it is part of, as well. As a consequence, a coherent mode of describing

the expected interaction patterns is necessary, together with means of mediating between heterogeneous patterns.

The Web Service Modeling Ontology (WSMO) working group started an initiative for standardizing various aspects related to Semantic Web Services. The objective of WSMO and its surrounding efforts is to define a coherent technology for Semantic Web Services by providing means for (semi-)automatic discovery, composition and execution of Web Services which are based on logical inference mechanisms. Web Service Execution Environment (WSMX) [3] is a reference implementation for WSMO, a proof of concept for this specification, aiming to provide reference implementations for the main tasks related to Web Services as envisioned by WSMO.

In this context we propose a solution able to cope with the differences in the way a requester wants to consume the functionality of a Web Service and the way this functionality is made available by the Web Service to the requester. We use WSMO choreography to describe the expected/requested behavior of the two parties, which is in fact a formalization of their public business processes. Using these descriptions and the services of a data mediator (to solve data heterogeneity problems) we introduce the process mediator, a system able to adjust the two parties' behavior and to enable their communication.

In this paper we will present WSMX approach for process mediation, and the WSMX Process Mediator component. Section 2 presents the motivation for developing such a mediator and Section 3 briefly presents WSMO and WSMX. The WSMX Process Mediator is presented in Section 4, followed by an example (Section 5). The final parts of this document present some related efforts in this area (Section 6) and conclusions (Section 7).

2 Motivation

2.1 Overview

The simple scenario of invoking a (Semantic) Web Service may become extremely complicated in a heterogeneous environment such as the existing web.

Usually the client has its own communication pattern (expressing how it wants to communicate with a service) that in general is different from the one used by the corresponding Web Service (which expresses how the service wants to be invoked). As a consequence the two parties will not be able to directly communicate, even if they can understand the same data formats. In order to communicate they must be able either to redefine their communication patterns (at least one of them has to) or to use an external mediation system as part of the process. The first solution is generally a very expensive one implying changes in the entities' business logic, and it is not suitable in a dynamic environment since every participant would have to readjust its pattern (through re-programming) each time it gets involved in a new partnership. As a consequence, the role of the mediator system will be to compensate the communication patterns in order to obtain equivalent processes.

2.2 Problem Definition

A set of assumptions are made regarding the two parties to mediate between:

- Each of the parties have to make public the expected/requested way of inter-operating with its partner. Conform to WSMO these represent choreography descriptions and they are included in the goal's interface and in the Web Service's description.
- The involved parties have to refer from their choreographies the ontologies they used to describe their domain. Furthermore these ontologies have to be available and the heterogeneity problems between them resolved by a Data Mediator. This implies that a failure of the Data Mediator in solving the data heterogeneity problems has as a direct effect the failure of the Process Mediator.
- The messages exchanged between the two parties have to contain data represented in terms of the used ontologies, that is, ontology instances.

The scope of the Process Mediator is to make this conversation possible by the use of different technics as message blocking, message splitting or aggregation, acknowledgements generation and so on. The process mediator is part of WSMX and it can make use of all the functionalities provided by WSMX regarding message receiving and sending, keeping track of the ongoing conversation, access various Data Mediators, resources and so on.

3 WSMO and WSMX

The Web Service Modeling Ontology (WSMO) is a formal ontology for describing various aspects related to Semantic Web Services.

WSMO defines four main modeling elements for describing several aspects of Semantic Web Services: *ontologies*, *Web Services*, *goals* and *mediators*. In what follows, we will describe all these elements, insisting on their importance in reaching a truly Semantic Web Service technology.

As defined in [6], *ontologies* are formal explicit specifications of shared conceptualizations. In WSMO they represent key elements, having a twofold purpose: firstly they define the information's formal semantics and secondly, they allow to link machine and human terminologies. The WSMO ontologies give meaning to the other elements (Web Services, goals and mediators), and provide common semantics, understandable by all the involved entities (both humans and machines).

In WSMO, requestors of a service express their objectives as *goals*, which are high level descriptions of concrete tasks. Every requestor expresses its goal in terms of its own ontology, which, on one hand provides the means for a human user to understand the goal, and on the other hand, allows a machine to interpret it as part of the requestor's ontology. Another advantage of using the goals is that the requestor only has to provide a declarative specification of what it wants, and does not need to have a fixed relation with the Web Service or

to browse through an UDDI registry for finding Web Services that provide the appropriate capability.

In order for this goal to be accomplished, the requestor (by means of its information system) has to find an appropriate *Web Service* which may fulfill the required task. Similar to the way the requestor declares its goal, every Web Service has to declare its capability (that is, what it is able to accomplish) in terms of its own ontology. If the requestor of the service and the Web Service that offers it use the same ontology the matching between the goal and the capability can be directly established. Unfortunately, in most of the cases they use different ontologies, and the equivalence between the goal and the capability can be determined only if a third party is consulted for determining the similarities between the two ontologies. Another problem that may appear is the impossibility of the requester and of the provider of the service to communicate with each other, the reason for this being the heterogeneity of their communication protocols. For these reasons, WSMO introduces the fourth key modeling element: the *mediators*, which have the task of overcoming the heterogeneity problems, both at data level and at communication level.

The Web Service Execution Environment (WSMX) is the reference implementation for WSMO, designed to allow dynamic discovery, invocation and composition of Web Services. WSMX offers complete support for interacting with Semantic Web Services. In addition, WSMX supports the interaction with non-WSMO, but classical Web Services ensuring that a seamless interaction with existing Web Services is possible.

By using WSMX, if two partners want to interoperate with each other, they only have to expose their own functionality and to consume the functionality offered by their partners. Additionally, there can be the case that one entity wants to get involved in ad-hoc business processes without knowing its partner beforehand. In order to accommodate this situation, WSMX offers mechanisms and strategies for dynamic discovery of those entities that expose the desired capability. Furthermore, it extends this search in order to find out if multiple partners are able to fulfill the requester goal, by composing the offered sub-functionalities.

In any of the functionalities offered by WSMX (discovery, invocation and composition) mediation can be needed at both data [8] and process level (behavioral level) [2]. In the following chapter we will describe the WSMX process mediation approach.

4 Process Mediation Based on WSMO Choreographies

For addressing the problem of process mediation we have to define first what a process means, and how we represent a WSMX process.

We adopt the standard definition of a process: collection of activities designed to produce a specific output for a particular customer, based on a specific inputs [1], an activity being a function, or a task that occurs over time and has recognizable results.

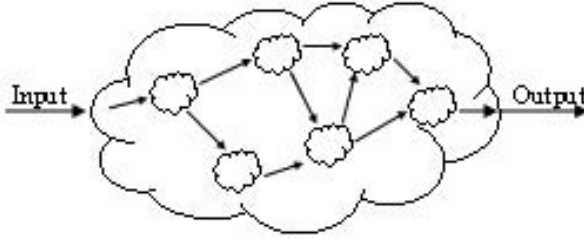


Fig. 1. Process consisting of multiple processes

Depending on the considered level of granularity, each process can be seen as being composed of different, multiple processes. The smallest process possible consists of only one activity. Figure 1 illustrates a process obtain by combining multiple processes. The output of one process (or more processes) is considered the input of one or many other processes.

One can distinguish between two type of processes: *private processes*, which are carried out internally by an organization, and usually are not visible to any other entity, and *public processes*, which are defining the behavior of the organization in collaboration with other entities [5]. From the process mediation point of view we are interested only in the public processes, the private process not being visible to the exterior, can not be the object of WSMX mediation.

In what follows, we will define the WSMX public process representation, the problems this mediator intends to solve, the Process Mediator's interactions with other WSMX components, and we will describe step by step the process mediation algorithm.

4.1 Process Representation

WSMX process representation is similar with the WSMO choreography [9] definition. This dependency is a straight forward one considering that WSMX Process Mediator is dealing with the communication patterns heterogeneity, and that the WSMO choreography describes the behavior of the service from a user point of view (that is, how the user should interact with the Web Service in order to consume its functionality). In terms of WSMX, every choreography represents a public process, which means that WSMO choreography is a subclass of WSMX process.

In order for this paper to be self-contained we describe in the next paragraphs the main features of WSMX processes, i.e. the main features of WSMO choreography, as described in [9].

The representation of a WSMX business process is based on the Abstract State Machine [7] methodology, and it inherits the core principles of ASMs:

- it is state-based;
- it represents a state by an algebra;
- it models state changes by guarded transition rules that change the values of functions and relations defined by the signature of the algebra.

A WSMX process consists of *states* and *guarded transitions* [9]. A state is described by a WSMO ontology, and is obtained from the ontology used by the owner of the process by:

- subclassifying all the concepts that the owner needs to make public in order to enable the communication, and
- adding an additional attribute **mode**, which shows who has the right of modifying the instances of the concept. This attribute can take the values:
 - static** - the extensions of the concept can not be changed;
 - controlled** - the extensions of the concept can only be changed by its owner;
 - in** - the extensions of the concept can only be changed by the environment; the environment should have **write** access over them;
 - shared** - the extensions of the concept can be changed by its owner and by the environment; the environment should have **read/write** access over them;
 - out** - the extensions of the concept can only be changed by its owner; the environment should have **read** access over them.

The guarded transitions (transition rule) are used to express changes of states by means of rules, expressible in the following form:

if Cond **then** Updates

Cond is an arbitrary Web Service Modeling Language (WSML) [4] axiom, formulated in the given signature of the state.

The **Updates** consist of arbitrary WSMO Ontology instances.

In the Semantic Web Services context the level of granularity for representing a certain process is strictly up to the owner of that process. Each action can be represented as a transition, which is the most detailed level, or more actions can be modelled using only one transition.

4.2 Addressed Problems

Usually a business communication consists of more than one exchange message, and as a consequence, finding the equivalences between the message exchange patterns of the two (or more) parties is not at all a trivial task. Intuitively, the easiest way of doing this is to first determine the mismatches, and then search for a solution of eliminating them. [5] identifies three possible cases that may appear during the message exchange:

Precise match. The two partners have exactly the same pattern in realizing the business process, which means that each of them sends the messages in exactly the order the other one requests them. In this ideal case the communication can take place without using a Process Mediator.

Resolvable message mismatch. This case appears when the two partners use different exchange partners, and several transformations have to be performed in

order to resolve the mismatches. For example when one partner sends multiple instances in a single message, but the other one expects them separately, the mediator can break the initial message, and send the instances one by one.

Unresolvable message mismatch. In this case, one of the partners expects a message that the other one do not intend to send. Unless the mediator can provide this message, the communication reaches a dead-end (one of the partners is waiting indefinitely).

In order to communicate two partners have to either define equivalent processes, or to use an external mediation system as part of the communication process. The mediator's role will be to transform the clients messages and/or Web Services messages, in order to obtain a sequence of equivalent processes.

As illustrated above, not all the communication mismatches can be solved by using a mediator, but only some of them. A list containing the initial set of resolvable mismatches that our mediator intends to address is provided below.

Stopping an unexpected message (Figure 2. a)): in case one of the partners sends a message that the other one does not want to receive, the mediator should just retain and store it. This message can be send later, if needed, or it will just be deleted after the communication ends.

Inverting the order of messages (Figure 2. b)): in case one of the partners sends the messages in a different order than the one the other partner wants to receive them. The messages that are not yet expected will be stored and sent when needed.

Splitting a message (Figure 2. c)): in case one of the partners sends in a single message multiple information that the other one expects to receive in different messages.

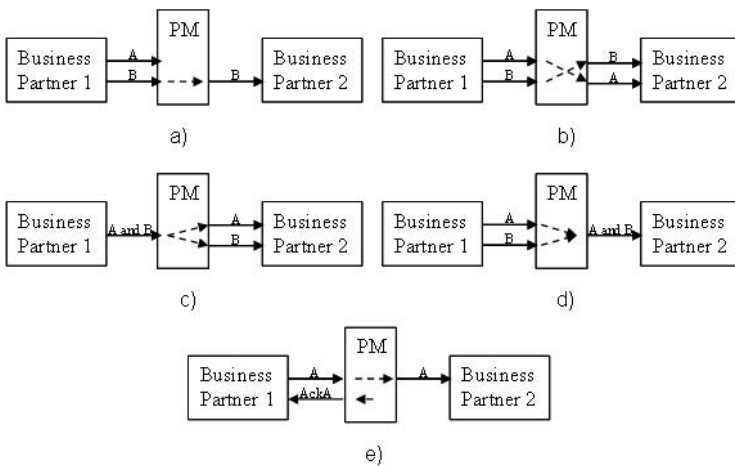


Fig. 2. Addresses Mismatches

Combining messages (Figure 2. d)): in case one of the partners expects a single message, containing information sent by the other one in multiple messages.

Sending a dummy acknowledgement (Figure 2. e)): in case one of the partners expects an acknowledgement for a certain message, and the other partner does not intend to send it, even if it receives the message.

4.3 WSMX Process Mediator

WSMX process mediation is concerned with determining how two public processes can be combined in order to provide certain functionality. In other words, how two business partners can communicate, considering their public processes.

When WSMX receives a message, either from the requestor of the service or from a Web Service, it has to check if it is the first message in a conversation. If it is the first, WSMX creates copies (instances) of both the sender and the targeted business partner choreographies, and stores these instances in a repository, together with a uniquely identifier of the conversation. If it is not the first message of a conversation, WSMX has to determine the conversation id. These computations performed on the message are done by two WSMX components, Communication Manager and Choreography Engine. Their descriptions are not included in this article, since they are not relevant from the process mediation's point of view; more information about various WSMX components can be found in [10].

After the id of the conversation is obtained, the Process Mediator (PM) receives it, together with the message; the message consists of instances of concepts from the sender's ontology. Based on the id, the PM loads the two choreography instances from the WSMX Repository, by invoking the WSMX Resource Manager. All the transformations performed by the PM will be done on this instances. In case different ontologies have been used for modeling the two choreographies, the PM has to invoke an external Data Mediator for transforming the message in terms of the target ontology.

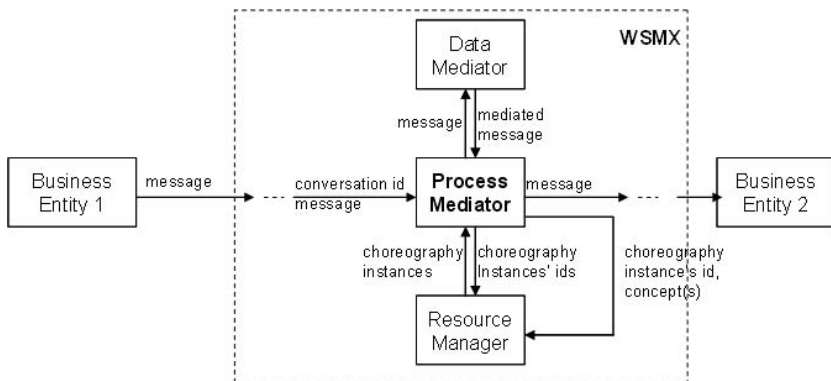


Fig. 3. Process Mediator interactions

After various internal computations (described in the next chapter) the PM determines if based on the incoming message it can generate any message expected by either one of the partners. The generation of any message determines a transformation in the choreography instance of the party that receives that message. Simultaneously with sending the message, the process mediator should update the choreography instances and reevaluate all the rules, until no further updates are possible.

The interactions between the Process Mediator and other WSMX components is represented in Figure 3.

4.4 Execution

The Process Mediator is triggered when it receives a message and a conversation id. The message contains instances of concepts; the conversation id uniquely identifies the instances of the choreographies involved in the communication.

After being invoked, the PM performs the following steps:

1. Loads the two choreography instances from the repository.
2. Adds the instances contained in the message to the corresponding choreography instance (the sender's choreography instance); this step is needed considering that the choreography instances contains the information prior to the transmission of the current message.
3. Mediates the incoming instances in terms of the targeted partner ontology, and checks if the targeted partner is expecting them, at any phase of the communication. This is done by checking the value of the `mode` attribute, for the mediated instances' owner. If this attribute is set to `in` or `shared` for a certain concept, than this concept's instances may be needed at some point in time. The instances expected by the targeted partner are stored in an internal repository.
4. For all the instances from the repository, the PM has to check if they are expected at this phase of the communication, which is done by evaluating the transition rules. The evaluation of a rule will return the first condition that can not be fulfilled, that is, the next expected instance for that rule. This means that an instance is expected if it can trigger an action (not necessary to change a state, but to eliminate one condition for changing a state).

The possibility that various instances from this repository can be combined in order to obtain a single instance, is also considered.

5. Each time the PM determines that one instance is expected, it sends it, deletes it from the repository, updates the targeted partner choreography instance, and restarts the evaluation process (step 4). When a transition rule can be executed, it is marked as such and not re-evaluated at further iterations. The PM only checks if a transition rule can be executed since it can not update any of the two choreography instances without receiving input from one of the communication partner. By evaluating a rule, the PM determines that one of the business partner can execute it, without expecting any other inputs.

This process stops when, after performing these checkings for all the instances from the repository, no new message is generated.

6. For each instance forwarded to the targeted partner, the PM has to check if the sender is expecting an acknowledgement. If the sender expects an acknowledgement, but the targeted partner does not intend to send it, the PM generate a dummy acknowledgement and sends it. Simultaneously, it updates the sender's choreography instance.

7. The PM checks all the sender's rules and marks the ones that can be executed.

8. The PM checks the requestor's rule, to see if all of them are marked; when all are marked, the communication is over and PM deletes all the data regarding this conversation, from both its internal repository and WSMX repository.

This algorithm is implemented by the PM in order to solve the communication heterogeneity problem.

5 Examples

We consider a Virtual Travel Agency (VTA) service, able to provide on-line tickets for certain routes, and a client who wants to invoke this service. For keeping this example as simple as possible, we will present only parts of the two choreographies, but these parts are conclusive enough to illustrate how the Process Mediator works. Additionally, we will consider that some of the concepts have exactly the same semantic for both the service and the client.

5.1 Requestor and Provider's Choreographies

We consider that the two participants have the following concepts in their internal ontologies:

- **station** - the concept of a station, whose instances can be the starting point or the destination of a trip;
- **date** - the instance of this concept represents the date the trip should begun;
- **time** - an instance of this concept represents the departure time;
- **price** - the price of a certain trip.

Additionally, the requestor's ontology contains the concept **myRoute**, with the following signature¹:

```
concept myRoute
  nonFunctionalProperties
    dc:description hasValue "concept of myRoute, containing the source
      and the destination locations, and the date of the trip"
    mode hasValue out
  endNonFunctionalProperties
  sourceLocation ofType station
  destinationLocation ofType station
  onDate ofType date
```

¹ All the concepts described in this section are WSMO concepts and they are modelled using Web Service Modeling Language (www.wsmml.org).

The choreography of the requestor states that the attribute `mode` has the value `out` for the concept `myRoute`, which means that the client will send the instance of this concept to the environment. `mode` has the value `in` for `time` and `price`, since the client expects to receive information about the departure time and the price of the trip from the service, and `controlled` for `station` and `date` (only the client can decide the starting and the destination point of his trip, as well as the date he wants to travel).

Additionally, it's choreography includes the following rules:

```
?x [sourceLocation hasValue ?sourceLocation_,
    destinationLocation hasValue ?destinationLocation_,
    onDate hasValue ?onDate_]
  memberOf myRoute <- ?sourceLocation memberOf station and
    ?endLocation memberOf station and ?onDate memberOf date.
```

The above rule creates an instance of `myRoute`, assuming that two instances of `station` and an instance of `date` are already created; since both `station` and `date` have the value of `mode` set to `controlled`, the requestor does not expect any input in order to create the instance of `myRoute`.

```
?x memberOf time <- ?myRoute memberOf myRoute.
```

An instance of `time` is expected, after the instance of `myRoute` was sent to the service.

```
?x memberOf price <- ?myRoute memberOf myRoute.
```

An instance of `price` is expected, after the instance of `myRoute` was sent to the service.

As shown by these rules, the client will first send an instance of `myRoute`, and then expects to receive an instance of `time` and `price`. There are no restrictions regarding the order of receiving the `time` and `price` instances.

The service also has some additional concepts in its choreography: `route` and `routeOnDate`, with the following signatures:

```
concept route
  nonFunctionalProperties
    dc#description hasValue "concept of route, having two attributes of type
      station which show the starting and the ending point of the route"
    mode hasValue out
  endNonFunctionalProperties
  sourceLocation ofType station
  destinationLocation ofType station

concept routeOnDate
  nonFunctionalProperties
    dc#description hasValue "concept of route on a certain date, containing the
      containing the route, the date, the departure time and the price of a ticket"
    mode hasValue out
  endNonFunctionalProperties
  forRoute ofType route
  onDate ofType date
  onTime ofType time
  forPrice ofType price
```

From the service's concepts, the attribute `mode` has the value `in` only for `station` and `date`, and the value `out` for `route` and `routeOnDate`. The other concepts have the `mode` set to `static`, which means that their instances' values can not be changed during the communication process.

The service's choreography includes the following rules:

```
?x [startLocation hasValue ?startLocation_, endLocation hasValue ?endLocation_]
  memberOf route <- ?startLocation_ memberOf station and ?endLocation_ memberOf station.
```

The above rule states that an instance of **route** can be created only after two instances of **station** exists; since the concept **station** has the mode set to **in**, this instances need to be provided by the environment; the instance of **route** will be sent to the requestor of the service.

```
?x [forRoute hasValue ?forRoute_, onDate hasValue onDate_,
    onTime hasValue ?onTime_, forPrice hasValue ?forPrice_]
  memberOf routeOnDate<- ?forRoute_ memberOf vtasc#route and ?onDate_ memberOf vtasc#date and
    ?onTime_ memberOf vtasc#time and ?forPrice memberOf xsd#integer.
```

This rule expresses the fact that an instance of **routeOnDate** is created, assuming that instances of **route**, **date**, **time** and **price** already exist; since only **date** has the mode set to **in**, it is the only instance expected from the environment.

5.2 Communication Process

In this chapter we illustrate step by step the communication process between the VTA service provider and requestor:

1. The requestor initiates the communication by sending an instance of **myRoute**.

2. PM translates this instance in terms of the service's ontology, obtaining two instances of **station** and one of **date**, which it stores in its internal repository.

3. Conform to the provider's choreography, all these three instances are expected, but the guarded transitions show that only one of them (one instance of **station**) is expected at this phase. Since the targeted choreography does not specify which one of the **station**'s instances is expected, the PM randomly sends one of them, and deletes it from the repository.

The evaluation of the transition rules starts again for the rest of the instances from the repository, and the second instance of **station** is sent and deleted.

4. PM evaluates the requestor's rules, and marks the first of them, which means it will not be reevaluated during further iterations.

5. Internally, the provider creates the instance of **route**, which is sent to WSMX.

6. After translating the **route**'s instance in terms of the requestor's ontology, and analyzing the two choreographies, the PM discards the instance of **route** (nobody is expecting any information contained by that instance) and the mediated instances. By evaluating the transition rules PM determines that the provider expects the previously stored instance of **date**; it sends it and deletes it from its internal repository.

PM marks the first rule from the service's choreography, which means it will not reevaluate it at further iterations.

7. PM marks the first rule from the requestor's choreography.

8. PM checks if all requestor's rules are marked; since there are still unmarked rules, the communication is not over yet.

9. The provider creates an instance of `routeOnDate` and sends it to WSMX.

10. PM translates the `routeOnDate` in terms of the requestor's ontology in two instances of `station`, an instance of `time` and one of `price`. Nobody is expecting instances of `station` anymore, so these two can be deleted. The `price` and `time` instances are sent to the requestor; the order of sending them is not specified in the requestor's choreography, so the PM randomly selects one, sends it to the requestor, and deletes it from the repository; the corresponding rule is marked.

The second instance is sent and deleted at the second evaluation of the instances contained by the repository. The rule triggered by sending this instance is marked.

11. PM evaluates the service's rules and mark the second one.

12. PM checks if all requestor's rules are marked; since they are, the communication is over.

13. PM deletes the two choreography instances (it should also delete any instances from its internal repository, but in this case there are none).

6 Related Work

Processes mediation is still a poorly explored research field, in the context of Semantic Web Services. The existing work represents only visions of mediator systems able to resolve in a (semi-) automatic manner the processes heterogeneity problems, without presenting sufficient details about their architectural elements. Still, these visions represent the starting points and valuable references for the future concrete implementations.

Two integration tools, *Contivo*² and *CrossWorlds*³ seemed to be the most advanced ones in this field.

Contivo is an integration framework which uses metadata representing messages organized by semantically defined relationships. One of its functionalities is that it is able to generate transform code based on the semantic of the relationships between data elements, and to use this code for transforming the exchange messages. However, *Contivo* is limited by the use of a purpose-built vocabulary and of pre-configured data models and formats.

CrossWorlds is an IBM integration tool, meant to facilitate the B2B collaboration through business processes integration. It may be used to implement various e-business models, including enhanced intranets (improving operational efficiency within a business enterprise), extranets (for facilitating electronic trading between a business and its suppliers) and virtual enterprises (allowing enterprises to link to outsourced parts of its organization). The disadvantage of this tool is that different applications need to implement different collaboration and connection modules, in order to interact. As a consequence, the integration of a new application can be done only with additional effort.

² <http://www.contivo.com/>

³ <http://www.sars.ws/hl4/ibm-crossworlds.html>

Through our approach we aim to provide dynamic mediation between various parties using WSMO for describing goals and Web Services. As described in this paper this is possible without introducing any hard-coded transformations.

7 Conclusions

In this document we proposed an approach and a mechanism for coping with the processes heterogeneity problem. The processes we are addressing in this paper are the public processes of business entities (services or a requestors of services), which express their public processes as WSMO choreographies.

The proposed approach is based on the semantic description of the Web Services and of their behavior, as well as on how a requestor that wants to interact with a Web Service express the expected behaviour of the Web Service.

We showed that an algorithm that considers the concepts' definitions and evaluates the transition rules from the two choreographies can resolve (some of the) heterogeneity problems of the collaborating parties.

References

1. Business Process Trends Glossary. http://www.bptrends.com/resources_glossary.cfm, 2003.
2. E. Cimpian and A. Mocan. Process Mediation in WSMX. Technical report, WSMX Working Draft, <http://www.wsmo.org/TR/d13/d13.7/v0.1/>, March 2005.
3. E. Cimpian, M. Moran, E. Oren, T. Vitvar, and M. Zaremba. Overview and Scope of WSMX. Technical report, WSMX Working Draft, <http://www.wsmo.org/TR/d13/d13.0/v0.2/>, February 2005.
4. J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Technical report, WSML Working Draft, <http://www.wsmo.org/TR/d16/d16.1/v0.2/>, March 2005.
5. D. Fensel and C. Bussler. The web service modeling framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
6. T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–229, 1993.
7. Y. Gurevich. Evolving algebras 1993: Lipari Guide. In Egon Börger, editor, *Specification and Validation Methods*, pages 9–37. Oxford University Press, 1994.
8. A. Mocan and E. Cimpian. WSMX Data Mediation. Technical report, WSMX Working Draft, <http://www.wsmo.org/TR/d13/d13.3/v0.2/>, March 2005.
9. D. Roman, J. Scicluna, C. Feier, M. Stollberg, and D. Fensel. Ontology-based Choreography and Orchestration of WSMO Services. Technical report, WSMO Working Draft, <http://www.wsmo.org/TR/d14/v0.1/>, March 2005.
10. M. Zaremba, M. Moran, and T. Haselwanter. WSMX Architecture. Technical report, WSMX Working Draft, <http://www.wsmo.org/TR/d13/d13.4/v0.2/>, March 2005.

An Abstract Machine Architecture for Web Service Based Business Process Management

Roozbeh Farahbod, Uwe Glässer, and Mona Vajihollahi

School of Computing Science,
Simon Fraser University,
Burnaby, B.C., Canada
{rfarahbo, glaesser, mvajihol}@cs.sfu.ca

Abstract. We define an abstract operational model of the Business Process Execution Language for Web Services (BPEL) based on the *abstract state machine* (ASM) formalism. That is, we abstractly model dynamic properties of the key language constructs through the construction of a *BPEL abstract machine*. Specifically, we present the *process execution model* and the underlying *execution lifecycle* of BPEL activities. The goal of our work is to provide a precise and well defined semantic framework for establishing the key language attributes. To this end, the BPEL abstract machine forms a comprehensive and robust formalization closely reflecting the view of the informal language definition.

1 Introduction

In this paper, we present an abstract operational model of the Business Process Execution Language for Web Services (BPEL4WS) [1] proposed by OASIS [2] as a future standard for the e-business world. BPEL4WS, or BPEL for short, provides distinctive expressive means for describing the process interfaces of Web based business protocols and builds on existing standards and technologies for Web Services; specifically, it is defined on top of the service interaction model of W3C's Web Services Description Language (WSDL) [3].

Based on the *abstract state machine* (ASM) formalism [4], we define a *BPEL abstract machine*, called $BPEL_{AM}$, as a concise and robust semantic framework for establishing the key language attributes in a precise and well defined form. That is, we model the dynamic properties of the Web Services interaction model of a BPEL business process in terms of finite or infinite abstract machine *runs*. The concurrent and reactive nature of Web Services and the need for dealing with time related aspects in coordinating distributed activities call for an asynchronous execution model with an abstract notion of real time. Thus, the semantic foundation for the $BPEL_{AM}$ model is a *distributed real-time ASM*. The resulting model captures the dynamic properties of the key BPEL language constructs defined in the language reference manual [1], henceforth called LRM, including concurrent control structures, dynamic creation and termination of ser-

vice instances, communication primitives, message correlation, event handling, and fault and compensation handling.

The goal of our work is twofold. First and foremost, BPEL_{AM} provides a firm semantic foundation, a blueprint of the language design, for checking consistency and validity of semantic properties. Formalization is crucial for identifying and eliminating deficiencies that easily remain hidden in the informal language definition of the LRM [2, Issue #42]: “*There is a need for formalism [...] Empirical deduction is not sufficient.*”

Second, we address pragmatic issues resulting from previous experience with other industrial standards, including the ITU-T language SDL [5] and the IEEE language VHDL [6]. An important observation is that sensible use of formal techniques and supporting tools for practical purposes such as standardization calls for a gradual formalization of abstract requirements with a degree of detail and precision as needed [5]. To avoid a gap between the informal language definition and the formal semantics, the ability to model the language definition *as is* without making compromises is crucial. Consequently, we adopt here the LRM view and terminology, effectively formalizing the intuitive understanding of BPEL as directly as possible and in a comprehensible and objectively verifiable form.

Our BPEL_{AM} provides what is called an *ASM ground model* [4, 7] of BPEL. Intuitively, ground models serve as blueprints for establishing functional software requirements — including elicitation, clarification and documentation — so that one can reason about vital aspects of system behavior prior to building a system. Constructing such a ground model means a major effort, especially, as a clear architectural model, which is central for dealing with complex semantic issues, is often missing in the current language definition.

The paper is organized as follows. Section 1.1 briefly summarizes the formal semantic framework. Section 2 introduces the core of the hierarchically defined BPEL_{AM} , and Section 3 addresses important extensions to the BPEL_{AM} core. Section 4 discusses related work, and Section 5 concludes the paper.

1.1 Distributed Real-Time ASM

We briefly outline the formal semantic framework at an intuitive level of understanding using common notions and structures from discrete mathematics and computing science. For details, we refer to the existing literature on the theory of abstract state machines [8] and their applications [4].¹

We focus here on the asynchronous ASM model, called distributed abstract state machine (DASM), as formal basis for modeling concurrent and reactive system behavior in terms of abstract machine *runs*. A DASM M is defined over a given vocabulary V by its program P_M and a non-empty set I_M of initial states. V consists of symbols denoting the various semantic objects and their relations in the formal representation of M , where we distinguish *domain symbols*, *function symbols* and *predicate symbols*. Symbols that have a fixed interpretation regardless of the state of M are called *static*; those that may have different in-

¹ See also the ASM Web site at www.eecs.umich.edu/gasm.

interpretations in different states of M are called *dynamic*. A state S of M yields a valid interpretation of all the symbols in V .

Concurrent control threads in an execution of P_M are modeled by a dynamic set AGENT of autonomously operating *agents*. Agents of M interact with each other by reading and writing shared locations of global machine states, where the underlying semantic model regulates such interactions so that potential conflicts are resolved according to the definition of *partially ordered runs* [4].

P_M consists of a statically defined collection of agent programs, each of which defines the behavior of a certain *type* of agent in terms of state transition rules. The canonical rule consists of a basic update instruction of the form $f(t_1, t_2, \dots, t_n) := t_0$ where f is an n -ary dynamic function symbol and the t_i s ($0 \leq i \leq n$) are terms. An update instruction specifies a pointwise function update, i.e., an operation that replaces an existing function value by a new value to be associated with the given arguments. Complex rules are formed by means of simple rule constructors.

2 BPEL Abstract Machine

This section introduces the core BPEL_{AM} architecture and underlying abstraction principles. Starting with a brief characterization of the key language features defined in [1], we describe the process execution model and its decomposition into *execution lifecycles* of basic and structured activities. Based on the abstract architectural view, we model the *pick* activity as a concrete example of a structured activity involving concurrency and real-time among other aspects. The architectural view, the decomposition into execution lifecycles, and the model of *pick* are new and not contained in [9].

BPEL introduces a stateful model of Web Services interacting with one another by exchanging sequences of messages. A business process and its partners are defined by a collection of abstract WSDL services based on the WSDL model for message interaction. The major parts of a BPEL process definition consist of (1) *partners* of the business process, i.e. Web services that this process interacts with, (2) a set of *variables* that keep the state of the process, and (3) an *activity* defining the logic of interactions between the process and its partners. Activities that can be performed by a business process are categorized into *basic* activities, *structured* activities and *scope-related* activities. Basic activities perform simple operations like *receive*, *reply*, *invoke* and others. Structured activities impose an execution order on a collection of activities and can be nested. Scope-related activities serve for defining logical units of work and encapsulating the reversible behavior of each such unit.

Dynamic Process Creation. A BPEL process definition serves as a template for creating business process instances. Process creation is implicit and is done by defining a *start activity* — either a *receive* or a *pick* activity that is annotated with ‘*createInstance = yes*’ — causing a new process instance to be created upon receiving a matching message. That is, when a new instance of a business

process is created, it starts its execution by receiving the message that triggered its creation.

Correlation and Data Handling. A Web service consists of a number of business process instances; thus, the messages arriving at a specific port must be delivered to the correct process instance according to the state of each process instance. BPEL introduces a generic mechanism for dynamic binding of messages to process instances, called *correlation*. The data handling features of BPEL facilitate dealing with stateful interactions by providing the ability to keep track of the internal state of each business process instance.

Long Running Business Transactions. Business processes normally perform transactions with non-negligible duration involving local updates at business partners. When an error occurs, it may be required to reverse the effects of some or even all of the previous activities. This is known as *compensation*. The ability to compensate the effects of previous activities in case of an exception enables so-called Long-Running (Business) Transactions (LRTs).

In the process of building the BPEL_{AM} model, we have extracted the key language requirements from the LRM in form of *requirement lists* making these requirements accessible for further extensions, validation, and verification of the model, and also to facilitate finding inconsistencies and ambiguities in the LRM. For a complete list of these requirements see [10].

2.1 Abstract Machine Architecture

The BPEL_{AM} architecture is composed of three basic building blocks, referred to as *core*, *data handling extension*, and *fault and compensation extension*. The *core* handles dynamic process creation/termination, communication primitives, message correlation, concurrent control structures, as well as the following activities: *receive*, *reply*, *invoke*, *wait*, *empty*, *sequence*, *switch*, *while*, *pick* and *flow*. The *core* does not consider data handling, fault handling, and compensation behavior; rather these aspects are treated as extensions to the core (see Section 3). Together with the *core*, the extensions form the complete BPEL_{AM} .

The vertical organization of the machine architecture consists of three layers, called *abstract* model, *intermediate* model and *executable* model. The abstract model formally sketches the behavior of the key BPEL constructs and introduces the overall organization of the abstract machine architecture. The intermediate model, obtained as the result of the first refinement step, provides a comprehensive formalization as required for establishing of and reasoning about key language properties. Finally, the executable model provides an abstract executable semantics implemented in AsmL [11]. A GUI facilitates experimental validation through simulation and animation of abstract machine runs.

A BPEL document abstractly defines a Web service consisting of a collection of business process instances. Each such instance interacts with the external world through two interface components, called *inbox manager* and *outbox manager*. The inbox manager handles all the messages that arrive at the Web service. If a message matches a request from a local process instance waiting for that

message, it is forwarded to this process instance. Additionally, the inbox manager also deals with new process instance creation. The outbox manager, on the other hand, forwards outbound messages from process instances to the network.

Inbox manager, outbox manager, and process instances are modeled by three different types of DASM agents: the *inbox manager agent*, the *outbox manager agent*, and one uniquely identified *process agent* for each of the process instances.

In the following sections, we model the behavior of the inbox manager and the process instances in terms of the execution lifecycles of basic and structured BPEL activities. For a comprehensive definition of the formal model see [10, 12].

2.2 Inbox Manager

The LRM does not explicitly address the mechanism for assigning inbound messages to matching business process instances but provides only loose guidelines basically leaving this problem to the engine (or implementation). We contend that the message assignment mechanism is essential for defining the semantics of activities that receive messages, such as *receive* and *pick*. Hence, we collect the scattered LRM requirements on inbound messages (see the requirement lists in [10, App. A]) and combine them to model the behavior of the inbox manager. In our model, the inbox manager is the entity responsible for assigning inbound messages to matching process instances.

The inbox manager agent operates on the *inbox space*, a possibly empty set of inbound messages. In each computation step, it attempts to assign a message to a matching process instance. The predicate $correspond(p, dsc, m)$ holds if message m can be assigned to process instance p according to the information specified by the input descriptor dsc . An *input descriptor*² contains information on the waiting input operation and the waiting agent. If the matching is successful, the message is assigned to the process instance by the `AssignMessage` rule which is further defined in the intermediate model [12, 10].

Another major issue deserving attention is process creation. The LRM states [1, Section 6.4] that “*the creation of a process instance in BPEL4WS is always implicit; activities that receive messages (that is, receive activities and pick activities) can be annotated to indicate that the occurrence of that activity causes a new instance of the business process to be created. [...] When a message is received by such an activity, an instance of the business process is created if it does not already exist.*” Therefore, the execution of such an input activity (called start activity) is accompanied by the creation of the corresponding process instance. In other words, a new process instance is created by execution of its first activity. So the question is, how can an activity of a process instance be executed before the process instance is created? Although this approach is somewhat unconventional, the LRM does not further clarify process creation. However, because of the importance of process creation in the lifecycle of business processes, we cap-

² In BPEL_{AM}, input activities (such as *receive* and *pick*) add an input descriptor to the *waitingSetForInput* for every message they expect to receive.

ture this requirement *as is* in the formal model. This is done by introducing the notion of a *dummy process instance* in the inbox manager.

Basically, the dummy process instance is not different from other process instances in its nature. However, there is always one and only one such process instance which is waiting on its start activity. The inbox manager creates a new process instance whenever a matching message arrives for a start activity of the dummy process. Modeling process instance creation is simplified by introducing a nullary function *dummy* identifying the dummy process instance. By receiving the first matching message, the dummy process instance becomes a normal running process instance and a new dummy process instance will be created automatically by the inbox manager updating the value of *dummy* accordingly.

The DASM program given below specifies the behavior of the inbox manager, where *self* refers to an inbox manager agent.

```

InboxManagerProgram  $\equiv$  InboxManager
  if inboxSpace(self)  $\neq \emptyset$  then
    choose  $p \in \text{PROCESS}, m \in \text{inboxSpace}(self),$ 
       $dsc \in \text{waitingSetForInput}(p)$  with correspond(p, dsc, m)
      AssignMessage(p, dsc, m)
      if  $p = \text{dummyProcess}$  then // process instance creation
        new newDummy : PROCESS
        dummyProcess := newDummy
  where
    correspond(p : PROCESS, dsc : INPUT_DSCRIP, m : MESSAGE)  $\equiv$ 
      match(p, dscOperation(dsc), m) \wedge \text{waitingOnIO}(dscAgent(dsc), p)
      // waitingOnIO confirms the agent is still waiting (no fault, no termination)

```

The behavior of the inbox manager also addresses a loose end in the LRM. According to the LRM, a receive activity is a “*blocking activity in the sense that it will not complete until a matching message is received by the process instance.*” [1, Section 11.4] Therefore, it is implicitly assumed that a matching message will arrive after the corresponding receive activity has been executed, and it is not clear what happens when a message arrives before the corresponding receive activity is executed. Indeed, such a message can be regarded pessimistically (e.g., discarded) or optimistically (e.g., stored in a buffer), each of which giving rise to a different implementation of the language. Thus, it is certainly important for the LRM to provide a comprehensive description of the message assignment mechanism. For a more detailed discussion of the inbox manager and the associated issues of the LRM, the reader is referred to [12]. In BPEL_{AM}, all incoming messages are buffered before being processed.

2.3 Activity Execution Lifecycle

Intuitively, the execution of a process instance is decomposed into a collection of execution lifecycles for the individual BPEL activities. We therefore introduce *activity agents*, created dynamically by process agents, for executing structured activities. Each activity agent dynamically creates additional activity agents for

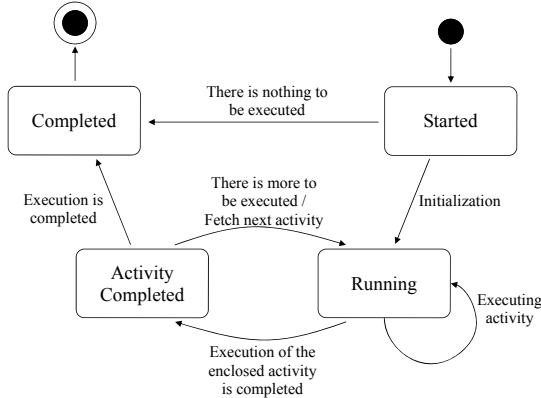


Fig. 1. Activity Execution Lifecycle: BPEL_{AM} core

executing nested, structured activities. Similarly, it creates auxiliary activity agents for dealing with concurrent control threads (like in *flow* and *pickFor* for instance, to concurrently execute a set of activities, a flow agent assigns each enclosed activity to a separate *flow thread agent* [9]). At any time during the execution of a process instance, the DASM agents running under control of this process agent form a tree structure where each of the sub-agents monitors the execution of its child agents (if any) and notifies its parent agent in case of normal completion or fault. This structure provides a general framework for execution of BPEL activities. The DASM agents that model BPEL process execution are jointly called *kernel agents*. They include process agents and subprocess agents. In the *core*, however, subprocess agents are identical to activity agents.

Figure 1 illustrates the normal activity execution lifecycle of kernel agents in the BPEL_{AM} core. When created, a kernel agent is in the *Started* mode. After initialization, the kernel agent starts executing its assigned task by switching its mode to *Running*. Upon completion, the agent switches its mode to *Activity-Completed* and decides (based on the nature of the assigned task) to either return to the *Running* mode or finalize the execution and become *Completed*. Activity agents that may execute more than one activity (like *sequence*) or execute one activity more than once (like *while*) can switch back and forth between the two modes *Activity-Completed* and *Running*.

2.4 Pick Activity

A *pick* activity identifies a set of events and associates with each of these events a certain activity. Intuitively, it waits on one of the events to occur and then performs the respective activity; thereafter, the *pick* activity no longer accepts any other event.³ There are basically two different types of events: *onMessage* events and *onAlarm* events. An *onMessage* event occurs as soon as a related

³ Regarding the case that several events occur at a time, the LRM is somewhat loose declaring that the choice “is dependent on both timing and implementation.” [1]

message is received, whereas an onAlarm event is triggered by a timer mechanism waiting ‘for’ a certain period of time or ‘until’ a certain deadline is reached.

In BPEL_{AM}, each *pick* activity is modeled by a separate activity agent, called *pick agent*. A pick agent is assisted by two auxiliary agents, a *pick message agent* that is waiting for a message to arrive, and a *pick alarm agent* that is watching a timer. We formalize the semantics of the *pick* activity in several steps, each of which addresses a particular property, and then compose the resulting DASM program, called *PickProgram*, in which *self* refers to a pick agent executing the program.

PickProgram ≡

Pick Agent

```

case execMode(self) of
  Started → PickAgentStarted
  Running → PickAgentRunning
  ActivityCompleted → FinalizePickAgent
  Completed → stop self

```

When created, the pick agent is in the *Started* mode and initializes its execution by creating a pick alarm agent and a pick message agent. It then switches its mode to *Running* and waits for an event to occur — either a message arrives or a timer expires.

PickAgentRunning ≡

Pick Agent

```

if normalExecution(self) then
  onsignal s : AGENT_COMPLETED
    execMode(self) := ActivityCompleted
  otherwise
    if chosenAct(self) = undef then
      choose dsc ∈ occurredEvents(self) with MinTime(dsc)
        chosenAct(self) := onEventAct(edscEvent(dsc))
        // onEventAct is the activity associated with an event
    else
      ExecuteActivity(chosenAct(self))

```

Depending on the event type, either the pick message agent or the pick alarm agent notifies the pick agent by adding an *event descriptor* to the *occurredEvents* set of the pick agent. An event descriptor contains information on the event such as the time of its occurrence. When an event occurs, the pick agent updates the function *chosenAct* (with initial value *undef*) with the activity associated with the event. Once the activity is chosen (*chosenAct*(*self*) ≠ *undef*), the pick agent performs the chosen activity and remains *Running* until the execution of the chosen activity is completed as indicated by a predicate *chosenActCompleted*. It then switches its execution mode to *Activity-Completed*.

Finalizing a running pick agent includes informing its parent agent that the execution is completed and changing the execution mode to *Completed*. As illustrated in Figure 1, the *Completed* mode leads to the agent’s termination.

Due to the space limitations, we do not show here the definitions of `PickAgent-Started`, `FinalizePickAgent`, as well as the programs of the pick message and the pick alarm agents, but refer to [10, 13] for a complete description.

3 Extensions to the $\text{BPEL}_{\mathcal{AM}}$ Core

Our two-dimensional refinement approach [10] facilitates refinements of the *core* to capture additional aspects of BPEL through incremental extensions [14], and enables step by step elucidation of the extensions through a combination of data refinement and procedural refinement approaches [10, 14]. For a clear separation of concerns and also for robustness of the formal semantic model, the aspects of data handling, fault handling and compensation behavior are carefully separated from the core of the language. To this end, the core of $\text{BPEL}_{\mathcal{AM}}$ provides a basic, yet comprehensive, model for *abstract processes* in which data handling focuses on protocol relevant data in the form of correlations while payload data values are left unspecified [1].

Compensation and fault handling behavior is a fairly complex issue in the definition of BPEL. An in-depth analysis in fact shows that the semantics of fault and compensation handling, even when ignoring all the syntactical issues, is related to more than 40 individual requirements spread out all over the LRM. These requirements (some of them comprise up to 10 sub-items) address a variety of separate issues related to the core semantics, general constraints, and various special cases (see [10, App. A]). While most of these requirements are defined with painstaking accuracy, such definitions are not free of ambiguities and imprecisions inherent to natural languages. Consequently, complementary formal descriptions are vital for turning abstract requirements into precise specifications. Specifically, they are beneficial since: 1) analyzing a requirement to construct a formal specification often provides a different view to the requirement (and to the system under inspection) potentially uncovering possible problems, such as ambiguities, inconsistencies and loose ends; 2) formalization of requirements along with their informal description (the idea of *literate specifications* [15]) provides a sensible way of gaining precision without loosing intelligibility.

A thorough treatment of the extensions is beyond the space limitations of this paper. We present an overview of the fault handling behavior in the following sections and refer to [10] for a comprehensive description of non-trivial issues.

3.1 Scope Activity

The *scope* activity is the core construct of data handling, fault handling, and compensation handling in BPEL. A *scope* activity is a wrapper around a logical unit of work (a block of BPEL code) that provides local variables, a fault handler, and a compensation handler. The fault handler of a scope is a set of *catch* clauses defining how the scope should respond to different types of faults. A compensation handler is a wrapper around a BPEL activity that compensates the effects of the execution of the scope. Each scope has a primary activity which defines the normal behavior of the scope. This activity can be any basic

or structured activity. BPEL allows scopes to be nested arbitrarily. In $BPEL_{AM}$, we model scopes by defining a new type of activity agents, called *scope agents*.

Fault handling in BPEL can be conceived as a mode switch from the normal execution of the process [1]. When a fault occurs in the execution of an activity, the fault is thrown up to the innermost enclosing scope. If the scope handles the fault successfully, it sends an *exited* signal to its parent scope and ends gracefully, but if the fault is re-thrown from the fault handler, or a new fault has occurred during the fault handling procedure, the scope sends a *faulted* signal along with the thrown fault to its parent scope. The fault is thrown up from scopes to parent scopes until a scope handles it successfully. A successful fault handling switches the execution mode back to normal. If a fault reaches the global scope, the process execution terminates [1].

The normal execution lifecycle of the process execution model (Figure 1) needs to be extended to comprise the fault handling mode of BPEL processes. The occurrence of a fault causes the kernel agent (be it an activity agent or the main process) to leave its normal execution lifecycle and enter a fault handling lifecycle. Figure 2 illustrates the extended execution lifecycle of BPEL activities.

In $BPEL_{AM}$, whenever a sub-process agent encounters a fault, the agent leaves its normal execution mode and enters the *Execution-Fault* mode. If this agent is not a scope agent, it informs its parent agent of the fault and stays in the *Execution-Fault* mode until it receives a notification for termination. On the other hand, if the faulted agent is a scope agent, it terminates its enclosing activity, creates a fault handler, assigns the fault to that handler, and switches to the *Fault-Handling* mode. If the fault handler finishes successfully, the scope agent enters the *Exited* mode indicating that this agent exited its execution with a successful fault handling process. The difference between a *scope* which has finished its execution in the *Completed* mode and a *scope* that has finished in the *Exited* mode is reflected by the way scopes are compensated, which we do not further address in this paper.

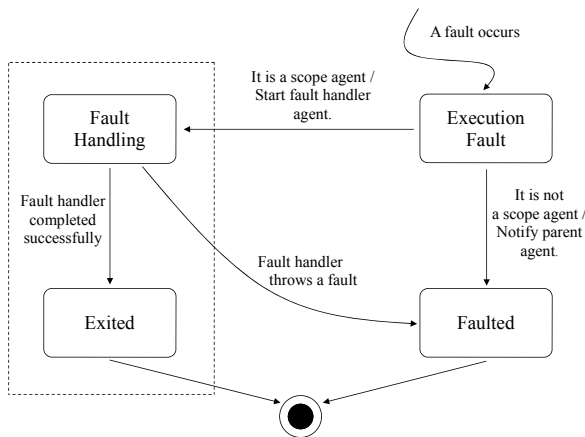


Fig. 2. Activity Execution Lifecycle: Fault Handling

3.2 Pick Activity: Extended

The structured activities of the *core* (activity agents) are also refined to capture the fault handling behavior of BPEL. The well-defined activity execution life-cycle of BPEL_{AM} (Figures 1 and 2) along with the fact that the fault handling behavior of BPEL is mostly centered in the *scope* activity, enable us to generally extend the behavior of structured activities by defining two new rules: **HandleExceptionsInRunningMode** and **WaitForTermination**. As an example, the pick agent program of Section 2.4 is refined as follows:

PickProgram \equiv

PickProgram_{core}

case *execMode*(*self*) **of**

Running \rightarrow HandleExceptionsInRunningMode

ExecutionFault \rightarrow WaitForTermination

Faulted \rightarrow **stop** *self*

Pick Activity Extended

Activity agents react to a fault by informing their parent agent of the fault and stay in the *Execution-Fault* mode until they receive a notification for termination. If the parent agent is not a scope agent, the parent agent reacts in the same way and the fault is passed upwards until it reaches a scope agent. The scope agent handles the fault as described in Section 3.1, and sends a termination notification to its child agent. Upon receiving the notification, a sub-process agent that is waiting for a termination notification in turn passes it to its child agents (if any) and enters the *Faulted* mode, where it then terminates. If a sub-process agent receives a termination notification while in its normal execution mode, it first enters the *Execution-Fault* mode and then reacts as if it were waiting for the notification.

The normal execution of activity agents in the *Running* mode is extended by the following rule. The *faultExtensionSignal* predicate holds only if the agent has received a signal related to fault and compensation handling. The *normalExecution* predicate in PickAgentRunning (Section 2.4) is defined as the negation of this predicate which facilitate conservative refinement of the core model.

In the *Execution-Fault* mode, if a termination notification is received, the pick agent terminates its enclosing activity and goes to the *Faulted* mode. Analogously to the *Completed* mode, sub-process agents terminate their execution in the *Faulted* mode. For the complete extended pick agent program see [13].

HandleExceptionsInRunningMode \equiv

if *faultExtensionSignal*(*self*) **then**

onsignal *s* : AGENT_EXITED

execMode(*self*) := *ActivityCompleted*

otherwise

onsignal *s* : AGENT_FAULTED

TransitionToExecutionFault(*fault*(*s*))

otherwise

onsignal *s* : FORCED_TERMINATION

Structured Activity Extended

```

faultThrown(self) := fault(s)
PassForcedTerminationToChildren(fault(s))
execMode(self) := emExecutionFault

```

The LRM does not precisely specify how activity termination (due to a fault) takes place. It states that when a fault occurs in a *scope*, the fault handler begins by implicitly terminating all activities inside the *scope*. Further, in [1, Appendix A] on standard faults, the LRM states that *forcedTermination* is used by a scope to terminate its enclosing activities. However, it is not clear how the *forcedTermination* fault is used to terminate enclosing activities: it is not stated whether the faulted activities should wait for the *forcedTermination* fault when they encounter a fault, or they should terminate spontaneously. See [10] for an example and more details on this discussion.

4 Related Work

There are various research activities to formally define, analyze, and verify Web Services orchestration languages. A group at Humboldt University is working on formalizations of BPEL for analysis, graphics and semantics [16]. In [17], they translate a small business process into a Petri net model without addressing fault handling, compensation handling, and timing aspects. The ultimate goal is versification of business processes, however, the feasibility of verifying larger business processes is still subject to future work. The ASM semantic model in [18] closely follows our work in [19, 10, 13] with minor technical differences in handling basic activities and variables. The core of BPEL_{AM} was first introduced in [9]. In [20], elaboration and refinement of the core is briefly discussed.⁴

The SPIN model-checker is used for verification [21] by translating Web Services Flow Language (WSFL) descriptions into Promela. The approach in [22] is based on Petri-nets, while [23] uses a process algebra to derive a structural operational semantics of BPEL as a formal basis for verifying properties of the specification. In [24], BPEL processes are translated to Finite State Process (FSP) models and compiled into a Labeled Transition System (LTS) which is used as a basis for verification.

Various research has been done to evaluate the capabilities and limitations of different languages proposed for Web services composition. Notably, van der Aalst et al. presented a pattern-based analysis of BPEL [25], and BPML and WSCI [26] based on a collection of workflow and communication patterns which allows comparing the capabilities and limitation of these languages.

5 Conclusions

We propose a BPEL abstract machine as a well-defined formal framework for establishing the key semantic concepts of BPEL. The hierarchical and modular

⁴ Neither [9] nor [20] present details of the BPEL_{AM} architecture such as the process execution model and the inbox manager. Also, the two other papers do not discuss the problems we encountered in the LRM.

structure of the abstract machine architecture supports a gradual formalization of complex requirements and ensures a clear separation of concerns. As a result of building this ASM ground model, we actually discovered a number of weak points in the LRM, some of them are addressed in this paper (Sections 2.2 and 3.1); for a complete list see [12, 10].

The focus in this paper is on formal specification rather than verification, understanding the former as a prerequisite for the latter. Beyond reasoning about the language design and checking consistency and validity of semantic properties, the abstract machine also serves as a platform for experimental validation. As result of the final refinement step, we obtain an abstract executable semantics encoded in *AsmL*, an industrial design language [11]. Experimental validation of high-level design specifications clearly offers additional benefits in eliminating deficiencies prior to low-level coding.

The dynamic nature of standardization calls for flexibility and robustness of the formalization approach. To this end, we feel that the ASM formalism and abstraction principles offer a good compromise between practical relevance and mathematical elegance — already proven useful for practical purposes in other standardization contexts [5].

Acknowledgements. The authors would like to thank the four anonymous reviewers for their detailed and valuable comments and suggestions incorporated into the final version of this paper.

References

1. Andrews, T., et al.: Business process execution language for web services version 1.1 (2003) Last visited Feb. 2005, <http://ifr.sap.com/bpel4ws/>.
2. Organization for the Advancement of Structured Information Standards (OASIS): WS BPEL issues list. (2004) <http://www.oasis-open.org>.
3. W3C: Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language. (2003) Last visited May 2004, <http://www.w3.org>.
4. Börger, E., Stärk, R.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag (2003)
5. Glässer, U., Gotzhein, R., Prinz, A.: The formal semantics of SDL-2000: status and perspectives. *Comput. Networks* **42** (2003) 343–358
6. Börger, E., Glässer, U., Müller, W.: Formal Definition of an Abstract VHDL'93 Simulator by EA-Machines. In Delgado Kloos, C., Breuer, P.T., eds.: *Formal Semantics for VHDL*. Kluwer Academic Publishers (1995) 107–139
7. Börger, E.: The ASM ground model method as a foundation for requirements engineering. In: *Verification: Theory and Practice*. (2003) 145–160
8. Gurevich, Y.: Sequential Abstract State Machines Capture Sequential Algorithms. *ACM Transactions on Computational Logic* **1** (2000) 77–111
9. Farahbod, R., Glässer, U., Vajihollahi, M.: Specification and Validation of the Business Process Execution Language for Web Services. In: *Proc. of the 11th Int'l Workshop on Abstract State Machines*, Springer-Verlag (2004)
10. Farahbod, R.: Extending and refining an abstract operational semantics of the web services architecture for the business process execution language. Master's thesis, Simon Fraser University, Burnaby, Canada (2004)

11. Glässer, U., Gurevich, Y., Veanes, M.: An abstract communication architecture for modeling distributed systems. *IEEE Trans. on Soft. Eng.* **30** (2004) 458–472
12. Vajihollahi, M.: High level specification and validation of the business process execution language for web services. Master's thesis, Simon Fraser University, Burnaby, Canada (2004)
13. Farahbod, R., Glässer, U., Vajihollahi, M.: Abstract Operational Semantics of the Business Process Execution Language for Web Services. Technical Report SFU-CMPT-TR-2005-04, Simon Fraser University (2005) Revised version of SFU-CMPT-TR-2004-03, April 2004.
14. Börger, E.: The ASM Refinement Method. *Formal Aspects of Computing* (2003) 237–257
15. Johnson, C.W.: Literate specifications. *Software Engineering Journal* **11** (1996) 225–237
16. Martens, A.: Analysis and re-engineering of web services. To appear in 6th International Conference on Enterprise Information Systems (ICEIS'04) (2004)
17. Schmidt, K., Stahl, C.: A petri net semantic for BPEL4WS - validation and application. In Kindler, E., ed.: *Proceedings of 11th Workshop on Algorithms and Tools for Petri Nets.* (2004)
18. Fahland, D., Reisig, W.: ASM-based semantics for BPEL: The negative control flow. In: *Proc. of the 12th Int'l Workshop on Abstract State Machines.* (2005)
19. Farahbod, R., Glässer, U., Vajihollahi, M.: Specification and Validation of the Business Process Execution Language for Web Services. Technical Report SFU-CMPT-TR-2003-06, Simon Fraser University (2003)
20. Farahbod, R., Glässer, U., Vajihollahi, M.: A formal semantics for the business process execution language for Web Services. In Bevinakoppa, S., et al., eds.: *Web Services and Model-Driven Enterprise Information Systems*, Portugal, INSTICC Press (2005) 144–155
21. Nakajima, S.: Model-checking verification for reliable web service. In: *OOPSLA 2002: Workshop on Object-Oriented Web Services.* (2002)
22. Martens, A.: *Verteilte Geschäftsprozesse - Modellierung und Verifikation mit Hilfe von Web Services.* PhD thesis, Humboldt University of Berlin, Germany (2003)
23. Koshkina, M., van Breugel, F.: Verification of Business Processes for Web Services. Technical Report CS-2003-11, York University (2003)
24. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Compatibility verification for web service choreography. In: *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, IEEE Computer Society (2004) 738–741
25. van der Aalst, W., Dumas, M., ter Hofstede, A., Wohed, P.: Analysis of web services composition languages: The case of bpel4ws. *1st Web Services Quality Workshop (WQW 2003)* (2003)
26. van der Aalst, W., Dumas, M., ter Hofstede, A., Wohed, P.: Pattern-Based Analysis of BPML (and WSCI). Technical Report FIT-TR-2002-05, Queensland University of Technology (2002)

Preface

(BPI 2005)

Business process intelligence (BPI) is an emerging area that has been increasing in importance during the last few years as a result of the pressing need for companies to improve the business processes underlying their business operations so as to better meet their business goals. A number of groups in different research areas are working on technologies to support different aspects of BPI, even if they do not call it this.

Many other names exist for such technologies, and there is confusion concerning the exact meaning of terms like BAM (business activity monitoring), BOM (business operations management), BPM (business performance management), among others. The reality is that there is much overlap among techniques and tools supporting all these technologies. The realization of the first workshop on BPI gave the opportunity to start consolidating this field while at the same time building a community that recognizes BPI as an area encompassing all these technologies whose end goal is the improvement of enterprise business operations.

Broadly speaking we can say that BPI is the application of business intelligence to business processes so as to improve different aspects of how such processes are being conducted. Some of these aspects include:

- Process discovery: this refers to the analysis of enterprise operations in order to derive the process models that these operations obey. It may be useful for users to better understand their operations and it can be the first step that leads to supporting the process with a workflow tool. It can also be used to reengineer an existing process model to make it more efficient.

- 'Intelligent' process analysis: this refers to the analysis of business process execution to discover interesting correlations, e.g., between process data and resources and business metrics, to perform capacity planning, or to identify the causes of low-quality process executions. For example, users may be interested in discovering under which situations a certain exception is raised, or the process follows a certain path, or leads to a certain outcome.

- Prediction: besides analyzing the value of business metrics and understanding, among other things, the causes of low-quality process executions, BPI aims at predicting critical situations (e.g., an exception, or a delay) on a running process instance before it actually happens. Ideally, predictions are made at the early stages of execution of a process instance, and are then refined as the execution progresses and more data becomes available.

- Exception handling: once a problem has been recognized (or predicted), another goal of BPI is to assist the analyst in making decisions to address the problem. This may be, for example, based on mining how similar problems have been successfully handled in the past.

- Static and dynamic optimization: on the static side, the intelligent analysis described above may lead to the identification of areas of optimization for a process, for example, in terms of different sizing of resource pools, different resource assignment criteria, and the like. BPI offers support for optimizing the process configuration to improve upon those areas. On the dynamic side, ideally one could think of an intelli-

gent component that constantly manages and supervises each process instance (in a controlled way), for example, by having influence in routing and task assignment decisions in order to maximize certain business objectives.

In spite of the wide variety of aspects addressed by BPI, in this first instance of the BPI workshop the submitted contributions, briefly described below, only covered a very reduced subset. The strongest focus was on process discovery and related subjects, perhaps because it is the most well-known area of BPI. This is evidence that BPI is still in its infancy and that the research community working on related fields needs to be made aware of the emergence of this new technology.

Acknowledgments

We wish to express a special thanks to the Program Committee members (Wil Van der Aalst, Boualem Benatallah, Fabio Casati, Jonhatan E. Cook, Peter Dadam, Marlon Dumas, Gianluigi Greco, Dimitrios Georgakopoulos, Mati Golani, Joachim Herbst, Cesare Pautasso, Shlomit S. Pinter, Michael Rosemann, Marek Rusinkiewicz, Pnina Soffer, Hans Weigand, Mathias Weske, and Michael zur Muhlen) for providing their technical expertise in reviewing the submitted papers and their valuable support in creating an interesting program.

We are particularly grateful to the keynote speaker, Boualem Benatallah, for delighting us with the interesting keynote “E-Business Automation, Web Services, and Design Intelligence.”

Last but not least, we thank all the authors of the accepted papers for sharing their work and experiences in this workshop.

Finally, we want to express our sincere appreciation to the BPM 2005 Workshops Chair, Chris Bussler, for his support in the organization of the workshops and the proceedings.

Malu Castellanos
Ton Weijters

Organization

Executive Committee

Organizers	Malu Castellanos, Hewlett-Packard Labs, USA Ton Weijters, U. of Eindhoven, The Netherlands
Publication & Coordination Chair	Manolo Garcia-Solaco, CMU-West, USA
Program Chairs	Malu Castellanos, Hewlett-Packard Labs, USA Ton Weijters, U. of Eindhoven, The Netherlands

Program Committee

Wil Van der Aalst, University of Eindhoven
Boualem Benatallah, University of New South Wales, Australia
Fabio Casati, Hewlett-Packard, USA
Jonhatan E. Cook, New Mexico State University, USA
Peter Dadam, University of Ulm, Germany
Marlon Dumas, Queensland University of Technology, Australia
Gianluigi Greco, University of Calabria, Italy
Dimitrios Georgakopoulos, Telcordia Technologies, Austin, USA
Mati Golani, Technion, Israel
Joachim Herbst, DaimlerChrysler Research and Technology, Germany
Cesare Pautasso, ETH Zurich, Switzerland
Shlomit S. Pinter, IBM Haifa Research Lab, Israel
Michael Rosemann, Queensland University of Technology, Australia
Marek Rusinkiewicz, Telcordia Research, USA
Pnina Soffer, Haifa University, Israel
Hans Weigand, Infolab, Tilburg University, The Netherlands
Mathias Weske, Hasso Plattner Institute at University of Potsdam, Germany
Michael zur Muhlen, Stevens Institute of Technology, USA

Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models

A. Rozinat^{1,2} and W.M.P. van der Aalst¹

¹ Department of Technology Management, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands

w.m.p.v.d.aalst@tm.tue.nl

² Department of Business Process Technology, HPI University of Potsdam,
P.O. Box 900460, D-14440, Potsdam, Germany

a.rozinat@tm.tue.nl

Abstract. Most information systems log events (e.g., transaction logs, audit trails) to audit and monitor the processes they support. At the same time, many of these processes have been explicitly modeled. For example, SAP R/3 logs events in transaction logs and there are EPCs (Event-driven Process Chains) describing the so-called reference models. These reference models describe how the system should be used. The co-existence of event logs and process models raises an interesting question: “Does the event log *conform* to the process model and vice versa?”. This paper demonstrates that there is not a simple answer to this question. To tackle the problem, we distinguish two dimensions of conformance: *fitness* (the event log may be the result of the process modeled) and *appropriateness* (the model is a likely candidate from a structural and behavioral point of view). Different metrics have been defined and a *Conformance Checker* has been implemented within the ProM Framework.

1 Introduction

New legislation such as the Sarbanes-Oxley (SOX) Act [22] and increased emphasis on corporate governance and operational efficiency has triggered the need for improved auditing systems. To audit an organization, business activities need to be monitored. Buzzwords such as BAM (Business Activity Monitoring), BOM (Business Operations Management), BPI (Business Process Intelligence) illustrate the interest of vendors to support the monitoring and analysis of business activities. The close monitoring of processes can be seen as a second wave following the wave of business process modeling and simulation. In the first wave the emphasis was on constructing process models and analyzing them. The many notations (e.g., Petri nets, UML activity diagrams, EPCs, IDEF, BPMN, and not to mention the vendor or system specific notations) illustrate this. This creates the interesting situation where processes are being monitored while at the same time there are process models describing these processes. The focus of this paper is on *conformance*, i.e., “Is there a good match between the recorded events and

the model?”. A term that could be used in this context is “business alignment”, i.e., are the real process (reflected by the log) and the process model (e.g., used to configure the system) aligned properly.

Most information systems, such as WFM, ERP, CRM, SCM, and B2B systems, provide some kind of *event log* (also referred to as transaction log or audit trail) [6]. Typically such an event log registers the start and/or completion of activities. Every event refers to a case (i.e., process instance) and an activity, and, in most systems, also a timestamp, a performer, and some additional data. In this paper, we only use the first two attributes of an event, i.e., the identity of the case and the name of the activity. Meanwhile, any organization documents its processes in some form. The reasons for making these process models are manifold. Process models are used for communication, ISO 9000 certification, system configuration, analysis, simulation, etc. A process model may be of a *descriptive* or of a *prescriptive* nature. Descriptive models try to capture existing processes without being normative. Prescriptive models describe the way that processes should be executed. In a Workflow Management (WFM) system prescriptive models are used to enforce a particular way of working using IT [2]. However, in most situations prescriptive models are not used directly by the information system. For example, the reference models in the context of SAP R/3 [18] and ARIS [23] describe the “preferred” way processes should be executed. People actually using SAP R/3 may deviate from these reference models.

In this paper, we will use Petri nets [13] to model processes. Although the metrics are based on the Petri net approach, the results of this paper in general can be applied to any modeling language that can be equipped with executable semantics. An event log is represented by a set of event sequences, also referred to as traces. Each case in the log refers to one sequence. The most dominant requirement for conformance is *fitness*. An event log and Petri net “fit” if the Petri net can generate each trace in the log. In other words: the Petri net should be able to “parse” every event sequence. We will show that it is possible to quantify fitness, e.g., an event log and Petri net may have a fitness of 0.66. Unfortunately, a good fitness does not imply conformance. As we will show, it is easy to construct Petri nets that are able to parse any event log. Although such Petri nets have a fitness of 1 they do not provide meaningful information. Therefore, we introduce a second dimension: *appropriateness*. Appropriateness tries to capture the idea of *Occam’s razor*, i.e., “one should not increase, beyond what is necessary, the number of entities required to explain anything”. Clearly, this dimension is not as easy to quantify as fitness. We will distinguish between *structural appropriateness* (if a simple model can explain the log, why choose a complicated one) and *behavioral appropriateness* (the model should not be too generic). Using examples, we will show that both the structural and behavioral aspects need to be considered to measure appropriateness adequately.

To actually measure conformance, we have developed a tool called *Conformance Checker*. It is part of the *ProM framework*¹, which offers a wide range of tools related to process mining, i.e., extracting information from event logs [6].

¹ Both documentation and software can be downloaded from www.processmining.org.

This paper is organized as follows. Section 2 introduces a running example that will be used to illustrate the concept of conformance. Section 3 discusses the need for two dimensions. The fitness dimension is discussed in Section 4. The appropriateness dimension is elaborated in Section 5. Section 6 shows how these properties can be verified using the conformance checker in ProM. Finally, some related work is discussed and the paper is concluded.

2 Running Example

The example model used throughout the paper concerns the processing of a liability claim within an insurance company (cf. Figure 1(a)).

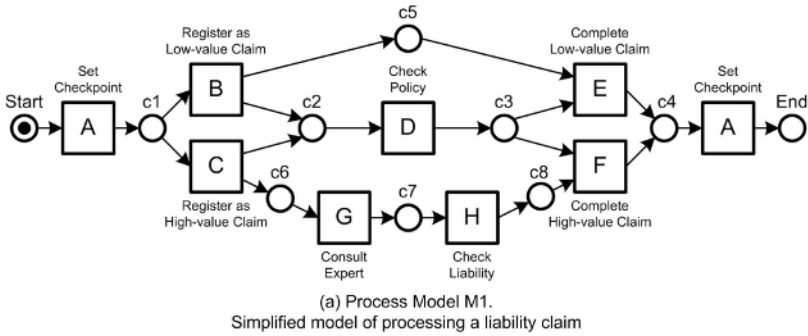
At first there are two tasks bearing the same label “Set Checkpoint”. This can be thought of as an automatic backup action within the context of a transactional system, i.e., activity *A* is carried out at the beginning to define a rollback point enabling atomicity of the whole process, and at the end to ensure durability of the results. Then the actual business process is started with the distinction of low-value claims and high-value claims, which get registered differently (*B* or *C*). The policy of the client is checked anyway (*D*) but in the case of a high-value claim, additionally, the consultation of an expert takes place (*G*), and then the filed liability claim is being checked in more detail (*H*). Finally, the claim is completed according to the former choice between *B* and *C* (i.e., *E* or *F*).

Figures 1(b)-(d) show three example logs for the process described in Figure 1(a) at an aggregate level. This means that process instances exhibiting the same event sequence are combined as a logical log trace, memorizing the number of instances to weigh the importance of that trace. That is possible since only the control flow perspective is considered here. In a different setting like, e.g., mining social networks [5], the resources performing an activity would distinguish those instances from each other.

Note that none of the logs contains the sequence *ACGHDFEA*, although the Petri net model would allow this. In fact it is highly probable that a log does not exhibit all possible sequences of concurrent behavior, since, e.g., the duration of activities or the availability of suitable resources may render some sequences very unlikely to occur. With respect to the example model one could think of task *D* as a standard task requiring a very low specialization level and task *G* and *H* as highly specialized and time-consuming checks, so that finishing *G* and *H* before *D* may not happen in a given log.

3 Two Dimensions of Conformance: Fitness and Appropriateness

Measurement can be defined as a set of rules to assign values to a real-world property, i.e., observations are mapped onto a numerical scale. In the context of conformance testing this means to weigh the “distance” between the behavior actually observed in the workflow log and the behavior described by the process model. If the distance is zero, i.e., the real business process exactly complies with



No. of Instances	Log Traces
4070	ABDEA
245	ACDGHFA
56	ACGDHFA

(b) Event Log L1

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

(c) Event Log L2

No. of Instances	Log Traces
24	BDE
7	AABHF
15	CHF
6	ADBE
1	ACBGDFAA
8	ABEDA

(d) Event Log L3

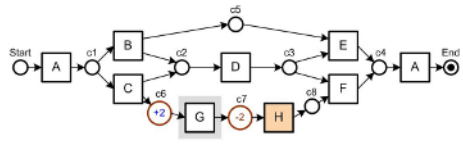
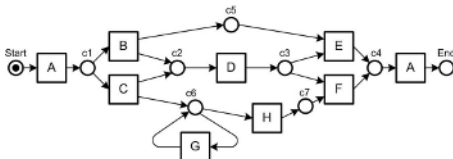
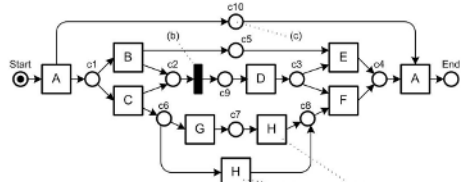
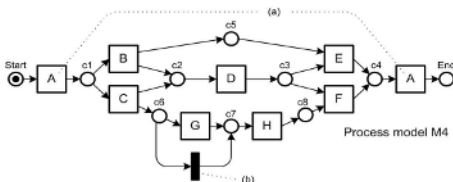
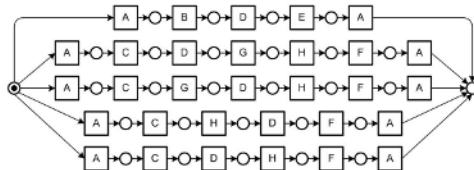
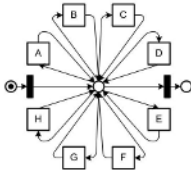


Fig. 1. Example models and logs

the specified behavior, one can say that the log *fits* the model. With respect to the example model $M1$ this seems to apply for event log $L1$, since every log trace can be associated with a valid path from *Start* to *End*. In contrast, event log $L2$ does not match completely as the traces $ACHDFA$ and $ACDHFA$ lack the execution of activity G , while event log $L3$ does not even contain one trace corresponding to the specified behavior. Somehow $L3$ seems to fit “worse” than $L2$, and the degree of fitness should be determined according to this intuitive notion of conformance, which might vary for different settings.

But there is another interesting—rather qualitative—dimension of conformance, which can be illustrated by relating the process models $M2$ and $M3$, shown in Figure 1(e) and (f), to event log $L2$. Although the log fits both models quantitatively, i.e., the event streams of the log and the model can be matched perfectly, they do not seem to be *appropriate* in describing the insurance claim administration.

The first one is much too generic as it covers a lot of extra behavior, allowing for arbitrary sequences containing the activities A, B, C, D, E, F, G , or H , while the latter does not allow for more sequences than those having been observed but only lists the possible behavior instead of expressing it in a meaningful way. Therefore, it does not offer a better understanding than can be obtained by just looking at the aggregated log. We claim that a “good” process model should somehow be minimal in structure to clearly reflect the described behavior, in the following referred to as *structural appropriateness*, and minimal in behavior to represent as closely as possible what actually takes place, which will be called *behavioral appropriateness*.

Apparently, conformance testing demands for two different types of metrics, which are:

- *Fitness*, i.e., the extent to which the log traces can be associated with execution paths specified by the process model, and
- *Appropriateness*, i.e., the degree of accuracy in which the process model describes the observed behavior, combined with the degree of clarity in which it is represented.

4 Measuring Fitness

Different ways are conceivable to measure the fit between event logs and process models. A rather naive approach would be to generate all execution sequences allowed by the model and then compare them to the log traces using string distance metrics. Unfortunately the number of firing sequences increases very fast if a model contains parallelism and might even be infinite if we allow for loops. Therefore, this is of limited applicability.

Another possibility is to replay the log in the model and somehow measure the mismatch, which subsequently is described in more detail. The replay of every logical log trace starts with marking the initial place in the model and then the transitions that belong to the logged events in the trace are fired one after another. While doing so we count the number of tokens that had to be

created artificially (i.e., the transition belonging to the logged event was not enabled and therefore could not be *successfully executed*) and the number of tokens that had been left in the model, which indicates the process not having *properly completed*.

Metric 1 (Fitness). Let k be the number of different traces from the aggregated log. For each log trace i ($1 \leq i \leq k$) n_i is the number of process instances combined into the current trace, m_i is the number of missing tokens, r_i is the number of remaining tokens, c_i is the number of consumed tokens, and p_i is the number of produced tokens during log replay of the current trace. The token-based fitness metric f is formalized as follows:

$$f = \frac{1}{2} \left(1 - \frac{\sum_{i=1}^k n_i m_i}{\sum_{i=1}^k n_i c_i} \right) + \frac{1}{2} \left(1 - \frac{\sum_{i=1}^k n_i r_i}{\sum_{i=1}^k n_i p_i} \right)$$

Note that, for all i , $m_i \leq c_i$ and $r_i \leq p_i$, and therefore $0 \leq f \leq 1$. Using the metric f we can now calculate the fitness between the event logs $L1$, $L2$, $L3$, and the process description $M1$, respectively. The first event log $L1$ shows three different log traces that all correspond to possible firing sequences of the Petri net with one initial token in the *Start* place. Thus, there are neither tokens left nor missing in the model during log replay and the fitness measurement yields $f(M1, L1) = 1$. Replaying the event log $L2$ fails for the last two traces $ACHDFA$ and $ACDHFA$, since the model requires activity G being performed before activating task H . Therefore, in both cases one token remains in place $c6$, and one token needs to be created artificially in place $c7$ for firing transition H (i.e., $m_1 = r_1 = m_2 = r_2 = m_3 = r_3 = 0$, and $m_4 = r_4 = m_5 = r_5 = 1$). Counting the tokens being produced and consumed in the Petri net model (i.e., $c_1 = p_1 = 7$, and $c_2 = c_3 = p_2 = p_3 = 9$, and $c_4 = c_5 = p_4 = p_5 = 8$), and with the number of process instances per trace, given in Figure 1(c), the fitness can be measured as $f(M1, L2) \approx 0.995$. For the last event log $L3$ the fitness measurement yields $f(M1, L3) \approx 0.540$.

Besides measuring the degree of fitness pinpointing the site of mismatch is crucial for giving useful feedback to the analyst. In fact, the place of missing and remaining tokens during log replay can provide insight into problems, such as Figure 1(j) visualizes some diagnostic information obtained for event log $L2$. Because of the remaining tokens (whose amount is indicated by a + sign) in place $c6$ transition G has stayed enabled, and as there were tokens missing (indicated by a - sign) in place $c7$ transition H has failed seamless execution. Regarding evaluation of potential alignment procedures, there rather should be created a possibility to skip activity G in the model than considering the expert consultation missing in almost half of the high-value claims that took place; however, a final interpretation could only be given by a domain expert from the insurance company.

Note that this replay is carried out in a non-blocking way and from a log-based perspective, i.e., for each log event in the trace the corresponding transition is fired, regardless whether the path of the model is followed or not. This leads

to the fact that—in contrast to directly comparing the event streams of models and logs—a concatenation of missing log events is punished by the fitness metric f just as much as a single one, since it could always be interpreted as a missing link in the model.

As a prerequisite of conformance analysis model tasks must be associated with the logged events, which may result in *duplicate tasks*, i.e., multiple tasks that are mapped onto the same kind of log event, and *invisible tasks*, i.e., tasks that have no corresponding log event. Duplicate tasks cause no problems during log replay as long as they are not enabled at the same time and can be seamlessly executed, but otherwise one must enable and/or fire the right task for progressing properly. Invisible tasks are considered to be lazy, i.e., they are only fired if they can enable the transition in question. In both cases it is necessary to partially explore the state space of the model. A detailed description is beyond the scope of this paper but the interested reader is kindly referred to [21].

5 Measuring Appropriateness

Generally spoken, determining the degree of appropriateness of a workflow process model strongly depends on subjective perception, and is highly correlated to the specific purpose. There are aspects like the proper semantic level of abstraction, i.e., the granularity of the described workflow actions, which can only be found by an experienced human designer. The notion of appropriateness addressed by this paper rather relates to the control flow perspective and therefore is approachable to measurement but there still remains a subjective element.

The overall aim is to have the model clearly reflect the behavior observed in the log, whereas the degree of appropriateness is determined by both structural properties of the model and the behavior described by it. Figure 1(g) shows M_4 , which is a good model for the event log L_2 as it exactly generates the observed sequences in a structurally suitable way.

In the remainder of this section, both the structural and the behavioral part of appropriateness are considered in more detail.

5.1 Structural Appropriateness

The desire to model a business process in a compact and meaningful way is difficult to capture by measurement. As a first indicator we will define a simple metric that solely evaluates the size of the graph and subsequently some constructs that may inflate the structure of a process model are considered.

Metric 2 (Structural Appropriateness). *Let L be the set of labels, and N the set of nodes (i.e., places and transitions) in the Petri net model, then the structural appropriateness metric a_S is formalized as follows:*

$$a_S = \frac{|L| + 2}{|N|}$$

Given the fact that a business process model is expected to have a dedicated *Start* and *End* place, the graph must contain at least one transition for every task label, plus two places (the start and end place). In this case $|N| = |L| + 2$ and the metric a_S yields the value 1. The more the size of the graph is growing, e.g., due to additional places, the measured value moves towards 0.

Calculating the structural appropriateness for the model $M3$ yields $a_S(M3) \approx 0.170$, which is a very bad value caused by the many duplicate tasks. For the good model $M4$ the metric yields $a_S(M4) = 0.5$. With $a_S(M5) \approx 0.435$ a slightly worse value is calculated for the behaviorally (trace) equivalent model $M5$ in Figure 1(h), which is now used to consider some constructs that may decrease the structural appropriateness a_S .

(a) *Duplicate tasks.* Duplicate tasks that are used to list alternative execution sequences tend to produce models like the extreme $M3$. $M5(a)$ indicates an example situation in which a duplicate task is used to express that after performing activity C either the sequence GH or H alone can be executed. $M4(b)$ describes the same process with the help of an invisible task, which is only used for routing purposes and therefore not visible in the log. One could argue that this model supports a more suitable perception namely activity G is not obliged to execute but can be skipped, but it somehow remains a matter of taste. However, excessive usage of duplicate tasks for listing alternative paths reduces the appropriateness of a model in preventing desired abstraction. In addition, there are also duplicate tasks that are necessary to, e.g., specify a certain activity taking place exactly at the beginning and at the end of the process like task A in $M4(a)$.

(b) *Invisible tasks.* Besides the invisible tasks used for routing purposes like, e.g., shown in $M4(b)$, there are also invisible tasks that only delay visible tasks, such as the one depicted in $M5(b)$. If they do not serve any model-related purpose they can simply be removed, thus making the model more concise.

(c) *Implicit places.* Implicit places are places that can be removed without changing the behavior of the model. An example for an implicit place is given in $M5(c)$. Again, one could argue that they should be removed as they do not contribute anything, but sometimes it can be useful to insert such an implicit place to, e.g., show document flows. Note that the place $c5$ is not implicit as it influences the choice made later on between E and F . Both $c5$ and $c10$ are *silent places*, with a silent place being a place whose directly preceding transitions are never directly followed by one of their directly succeeding transitions (i.e., the model is unable to produce an event sequence containing BE or AA). Mining techniques by definition are unable to detect implicit places, and have problems detecting silent places.

5.2 Behavioral Appropriateness

Besides the structural properties that can be evaluated on the model itself appropriateness can also be examined with respect to the behavior recorded in the log. Assuming that the log fits the model, i.e., the model allows for all the

execution sequences present in the log, there remain those that would fit the model but have not been observed. Assuming further that the log satisfies some notion of completeness, i.e., the behavior observed corresponds to the behavior that should be described by the model, it is desirable to represent it as precisely as possible. When the model gets too general and allows for more behavior than necessary (like in the “flower” model $M2$) it becomes less informative in actually describing the process.

One approach to measure the amount of possible behavior is to determine the mean number of enabled transitions during log replay. This corresponds to the idea that for models clearly reflecting their behavior, i.e., complying with the structural properties mentioned, an increase of alternatives or parallelism and therefore an increase of potential behavior will result in a higher number of enabled transitions during log replay.

Metric 3 (Behavioral Appropriateness). *Let k be the number of different traces from the aggregated log. For each log trace i ($1 \leq i \leq k$) n_i is the number of process instances combined into the current trace, and x_i is the mean number of enabled transitions during log replay of the current trace (note that invisible tasks may enable succeeding labeled tasks but they are not counted themselves). Furthermore, m is the number of labeled tasks (i.e., does not include invisible tasks, and assuming $m > 1$) in the Petri net model. The behavioral appropriateness metric a_B is formalized as follows:*

$$a_B = 1 - \frac{\sum_{i=1}^k n_i(x_i - 1)}{(m - 1) \cdot \sum_{i=1}^k n_i}$$

Calculating the behavioral appropriateness with respect to event log $L2$ for the model $M2$ yields $a_B(M2, L2) = 0$, which indicates the arbitrary behavior described by it. For $M4$, which exactly allows for the behavior observed in the log, the metric yields $a_B(M4, L2) \approx 0.967$. As an example it can be compared with the model $M6$ in Figure 1(i), which additionally allows for arbitrary loops of activity G and therefore exhibits more potential behavior. This is also reflected by the behavioral appropriateness measure as it yields a slightly smaller value than for the model $M4$, namely $a_B(M6, L2) \approx 0.964$.

5.3 Balancing Fitness and Appropriateness

Having defined the three metrics f , a_S , and a_B , the question is now how to put them together. This is not an easy task since they are partly correlated with each other. So the structure of a process model may influence the fitness metric f as, e.g., due to inserting redundant invisible tasks the value of f increases because of the more tokens being produced and consumed while having the same amount of missing and remaining ones. But unlike a_S and a_B the metric f defines an optimal value 1.0, for a log that can be parsed by the model without any error.

Therefore we suggest a conformance testing approach carried out in two phases. During the first phase the fitness of the log and the model is ensured, which means that discrepancies are analyzed and potential corrective actions are

undertaken. If there still remain some tolerable deviations, the log or the model should be manually adapted to comply with the ideal or intended behavior, in order to go on with the so-called appropriateness analysis. Within this second phase the degree of suitability of the respective model in representing the process recorded in the log is determined.

Table 1. Diagnostic results

	$M1$	$M2$	$M3$	$M4$	$M5$	$M6$
$L1$	$f = 1.0$	$f = 1.0$	$f = 1.0$	$f = 1.0$	$f = 1.0$	$f = 1.0$
	$a_S = 0.5263$	$a_S = 0.7692$	$a_S = 0.1695$	$a_S = 0.5$	$a_S = 0.4348$	$a_S = 0.5556$
	$a_B = 0.9740$	$a_B = 0.0$	$a_B = 0.9739$	$a_B = 0.9718$	$a_B = 0.9749$	$a_B = 0.9703$
$L2$	$f = 0.9952$	$f = 1.0$	$f = 1.0$	$f = 1.0$	$f = 1.0$	$f = 1.0$
	$a_S = 0.5263$	$a_S = 0.7692$	$a_S = 0.1695$	$a_S = 0.5$	$a_S = 0.4348$	$a_S = 0.5556$
	$a_B = 0.9705$	$a_B = 0.0$	$a_B = 0.9745$	$a_B = 0.9669$	$a_B = 0.9706$	$a_B = 0.9637$
$L3$	$f = 0.5397$	$f = 1.0$	$f = 0.4947$	$f = 0.6003$	$f = 0.6119$	$f = 0.5830$
	$a_S = 0.5263$	$a_S = 0.7692$	$a_S = 0.1695$	$a_S = 0.5$	$a_S = 0.4348$	$a_S = 0.5556$
	$a_B = 0.8909$	$a_B = 0.0$	$a_B = 0.8798$	$a_B = 0.8904$	$a_B = 0.9026$	$a_B = 0.8894$

Regarding the example logs given in Figure 1(b)-(d) this means that we only evaluate the appropriateness measures of those models having a fitness value $f = 1.0$ (cf. Table 1) and therefore completely discard event log $L3$, which only fits the trivial model $M2$, and the model $M1$ for event log $L2$. For event log $L1$ and $L2$ we now want to find the most adequate process model among the remaining ones, respectively. Given the fact that neither the structural appropriateness metric a_S nor the behavioral appropriateness metric a_B defines an optimal point (note that for the process model $M2$ the a_S value is very high while the a_B value is very low and vice versa for the other extreme model $M3$) they both must be understood as an indicator to be maximized without decreasing the other. A possible outcome of such a qualitative analysis could be that $M1$ is selected for $L1$ while $M4$ is selected for $L2$. A more in-depth evaluation can be found in a technical report on conformance testing [21].

6 Adding Conformance to the ProM Framework

The main concepts discussed in this paper have been implemented in a plug-in for the ProM Framework. The conformance checker replays an event log within a Petri net model in a non-blocking way while gathering diagnostic information that can be accessed afterwards. It calculates the token-based fitness metric f , taking into account the number of process instances represented by each logical log trace, the structural appropriateness a_S , and the behavioral appropriateness a_B . Furthermore, the diagnostic results can be visualized from both a log-based and model-based perspective.

During log replay the plug-in takes care of invisible tasks that might enable the transition to be replayed next, and it is able to deal with duplicate tasks.

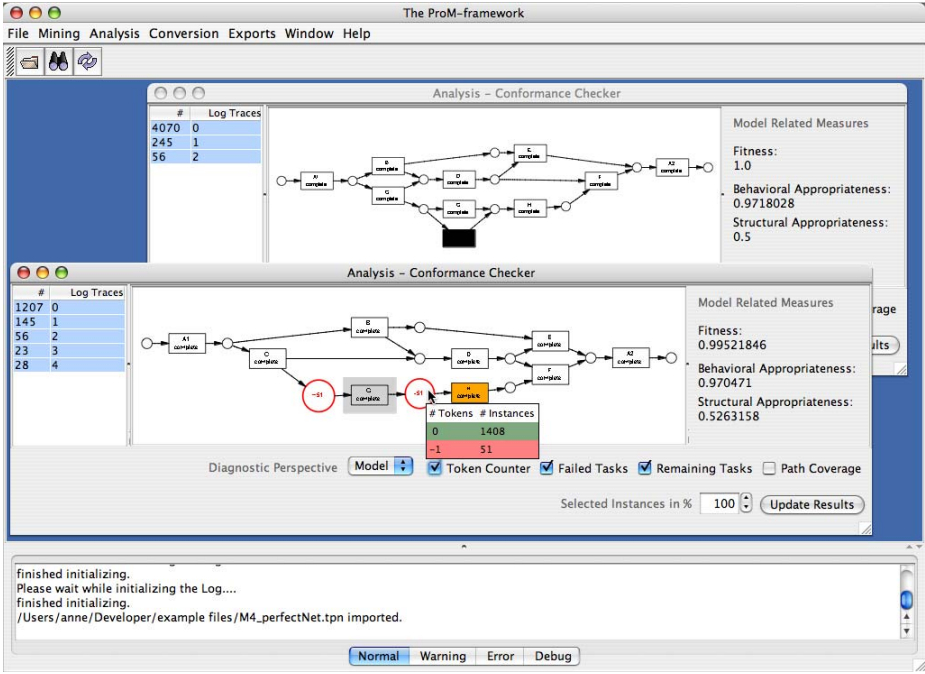


Fig. 2. Screenshot of the conformance analysis plug-in

The lower part of Figure 2 shows the result screen of analyzing the conformance of event log L_2 and process model M_1 . As discussed before, for replaying L_2 the model lacks the possibility to skip activity G , which also becomes clear in the visualization of the model augmented with diagnostic information. In the other window the process specification M_4 is measured to fit with event log L_1 .

7 Related Work

The work reported in this paper is closely related to earlier work on process mining, i.e., discovering a process model based on some event log. The idea of applying process mining in the context of workflow management was first introduced in [9]. Cook and Wolf have investigated similar issues in the context of software engineering processes using different approaches [11]. Herbst and Karagiannis also address the issue of process mining in the context of workflow management using an inductive approach [17]. They use stochastic task graphs as an intermediate representation and generate a workflow model described in the ADONIS modeling language. Then there are several variants of the α algorithm [8,25]. In [8] it is shown that this algorithm can be proven to be correct for a large class of processes. In [25] a heuristic approach using rather simple metrics is used to construct so-called “dependency/frequency tables” and “dependency/frequency graphs”. This is used as input for the α algorithm. As a

result it is possible to tackle the problem of noise. For more information on process mining we refer to a special issue of *Computers in Industry* on process mining [7] and a survey paper [6]. Given the scope of this paper, we are unable to provide a complete listing of the many papers published in recent years.

The work of Cook et al. [12,10] is closely related to this paper. In [12] the concept of process validation is introduced. It assumes an event stream coming from the model and an event stream coming from real-life observations, both streams are compared. Here the time-complexity is problematic as the state-space of the model needs to be explored. In [10] the results are extended to include time aspects. The notion of conformance has also been discussed in the context of security [3], business alignment [1], and genetic mining [4]. However, in each of the papers mentioned only fitness is considered and appropriateness is mostly ignored. (Note that more recent work on genetic mining also includes “penalties” for “too much behavior” [19].) In [14] the process mining problem is faced with the aim of deriving a model which is as compliant as possible with the log data, accounting for fitness (called completeness) and also behavioral appropriateness (called soundness). In [24] case-based reasoning is applied to explicitly record information about non-compliant cases, which can be re-used for potential adaptations of the business process model.

Process mining and conformance testing can be seen in the broader context of Business (Process) Intelligence (BPI) and Business Activity Monitoring (BAM). In [15,16] a BPI tool set on top of HP’s Process Manager is described. The BPI tool suite includes a so-called “BPI Process Mining Engine”. In [20] Zur Muehlen describes the PISA tool which can be used to extract performance metrics from workflow logs. Similar diagnostics are provided by the ARIS Process Performance Manager (PPM). The latter tool is commercially available and a customized version of PPM is the Staffware Process Monitor (SPM), which is tailored towards mining Staffware logs. Note that none of the latter tools is supporting conformance testing. The focus of these tools is often on performance measurements rather than monitoring (un)desirable behavior.

8 Conclusion

Given the presence of both process models and event logs in most organizations of some complexity, it is interesting to investigate the notion of conformance as it has been defined in this paper. Conformance is an important notion in the context of business alignment, auditing (cf. Sarbanes-Oxley (SOX) Act [22]), and business process improvement. Therefore, the question “Does the event log *conform* to the process model and vice versa?” is highly relevant.

We have shown that conformance has two dimensions: fitness and appropriateness. Fitness can be captured in one metric (f). For measuring appropriateness we introduced two metrics: structural appropriateness a_S and behavioral appropriateness a_B . Together these three metrics allow for the quantification of conformance. The metrics defined in this paper are supported by the *Conformance Checker*, a tool which has been implemented within the ProM Framework.

An interesting direction for future research is to exploit log data on a more fine-grained level (e.g., stating the *start* and *end* of activities) and to include other perspectives such as time, data, and resources. For example, in some application the timing of an event is as important as its occurrence.

Acknowledgements

The authors would like to thank Ton Weijters, Boudewijn van Dongen, Ana Karla Alves de Medeiros, Minseok Song, Laura Maruster, Eric Verbeek, Monique Jansen-Vullers, Hajo Reijers, Michael Rosemann, Huub de Beer, Peter van den Brand, et al. for their on-going work on process mining techniques.

References

1. W.M.P. van der Aalst. Business Alignment: Using Process Mining as a Tool for Delta Analysis. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the 5th Workshop on Business Process Modeling, Development and Support (BPMDS'04)*, volume 2 of *Caise'04 Workshops*, pages 138–145. Riga Technical University, Latvia, 2004.
2. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
3. W.M.P. van der Aalst and A.K.A. de Medeiros. Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance. In N. Busi, R. Gorrieri, and F. Martinelli, editors, *Second International Workshop on Security Issues with Petri Nets and other Computational Models (WISP 2004)*, pages 69–84. STAR, Servizio Tipografico Area della Ricerca, CNR Pisa, Italy, 2004.
4. W.M.P. van der Aalst, A.K.A. de Medeiros, and A.J.M.M. Weijters. Genetic Process Mining. In G. Ciardo and P. Darondeau, editors, *26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005)*, volume 3536 of *Lecture Notes in Computer Science*, pages 48–69, 2005.
5. W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering Interaction Patterns in Business Processes. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 244–260. Springer-Verlag, Berlin, 2004.
6. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
7. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
8. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
9. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.

10. J.E. Cook, C. He, and C. Ma. Measuring Behavioral Correspondence to a Timed Concurrent Model. In *Proceedings of the 2001 International Conference on Software Maintenance*, pages 332–341, 2001.
11. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
12. J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. *ACM Transactions on Software Engineering and Methodology*, 8(2):147–176, 1999.
13. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.
14. G. Greco, A. Guzzo, L. Pontieri, and D. Saccá. Mining Expressive Process Models by Clustering Workflow Traces. In *Proc of Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference (PAKDD 2004)*, pages 52–62, 2004.
15. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004.
16. D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In P. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 159–168. Morgan Kaufmann, 2001.
17. J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.
18. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley, Reading MA, 1998.
19. A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic Process Mining: A Basic Approach and its Challenges. In M. Castellanos and T. Weijters, editors, *First International Workshop on Business Process Intelligence (BPI'05)*, pages 46–57, Nancy, France, September 2005.
20. M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
21. A. Rozinat and W.M.P. van der Aalst. Conformance Testing: Measuring the Alignment Between Event Logs and Process Models. BETA Working Paper Series, WP 144, Eindhoven University of Technology, Eindhoven, 2005.
22. P. Sarbanes, G. Oxley, and et al. Sarbanes-Oxley Act of 2002, 2002.
23. A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.
24. B. Weber, M. Reichert, S. Rinderle, and W. Wild. Towards a Framework for the Agile Mining of Business Processes. In M. Castellanos and T. Weijters, editors, *First International Workshop on Business Process Intelligence (BPI'05)*, pages 36–45, Nancy, France, September 2005.
25. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

Mining Staff Assignment Rules from Event-Based Data

Linh Thao Ly¹, Stefanie Rinderle¹, Peter Dadam¹, and Manfred Reichert²

¹ Dept. DBIS, University of Ulm, Germany
{thao.ly, rinderle, dadam}@informatik.uni-ulm.de
² IS Group, University of Twente, The Netherlands
m.u.reichert@cs.utwente.nl

Abstract. Process mining offers methods and techniques for capturing process behaviour from log data of past process executions. Although many promising approaches on mining the control flow have been published, no attempt has been made to mine the staff assignment situation of business processes. In this paper, we introduce the problem of mining staff assignment rules using history data and organisational information (e.g., an organisational model) as input. We show that this task can be considered an inductive learning problem and adapt a decision tree learning approach to derive staff assignment rules. In contrast to rules acquired by traditional techniques (e.g., questionnaires) the thus derived rules are objective and show the staff assignment situation at hand. Therefore, they can help to better understand the process. Moreover, the rules can be used as input for further analysis, e.g., workload balance analysis or delta analysis. This paper presents the current state of our work and points out some challenges for future research.

1 Introduction

While great effort has been spent on researching the control flow aspect of business processes, organisational aspects of processes have often been neglected. In particular, the link between the process and the organisational elements is less understood [1]. However, in order to fully understand a business process, it is also necessary to know by whom the activities of the process are performed. This especially applies when *Workflow Management Systems* (WfMS) should be employed in order to support the process execution. In WfMSs rules which are based on organisational concepts, e.g., roles, are often used to assign work items to staff members (staff assignment rules). Staff assignment rules define, to a certain extent, the profile of agents capable of or eligible for performing an activity. For example, for performing the activity “create bills” agents have to possess the role “book-keeper” and additionally need to have “computer skills”. Properties not referred to in the staff assignment rule have don’t-care semantics. An agent might have those properties or not.

The traditional way of acquiring staff assignment rules (or process knowledge in general) is by means of questionnaires or interviews. However, these techniques are very cost-intense and error-prone. Furthermore, the results acquired

by applying traditional techniques are rather subjective and need not necessarily reflect the staff assignment situation at hand. Therefore, it makes sense to support the process engineer in acquiring staff assignment rules by providing objective rules as a complement to the results acquired by traditional methods.

In this paper, we introduce the task of deriving staff assignment rules from log data of past process executions and organisational information. We denote this as *staff assignment mining*.

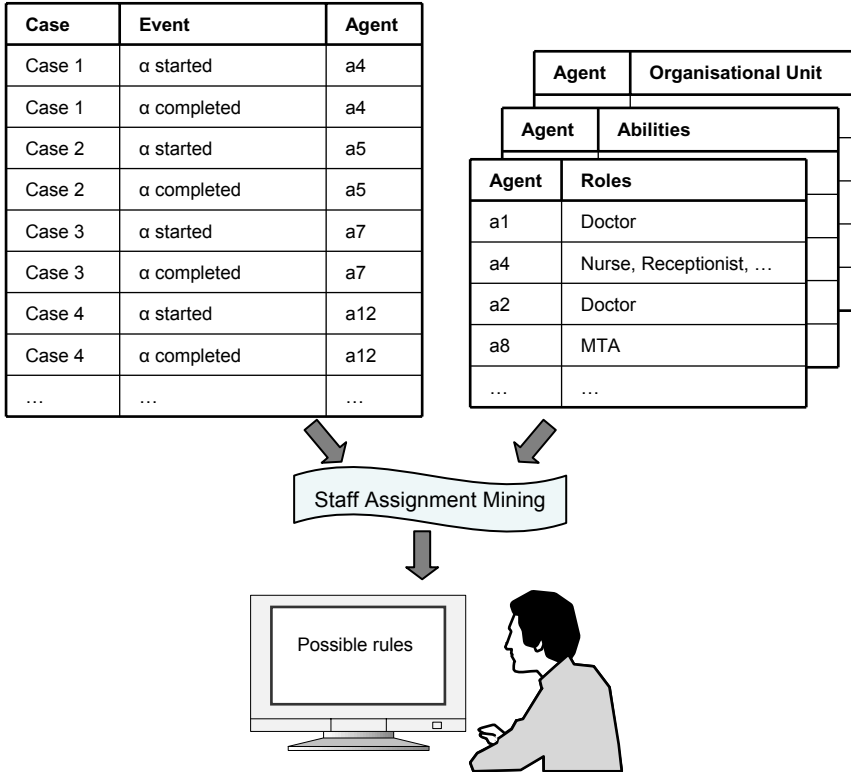


Fig. 1. The process engineer is supported by proposing a set of possible staff assignment rules for a given activity

Workflow Management Systems but also other process-oriented systems, e.g., *Enterprise Resource Planning Systems* (ERP) like SAP, log all events which occur while a process instance (a *case*) is executed. The log data, also called audit trail or history data, typically contain information about the start and the end of an activity but also about the agent who performed the activity. By combining this log data with organisational information (for instance, from an organisational model), e.g., the roles that staff members have, objective staff assignment rules can be derived (cf. Fig. 1).

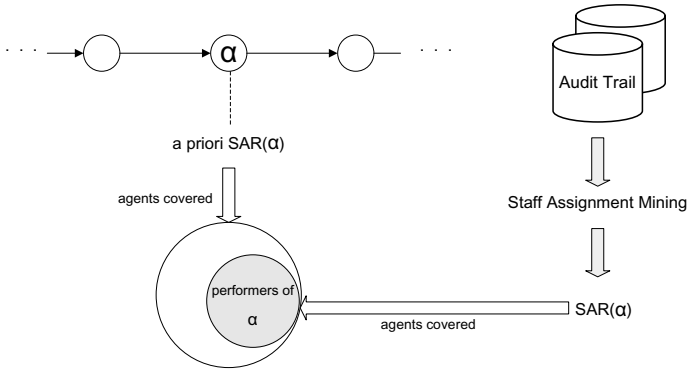


Fig. 2. Using staff assignment mining for delta-analysis

Fig. 1 shows some log data of the execution of an example activity α in different cases. Abstracting from concrete events we will refer to a performer of an activity as an agent who started and completed an instance of the activity.

By using organisational concepts (e.g., roles) for separating the performers of the activity from the non-performers¹ meaningful staff assignment rules can be derived. If we for instance find out that all performers of the example activity α , e.g., a4, a5 etc., have the role “doctor” while all non-performers do not, it is likely that the role “doctor” is a key property for performing α . Thus, the staff assignment rule for α could demand that all agents need to possess the role “doctor”.

Our objective is to derive staff assignment rules for a given activity such that:

- the rules are consistent to the audit trail data
- the rules identify the actual set of performers of the activity
- the rules are general such that they cover the essential profile of the performers

The derived staff assignment rules reveal the actual profile of the performers and thus, reveal the staff assignment situation at hand. Therefore, they can be used as input for further analysis, particularly delta-analysis (cf. Fig. 2). If an a priori staff assignment rule of the example activity α (SAR(α)) in Fig. 2 is, for instance, more general than the rule derived, this indicates that only a subset of the agents identified by the a priori rule really performed instances of α . This might indicate that the work item is delivered to work queues of staff members who never performed the activity. This, of course, can be intended. However, it might also indicate that the staff assignment situation has changed and that the a priori rule is obsolete. Besides a better understanding of the process behaviour knowing the actual staff assignment situation at hand also allows for incrementally defining staff assignment rules.

This paper is organised as follows. After addressing related work in the process mining context in Sect. 2 we will refer to the problem of learning staff

¹ Agents who did not perform the activity.

assignment rules. For this purpose an organisational meta-model and an appropriate representation of staff assignment rules are introduced in Sect. 3. Then, the problem of learning staff assignment rules is addressed in Sect. 4. Finally, Sect. 5 concludes the paper.

2 The Process Mining Context

Besides staff assignment rules, many information can be derived from the audit trail data of process executions. *Process mining* deals with developing methods and techniques to capture the process behaviour from audit trail data. In particular, a structured process description can be derived using *process mining* techniques. The derived process model can be used as input for further analysis. For instance, delta-analysis can be applied to detect discrepancies between the a priori and the derived process model.

Many papers on *process mining* have been published recently (e.g., [6, 7, 10, 12, 14, 20, 11]). Most of them focus on mining the control flow (*control flow mining*). Only few papers consider organisational aspects. For a survey on *process mining* approaches the reader is referred to [3, 2].

Approaches integrating organisational aspects can be divided into two categories. The first category concentrates on relations between agents involved in the process [4, 5]. The second category focuses on the relations between a process and organisational concepts. Our approach presented in this paper falls in the second category.

In [4, 5] van der Aalst and Song introduce an approach for mining social networks from log data. The authors define four categories of metrics expressing potential relationships between agents (e.g., metrics based on joint activities). Using these metrics sociograms are derived which can be further used for *social network analysis*. This work can be considered an important contribution to *enterprise social networks analysis*.

The authors also mentioned the possibility of “guessing” organisational structures, in particular, guessing roles of agents. Agents performing the same activities are assigned the same roles. However, in [4, 5], van der Aalst and Song do not consider the use of additional organisational information in this context. In addition, it seems that this was just a suggestion since no further work on this aspect has been published. At our best knowledge, no other work on mining the relations between the process and the organisation is available to date.

Staff assignment mining is a novel facet of *process mining* and can be smoothly integrated in the mining process. We consider our approach a complement to current process mining efforts.

3 The Organisational Meta-model

As a starting point, we use a simple but yet powerful organisational meta-model to describe organisational concepts (cf. Fig. 3). However, the approach presented in this paper is not restricted the meta-model and the organisational

concepts presented here. In fact, our approach can be directly applied for other organisational meta-models and further organisational concepts as well. In order to present our approach, however, a meta-model is needed to specify the organisational concepts used. The meta-model uses the following organisational

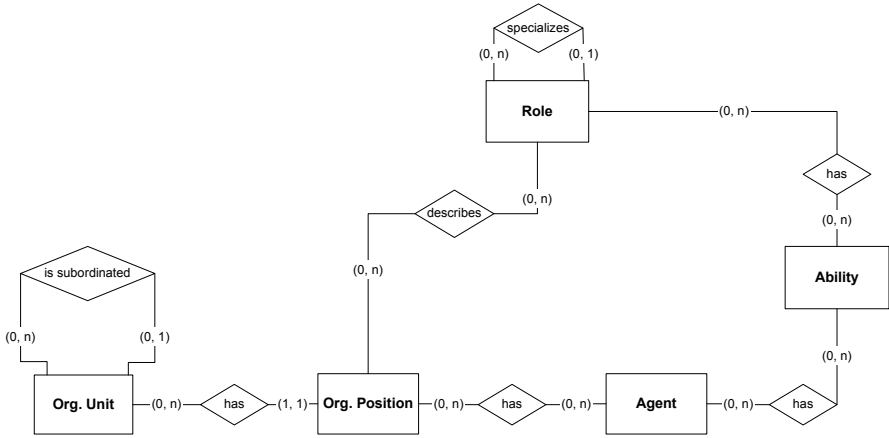


Fig. 3. The organisational meta-model used

concepts: agents, organisational units, roles, abilities, and organisational positions. Since the latter one is considered to be assigned to only one agent (except for time-sharing aspects) we neither consider agent objects nor organisational positions when deriving rules, since those rules would not represent a general profile. Abilities can be assigned to agents directly or indirectly via roles. Being assigned a certain role an agent also has all abilities, i.e. capabilities and privileges, associated with that role. For example, the role “receptionist” has the ability “computer skills”. Organisational positions, e.g., “1st book-keeper”, can be interpreted as an instantiation of a set of roles. Due to space limitations we cannot go into detail on the meta-model. For further information the reader is referred to [18]. Table 1 shows an example of an organisation model based on this meta-model. We will refer to this example later.

Based on the organisational meta-model, staff assignment rules (abbr.: SAR) can refer to organisational entities in a manner similar to disjunctive normal forms (DNF) in order to define the profile of the performers. A SAR of the example activity α ($SAR(\alpha)$) is given below, where a certain role (ability) is specified by R (A). This rule would identify the agents a4, a5, a7, a11 and a12 from Tab. 1.

SAR(α):
 (R = 'receptionist' AND A = 'english')
 OR
 (R = 'receptionist' AND A = 'french')

Table 1. Example of an organisational model. The first part lists agents and respective organisational entity, position and roles (MTA stands for medical-technical assistant). The second part lists agents and respective abilities. Abilities directly assigned to the agent are marked with an asterisk.

Agent	Org. unit	Org. position	Roles
A1	Clinical Centre	1st Doctor	Doctor
A2	Clinical Centre	2nd Doctor	Doctor
A3	Clinical Centre	3rd Doctor	Doctor
A4	Clinical Centre	1st Nurse	Nurse, Receptionist, Book-keeper
A5	Clinical Centre	2nd Nurse	Nurse, Receptionist, Book-keeper
A6	Clinical Centre	3rd Nurse	Nurse, Receptionist, Book-keeper
A7	Clinical Centre	4th Nurse	Nurse, Receptionist, Book-keeper
A8	Clinical Centre	1st MTA	MTA
A9	Clinical Centre	2nd MTA	MTA
A10	Clinical Centre	3rd MTA	MTA
A11	Clinical Centre	1st Secretary	Secretary, Receptionist, Book-keeper
A12	Clinical Centre	2nd. Secretary	Secretary, Receptionist, Book-keeper

Agent	Abilities
A1	Computer skills*, Take blood sample, Issue prescription, English*
A2	Computer skills*, Take blood sample, Issue prescription
A3	Computer skills*, Take blood sample, Issue prescription, English*
A4	Computer skills, Take blood sample, English*, French*
A5	Computer skills, Take blood sample, English*
A6	Computer skills, Take blood sample
A7	Computer skills, Take blood sample, French*
A8	English*
A9	English*
A10	
A11	Computer skills, French*
A12	Computer skills, English*

Note that it is also possible to use negative qualifications by using NOT, in the sense of demanding that an agent must not have certain properties. If an agent is not related to an organisational entity, then he is considered to have negative qualifications concerning these entities. Though organisational entities may have many attributes (for example, the ability “english” may have the attribute “level” with values ranging from “beginner” to “expert”) we abstract from attributes other than the name of the entity.

Rules in the form described above can be used to define the profile of appropriate performers of a workflow activity but also to define access rules or constraints for any kind of information system and objects (e.g., to control the access to electronic documents) [19, 22].

4 Learning Staff Assignment Rules

In this section, we present our approach for deriving meaningful staff assignment rules using audit trail data and organisational information based on the meta-model described before.

4.1 Decision Tree Learning

Since staff assignment rules are supposed to identify the performers of a given activity, the question is to determine combinations of organisational properties that distinguish performers from non-performers. Thus, the problem of deriving staff assignment rules can be interpreted as an inductive learning task, particularly learning from positive and negative examples. Unlike with *control flow mining*, negative examples are directly given for our problem: every non-performer can serve as a negative example. First, we define the notion of positive and negative examples for this learning problem.

Definition 1 (Positive/Negative Examples). *Let A be a set of agents, and let X be the total set of activities. Then $performer(x,a)$ is a classification function which determines whether a given agent $a \in A$ has worked on any instance of activity $x \in X$ or not:*

$$performer : X \times A \rightarrow \{\text{True}, \text{False}\}$$

$$performer(x,a) = \begin{cases} \text{True} & \text{if } a \text{ has performed an instance of } x \\ \text{False} & \text{otherwise} \end{cases}$$

An 'example' is a triple $(x,a,performer(x,a))$. We further distinguish between positive examples, i.e., (x,a,True) , and negative examples, i.e., (x,a,False) . Note that due to this definition, agents performing x multiple times will be associated with a respective number of examples. For every non-performer a negative example can be generated.

Table 2 shows a set of examples referring to the agents from our organisational model depicted in Tab. 1 and our example activity α (cf. Fig. 1). Since we refer to α , our running example throughout this paper, the activity information is omitted in Tab. 2.

Based on the examples the objective is to derive rules which approximates the classification function $performer$. This problem belongs to *supervised learning* [13] since we have predefined classes (performers and non-performers).

Various learning methods can be applied to solve this learning problem. We have chosen to adapt *decision tree learning* [9]. *Decision tree learning* is one of the most widely-used methods of inductive inference. It can be employed for attribute-based learning of disjunctive concepts. This method is simple and explicitly facilitates graphical representations. This constitutes an advantage when developing a user-friendly graphical interface for a respective staff assignment mining tool. Furthermore, decision tree learning also incorporates methods for

Table 2. A set of examples. The agents a1, a2, a6, a8, a9, and a10 did not perform α while the agents a4, a5, a7, a11, and a12 did.

Agent a	performer(α ,a)
a1	False
a2	False
a3	False
a4	True
a5	True
a6	False
a7	True
a8	False
a9	False
a10	False
a11	True
a12	True

handling noise data and continuous attribute values. Continuous attribute values do not occur with our preliminaries. However, this will be an important feature when we will extend our approach to consider attributes of organisational entities as well.

Staff assignment rules can be derived by growing decision trees (cf. Fig. 4). Starting at the root node, an organisational entity is chosen as testing attribute in order to separate the positive from the negative examples. In Fig. 4 $R = \text{'receptionist'}$ was chosen as the first attribute. (Which attributes are chosen and in which order is discussed in the following.) Depending on whether they are related to an organisational entity, examples (i.e., agents) are assigned to the “yes”-child-node or “no”-child-node, respectively. Note that for every agent, it can be determined whether the agent is related to an organisational entity or not. This procedure is repeated recursively for the child nodes until only examples from one class, indicated by the ‘+’ and the ‘-’ set in Fig. 4, are left, or there are no attributes left. The ‘+’ set represents the class of performers while the ‘-’ set represents the class of non-performers.

All entities of an organisational model can be used as testing attributes. For example, the set of possible attributes shown below can be derived from the organisational model described in Tab. 1. A certain organisational unit is specified by OU.

```
{OU = 'clinical centre', R = 'nurse', R = 'doctor',
  R = 'receptionist', ...}
```

From a decision tree if-then-rules or rules in DNF can be easily derived. The conjunction of attribute values of a path from a leaf-node with the target class to the root represents the if-part of the if-then-rule or a disjunction element of the DNF.

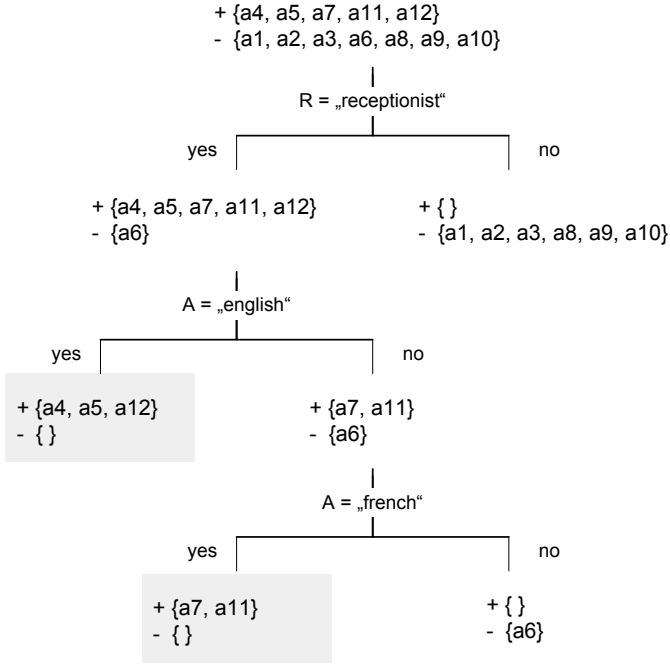


Fig. 4. A decision tree for the example set for activity α from Tab. 2. From this decision tree the rule $\text{SAR}(\alpha)$: $(R = \text{'receptionist'} \text{ AND } A = \text{'english'}) \text{ OR } (R = \text{'receptionist'} \text{ AND } A = \text{'french'})$ can be derived.

However, our objective is to mine general profiles of performers with as less conjunction elements as possible. Finding decision trees representing minimal rules is of NP-hard complexity [16]. Therefore, a greedy search strategy using the metrics *information gain* [17, 16] for guiding the search, i.e. choosing an attribute, is applied. The *information gain* metrics is based on entropy calculations. The formulas for calculating the entropy and *information gain* are given below. S denotes an example set, a an attribute, and p_+ and p_- indicate the proportion of positive and negative examples respectively. S_{yes} and S_{no} are the example sets assigned to the “yes”- or the “no”-child of the node belonging to S , respectively.

The entropy is a metrics for the homogeneity of a set. At each separation step the attribute with the best *information gain* value is chosen. Thus, the decision tree algorithm tries to achieve the best split of the remaining example set in every step.

$$\text{entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad (1)$$

$$\text{information gain}(S, a) = \text{entropy}(S) - \frac{|S_{yes}|}{|S|} \text{entropy}(S_{yes}) - \frac{|S_{no}|}{|S|} \text{entropy}(S_{no}) \quad (2)$$

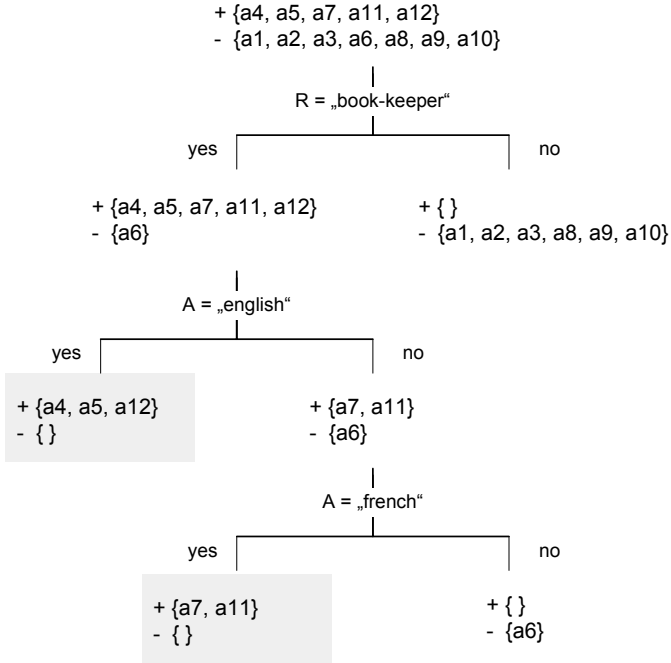


Fig. 5. An alternative decision tree for the example data. From this decision tree the rule SAR(α): R = ('book-keeper' AND A = 'english') OR (R = 'book-keeper' AND A = 'french') can be derived.

The decision tree in Fig. 4 was generated using *information gain*. For further information on decision trees and metrics please refer to [17, 16, 15].

Generally, more than one decision tree can often be derived which fit the input data. This also applies to our example set from Tab. 2. Therefore, it is important to offer a set of potential rules to the process engineer. The process engineer can then, after an evaluation process, decide which of the rules are useful. In order to extract more than one rule, backtracking is needed. Again, *information gain* can be used for choosing the suitable attributes. Instead of using only the best separating attribute, the *k*-best attributes can be used, whereas *k* is a configurable parameter. Figure 5 shows an alternative decision tree using the second best testing attribute (R= 'book-keeper') at root level. Note that besides the trees shown here even more decision trees can be derived from our example set by choosing another *k* and/or allowing backtracking on other than the root level.

4.2 Advanced Issues

Attributes, i.e. organisational entities, are, with regard to the given organisational meta-model, not necessarily independent. In fact, attributes can be related

to each other in different ways. Organisational units can consist of other units. Thus, any agent in the sub-unit also belongs to the superordinate unit. Furthermore, roles can be in a specialisation/generalisation relationship to other roles. For example, the role “nurse” can have the role “lead nurse” as specialisation, which inherits all privileges and abilities of the role “nurse”. Thus, having the role “lead nurse” directly implies having the role “nurse”.

Another case of dependent attributes occurs with roles and abilities. Roles can imply abilities but not vice versa. Since every organisational entity is represented by an attribute, those dependencies need not be handled separately. However, when an attribute is selected for separation, all attributes implied by it need not be used as testing attributes because they would not achieve a further separation. Therefore, these attributes can be excluded from the set of remaining attributes for this path. This helps reducing the amount of attributes to test.

Moreover, it will typically be possible to confine the set of relevant organisational entities as well as relevant examples in advance. On the one hand, it is often possible to exclude certain organisational entities. For the activity “create bills”, for instance, the ability “take a blood sample” is fairly uninteresting. Therefore, the ability “take a blood sample” can be excluded from the set of testing attributes since staff assignment rules based on this attribute would not make sense anyway.

On the other hand, a basic set of qualifications necessary for performing an activity is often already known in advance. For example, for activity “examine the patient” the role “doctor” is required. Agents, who do not have the required qualifications, can be excluded from the example set. This helps reducing the amount of examples. The task of excluding attributes in advance or selecting qualifications, which agents need to possess, should be performed by the process engineer.

4.3 Dealing with Noise

“Perfect” process executions and perfect log data as in the previous examples, however, cannot be taken for granted. Hence, we also have to deal with exceptional cases and noise data. Exceptional cases are considered to be cases, where agents perform the activity although they are not eligible to do so, e.g., as a replacement. Replacement performers do not necessarily have the profile of regular performers. Therefore, the decision tree might not reveal the profile of regular performers.

Noise data occur when, for instance, a wrong agent is logged as the performer of the activity. In order to account for those cases, threshold values are introduced. Concerning performers, the frequency of their occurrence in the example set can be used as an indication of whether they are regular performers or not. If agent a4, for instance, executed activity α twice while all other agents who executed α did so a lot more often, this indicates that agent a4 is not a regular performer of activity α . Thus, performers executing α less often than a given threshold value can be removed from the example set in advance. Threshold values can also be used for post-pruning the decision tree. For example, nodes

where the proportion of positive examples is less than a threshold value can be transformed into a leaf-node.

Since it is our objective to identify the actual performer set, pruning the tree based on negative examples should mainly be used in order to account for minor errors of the organisational model, e.g., non-performer agents were assigned spurious properties making them more difficult to separate from the positive examples. In a post-pruning operation nodes where the amount of negative examples is less than a threshold value can be transformed into a leaf-node. For further information on pruning decision trees the interested reader is referred to [17, 9].

In contrast to *control flow mining* we use organisational information (organisational model) as input data, in addition to the audit trail data. Thus, the quality of the derived rules highly depends on the quality of the organisational model. As aforementioned, minor errors in the organisational model can be compensated using threshold values. Nevertheless, the organisational model needs to be complete, in the sense of that all relevant organisational entities are modelled. Mistakes or incompleteness of the organisational model may lead to less meaningful rules. However, the derived rules at least reveal the actual situation.

5 Conclusion and Outlook

In this paper, we concentrated on a new aspect of *process mining*: mining staff assignment rules. We have shown that the problem of deriving staff assignment rules using information from audit trail data and organisational information (e.g., an organisational model) as input can be interpreted as an inductive learning problem. Therefore, machine learning techniques can be adapted in order to solve the problem. In particular, we have used decision tree learning to derive meaningful staff assignment rules. Thus, it is possible to provide staff assignment information about activities enabling a better understanding of the underlying process.

However, enhancements and alternative learning methods have to be considered. Instead of using *information gain* as the metrics for guiding the search, another metrics, which prefers positive qualifications of positive examples, can be applied. This may lead to better results since performers' profiles are typically defined by positive rather than negative properties. Furthermore, another way of dealing with dependent attributes may also be considered, e.g., by incorporating a reasoning-component. In addition, alternative learning methods are interesting subjects of study. In particular, inductive logic programming [21] and mining association rules [13] seem to be interesting in this context. Alternative learning techniques will be an important subject for future research.

Besides possible enhancements of the mining procedure itself, many other interesting questions concerning *staff assignment mining* have arisen, e.g., dealing with dependent staff assignment rules (i.e., the performer working on activity d should be always the same as the one who worked on a preceding activity c), and combining different mining perspectives (e.g., for credit amounts greater than

50000 € other agents are needed). Furthermore, for comprehensively supporting the process engineer, validation features for staff assignment rules also seem very useful in a respective tool.

We are currently working on an implementation of our approach as a plug-in for the ProM framework² [8]. ProM is a *process mining* tool where particular approaches can be incorporated as plug-ins. For further information on ProM, please refer to [8].

Though tests on real data sets are still required, we believe that the first steps are taken to mine the relations between the process and organisational structures.

References

1. zur Mühlen, M.: Organizational Management in Workflow Applications. In: Information Technology and Management, 5(3-4) (2004), 271–291
2. v.d. Aalst, W., Weijters, A.: Process Mining: A Research Agenda. In: Computers in Industry, 53(3) (2004), 231–244
3. v. d. Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow mining: A survey of issues and approaches. In: Data and Knowledge Engineering, 47(2) (2003), 237–267
4. v. d. Aalst, W., Song, M.: Mining Social Networks: Uncovering Interaction Patterns in Business Processes. In: BPM'04, Potsdam. Lecture Notes in Computer Science, Vol. 3080 (2004), 244–260
5. v. d. Aalst, W., Song, M.: Discovering Social Networks from Event Logs. BETA Working Paper Series, WP 116, Eindhoven University of Technology, The Netherlands (2004)
6. v. d. Aalst, W., Weijters, A., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. In: IEEE Transactions on Knowledge and Data Engineering, Vol. 16(9) (2004), 1128–1142
7. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.: Genetic Process Mining. In: 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005). Lecture Notes in Computer Science, Vol. 3536. Springer Verlag (2005), 48–69
8. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds): 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005). Lecture Notes in Computer Science, Vol. 3536 (2005), 444–454
9. Breslow, L., Aha, D.: Simplifying decision trees: a survey. In: Knowledge Engineering Review, 12(1) (1997), 1–40
10. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Mining Expressive Process Models by Clustering Workflow Traces. In: PAKDD, (2004), 52–62
11. Cook, J., Wolf, L.: Discovering Models of Software Processes from Event-Based Data. In: ACM Transactions on Software Engineering and Methodology, 7(3) (1998), 215–249

² ProM can be obtained from the web: www.processmining.org

12. Golani, M., Pinter, S.: Generating a Process Model from a Process Audit Log. In: Business Process Management. Lecture Notes in Computer Science, Vol. 2678 (2003), 136–151
13. Hand, D., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press (2001)
14. Herbst, J.: A Machine Learning Approach to Workflow Management. In: ECML, (2000), 183–194
15. Mitchell, T.: Machine Learning. McGraw-Hill (1997)
16. Quinlan, J. R.: C4.5: Programs for Machine Learning Morgan Kaufmann Publishers, Inc. (1993)
17. Quinlan, J. R.: Learning Decision Tree Classifiers. In: ACM Computing Surveys, 28(1) (1996), 71–72
18. Reichert, M.; Wiedemuth–Catrinescu, U.; Rinderle, S.: Evolution of Access Control in Information Systems. In: Proc. Conf. EGP'04, Klagenfurt (2004), 100–114 (in German)
19. Rinderle, S.; Reichert, M.: On the Controlled Evolution of Access Rules in Information Systems. In: Proc. 13th Int'l Conf. on Cooperative Information Systems (CoopIS'05), Agia Napa, Cyprus (2005) (to appear)
20. Schimm, G.: Mining Exact Models of Concurrent Workflows. In: Process/Workflow Mining, Computers in Industry, 53 (3), Elsevier (2004), 265–281
21. Lavrac, N., Dzeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York 53 (1994)
22. Weber, B., Reichert, M., Wild, W., Rinderle, S.: Balancing Flexibility and Security in Adaptive Process Management Systems. In: Proc. 13th Int'l Conf. on Cooperative Information Systems (CoopIS'05), Agia Napa, Cyprus (2005) (to appear)

Towards a Framework for the Agile Mining of Business Processes

Barbara Weber¹, Manfred Reichert², Stefanie Rinderle³, and Werner Wild⁴

¹ Quality Engineering Research Group, University of Innsbruck, Austria
`Barbara.Weber@uibk.ac.at`

² Information Systems Group, University of Twente, The Netherlands
`m.u.reichert@cs.utwente.nl`

³ Dept. Databases and Information Systems, University of Ulm, Germany
`rinderle@informatik.uni-ulm.de`

⁴ Evolution Consulting, Innsbruck, Austria
`werner.wild@evolution.at`

Abstract. In order to support business processes effectively, their implementation by a process management systems (PMS) must be as close to the real world's processes as possible. Generally, it is not sufficient to analyze and model a business process only once, and then to handle respective business cases according to the defined model for a long period of time. Instead, process implementations must be quickly adaptable to changing needs. A PMS should enable process instance changes and provide facilities for analyzing these instance-specific changes in order to derive optimized process models. In this paper we introduce a framework for the agile mining of business processes which supports the whole process life cycle in an integrated way. Our framework is based on process mining techniques, adaptive process management, and conversational case-based reasoning. On the one hand, it allows annotating execution and change logs with semantical information to gather information about the reasons for ad-hoc deviations, which can then be analyzed by the process engineer (with support from the PMS). On the other hand, it enables the process engineer to adapt process models based on the outcome of these analyses and to migrate related process instances to the new model.

1 Introduction

Companies are developing a growing interest in aligning their information systems in a process-oriented way. Recently, process mining, in particular Delta analysis [1,2], has been proposed to improve business alignment [3]. If no process models are available yet, process mining techniques can be applied to identify repetitive process fragments. However, if the business processes are already captured in process models, Delta analysis can help to detect discrepancies between the modeled and the observed execution behavior of a process. Though process execution logs can be used to reveal malfunctions or bottlenecks, they do not provide any semantical information about the reasons for the observed

discrepancies. In respect to the optimization of the process models these logs therefore provide only limited information to process engineers. Furthermore, current PMS do not sufficiently support process engineers to incorporate the results of a Delta analysis into an improved process model and to smoothly migrate running process instance to the new model version.

Obviously, the practical benefit of process mining depends on the quality of the available log data. In PMS, for instance, respective execution logs can only reflect situations the PMS is able to handle. Particularly, if the PMS does not support process instance changes it has to be bypassed in exceptional situations (e.g., by executing unplanned process activities outside the scope of the PMS). Consequently, the PMS is unaware of the applied deviations and thus unable to log information about them. This missing traceability of process instance changes significantly limits the benefits of process mining and Delta analysis approaches.

Continuous process improvement requires adaptive PMS which enable authorized users to flexibly deviate from premodeled processes as needed. Since this results in more meaningful execution logs, which implicitly reflect the applied process instance changes, process mining and Delta analysis approaches become more useful. If such analyses result in an optimized version of a process model, adaptive PMS support the process engineer to quickly implement the model change and smoothly migrate running process instances to the new model.

In addition to process execution logs, adaptive PMS maintain information about applied instance modifications in change logs. Minimally, a change log should keep syntactical information about the kind and the context of the applied changes (e.g., the type and position of a dynamically inserted process activity). While this information is useful for process mining, it is not sufficient to effectively support process optimization efforts, process engineers also need semantical information about the reasons for the change. This is particularly important if the same or similar instance changes happen over and over again. Assume that in a patient treatment process an unplanned lab test is dynamically added for a considerable number of process instances. Then the respective change logs should also reflect information about the semantical context of the applied instance changes (e.g., that insertions have been mainly performed for patients older than 40 years and suffering from diabetes).

In this paper we introduce a framework for the agile mining of business processes which supports the whole process life cycle in an integrated way and fosters continuous and quick adaptation to change. Our framework is based on process mining [3], adaptive process management (PM) [4,5], and, to close the semantic gap, conversational case-based reasoning (CCBR) [6]. CCBR is an interactive extension of the case-based reasoning (CBR) paradigm [7]. In particular, we combine the advantages of these three approaches in an integrated prototype. We enhance execution and change logs with semantical information, which is then analyzed by the process engineer with support from the PMS to provide knowledge about the context of and the reasons for discrepancies between process models and related instances. This semantical information is also reused to support users when similar deviations become necessary. Finally, our framework

enables the process engineer to adapt process models based on the outcome of process mining efforts and to smoothly migrate related process instances to the new model version.

Section 2 describes building blocks for the agile mining of business processes. Section 3 gives an overview of our framework and Section 4 discusses a sample application. The paper closes with a summary and an outlook in Section 5.

2 Building Blocks for Agile Mining of Business Processes

This section summarizes main characteristics of our framework's building blocks: process mining, case-based reasoning, and adaptive PM.

2.1 Process Mining

Process mining denotes the extraction of process knowledge from log data related to past process and application executions (cf. Fig. 1). Respective logs are usually provided by workflow systems, but also by other process-oriented applications, like enterprise resource planning or supply chain management systems. Typically, all these systems log event-based data (e.g., related to the start or completion of task executions) together with additional context information (e.g., about actors).

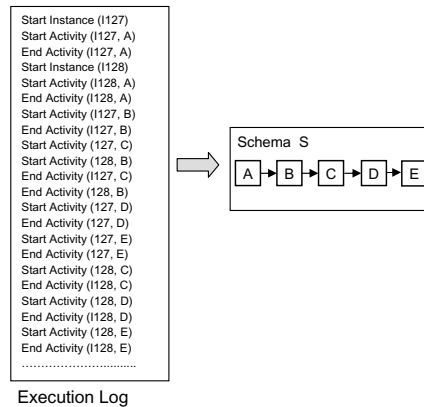


Fig. 1. Process Mining

The main objective of process mining is to effectively use automatically collected data in order to gain process knowledge from the logs. In particular, process mining extracts formal process models from execution logs [3,8,9,10,11]. So far, the focus has been put on issues related to control-flow mining. However, first approaches have been developed which use event-based data for mining organizational and performance aspects as well [12,13]. Process mining is generally

considered as an alternative to interviews or questionnaires to acquire process knowledge. Its results can be used for further analysis. For instance, Delta analysis [1,2] compares existing process models with the results of process mining in order to detect discrepancies between the modeled and the observed execution behavior. This information is then used to improve the process model.

2.2 Case-Based Reasoning

Case-based reasoning (CBR) is a contemporary approach to problem solving and learning [7]. New problems are dealt with by drawing on past experiences – described in cases – and by adapting their solutions to the new problem situation (cf. Fig. 2).

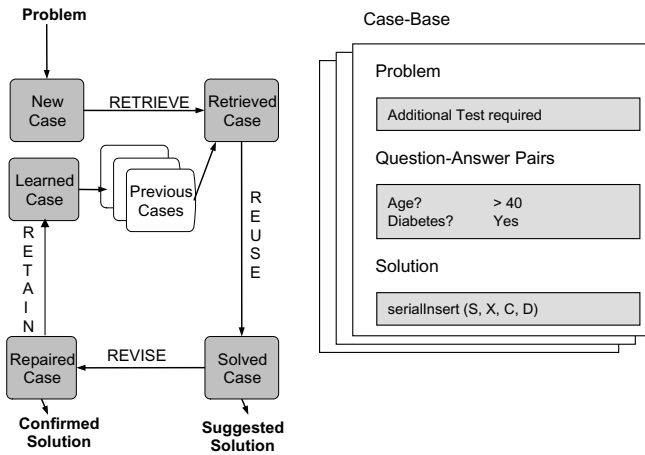


Fig. 2. Case-Based Reasoning

Reasoning based on past experiences is a powerful and frequently applied way to solve problems by humans [14]. A physician, for example, remembers previous cases to determine the disease of a newly admitted patient. A case is a contextualized piece of knowledge representing an experience [7], which typically consists of a problem description and the corresponding solution.

Our framework applies conversational CBR, an extension of the CBR paradigm, which actively involves users in the inference process [15]. CCBR systems are interactive systems that, via a mixed-initiative dialogue, guide users through a question-answering sequence in a case retrieval context. Unlike traditional CBR, CCBR neither requires users to provide a complete a priori problem specification for case retrieval nor to provide knowledge about the relevance of each feature for problem solving. Instead, the system assists users in finding relevant cases by presenting a set of questions to assess a given situation. Furthermore, it guides users who may supply already known information on their

initiative. Therefore, CCBR is especially suitable for handling exceptional or unanticipated situations which cannot be dealt with in a fully automated way.

CBR has been applied to PM for several purposes [16]. Our research prototype CBRFlow, for example, uses CCBR to perform ad-hoc changes of single process instances, to memorize these changes, and to support their reuse in similar future situations [16].

2.3 Adaptive Process Management

Adaptive PM technology increases the flexibility of process-oriented information systems by enabling (dynamic) changes of different process aspects (e.g., control and data flow). Such process changes can be performed at two levels – the *process type* and the *process instance* level [5,17] (cf. Fig. 3).

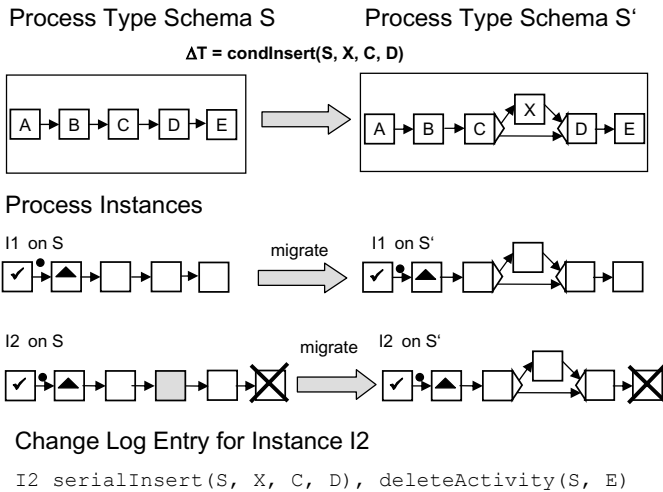


Fig. 3. Adaptive Process Management

When a process template or, more precisely, the *process type schema* representing this template is changed, the respective changes should be propagated to already running process instances [18]. This has to be done in a correct and consistent manner, and respective *migration procedures* must efficiently handle a high number of concurrently running process instances. In this context it must be possible to propagate process type changes to both unbiased and biased process instances. We denote process instances as *unbiased* if they are running according to the original process type schema they were derived from, whereas process instances are denoted as *biased* if they have been individually modified (e.g., due to an ad-hoc change) [17].

Instance-specific ad-hoc changes (e.g., switching the order of two activities or adding new ones) must be performed to deal with exceptional situations [4].

Usually, they are stored in change logs, which provide information about the type of the applied change and related context information (e.g., the position of a newly inserted activity). This log information contributes to the analysis of potential conflicts between process type and process instance changes as well as to the documentation of the instance history. The described functionality has been implemented in our ADEPT PMS [4,5,17,18].

3 Framework for Agile Mining of Business Processes

The implementation of a framework for the agile mining of business processes raises a number of challenges. This section outlines basic requirements (Section 3.1), gives an overview of the designed framework (Section 3.2), and describes the current state of its implementation (Section 3.3).

3.1 Requirements for an Agile Process Mining Framework

An agile process mining framework must provide comprehensive facilities for business process monitoring and must allow quick responses to observed discrepancies between the modeled and the executed processes. In particular, process engineers should be supported in learning from previous (ad-hoc) instance changes and in deriving optimized process models from log data.

When performing a process instance change information about the reasons for the change should be kept in a case-base. This can be done by either adding a new case (when no similar change has been applied before) or by reusing an existing one (representing a previously applied, similar ad-hoc change). A pure syntactical approach is not sufficient to stimulate the reuse of change information, semantical information about the changes must be maintained as well. Consider, for example, a patient treatment process and assume that an additional activity (i.e., a lab test) has been frequently inserted for patients older than 40 years and suffering from diabetes. If such context information is explicitly maintained, respective cases can be reused in similar situations and optimized process models can be derived from instance change logs.

To continuously optimize business processes, extended mining techniques must be provided. They should use information from execution and change logs as well as the semantical information (cases) associated with the changes. When similar ad-hoc changes occur frequently enough (i.e., their occurrence exceeds a certain threshold), process engineers should be notified and assisted in optimizing the process model. Semantical change information must be represented in such a way that useful suggestions can be made. For example, assume that our lab test activity has been inserted for a significant number of process instances in the context just mentioned. When moving this change to the process type level the simple insert operation (used at the instance level) cannot be applied directly as the additional lab test activity shall only be performed if the specific conditions (older than 40 years, diabetes) are met. Therefore, the PMS should be

able to translate instance changes and related semantical information to process type changes (i.e., to respective transformations of the process type schema) and to suggest them to the process engineer.

Process engineers should not only be supported in deriving new schema versions from log data, but also in migrating already running instances to a modified schema. For this, the PMS has to check whether these instances are compliant with this new schema version or not. Depending on whether an instance is biased or unbiased (cf. Section 2.3) and depending on the degree of overlap between process type and process instance change, different migration strategies must be supported by the PMS [19]. Furthermore, the system has to migrate the semantical information associated with the process schema as well (i.e., the cases representing ad-hoc changes on instances of this schema), as only information not yet covered by the new process schema must be migrated. Information on the ad-hoc change which triggered the process type evolution should be omitted, i.e., those cases should be dropped from the new case-base version.

3.2 Overview of the Framework

In order to meet these requirements and to reflect a company's business processes adequately, process models must be continuously monitored and adapted to changing needs. For this, both execution and change logs must be enriched with semantical information. As illustrated in Fig. 4, different information sources (i.e., execution logs, change logs, and case-bases) are relevant in this context. These sources must be continuously evaluated and changes to the process model should be triggered when discrepancies between it and its instances occur frequently. Based on the information maintained in the execution logs, in the change logs, and in the case-base, the process engineer can then adapt the process model and migrate related process instances by using adaptive PM technology (cf. Section 2.3).

Execution and change logs provide the syntactical information on what happened, i.e., which activities were executed and which deviations occurred at what time. CCBR (cf. Section 2.2) is used to provide semantical information on why changes happened. More precisely, experiences from previous changes are stored as cases in a case-base. A case represents a concrete ad-hoc modification of one or more process instances, it consists of a textual problem description briefly explaining the problem that made the deviation necessary, a set of question-answer pairs, and a solution part. The question-answer pairs describe the reasons and the context of the ad-hoc deviation and the solution part reflects the concrete change operations that have been applied to the respective instance(s) to perform the desired ad-hoc change by using services of the adaptive PMS (for a more detailed description see [20,21]).

In Fig. 4 the mining of the process execution log reveals that activities A, B, C, D and E are always executed in sequence. The change log further indicates that for some process instances running on schema S the additional activity X was dynamically inserted between activities C and D. However, the change log does not provide any semantical information on why activity X was inserted. By

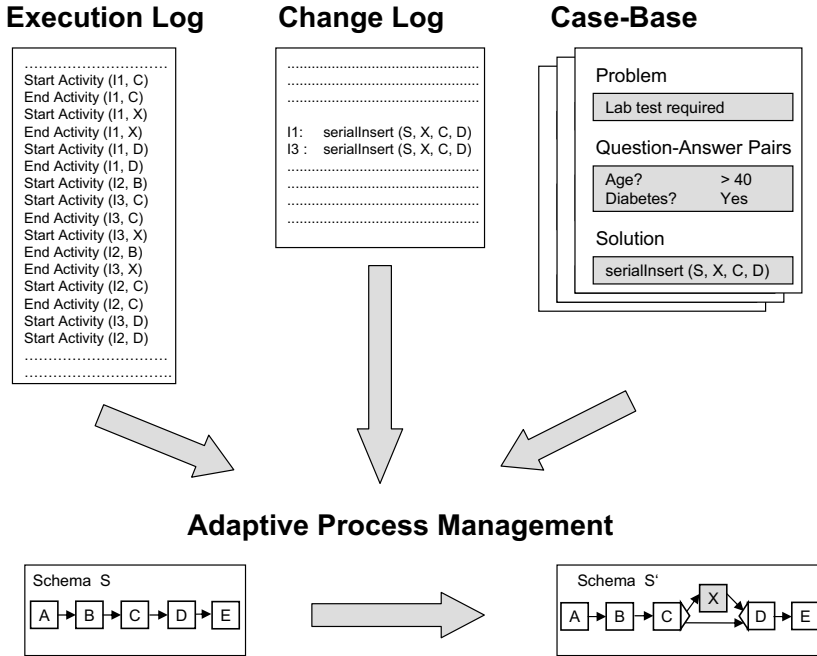


Fig. 4. Information Sources Triggering Process Evolution

analyzing the cases in the case-base, the process engineer learns that activity X was primarily added and executed for patients older than 40 years who suffers from diabetes. Based on this information he can derive a new process schema containing the additional activity X for patients matching these two conditions. Finally, after the new version of the process type schema is released, process instances can be migrated to the new schema version.

In the following we describe how the building blocks (cf. Section 2) and their respective information sources (i.e., execution log, change log and case-bases) work together to complete the process life cycle (cf. Fig. 5). An initial process schema can either be created by applying process mining techniques (i.e., by observing process and/or task executions) or by business process analysis (1). During run-time new process instances are created from such a predefined process schema (2). In general, process instances are then executed according to their process type schema and execution logs are written (3). However, when deviations from the predefined schema become necessary (e.g. due to exceptions or unanticipated situations) process actors must be able to deviate from it. They can either specify a new ad-hoc deviation and document the reasons for their changes in the CCBR subsystem, or reuse a previously specified ad-hoc modification from the case-base (4). In both situations an appropriate change log entry is written (5) and the execution continues. When a particular ad-hoc modification is frequently reused, the process engineer is notified to perform a process

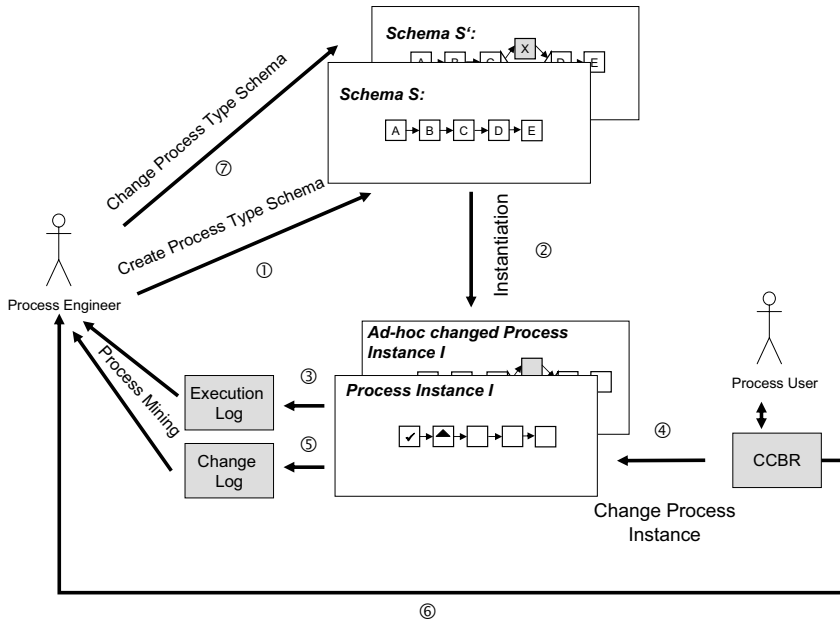


Fig. 5. Integrated Process Life Cycle Support

type change (6). Both execution and change logs are needed to know how often a particular schema has been instantiated and how frequently deviations occurred. The process engineer can then perform a schema evolution (7), as supported by the adaptive PMS, and, if possible, migrate running instances to the new schema version.

Thus, combining process mining and CCBR techniques with adaptive PM allows full process life cycle support and the seamless integration of the detected discrepancies into a new, optimized process schema.

3.3 Current State of the Framework

Our ongoing research focuses on the integration of adaptive PM technology and CCBR. We currently work on the integration of the methods, concepts, and prototypes developed by our groups in the ADEPT and the CBRFlow projects.

ADEPT [4] is a next generation PMS that offers full functionality in respect to the modeling, analysis, execution, and monitoring of business processes [4,5,18]. To our best knowledge it is the only PMS providing full support for adaptive processes at both the process instance and the process type level. When performing process instance changes, ADEPT ensures consistency and correctness. In the context of process type changes it additionally allows to migrate running process instances efficiently to the new schema version [5,17,18]. A powerful prototype demonstrating this functionality exists and is used by several research groups and industry partners.

CBRFlow [16] supports users in performing simple ad-hoc instance changes and allows them to express the semantics of these changes by applying CCBR techniques. It documents the reasons for a process instance change and enables the user to reuse information about previously performed ad-hoc deviations when creating new ones for other instances of the same process type. Like with ADEPT, a powerful prototype exists.

By integrating ADEPT and CBRFlow significant synergies can be exploited, fostering integrated process life cycle support [20,21]. On the one hand the combined system provides a powerful process engine, supporting all kinds of changes in one system. On the other hand it enables the intelligent reuse of process instance changes and fosters deriving process type changes from collected information. Currently we implement a prototype which integrates both systems; future work will include process mining tools as well [22]. We will apply and evaluate our prototype in different application settings, including healthcare processes and emergent workflows (e.g., in the automotive sector).

4 Sample Application of the Framework

Contemporary PMS rely on predefined process models requiring large upfront investments to analyze and model business processes before process-aware information systems can be deployed. It is especially difficult to create adequate process models for knowledge-intensive processes (e.g., engineering processes in the automotive sector) comprising both routinized and non-routinized work. In some cases process schemes are already obsolete when their specification is "completed". However, in today's dynamic business environment not all eventualities and deviations can be considered in advance as requirements change or evolve over time. Covering too many details in a process schema early on raises the risk of including rarely needed, not yet needed or never needed parts. For these reasons process modeling on demand and focusing on core functionality offer promising perspectives to foster shortened modeling cycles and earlier delivery of productive systems.

To cope with these risks and to ensure that the modeled process schemes closely reflect a company's business processes, short iteration cycles must be supported. Instead of starting with a completely defined process schema, the process engineer can either develop a preliminary one with a clear focus on core functionality (e.g., covering the standard process, but not all alternative paths) or apply process mining techniques to extract process fragments from event logs.

When starting with an initial process model our framework can be used to iteratively evolve the model over time. Whenever necessary, extensions to the process schema can be performed in an ad-hoc way by using adaptive process management. CCBR is applied to document these deviations, to memorize them and to support their reuse. Case reuse in combination with execution/change logs is applied to trigger process type changes and to incorporate these deviations into respective process schemes (cf. Fig. 5).

The framework also provides support for evolving process fragments which have been extracted by process mining techniques. Initially, new process fragments may only reflect a partial view on a process, i.e., the derived models are not mature. However, they already represent meaningful process knowledge which can be evolved over time and be reused in a different context (e.g., to automate routine work or to derive more comprehensive process templates). For example, engineers may want to customize fragments to individual needs or combine them to come up with more complete process views reflecting complex processes. When adapting or extending process fragments in such a way, new process knowledge is created, which again could be reused, harvested, and linked with existing fragments.

In this scenario fragment creation, storage, reuse, composition, and execution are important tasks, our framework provides fundamental support to deal with them. Initially, process fragments are derived by applying process mining techniques; the resulting models can then be semantically annotated using CCBP before they are stored in the process repository. When a user wants to retrieve a fragment, he is guided through a question-answering sequence by the CCBP sub-system. When a fragment is selected, its reuse counter is increased. Additionally, quality ratings from users are aggregated in a reputation score. A special challenge is the execution of incomplete fragments which are to be completed or adapted; for this, adaptive process management technology is key to success. Altogether, our framework provides the needed adaptation and mining techniques to support such evolving and emergent processes.

5 Conclusion and Outlook

We have introduced a framework for the agile mining of business processes based on CCBP, adaptive PM and process mining. The main focus has not been put on the development of new concepts in these domains, but on the integration of existing methods, concepts and prototypes (ADEPT and CBRFlow) and on the support of business process intelligence. We have sketched the basic relationships between the different components, discussed technical requirements for providing an integrated solution, and drafted the approach we take. Currently we work on the implementation of the framework presented, starting with the integration of ADEPT and CBRFlow. In this context we have also developed concepts for process evolution and the migration of related case-bases [20,21]. In the next step we want to incorporate process mining tools into our framework.

References

1. Guth, V., Oberweis, A.: Delta analysis of petri net based models for business processes. In: Proceedings of International Conference on Applied Informatics. (1997) 23–32
2. v.d. Aalst, W.: Inheritance of business processes: A journey visiting four notorious problems. In: Petri Net Technology for Communication Based Systems. LNCS 2472 (2003) 383–408

3. v.d. Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow mining: A survey of issues and approaches. *Data and Knowledge Engineering* **27** (2003) 237–267
4. Reichert, M., Dadam, P.: ADEPT_{flex} - supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems* **10** (1998) 93–129
5. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases* **16** (2004) 91–116
6. Aha, D.W., Breslow, L., Muñoz-Avila, H.: Conversational case-based reasoning. *Applied Intelligence* **14** (2001) 9–32
7. Kolodner, J.L.: *Case-Based Reasoning*. Morgan Kaufmann (1993)
8. Golani, M., Pinter, S.S.: Generating a process model from a process audit log. In: *Proceedings of International Conference on Business Process Management (BPM'03)*, Eindhoven (2003) 136–151
9. Herbst, J.: An inductive approach to adaptive workflow systems. In: *Proc. Workshop Towards Adaptive Workflow Systems (CSCW'98)*, Seattle (1998)
10. de Medeiros, A., van der Aalst, W., Weijters, A.: Workflow mining: Current status and future directions. In: *Proceedings of International Conference on Cooperative Information Systems (CoopIS'03)*. LNCS 2888, Berlin (2003) 389–406
11. van Dongen, B., van der Aalst, W.: Multi-phase process mining: Building instance graphs. In: *Proceedings of International Conference on Conceptual Modeling (ER 2004)*. LNCS 3288, Berlin (2004) 362–376
12. van der Aalst, W., Song, M.: Mining social networks. uncovering interaction patterns in business processes. In: *Proceedings of International Conference on Business Process Management (BPM'04)*, Potsdam, Germany (2004) 244–260
13. van der Aalst, W., van Dongen, B.: Discovering workflow performance models from timed logs. In: *International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*. LNCS 2480, Berlin (2003) 45–63
14. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations and system approaches. *AI Communications* **7** (1994) 39–59
15. Aha, D.W., Muñoz-Avila, H.: Introduction: Interactive case-based reasoning. *Applied Intelligence* **14** (2001) 7–8
16. Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling adaptive workflow management through conversational case-based reasoning. In: *Proceedings of European Conference on Case-based Reasoning (ECCBR'04)*, Madrid (2004) 434–448
17. Rinderle, S., Reichert, M., Dadam, P.: On dealing with structural conflicts between process type and instance changes. In: *Proceedings of International Conference on Business Process Management (BPM'04)*, Potsdam (2004) 274–289
18. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems – a survey. *Data and Knowledge Engineering* **50** (2004) 9–34
19. Rinderle, S., Reichert, M., Dadam, P.: Disjoint and overlapping process changes: Challenges, solutions, applications. In: *Proceedings of International Conference on Cooperative Information Systems (CoopIS'04)*, Agia Napa (2004) 101–120
20. Weber, B., Rinderle, S., Wild, W., Reichert, M.: CCBDR-driven business process evolution. In: *Proceedings of International Conference on Case-Based Reasoning (ICCBR'05)*, Chicago (2005) 610–624
21. Rinderle, S., Weber, B., Reichert, M., Wild, W.: Integrating process learning and process evolution - a semantics based approach. In: *Proceedings of International Conference on Business Process Management (BPM'05)*. (2005) 252–267
22. *Process Mining Research: www.processmining.org* (2005)

Genetic Process Mining: A Basic Approach and Its Challenges

A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{a.k.medeiros, a.j.m.m.weijters, w.m.p.v.d.aalst}@tm.tue.nl

Abstract. One of the aims of process mining is to retrieve a process model from a given event log. However, current techniques have problems when mining processes that contain non-trivial constructs and/or when dealing with the presence of noise in the logs. To overcome these problems, we try to use genetic algorithms to mine process models. The non-trivial constructs are tackled by choosing an internal representation that supports them. The noise problem is naturally tackled by the genetic algorithm because, per definition, these algorithms are robust to noise. The definition of a good fitness measure is the most critical challenge in a genetic approach. This paper presents the current status of our research and the pros and cons of the fitness measure that we used so far. Experiments show that the fitness measure leads to the mining of process models that can reproduce all the behavior in the log, but these mined models may also allow for extra behavior. In short, the current version of the genetic algorithm can already be used to mine process models, but future research is necessary to always ensure that the mined models do not allow for extra behavior. Thus, this paper also discusses some ideas for future research that could ensure that the mined models will *always* only reflect the behavior in the log.

Keywords: process mining, genetic mining, genetic algorithms, Petri nets, workflow Petri nets.

1 Introduction

One of the aims of process mining is to *automatically* build a process model that describes the behavior contained in an event log. The models mined by process mining tools can be used as an objective starting point during the deployment of systems that support the execution of processes and/or as a feedback mechanism to check the prescribed process model against the enacted one. We illustrate how process mining techniques work using an example. Consider the event log shown in Table 1. This log shows the events for applying to get a license to ride motorbikes or drive cars. Note that applicants for different types of licenses do the same theoretical exam (task C) but different practical ones (tasks D or E). In other words, whenever task B is executed, task E is the only one that can be executed after the applicant has done the theoretical exam. This shows that there

Table 1. Where: X = “Apply for license”, A = “Attend to classes on how to ride motorbikes”, B = “Attend to classes on how to drive cars”, C = “Do theoretical exam”, D = “Do practical exam to ride motorbikes”, E = “Do practical exam to drive cars”, and Y = “Get result”

ID	Process instance
1	X, A, C, D, Y
2	X, B, C, E, Y
3	X, A, C, D, Y
4	X, B, C, E, Y
5	X, B, C, E, Y
6	X, A, C, D, Y

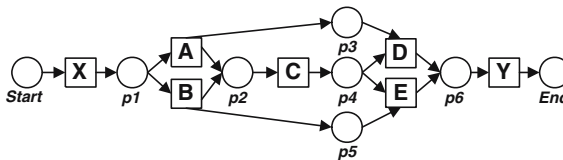


Fig. 1. Mined net for the log in Table 1

is a dependency between tasks B and E, and also between tasks A and D. Based on this log and these observations, process mining tools could retrieve the model in Figure 1. In this case, we are using Petri nets to depict this model. We do so because Petri nets [5, 10] will be used to explain how the semantics of our internal representation work.

Petri nets are a formalism to model concurrent processes. Graphically, Petri nets are bipartite directed graphs with two node types: *places* and *transitions*. Places represent conditions in the process. Transitions represent actions. Tasks in the event logs correspond to transitions in Petri nets. The state of a Petri net (or process for us) is described by adding tokens (black dots) to places. The dynamics of the Petri net is determined by the *firing rule*. A transition can be executed (i.e. an action can take place in the process) when all of its input places (i.e. pre-conditions) have at least a number of tokens that is equal to the number of directed arcs from the place to the transition. After execution, the transition removes tokens from the input places (one token is removed for every input arc from the place to the transition) and produces tokens for the output places (again, one token is produced for every output arc). Besides, the Petri nets that we consider have a single *start* place and a single *end* place. For the Petri net in Figure 1, in the initial state there is only one token in place *Start*. This implies that *X* is the only transition that can be executed in the initial state. When *X* executes (or fires), one token is removed from the place *Start* and one token is added to the place *p1*. In this marking, *A* or *B* are enabled to fire. If *A* fires, it consumes the token in *p1* and produces one token for *p2* and another for *p3*. Note that, although place *p3* has now one token, transition *D* cannot fire yet because the place *p4* is not marked. The next transitions to fire are respectively *C*, *D* and *Y*.

Current research in process mining [1, 2, 3, 4, 7, 8, 11] still has problems to discover process models with certain structural constructs and/or to deal with the presence of noise in the logs. The problematic constructs are: *non-free-choice*, *invisible tasks* and *duplicate tasks* [8]. Non-free-choice constructs combine synchronization and choice. The example in Figure 1 illustrates a non-free-choice construct involving the tasks D and E . The current techniques do not capture the dependency between the tasks $A - D$ and $B - E$. Invisible tasks are only used for routing purposes (for instance, to skip the execution of another task) and do not appear in the log. The current techniques do not mine models with unlabelled tasks. Duplicate tasks mean that multiple transitions have the same label in the original process model. The problem here is that most of the mining techniques treat these duplicate tasks as a single one. *Noise* can appear in two situations: event traces were somehow incorrectly logged or event traces reflect exceptional situations. Either way, most of the techniques will try to find a process model that can parse all the traces in the log. However, the presence of noise will hinder the correct mining of the most common behavior.

One of the reasons why the current techniques cannot completely cope with the above mentioned problematic constructs and/or with noisy logs is because their search is based on *local* information in the log. For instance, the α -algorithm (see [2] for details) uses only information about which tasks directly succeed or precede one another in the process instances. As a result, this algorithm does not capture the dependency in non-free-choice constructs. For example, the α -algorithm will never discover the Petri net in Figure 1 for the log in Table 1 because none of the process instances has the sub-trace “A,D” or “B,E”. Consequently, the α -algorithm will not link these tasks. To overcome these problems, our research uses *genetic algorithms* [6] to mine process models. The main motivation is to benefit from the *global* search that is performed by this kind of algorithms.

Genetic algorithms are adaptive search methods that try to mimic the process of evolution. These algorithms start with an initial population of individuals (in this case process models). Every individual is assigned a fitness measure to indicate its quality. In our case, an individual is a possible process model and the fitness is a function that evaluates how good the individual expresses the behavior in the log. Populations evolve by selecting the fittest individuals and generating new individuals using genetic operators such as *crossover* (combining parts of two of more individuals) and *mutation* (random modification of an individual).

When using genetic algorithms to mine process models, there are three main concerns. The first is to define the *internal representation*. The internal representation defines the search space of a genetic algorithm. The internal representation that we define and explain in this paper supports all the problematic constructs, except for duplicate tasks. The second concern is to define the *fitness measure*. In our case, the fitness measure evaluates the quality of a point (individual or process model) in the search space against the event log. A genetic algorithm searches for individuals whose fitness is maximal. Our fitness measure makes sure that individuals with a fitness that is equal to 1 will parse all the process instances (traces) in the log. The third concern relates to the *genetic operators* (crossover

and mutation) because they should ensure that all points in the search space defined by the internal representation may be reached when the genetic algorithm runs. This paper presents a genetic algorithm that addresses these three concerns.

The rest of the paper is organized as follows. Section 2 explains the internal representation that we use and its semantics. Section 3 explains how the genetic algorithm works. Section 4 discusses the experiments and results. This section also shows the results when the genetic algorithm uses some heuristics during the building of the initial population. Section 5 presents the current ideas to make sure that the returned model is “minimal”. Section 6 presents the conclusions and future work.

2 Internal Representation and Semantics

When defining the internal representation to be used by our genetic algorithm, the main requirement was that this representation should express the dependencies between the tasks in the log. In other words, the model should clearly express which tasks would enable the execution of other tasks. Additionally, it would be nice if the internal representation would be compatible with a formalism to which analysis techniques and tools exist. This way, these techniques could also be applied to the discovered models. Thus, one option would be to directly represent the individual (or process model) as a Petri net [10]. However, such a representation would require determining the number of places in every individual and this is not the core concern. It is more important to show the dependencies between the tasks. So, we defined an internal representation that is as expressive as Petri nets (from the task dependency perspective) but that focusses only on the dependencies between tasks. This representation is called *causal matrix*. Figure 2 illustrates in (i) the causal matrix¹ that expresses the same task dependencies that are in the “original Petri net”. The causal matrix shows which tasks enable the execution of other tasks via the matching of *input* (I) and *output* (O) condition functions. The sets returned by the condition functions I and O have *subsets* that contain the tasks in the model. Tasks in a same subset have an OR-split/join relation. Sets in different subsets have an AND-split/join relation. Thus, every I and O set expresses a conjunction of disjunctions. Additionally, a task may appear in more than one subset in a same set. As an example, for task D in the original Petri net in Figure 2 the causal matrix states that $I(D) = \{\{F, B, E\}, \{E, C\}, \{G\}\}$ because D is enabled by an AND-join construct that has 3 places. From top to bottom, the first place has a token whenever F or B or E fires. The second place, whenever E or C fires. The third place, whenever G fires. Similarly, the causal matrix has $O(D) = \{\}$ because D is executed last in the model. The following definition formally defines these notions.

Definition 1 (Causal Matrix). A *Causal Matrix* is a tuple $CM = (A, C, I, O)$, where

¹ Due to space limitations, Figure 2 shows a compact representation of this causal matrix.

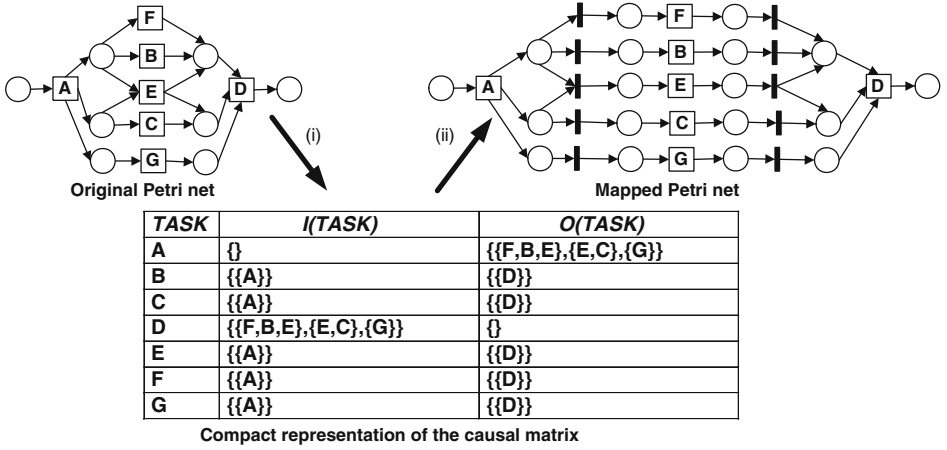


Fig. 2. Mapping of a PN with more than one place between two tasks (or transitions)

- A is a finite set of activities,
- $C \subseteq A \times A$ is the causality relation,
- $I \in A \rightarrow \mathcal{P}(\mathcal{P}(A))$ is the input condition function,²
- $O \in A \rightarrow \mathcal{P}(\mathcal{P}(A))$ is the output condition function,

such that

- $C = \{(a_1, a_2) \in A \times A \mid a_1 \in \bigcup I(a_2)\}$,³
- $C = \{(a_1, a_2) \in A \times A \mid a_2 \in \bigcup O(a_1)\}$,
- $C \cup \{(a_o, a_i) \in A \times A \mid a_o \overset{C}{\bullet} = \emptyset \wedge \overset{C}{\bullet} a_i = \emptyset\}$ is a strongly connected graph.

Any Petri net without duplicate tasks and without more than one place with the same input tasks and the same output tasks can be mapped to a causal matrix. Definition 2 formalizes such a mapping. The main idea is that there is a causal relation C between any two tasks t and t' whenever at least one of the *output* places of t is an *input* place of t' . Additionally, the I and O condition functions are based on the input and output places of the tasks. This is a natural way of mapping because the input and output places of Petri nets actually reflect the conjunction of disjunctions that these sets express.

Definition 2 ($\Pi_{PN \rightarrow CM}$). Let $PN = (P, T, F)$ be a Petri net. The mapping of PN is a tuple $\Pi_{PN \rightarrow CM}(PN) = (A, C, I, O)$, where

- $A = T$,
- $C = \{(t_1, t_2) \in T \times T \mid t_1 \bullet \cap \bullet t_2 \neq \emptyset\}$,

² $\mathcal{P}(A)$ denotes the powerset of some set A .

³ $\bigcup I(a_2)$ is the union of the sets in set $I(a_2)$.

- $I \in T \rightarrow \mathcal{P}(\mathcal{P}(T))$ such that $\forall t \in T \ I(t) = \{\bullet p \mid p \in \bullet t\}$,
- $O \in T \rightarrow \mathcal{P}(\mathcal{P}(T))$ such that $\forall t \in T \ O(t) = \{p \bullet \mid p \in t \bullet\}$.

The semantics of the causal matrix can be easily understood by mapping them back to Petri nets. This mapping is formalized in Definition 3. *Conceptually*, the causal matrix behaves as a Petri net that contains visible and invisible tasks. For instance, see Figure 2. This figure shows (i) the mapping of a Petri net to a causal matrix and (ii) the mapping from the causal matrix to a Petri net. The firing rule for the mapped Petri net is very similar to the firing rule of Petri nets in general. The only difference concerns the invisible tasks. Enabled invisible tasks can only fire if their firing enables a visible task. Similarly, a visible task is enabled if all of its input places have tokens or if there exists a *set of* invisible tasks that are enabled and whose firing will lead to the enabling of the visible task. Conceptually, the causal matrix keeps track of the distribution of tokens at a marking in the output places of the visible tasks. Every causal matrix starts with a token at the start place. We point out that, in Figure 2, although the mapped Petri net does not have the same *structure* of the original Petri net, these two nets are *behaviorally* equivalent. In other words, given that these two nets initially have a single token and this token is at the start place (i.e., the input place of A), the set of traces the two nets can generate is the same. Additionally, the invisible tasks in the mapped Petri net show that the causal matrix supports the representation of invisible tasks that are used, for instance, to skip tasks. A detailed explanation and formalization about the mappings in this section can be found in [9].

Definition 3 ($\Pi_{CM \rightarrow PN}^N$). Let $CM = (A, C, I, O)$ be a causal matrix. The naive Petri net mapping of CM is a tuple $\Pi_{CM \rightarrow PN}^N = (P, T, F)$, where

- $P = \{i, o\} \cup \{i_{t,s} \mid t \in A \wedge s \in I(t)\} \cup \{o_{t,s} \mid t \in A \wedge s \in O(t)\}$,
- $T = A \cup \{m_{t_1, t_2} \mid (t_1, t_2) \in C\}$,
- $F = \{(i, t) \mid t \in A \wedge \overset{C}{\bullet} t = \emptyset\} \cup \{(t, o) \mid t \in A \wedge t \overset{C}{\bullet} = \emptyset\} \cup \{(i_{t,s}, t) \mid t \in A \wedge s \in I(t)\} \cup \{(t, o_{t,s}) \mid t \in A \wedge s \in O(t)\} \cup \{(o_{t_1, s}, m_{t_1, t_2}) \mid (t_1, t_2) \in C \wedge s \in O(t_1) \wedge t_2 \in s\} \cup \{(m_{t_1, t_2}, i_{t_2, s}) \mid (t_1, t_2) \in C \wedge s \in I(t_2) \wedge t_1 \in s\}$.

3 Genetic Algorithm

In this section we describe the main building blocks of our genetic algorithm.

3.1 Initial Population

The initial population is randomly built by the genetic algorithm. As explained in Section 2, individuals are causal matrices. When building the initial population, we roughly follow Definition 1. Given a log, all individuals in any population of the genetic algorithm have the same set of activities (or tasks) A . This set contains the tasks that appear in the log. However, the causality relation C and the condition functions I and O are randomly built for every individual in the population. As a

result, the initial population can have any individual in the search space defined by a set of activities A . Note that the higher the amount of tasks that a log contains, the bigger this search space.

3.2 Fitness Calculation

Our fitness is based on the parsing of the event traces by individuals. For a noise-free log, the perfect individual should have fitness 1. For a noisy log, the perfect individual should have fitness close to 1 (since it would be able to parse most of the traces but it would have problems to parse the noisy traces). From this discussion, a natural fitness for an individual to a given log seems to be the number of properly parsed event traces⁴ divided by the total number of event traces. However, this fitness measure is too coarse because it does not give indication about how many parts of an individual are correct when the individual does not properly parse an event trace. So, we defined a more elaborate fitness function: when the task to be parsed is not enabled, the problems (e.g. number of missing tokens to enable this task) are registered and the parsing proceeds as if this task would be enabled. This *continuous* parsing semantic is more robust because it gives a better indication of how many tasks do or do not have problems during the parsing of a trace. The fitness function that our algorithm uses is in Definition 4. The notation used in this definition is as follows. $allParsedActivities(L, CM)$ gives the total number of tasks in the event log L that could be parsed without problems by the causal matrix (or individual) CM . $numActivitiesLog(L)$ gives the number of tasks in L . $allMissingTokens(L, CM)$ indicates the number of missing tokens in all event traces. $allExtraTokensLeftBehind(L, CM)$ indicates the number of tokens that were not consumed after the parsing stopped plus the number of tokens of the end place minus 1 (because of proper completion). $numTracesLog(L)$ indicates the number of traces in L . $numTracesMissingTokens(L, CM)$ and $numTracesExtraTokensLeftBehind(L, CM)$ respectively indicate the number of traces in which tokens were missing or tokens were left behind during the parsing.

Definition 4 (Fitness). *Let L be an event log. Let CM be a causal matrix. Then the fitness $F : L \times CM \rightarrow (-\infty, 1]$ is a function defined as*

$$F(L, CM) = \frac{allParsedActivities(L, CM) - punishment}{numActivitiesLog(L)}, \text{ where}$$

$$punishment = \frac{allMissingTokens(L, CM)}{numTracesLog(L) - numTracesMissingTokens(L, CM) + 1} + \frac{allExtraTokensLeftBehind(L, CM)}{numTracesLog(L) - numTracesExtraTokensLeftBehind(L, CM) + 1}$$

The fitness F gives a more detailed indication about how fit an individual is to a given log. The function $allMissingTokens$ penalizes (i) nets with OR-split where it should be an AND-split and (ii) nets with an AND-join where it should be an OR-join. Similarly, the function $allExtraTokensLeftBehind$ penalizes (i) nets with

⁴ An event trace is properly parsed by an individual if, for an initial marking that contains a single token and this token is at the start place of the mapped Petri net for this individual, after firing the visible tasks in the order in which they appear in the event trace, the end place is the only one to be marked and it has a single token.

AND-split where it should be an OR-split and (ii) nets with an OR-join where it should be an AND-join. Note that we weigh the impact of the *allMissingTokens* and *allExtraTokensLeftBehind* functions by respectively dividing them by the number of event traces minus the number of event traces with missing and left-behind tokens. The main idea is to promote individuals that correctly parse the more frequent behavior in the log. Additionally, if two individuals have the same *punishment* value, the one that can parse more tasks has a better fitness because its missing and left-behind tokens impact fewer tasks. This may indicate that this individual has more correct *I* and *O* condition functions than incorrect ones. In other words, this individual is a better candidate to produce offsprings for the next population (see Subsection 3.4).

3.3 Stop Criteria

The mining algorithm stops when (i) it finds an individual with fitness equals 1; or (ii) it computes n generations, where n is the maximum number of generation that is allowed; or (iii) the fittest individual has not changed for $n/2$ generations in a row.

3.4 Genetic Operators

We use *elitism*, *crossover* and *mutation* to build the individuals of the next generation. A percentage of the best individuals (the *elite*) is directly copied to the next population. The other individuals in the population are generated via crossover and mutation. Two parents produce two offsprings. To select parents, a *tournament* is played in which five individuals in the population are randomly drawn and the fittest one always wins. The *crossover rate* determines the probability that two parents undergo crossover. Crossover is a genetic operator that aims at recombining existing material in the current population. In our case, this material is the set of current causality relations in the population. The crossover operation should allow for the complete search of the space defined by the existing causality relation in a population. Given a set of causality relations, the search space contains all the individuals that can be created by any combination of a subset of the causality relations in the population. Thus, our crossover operator allows an individual to: lose tasks from the subsets in its *I/O* condition functions (but not necessarily causality relations because a same task may be in more than one subset of an *I/O* condition function), add tasks to the subsets in its *I/O* condition functions (again, not necessarily causality relations), exchange causality relations with other individuals, incorporate causality relations that are in the population but are not in the individual, lose causality relations, decrease the number of subsets in its *I/O* condition functions, *and/or* increase the number of subsets in its *I/O* condition functions. The crossover point of two parents is a randomly chosen task. Note that, after crossover, the number of causality relations for the whole population remains constant, but how these relations appear in the offsprings may be different from the parents.

After the crossover, the mutation operator takes place. The mutation operator aims at inserting new material in the current population. In our case, this means that the mutation operator may change the existing causality relations of a population. Thus, our mutation operator performs one of the following actions to the I/O condition functions of a task in an individual: (i) randomly choose a subset and add a task (in A) to this subset, (ii) randomly choose a subset and remove a task out of this subset, *or* (iii) randomly redistribute the elements in the subsets of I/O into new subsets. For example, consider the input condition function of task D in Figure 2. $I(D) = \{\{F, B, E\}, \{E, C\}, \{G\}\}$ can be mutated to (i) $\{\{F, B, E\}, \{E, C\}, \{G, D\}\}$ if task D is added to the subset $\{G\}$, (ii) $\{\{F, B, E\}, \{C\}, \{G\}\}$ if task E is removed from the subset $\{E, C\}$, or (iii) $\{\{F\}, \{E, C, B\}, \{G\}, \{E\}\}$ if the elements in the original $I(D)$ are randomly redistributed in a random chosen number of new subsets. Every task in an offspring may undergo mutation with the probability determined by the *mutation rate*.

4 Experiments and Results

As a first test for our genetic algorithm (GA), we applied it for *noise-free* event logs and checked if it could mine process models that contain *all* the behavior in these logs. In other words, the mined model should have the fitness $F = 1$. During the experiments, the genetic algorithm mined event logs from nets that contain 5, 7, 8, 12 and 22 tasks. These nets contain short loops, parallelism and/or non-free-choice constructs. Every event log has 1000 random executions of the nets. For each noise-free event-log, 10 runs of the genetic algorithm were executed. The populations had 500 individuals and were iterated for at most 100000 generations. The crossover rate was 1.0 and the mutation rate was 0.01. The elitism rate was 0.01. The initial population might contain duplicate individuals. All the experiments were run using the ProM framework, our tool set that can be obtained via www.processmining.org. We implemented the genetic algorithm described in this paper as a plug-in for this framework. This framework also supports a mapping from the internal representation to Petri nets (cf. Definition 3).

The results in Table 2 show that the GA could find an individual that can parse all log traces in most of the runs⁵. However, none of these individuals are equal to the original nets that were used to generate the event logs. This happens because, although the requirements that the fitness F captures are all *necessary* to ensure that the GA mines a process model that can parse all traces in the log, these requirements are not *sufficient* to ensure that the mined model will *always* give a good insight about what is happening in the log. The reason is that different models are able to parse all event traces and these models may allow for extra behavior that does not belong to the class of traces in the log. A class of traces defines all possible combinations of task orderings for a given set of tasks. For instance, the traces “a,b,c,d” and “a,c,b,d” belong to the same class

⁵ Note: The experiments for the log of the net with 22 tasks were run for at most 250 generations because they take too much time to complete. However, the obtained results suggest that the population was evolving towards the right direction.

Table 2. Results of the mining for 10 runs. “#” means “number of”. The columns BFE, WFE and MBF respectively show the Best Fitness Ever, the Worst Fitness Ever (i.e. the best one in the worst run) and the Mean Best Fitness (i.e. average over 10 runs)

# Tasks	Figure	# Runs $F = 1$	BFE	WFE	MBF	Mean # Generations	Original Found
5	3.a	9	1	0.989	0.998	5016	0
7	2	6	1	0.987	0.997	29259	0
7 (nfc)	1	10	1	1	1	511	0
8	3.b	10	1	1	1	5145	0
12	3.c	10	1	1	1	1831	0
22	4	0	0.931	0.537	0.739	249	0

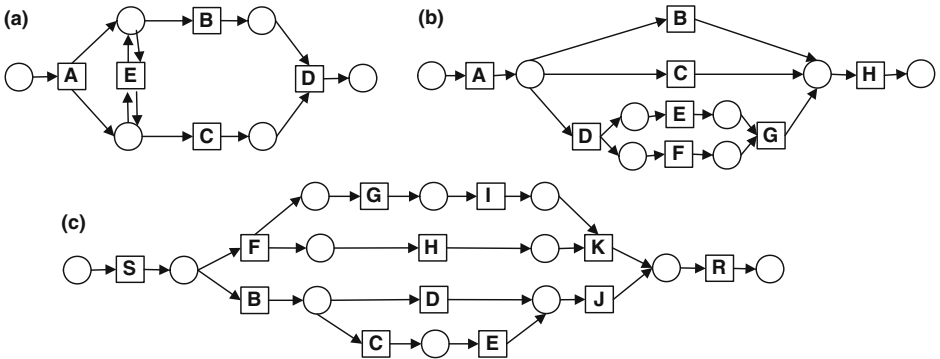


Fig. 3. The original nets with 5, 8 and 12 tasks

because they involve the same set of tasks $\{a, b, c, d\}$ and show that these tasks can be interleaved probably because they are in parallel or they are short loops.

Thus, the challenge we have now for our GA is: “How to ensure that the retrieved model that can parse all the traces does not allow for extra undesired behavior as well?”. To illustrate this we consider the nets shown in Figure 5. These models can also parse the traces in Table 1, but they allow for extra behavior. For instance, both models allow for the applicant to take the exam before attending to classes. To define a fitness measure to punish models that express more than it is in the log is especially difficult because we do not have negative examples. The logs show the allowed (positive) behavior, but they do not express the forbidden (negative) one.

A possible way to increase the probability that the GA will mine models that allow for none or little extra behavior is to use a hybrid version of evolutionary algorithm that uses heuristics in the search [6]. In our case, the hybrid version uses some heuristics to build the initial population. In short, the more often a task t is directly followed by a task t' (i.e. the subtrace “ t, t' ” appears in traces in the log), the higher the probability that individuals are built with a causality

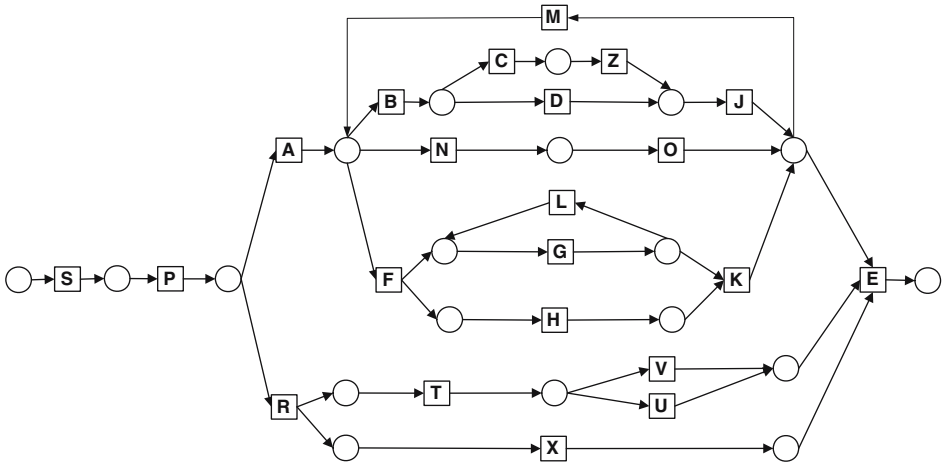


Fig. 4. The original net with 22 tasks

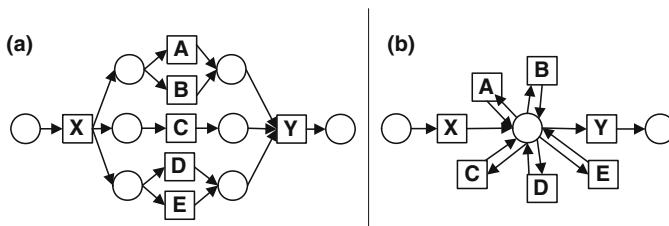


Fig. 5. Example of nets that can also reproduce the behavior for the log in Table 1. The problem here is that these nets allow for extra behavior that is not in the log.

Table 3. Results of the mining for 10 runs when heuristics are used to build the initial population

# Tasks	Figure	# Runs $F = 1$	BFE	WFE	MBF	Mean # Generations	Original Found
5	3.a	10	1	1	1	2	2
7	2	9	1	0.999	0.999	16323	4
7 (nfc)	1	10	1	1	1	0	0
8	3.b	10	1	1	1	2	9
12	3.c	10	1	1	1	1	10
22	4	1	1	0.972	0.989	192	0

relation from t to t' . Details about the heuristics can be found in [9]. With this setting, the genetic algorithm could also find, most of the time, an individual that could parse all traces in the log. Furthermore, the genetic algorithm sometimes

found an individual that is equal to the original net (see Table 3). Note that the hybrid genetic algorithm performed better for nets with few or no parallelism. Additionally, the use of heuristics hindered the discovering of the non-free-choice construct. This happens because the heuristics are based on local relations in the log. Note that there is no direct relation between tasks $A - D$ and $B - E$, and if we remove the places $p3$ and $p5$ from the net in Figure 1, the resulting net can also parse all traces that the net in Figure 1 can parse.

Another possible solution is to improve the fitness function to make sure that the mined model is not only *complete* (parses all traces in a log) but it is also *minimal* (does not allow for classes of traces that cannot be derived from the log). The next section presents some ideas on how to do so.

5 Challenges and Some Ideas

Although our fitness measure F (see Subsection 3.2) punishes individuals with wrong AND/OR-split/join constructs and individuals with missing arcs, it does not punish individuals that allow for additional behavior not present in the log. Thus, our challenge is to include in the fitness function F a measure that punishes for extra behavior.

One possible solution to punish an individual that allows for undesirable behavior could be to build the *coverability graph* [10] of the mapped Petri net for this individual and check the fraction of event traces this individual can generate that are not in the log. The traces that express different paths of execution for parallelism are not considered as extra behavior. The main idea in this approach is to punish the individual for every extra event trace it generates. Unfortunately, building the coverability graph is not very practical and it is unrealistic to assume that all possible behavior is present in the log.

Another possibility is to check, for every marking, the *number of visible tasks that are enabled*. Individuals that allow for extra behavior tend to have more enabled tasks than individuals that do not. For instance, the nets in Figure 5 have more enabled tasks in most reachable markings than the net in Figure 1. The main idea in this approach is to punish more the individuals that have more enabled visible tasks during the parsing of the log.

6 Conclusions and Future Work

In this paper we presented a genetic algorithm to mine process models. The internal representation allows for the mining of process models that contain non-free-choice and invisible tasks. The current version can search the space defined by the set of tasks in the log and return a process model (individual) that can parse all event traces, regardless of how the initial population is built. This means that our genetic operators (crossover and mutation) are working as expected. However, the fitness measure needs to be improved to make sure that the mined models only express the behavior in the log. Our main challenge here is how to cope with the lack of negative examples. We do not have logs that show the forbidden (negative)

behavior. Thus, the genetic algorithm has to work with what actually happened (positive examples), but it still should punish the individuals that allow for extra behavior that does not comply with the log. Some ideas to improve the fitness measure include (i) computing all the traces that an individual can produce (its coverability graph) or (ii) checking the amount of tasks that are enabled at every marking during the parsing of event traces. Our future work will focus on developing metrics to mine process models that are not only complete (express the behavior in the log), but are also minimal (do not allow for extra undesired behavior).

References

1. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
2. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
3. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
4. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
5. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
6. A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Natural Computing. Springer-Verlag, Berlin, 2003.
7. J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.
8. A.K.A. de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Workflow Mining: Current Status and Future Directions. In R. Meersman, Z. Tari, and D.C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 389–406. Springer-Verlag, Berlin, 2003.
9. A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Using Genetic Algorithms to Mine Process Models: Representation, Operators and Results. BETA Working Paper Series, WP 124, Eindhoven University of Technology, Eindhoven, 2004.
10. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
11. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

On Web Services Workflow Mining

Robert Gombotz and Schahram Dustdar

Distributed Systems Group,
Vienna University of Technology, Austria
{gombotz, dustdar}@infosys.tuwien.ac.at

Abstract. With the ever growing importance of the service-oriented paradigm in system architectures more and more (business) processes will be executed using service-oriented systems. Therefore, we believe that the ability to discover processes in loosely-coupled systems is essential in system optimization. Firstly, we briefly describe our previously introduced idea of Web Services Interaction Mining (WSIM) and then direct our attention on mining for workflows in logs provided by SOA. We thoroughly examine strategies in other fields of mining for their applicability in SOA. After that, we discuss logging possibilities in service-oriented systems and analyze mining opportunities with regards to the provided logs. As a case study we present a service-oriented system and its logging features. We conclude with a demonstration of how we successfully applied existing process mining strategies on this system's logs and present the results of that mining in the form of workflow models.

1 Introduction

With the emergence of Web services (WS) as a widely accepted standard, service-oriented architectures (SOA) increasingly gain attention, both, in research and in industry. The use of Web services facilitates the integration of formerly independent systems. In the future, service-oriented systems can be expected to gain more and more importance in information technology (IT). Designing new systems based on the service-oriented paradigm allows for the development and the deployment of loosely-coupled systems which promise to be more flexible and more easily adaptable when business process requirements change over time. Such quickly evolving systems demand for sophisticated and powerful monitoring in order to ensure correct system behavior and to allow for optimizing system characteristics, like performance or usability. In other fields, mining techniques are applied to gain additional knowledge about systems and data. In this paper, we argue that mining may also be applied to service-oriented systems.

In our previous work we have introduced the idea of Web services Interaction Mining (WSIM) [1,2]. In WSIM we make use of the findings in the fields of data mining and process mining and apply them to the world of Web services and service-oriented architectures. Mining describes the act of examining and analyzing large amounts of (log) data with the ultimate goal of knowledge discovery (KD). Consequently WSIM attempts to apply mining techniques on logs

provided by Web services and SOA infrastructures. We begin by presenting an overview of WSIM to outline the cornerstones of our approach.

1.1 Fundamentals of Web Services Interaction Mining

We currently develop our Web Services Interaction Mining approach with regards to three levels of abstraction that represent three complementary “views” on Web services.

The lowest level of abstraction is the *Web service operational* level. On this level only one Web service is to be examined. Its standing within a service-oriented system is not yet considered. Characteristics of a WS to be examined on this level include, but are not limited to, execution time, service usage, or service availability. Mining on this level can be as detailed as to only analyze single operations of a given WS.

The second level of abstraction is the *Web service interactions* level. On this level the focus is still on a single Web service but with regards to its interactions with other services. In these interactions the given WS may act as either provider or consumer. Therefore, analyses on this level may deal with WS conversation, i.e., in what order are operations invoked by service requestors, or WS composition, i.e., which services are consumed by a given composite WS [3]. Furthermore, a thorough analysis of past interaction with a given WS can reveal dependencies in service-oriented systems and provide the information necessary for an impact analysis when changes to a WS are planned. The result of mining on this level can be an interaction graph as presented in [1].

The highest level of abstraction is the *Web services workflow* level. Here, we attempt to identify workflows, or (business) processes, that are implemented using the functionalities of Web services. In that sense WSIM is positioned one step before WS orchestration. WS orchestration allows for the definition of workflows in an orchestration language such as WSBPEL [4] and for a monitored execution of that workflow using a BPEL engine [5,6]. However, WSIM deals with service-oriented systems that do not yet apply WS orchestration tools. A future application of WSIM on the workflow level might therefore be to generate abstract workflow definitions of discovered processes and thereby facilitate the work of workflow designers.

This paper focuses on the discovery of Web services workflows. We want to emphasize that we are working in environments where no means of WS orchestration, e.g. BPEL, are used.

The remainder of this paper is structured as follows. In Section 2 we examine other fields of mining and discuss their relevance for workflow mining in service-oriented architectures. In Section 3 we analyze logging possibilities in SOA and propose a scale of levels of logging as well as the mining opportunities on these levels. Section 4 describes a service-oriented system we have implemented as a motivating example. In Section 5 we describe the mining experiments that were performed in our case study. A conclusion is given in Section 6.

2 Related Work

In this section we present the current state-of-the-art in other fields of research which have influence on Web services Interaction Mining, the most relevant of which are Web usage mining (WUM) and process mining.

2.1 Web Usage Mining

With the ever growing importance of the World Wide Web (WWW) many researchers have directed their attention to developing means of managing, analyzing, and understanding the vast amounts of data provided on and by the web. These efforts are centered around applying data mining strategies to the content of Web sites, Web server log records, etc. Depending on the data that is analyzed and the desired outcomes of these analyses, web mining can be further broken down into sub-categories: *Web content mining*, *Web structure mining* and *Web usage mining* [7]. Web usage mining, and possibly Web content mining provide methods and approaches that can be integrated and used in Web services Interaction Mining.

Web usage mining is concerned with discovering Web access patterns in Web server logs in order to analyze user browsing behavior [8,9]. Most Web servers provide logging features, which record one log entry for each request received by the server. Such a log entry typically contains the following information [10,11]: the IP-address of the requesting host, a timestamp, the request line, e.g. "GET /index.html HTTP/1.0", and the HTTP status code returned to the client. Another important element a log entry may contain is the *referer* (sic). This is an identifier a client may include in his request indicating where he was referred from when requesting this resource, i.e., where the link was he clicked to request the current resource. Also, if user authentication is required to access a web site, the username is included in all log records of requests sent by that user.

In order to discover access patterns it is first necessary to group individual requests into user sessions, a process called session reconstruction. A session describes a user's visit to a Web site from opening the first page until the site is left. Once sessions have been identified, WUM mines in these sessions for reoccurring patterns. The knowledge gained is used to optimize or customize Web sites. Session reconstruction can be done by collecting requests sent from a given host and applying time constraints [12]. Such time-oriented heuristics pose limitations on the duration of the entire session (h1) and on the duration of a stay on one individual page (h2). The values for h1 and h2 are often fixed, e.g., to 30 and 10 minutes, respectively. Some argue that these values should be flexible and should also incorporate the structure and content of a web site [13,14].

2.2 Process Mining

Workflow mining or *process mining* (the terms may be used interchangeably), describes the effort to discover workflow patterns in a given set of log data. A

workflow is a reoccurring, ordered set of *activities* that are performed in order to fulfil a higher-level task. Each individual execution of that workflow is called an *instance* of the workflow, or a *case*. The goal of process mining is to analyze a so-called *workflow execution log* in order to construct a *workflow model* that best describes *all* recorded instances of that workflow. Log entries must include a workflow identifier and a case identifier, so that the recorded events can be mapped to the according workflow and case.

The greatest challenge in process mining is the construction of models for complex processes. Workflows may contain alternative flows, concurrencies, or loops [15]. Such complex patterns pose challenges to the development of efficient mining algorithms. In [16] van der Aalst et al. present the current state of process mining. In [17] the InWoLvE process mining system is presented which claims to be able to discover a wide range of process models. In [18] an algorithm for “mining exact models of concurrent workflows” is introduced. More issues are discussed in [19,20,15].

2.3 Applicability of Web Usage Mining and Process Mining in WSIM

Web usage mining will have an impact on our work of developing methods to discover workflows in WS-related logs. Comparable to the browsing through Web sites, Web service interactions can also be seen as isolated request-response transactions, embedded in larger-scale sessions made up of numerous such transactions. The records one can keep of WS interactions are somewhat similar to records found in Web server logs. Especially the scenarios addressed in [13,8], i.e., mining in Web server log records in the absence of session information, pose the same problems as the situation we are faced with in WSIM. However, the values used in time-oriented algorithms for user session reconstruction, namely h_1 and h_2 , may be suitable in human user interaction scenarios, where page requests often do happen in intervals of >1 minute. However, in service-oriented systems which are designed for machine-machine interactions, the time elapsed between two interactions may be far less. Therefore, new metrics will have to be found when applying WUM approaches in WSIM. Also, the structure of the system to be examined will have an impact of the values assumed for h_1 and h_2 , which is also one of the key statements in [13]. For example, in a system that performs a workflow of ordering products, time heuristics similar to those in Web usage mining may be suitable. On the other hand, in a system for chemical simulations, in which certain services compute parts of computationally intensive equations, interactions might happen in intervals of only a few milliseconds. Therefore, there are limitations to the possibilities of applying WUM in service-oriented systems.

Another disadvantage in the approaches of Web usage mining that it does not go as far as attempting to construct complete process models. The construction of such a model, however, is the main goal of WSIM on the workflow level. This goal is clearly shared with process mining. A major advantage of process mining is the availability of powerful algorithms, which can clearly be of use in WSIM research.

A draw-back of process mining is, in our opinion, the restrictive assumptions made about the richness of information available in workflow execution logs [16].

From our research we conclude that WSIM is located between WUM and process mining. Web usage mining attempts to find frequent traversal path patterns from a limited amount of log information. Process mining, on the other hand, tries to construct complete workflow models from very rich log data. The log data available in WSIM is comparable to that in Web server logs. However, the more steps are taken, and the earlier WSIM is considered in the development process of a service-oriented system, the richer the log information can become. Therefore, WSIM on the workflow level should both (a) apply Web usage mining techniques on service-oriented systems that provide little log information and (b) present methods for the development of more sophisticated logging in service-oriented systems, so that these systems may later be mined for exact workflow models with the assistance of process mining tools. Consequently, the quality of workflow models that can be discovered using WSIM will depend on the quality and richness of execution log data provided by the underlying system.

3 Web Service Logging

In this section we examine and formalize the logging possibilities in service-oriented architectures. The levels of logging vary in the richness of the information that is logged and in the additional development effort that is needed when implementing the respective features.

Level 1: Standard HTTP-server logging

The most commonly used log formats provided by Web servers are the *Common Log Format* and the *Combined Log Format* [10,11,21]. The log entry recorded in Apache Tomcat when a request is sent to a WS ExampleService may look as follows:

```
127.0.0.1 - - [15/Mar/2005:19:50:13 +0100]
"POST /axis/services/ExampleService HTTP/1.0" 200 819 "-" "Axis/1.1"
```

The log entry contains the requestor's IP address, a timestamp, the request line, the HTTP code returned by the server, i.e., 200 for OK, the size of the returned resource, and the User-Agent, i.e., Axis/1.1. The empty element, i.e. "-", indicates that no referer-information is available. Such log records allow for tracking of the service consumer, determining which service is called how often (but not which operation of the service), or analyzing service failure rates.

This level of logging can be achieved by simply enabling the respective Web server's logging facilities. The service-oriented environment is in no way affected.

Level 2: Logging of complete HTTP requests and responses

Alternatively to Web server logging, an HTTP listener may be used to record all traffic directed to a given port, which allows for logging of the complete HTTP

request and response traffic. This way the involved SOAP messages are also recorded since they are sent as part of the HTTP messages.

Achieving logging on level 2 requires an HTTP level logger that listens to a given port A and redirects to another port B. Also, it is necessary to reconfigure the HTTP server from the original port A, where service calls are expected to be received, to port B, to which the HTTP logger redirects. The service-oriented system is not affected. An existing open-source HTTP listener is Apache TCP Tunnel/Monitor [22].

Level 3: Logging at WS container level

On level 3 the logging is done inside the WS container, e.g., Apache Axis [23]. Unlike on levels 1 and 2, the logging is more flexible and can be configured, e.g., by only monitoring and logging certain WS. Also, HTTP requests not directed at Web services are ignored. The logging itself is achieved using SOAP intermediaries. In Apache Axis such intermediaries are called *handlers*, and they can be added to each deployed Web service's request and/or response flow.

This level of logging can be reached by developing SOAP intermediaries that log all SOAP traffic and integrating them into the service-oriented system. The WS themselves are not affected.

Level 4: Logging client activity

Logging on levels 1 through 3 involves provider-side logging only. Such scenarios are comparable to the situations addressed by Web usage mining where interaction with a single, central server is assumed. On level 4 logging should also be done on the consumer side, e.g., when a WS is a composite service and itself makes calls to other WS possibly deployed on other hosts. Level 4 logging, therefore, greatly expands the opportunities of workflow mining, since a system's activities as a client are also recorded. If a workflow spans over multiple hosts or systems, such interactions may be discovered if level logs are available.

Apache Axis allows for the use of client-side handlers which are invoked whenever the Axis API is used to consume services [23]. The system itself is not affected when level 4 logging is to be implemented.

In [24,25] the authors present a sophisticated logging architecture for service-oriented systems. Their logging facilities reach level 4 logging on the scale we propose. Some of the authors' goals they wish to reach using the log architecture are also addressed by WSIM on the operational level, such as service execution time and Web service usage.

Level 5: Providing for process information

As stated in subsection 2.3, successful mining for exact workflow models requires workflow information in log records. We have also shown that such information is not available in conventional Web server logs. One of the goals of WSIM is to make service-oriented systems fit for process mining by providing suitable logs.

Table 1. Summary of logging features and mining opportunities

Level	Logged information	Logging facilities	Mining opportunities
1	- consumer IP - invoked WS - timestamp - HTTP status code	- Web server	- service utilization - consumer tracking - failure rates
2	- level 1 + - SOAP request & response - timestamps	- HTTP listener & logger	- level 1 + - WS execution time - analysis of SOAP messages
3	- invoked WS & operation - SOAP request & response - timestamps	- WS container - SOAP handlers	- level 2
4	- level 3 + - consumer-side activity	- WS container - SOAP handlers	- level 3 + - client-side activity
5	- level 4 + - workflow information	- level 4 + - Web services	- level 4 + - Web services workflows

Therefore, the highest level of logging in SOA must provide for both a workflow identifier and a case identifier in each interaction that is logged. This in turn requires for additional design and implementation considerations when implementing service-oriented architectures. To be precise, Web services must “co-operate” in a sense that they are able to receive and forward information regarding the workflow they are currently part of. The exchange of that information between Web services can be done by using SOAP headers. Web services should process the workflow information and forward it to other WS it consumes in the process. If such measures are taken when designing Web services the recorded logs will allow the mining of exact models of complex workflows. Our future work includes the development of an API that facilitates the implementation of Web services which allow for level 5 logging.

4 Motivating Example

In this section we present a motivating example showing possible applications of WSIM. Although only simple, the example has been fully implemented and it demonstrates, on a small scale, what could be done with WSIM on a much larger scale in the not so distant future. We first present the scenario and then describe the mining process in Section 5.

The sample service-oriented system we have developed is a collection of 13 WS which are shown in Figure 1. We assume to have ownership over host 1, i.e., access to its logs, while host 2 and 3 are third-party owned.

The workflow is started by a client application making a call to the Order-Service (WSA), which triggers the execution of an application implementing the workflow by using the other Web services. The DataValidationService (WSB) is called to check if the customer is registered and if the given product is available

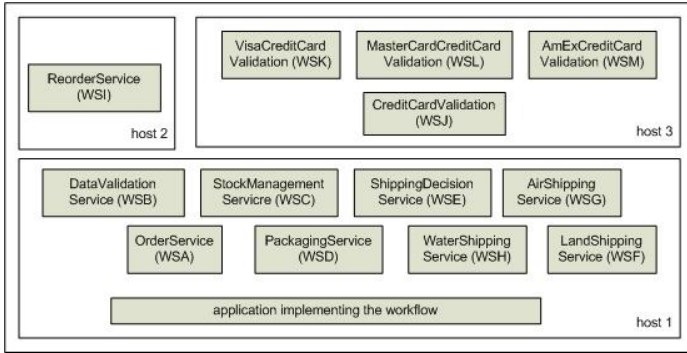


Fig. 1. Case Study Service oriented system

in the desired quantity. If the product is not available the StockManagementService (WSC) is called, which in turn makes a call to the ReorderService (WSI) on host 2. Furthermore, the DataValidationService (WSB) calls a CreditCardValidation (WSJ) on host 3. The CreditCardValidation-WS makes, depending on the company in the credit card information supplied, a call to either the Visa-, the MasterCard-, or the AmEx- Web service. Once order data and customer data have been successfully validated, the PackagingService (WSD) is invoked. When completed, the ShippingDecisionService (WSE) is called which determines the method of shipping and finally invokes one of the ShippingServices. This completes the workflow.

In addition to the example workflow we have implemented logging facilities which reach level 5 on our proposed scale, i.e., the Web services are able to receive, process, and forward workflow information. The SOAP enabling software used was Apache Axis [23], deployed on an Apache Tomcat servlet engine [21]. The logging is done by SOAP handlers on both the client and the server side which are invoked before and after the invocation of each WS. Therefore, all request and response messages are logged. Each log record is of the format

uniqueID - mappingID- targetWS - msgType - SOAPmessage - timestamp

where mappingID is an identifier mapping a SOAP request to its corresponding response, targetWS is the URL of the invoked service, msgType is the type of message, i.e., request or response, and SOAPmessage is the complete SOAP message that was sent.

5 The Mining Process

In our experiments the workflow was executed several times so that every possible flow of the process was performed at least once, e.g., in at least one instance of the workflow products were unavailable and the ReorderService had to be called. Once a sufficient amount of data was collected the log records were pre-processed to a format more suitable for data mining. For example, the recorded

SOAP messages were analyzed for workflow information in their headers and the invoked method was retrieved. Also, corresponding log records, i.e., request and corresponding response, were combined into one log record, so that one call to a Web service was represented by one record. The log format after data pre-processing was as follows:

```
workflowID - instanceID - targetWS - operation - SOAPrequest -
requestTimestamp - SOAPresponse - responseTimestamp
```

5.1 Scenario

In order to demonstrate the benefits of logging on level 5 consider the following scenario. The formerly independent hosts 1 - 3 in our case study are now under ownership of one virtual organization. Such a situation might occur when companies merge or when the independent hosts were previously managed by separate departments and are now put under the control of one central IT-department or outsourced. In such a situation it is beneficial when workflow information is available for processes that are performed using services deployed on (formerly) independent hosts.

5.2 Data Pre-processing for ProM

In our experiments we used ProM as the process mining tool [26]. ProM requires an input file in XML format which contains the information concerning processes and their instances. It should be noted that ProM works with event-based data in order to be able to discover complex workflow patterns. This means that not activities are recorded but rather events. The simplest events regarding an activity are **start** and **complete**. Therefore, the Web service execution logs have to be transformed into an event-based view. The XML format of a ProM - input file is roughly the following:

The root element is the `<WorkflowLog>` element and it has a number of `<Process>` sub-elements, each encapsulating execution data of *one* process, or workflow. A `<Process>` element has a number of `<ProcessInstance>` child elements. A `<ProcessInstance>` has numerous `<AuditTrailEntry>` child elements. An `<AuditTrailEntry>` represents one log record and contains an identifier of the activity, the event-type and a timestamp.

The pre-processed logs described in the previous subsection were traversed for log records belonging to a given workflow, i.e., “orderprocess” in our case. Because ProM requires log entries to be sorted by process instances, a list of instances was constructed, which was then traversed until all instances were processed. As an activity identifier we used the values `targetWS-operation` of the pre-processed logs. Furthermore, the receiving of a SOAP request was taken as the **start** event of an activity, the sending of the SOAP response was considered the **complete** event. Also, since both, provider- and consumer-side interactions were logged, care had to be taken that interactions were only considered once, in cases of provider and consumer residing on the same host.

5.3 Mining Results

The procedure of creating a ProM - compatible workflow log was performed several times with varying amounts of log data available. In one session only the logs collected at host 1 were used. This scenario may very well appear in the real world when system owners decide to upgrade their infrastructure to a BPEL - enabled environment.

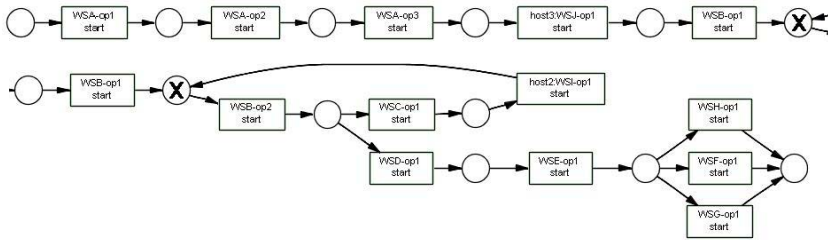


Fig. 2. Process model of host 1

The log data first had to be cleaned from interactions which were recorded twice, i.e., by both provider-side and consumer-side handlers. For all interactions with third party hosts the consumer-side entries made at host 1 were used to capture these interactions. From the information extracted from the logs the workflow model in Figure 2 was constructed using ProM. The workflow model turned out to be complete and correct.

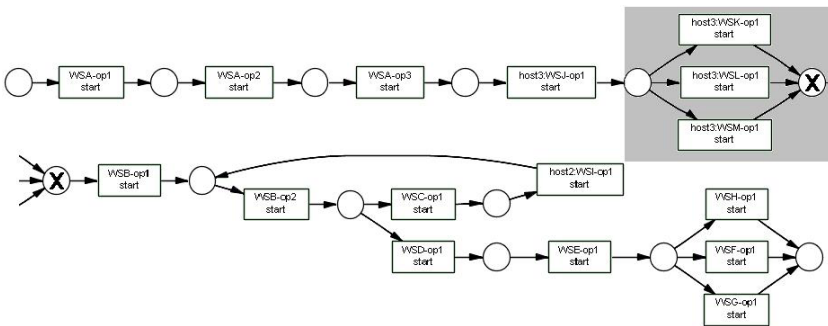


Fig. 3. Process model of all hosts

Note that the names of WS and operations were abbreviated in Figures 2 and 3 and only start-events were considered in order for the models to be displayable here. The abbreviations correspond to those used in Figure 1. Furthermore, the original images created by ProM had to be “cut in half”. The circles marked X are where the two halves should be joined.

In another experiment, we considered all logs collected on the three hosts. This scenario is imaginable when formerly independent hosts come into ownership of one virtual organization. Again, the data was pre-processed and cleaned from double recorded interactions. The workflow model in Figure 3 was constructed which now includes the workflow performed on host 3. Figure 3 demonstrates the benefits from providing logs at a level 5 of our scale. The portion of the model highlighted is the additional knowledge gained from including logs from host 3 in the mining process.

6 Conclusion and Future Work

In this paper we have presented our approach of Web services Interaction Mining (WSIM) with a focus on mining for Web services workflows. We have examined related work, of which we specifically point out Web usage mining and process mining. Since the possibilities of process mining in SOA seem limited using traditional logging facilities like Web server logs, we have presented a classification of service-oriented systems by the richness of the log data they provide. Depending on the level of logging, different methods of WSIM may be applied to a system. If level 5 logging is implemented, a service-oriented system may be mined for complete workflow information using already existing process mining tools such as ProM. As a case study we presented such a system and described the process of processing the log data to create logs understandable by ProM. We also showed the results achieved by mining the logs for workflows.

Since level 5 logging cannot be assumed in many service-oriented systems, our future work will be directed at developing strategies and algorithms for WSIM in systems which provide less log information. Specifically, we will examine WUM strategies and adapt them to the needs when mining in logs provided by service-oriented systems.

Our current research effort goes in two directions. On the one hand, we examine the possibilities of workflow mining in environments with limited log information. In such environments workflow mining may be impossible, which is why we currently focus on discovering frequent interaction patterns rather than complete workflows. On the other hand, we are examining strategies to generate workflow information in incomplete logs which may later allow for workflow mining. The most promising strategy at the moment seems to be that of session reconstruction in service-oriented systems.

References

1. Gombotz, R., Dustdar, S., Baina, K.: Web Services Interaction Mining. Technical Report TUV-1841-2004-16, Vienna University of Technology (2004) <http://www.infosys.tuwien.ac.at/Staff/sd/papers/TUV-1841-2004-16.pdf>.
2. Gombotz, R., Baina, K., Dustdar, S.: Towards Web Services Interaction Mining Architecture for e-commerce Applications Analysis. In: Proceedings of the Conference on E-Business and E-Learning. (2005)

3. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services - Concepts, Architectures and Applications. Springer-Verlag (2004)
4. OASIS: OASIS Web Services Business Process Execution Language (WSBPEL) TC (2004) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.
5. ActiveBPEL: ActiveBPEL Engine - Open Source BPEL Server (2004) <http://www.activebpel.org>.
6. Oracle: Oracle BPEL Process Manager (2004) <http://www.oracle.com/technology/products/ias/bpel/index.html>.
7. Patricio Galeas: Web Mining (2005) <http://www.galeas.de/webmining.html>.
8. Wu, K., Yu, P., Ballman, A.: SpeedTracer: A Web usage mining and analysis tool. IBM Systems Journal **37** (1998)
9. hypKNOWsis: hypKNOWsis ... making sense of your data (2004) <http://www.hypknowsys.org>.
10. Apache Software Foundation: Log Files - Apache HTTP Server (2005) <http://httpd.apache.org/docs-2.0/logs.html>.
11. Microsoft TechNet: IIS Logging Server Activity (2004) <http://www.microsoft.com/technet/archive/IIS3/iischp7.mspx> .
12. Spiliopoulou, M., Mobasher, B., Berendt, B., Nakagawa, M.: A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis. INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications (2003)
13. Zhang, J., Ghorbani, A.A.: The Reconstruction of User Sessions from a Server Log Using Improved Time-oriented Heuristics. In: Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04). (2004)
14. Berendt, B., Mobasher, B., Nakagawa, B., M.Spiliopoulou: The Impact of Site Structure and User Environment on Session Reconstruction in Web Usage Analysis. In: Proceedings of the 4th WebKDD 2002 Workshop at the ACM-SIGKDD Conference on Knowledge Discovery in Databases. (2002)
15. van der Aalst, W., Weijters, A.: Process mining: a research agenda. Computers in Industry **53** (2003) 245–264
16. van der Aalst, W., van Dongen, B., Herbst, J., Maruster, L., G.Schimm, Weijters, A.: Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering **47** (2003) 237–267
17. Herbst, J., Karagiannis, D.: Workflow mining with InWoLvE. Computers in Industry **53** (2003) 245–265
18. Schimm, G.: Mining exact models of concurrent workflows. Computers in Industry **53** (2003) 265–281
19. Pinter, S., Golani, M.: Discovering workflow models from activities' lifespans. Computers in Industry **53** (2003) 283–296
20. Cook, J., Du, Z., Chongbing, L., Wolf, A.: Discovering models of behaviour for concurrent workflows. Computers in Industry **53** (2003) 297–319
21. Apache Software Foundation: The Jakarta Site - Apache Jakarta Tomcat (2004) <http://jakarta.apache.org/tomcat/>.
22. Apache Software Foundation: Apache SOAP Documentation: User's Guide (2004) <http://ws.apache.org/soap/docs/>.
23. Apache Software Foundation: WebServices - Axis (2004) <http://ws.apache.org/axis/>.
24. Serra, S., Campos, L., Pires, P.: A Data Mart Approach for Monitoring Web Services Usage and Evaluating Quality of Services. In: XVIII Brazilian Symposium of Data Bases. (2003)

25. Serra, S., Campos, M., Pires, P., Campos, L.: Monitoring E-Business Web Services Usage through a Log Based Architecture. In: Proceedings of the IEEE Interantional Conference on Web Services (ICWS'04. (2004)
26. Technische Universiteit Eindhoven: Process Mining Reserach (2004) <http://www.processmining.org>.

Preface (ENEI 2005)

The aim of the Enterprise and Networked Enterprises Interoperability (ENEI 2005) workshop, organized in the framework of the 3rd International Conference on Business Process Modelling (BPM 2005), was to bring together researchers and practitioners to present and discuss the variety of practices, novel methods, automated support, architectures and technologies that may improve the ability of an enterprise “as a whole” to easily, correctly and safely render its existing as well as its future applications and software communication and cooperation.

In response to the ENEI 2005 call for papers, 23 regular papers and 5 short papers were submitted from 13 countries. Each paper was reviewed by three referees and 13 regular papers and 2 short papers were selected. This volume contains the revised versions of the selected papers organized according to the workshop sessions schedule.

It is a fact that enterprises need to communicate and collaborate and that networked business encounters recurrent difficulties due to the lack of interoperability between enterprise systems. Most of the research effort focused on studying how to make companies collaborate and communicate in the most effective and seamless way. Indeed, computer-supported integration and interoperability of enterprise applications and software have a growing need for language standardization in order to face the increased complexity of the enterprise ground business. The first session reported on related work in *Enterprise Modelling Languages for Enterprise Interoperability* from an industrial point of view, from a European project resulting on a unified enterprise modelling language and from international standardization initiatives. The papers focused on different and complementary issues related to a common enterprise modelling language. The paper “Exchange of Business Process Models Using the POP* Meta-model” presents a *proof of concept* of the POP* language. The paper “UEML 1.0 and UEML 2.0: Comparison, Benefits and Problems” focuses on the *methodology* to be used to define such a common language. The paper “Facilitating Interoperability: A Cross-Analysis of the Language UEML and the Standard ISO/DIS 19440” compares two candidate languages to be used as a common enterprise modelling language.

The interoperability problem is more crucial when one considers networked and extended enterprises. Indeed, enterprises are faced today with a situation similar to that of software engineering environments a couple of years ago: that is, enterprises are provided with collections of software coming from heterogeneous applications and software tools that were neither designed nor developed to favor their interaction and their cooperation. The second session focussed on Networked Enterprises Interoperability with contributions on architecture and enterprise reference models for collaborative networks that can be created among companies, people and societies in order to generate shared knowledge and wealth. Four papers were presented in this session: the first two were on frameworks and architectures for virtual enterprises, focusing on integration and interoperability. The proposed solutions pointed out key issues: (i) interoperability assessment and management, throughout an architecture

recognizing the concept of “Systems of Systems” (SoS), that is, following the author, the main abstraction for understanding the behaviors of networked enterprises; (ii) the concept of agents could be used to support all the Virtual Enterprise Lifecycle and provides the corresponding conceptual agent-based platform and hints for implementation. The other two papers describe more specific situations: distributed organization of scientific environments, and integration of CRM (customer relationship management) and SCM (supply chain management). The first one provides a possible organization for performing, in a distributed way, scientific experiments involving several laboratories. The second one points out the necessity of defining “business components” providing abstraction over the running systems such as SCM and CRM.

Due to the confusion of model contents and use, the reuse of existing enterprise models is limited. The inability of various aspects of an enterprise to be aware of its existing models further exacerbates this problem. One of the key drawbacks is the definition of services requirements for networked organizations to collaborate. The third session about Interoperability Requirements and Models dealt with such concerns in order to study the different frameworks for taking into account enterprise services heterogeneity. This session comprised five papers.

Some experiments of enterprise and networked enterprise interoperability are currently ongoing. The last session of the workshop encompassed three papers and dealt with Interoperability Applications and Case Studies to demonstrate the feasibility of technological solutions in different application domains (health care supply chain, travelling services, etc.).

It has been a great pleasure to work with the members of the Programme Committee and the additional reviewers who dedicated their time to review the submitted papers: we are indebted to all of them. We are also indebted to G. Berio (U. of Torino, Italy) and M. Petit (U. of Namur, Belgium), who moderated and reported on some of the workshop sessions, as we are indebted to the INTEROP network of excellence (European FP6 IST-508-011, <http://www.interop-noe.org>) for its support.

Nacer Boudjlida
Hervé Panetto

Organization

Workshop and Programme Committee Co-chairs

Boudjlida, Nacer
Panetto, Hervé

UHP Nancy 1, LORIA, France
UHP Nancy 1, CRAN, France

Organization Committee Members

Baïna, Salah
Chen, Dong

UHP Nancy 1, CRAN, France
UHP Nancy 1, LORIA, France

Programme Committee

Albani, Antonia
Baina, Karim
Bellahsène, Zohra

University of Augsburg, Germany
ENSIAS, Morocco
University of Montpellier, LIRMM,
France

Berio, Giuseppe
Boudjlida, Nacer
Boufaïda, Mahmoud

University of Torino, Italy
UHP Nancy 1, LORIA, France
University Mentouri, Constantine,
Algeria

Dubois, Eric
Eder, Johann
Jeusfeld, Manfred
Krogstie, John

CRP Henri Tudor, Luxembourg
Klagenfurt University, Austria
Tilburg University, The Netherlands
Norwegian Institute of Science and
Technology, Norway

Lenzerini, Maurizio

Università degli Studi di Roma "La
Sapienza", Italy

Molina, Arturo
Oquendo, Flavio

Tecnológico de Monterrey, Mexico
University of South Brittany at Vannes,
France

Ortiz, Angel
Panetto, Hervé
Perrin, Olivier
Petit, Michaël
Skaf-Molli, Hala
Tari, Zahir
Tsalgatidou, Aphrodite
Velardi, Paola

UP Valencia, Spain
UHP Nancy 1, CRAN, France
University of Nancy 2, LORIA, France
University of Namur, Belgium
UHP Nancy 1, LORIA, France
RMIT University Melbourne, Australia
NKU Athens, Greece

Whitman, Larry

Università degli Studi di Roma "La
Sapienza", Italy
University of Wichita, USA

Additional Referees

Berardi, Daniela

Castano, Sylvana

Craske, Gregory

Di Leva, Antonio

Horst, Pichler

Koncilia, Christian

Lillehagen, Frank

Mecella, Massimo

Molli, Pascal

Perepletchikov, Mikhail

Rey, Christophe

Sotiropoulou, Anya

Theotokis, Dimitrios

Tsagkani, Christina

Urso, Pascal

Zarour, Nacereddine

Zelm, Martin

Università degli Studi di Roma "La
Sapienza", Italy

University of Milan, Italy

RMIT University, Melbourne, Australia

University of Turin, Italy

Klagenfurt University, Austria

Klagenfurt University, Austria

Troux, Norway

Università degli Studi di Roma "La
Sapienza", Italy

UHP Nancy 1, LORIA, France

RMIT University, Melbourne, Australia

University of Montpellier, LIRMM,
France

NKUAthens, Greece

NKUAthens, Greece

NKUAthens, Greece

UHP Nancy 1, LORIA, France

University Mentouri, Constantine, Algeria

CIMOSA, Germany

Exchange of Business Process Models Using the POP* Meta-model

Reyes Grangel¹, Ricardo Chalmeta¹, Stefan Schuster², and Iñaki Peña²

¹ Grupo de Investigación en Integración y Re-Ingeniería de Sistemas (IRIS),
Dept. de Llenguatges i Sistemes Informàtics, Universitat Jaume I,
Campus del Riu Sec s/n, 12071 Castelló, Spain

{grangel, rchalmet}@uji.es

² European Software Institute (ESI), Parque Tecnológico de Zamudio # 204,
48170 Zamudio, (Bizkaia) Spain

{Stefan.Schuster, Inaki.Pena}@esi.es

Abstract. Enterprise Modelling, in general, and Business Process Modelling, in particular, have been used for decades for different purposes and with interesting results. However, a variety of problems can be identified in this context and many enterprises find it difficult to leverage the full potential and benefits of these technologies. One of the most important problems in this sense is the lack of interoperability among enterprises at the modelling level. Quite a lot of efforts has been carried out in this domain to improve enterprise interoperability at this level. The development of the POP* meta-model is one of these initiatives, which aim to establish a meta-model and a corresponding methodology that enable enterprises to exchange their enterprise models, despite the fact that they use different Enterprise Modelling Tools.

In this paper, we present a ‘proof of concept’ of the POP* meta-model focused on the process dimension, which is expected to further our understanding of how this meta-model can be used to exchange different business process models among the partners in networks of collaborative enterprises. Moreover, the work performed in this ‘proof of concept’ has been a valuable aid to validate and improve the development of the POP* meta-model.

1 Introduction

Enterprise Modelling is defined in [1] as the art of ‘externalising’ enterprise knowledge, that is to say, by representing the enterprise in terms of its organisation and dimensions (process, decision, product, resource, and so forth) [2]. Therefore, Enterprise Modelling enables enterprises to gain a much deeper knowledge and understanding of their business so that their objectives can be aligned with the market needs.

In the 70s, the first concepts of modelling were applied to the computer systems (E/R Model, DFD, and so forth), but the concept of Enterprise Modelling appeared in the USA at the beginning of the 80s, with the Computer Integrated Manufacturing (CIM) initiative. Examples of this initiative are the Integrated

Computer Aided Manufacturing (ICAM) Project carried out by the US Air Force or the Integrated Computer Aided Manufacturing-International (CAM-I) Project. In the mid 80s, different Enterprise Modelling Languages, such as GRAI or CIMOSA, emerged in Europe. In addition, numerous commercial tools appeared in the 90s to lend support to a great number of different modelling languages (ARIS ToolSet, FirstSTEP, METIS, KBSI Tools, MO²GO, e-MAGIM, and so forth.) [2].

Today, the use of Enterprise Modelling is widely extended and many languages, methodologies and tools related to Enterprise Modelling exist, even for modelling Virtual or Extended Enterprises [3]. Enterprise Modelling Languages provide constructs with which to describe and model peoples' roles, operational processes and functional contents, as well as support information, and production and management technologies. However, integration of the models generated with these languages is complicated, since tools for exchanging models generated with different languages do not exist [4,5,6,7]. In summary, the main problems with respect to Enterprise Modelling can be seen as lying along two axes [8]:

- **Horizontal:** the lack of interoperability between Enterprise Modelling Languages and their corresponding Enterprise Modelling Tools. Almost all languages of this sort are proprietary specifications and can only be implemented with specific tools designed for this purpose. This problem complicates the interoperability of enterprises at the conceptual level. The main solutions provided by the research community to address this problem are focused on defining a common exchange format. This was, for instance, the goal of the UEML Project [7] and one of the objectives of the INTEROP [6] and ATHENA [4] Projects.
- **Vertical:** the weak connection between enterprise models and the generation of software is one of the major reasons why enterprises develop only a few models, which, moreover, are rarely updated and are therefore not very successful in accomplishing their initial purposes. Initiatives, such as MDA [9] promoted by OMG and MDI within INTEROP [6], are intended to solve this kind of problems.

These same problems can also be observed in the business process context. The number of modelling techniques and tools available for supporting Business Process Modelling is growing rapidly, because of the increasing popularity of business process orientation [10]. In recent years, many advantages of using Business Process Modelling have been pointed out [11].

Nevertheless, collaborative enterprises face a number of problems when attempting to harvest the benefits of Business Process Modelling. A collaborative enterprise is an enterprise where teams work together across boundaries, e.g. life-cycle phases, sharing results and knowledge to improve their common understanding and enable better performance and higher quality results [12]. The main reason for this situation is the large number of techniques and tools [10] that support and that are used for Business Process Modelling; as a result, collaborative enterprises find it difficult to exchange business process models in an efficient way.

Taking the problem of interoperability as its main inspiration, the objective is to achieve a common format, like POP* or UEMML, which are valid initiatives allowing enterprises to exchange different kinds of models and to set up an environment in which existing models can be reused [4,5,6,7]. In particular, within the framework of the ATHENA Project [4], the POP* methodology was developed with the aim of solving this kind of problems and improving enterprise interoperability. In this context, then, this paper presents the work carried out in the ‘proof of concept’ of the POP* meta-model in order to validate it and it describes how the POP* meta-model could be used to exchange business process models among different partners from a process-oriented point of view.

The paper is organised as follows. Section 2 gives a brief description of the ATHENA Project as the framework in which this research was carried out, and also discusses the main issues regarding the POP* meta-model and especially its process dimension. Section 3 describes the research work performed and the main results obtained in the ‘proof of concept’ of the POP* meta-model. Finally, the main conclusions are outlined in section 4.

2 ATHENA Project

ATHENA (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications) is an Integrated Project sponsored by the European Commission in support of the Strategic Objective ‘Networked businesses and government’ set out in the IST 2003-2004 Work Programme of FP6 [4]. ATHENA aims to make a major contribution to interoperability by identifying and meeting a set of inter-related business, scientific and technical, and strategic objectives. In ATHENA, different Research and Development projects are executed in an integrated way. The research work presented in this paper was developed within the framework of one of these projects, called A1, and which focuses on ‘Enterprise Modelling in the Context of Collaborative Enterprises’.

The overall goal of this project is the development of methodologies, core languages and architectures as models, model-generated workplaces, services and execution platforms for establishing collaborative on-demand Extended Enterprises and Networked Organisations.

2.1 POP* Meta-model

One of the main goals of the A1 Project is to develop a methodology that provides a set of basic modelling constructs to support model exchange in the context of collaborative enterprises. The methodology includes [12]:

1. The **POP* meta-model**, which describes the set of basic modelling constructs defined and their relationships.
2. The **guidelines**, which describe the management and use of the POP* meta-model.

With respect to this goal and business process orientation, the work performed in the project has similar objectives, but at the same time a different scope, to other approaches like UEML [7] or BPDM [13]. Although the development of the POP* meta-model is based on the adoption of a holistic point of view of an enterprise which takes into account its different dimensions, that is to say, process, organisation, decision, and so forth, this first version is developed in a more comprehensive manner and focuses on the process dimension.

Moreover, POP* was developed taking into account how enterprises need to establish flexible relationships with other partners in order to achieve some competitive advantage, and also with a top-down approach that allows for definition of the constructs needed to depict the particular features of this kind of enterprises. On the other hand, the POP* meta-model was also developed with a bottom-up approach, which involved reviewing some of the most important Enterprise Modelling Languages like IEM, EEML, GRAI, and so forth, and as a result it covers the common concepts identified in these languages. However, the POP* meta-model is neither the merge of the meta-models of these specific Enterprise Modelling Languages, nor the addition of them, but the mapping of the main constructs of these languages in order to identify common concepts and to avoid redundancies. In this sense, the POP* meta-model is a valid mechanism with which to exchange enterprise models among partners in a collaborative enterprise that use different enterprise modelling platforms and languages.

Therefore, the POP* meta-model is a first, but necessary, step in order to achieve enterprise interoperability at the conceptual level. Furthermore, the POP* meta-model will be useful for developing the architecture specification of the Modelling Platform for Collaborative Enterprise (MPCE) within the ATHENA Project. This platform will facilitate the exchange of different kinds of enterprise models, based on the POP* meta-model, and allow them to be managed in a better fashion.

A thorough explanation of the POP* meta-model and its corresponding methodology can be found in [12]. This work includes the description of the POP* meta-model in its first version, with the dimensions defined so far:

- **Process dimension:** representing the activities and tasks carried out in an enterprise and the different objects that are needed to perform them.
- **Organisation dimension:** expressing the formal and informal organisational structures of an enterprise, as well as the different stakeholders and relationships that form part of this organisation.
- **Product dimension:** representing the products or services that an enterprise offers to the market.
- **Decision dimension:** expressing the decision-making process and the structure needed in an enterprise to perform it.
- **Infrastructure dimension:** depicting the ICT infrastructure of an enterprise.

Furthermore, it also provides guidelines illustrating the management and potential use of the POP* meta-model in a cross-organisational setting. The main goal of these guidelines is to explain how the POP* meta-model can be

used to exchange enterprise models among or inside enterprises that use different Enterprise Modelling Tools.

2.2 Process Dimension

The process dimension of the POP* meta-model, shown in Fig. 1, is concerned with the activities and elements needed to enact and execute processes in a collaborative enterprise. Its objective is to provide the basic constructs with which to model the tasks and the main enterprise objects that participate in these tasks with different roles, such as input, output, control, and so forth. The process dimension also supports the representation of the process flow, as well as conditions or associated decisions.

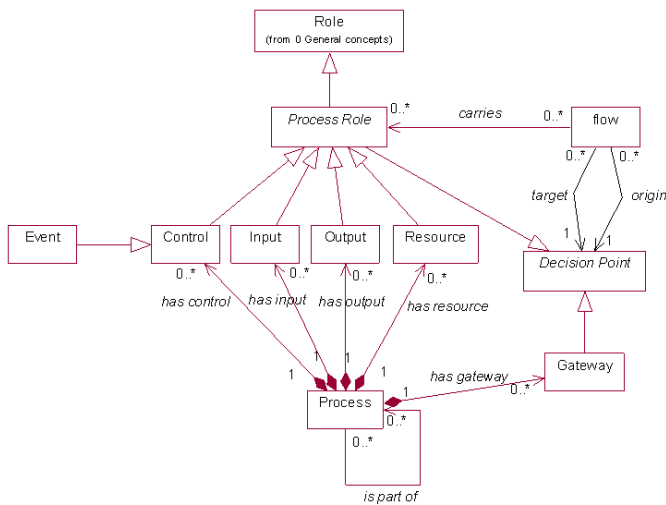


Fig. 1. POP* meta-model: process dimension

In this section, we present a brief description of the main constructs in the process dimension of the POP* meta-model (see Fig. 1). A complete description of these constructs can be found in [12].

- **Process:** this represents a task or an activity performed in an enterprise. A Process can be derived into different subprocesses in order to depict the desired level of detail.
- **Role/Process Role:** this is used to express the function of the diverse enterprise objects in the execution of a Process. Consequently, the subclasses of the Process Role are: Control, Input, Output and Resource.
- **Decision Point:** this depicts a conditional point used to solve the process flow and continuation, i.e., the process sequence. A Decision Point can be a Process Role, which can have an object attached to it, or a Gateway, which is a true decision point without attached object, and is owned by a process.

- **Flow:** this construct represents the connection of Processes across two Decision Points, which can be either Gateways or Process Roles played by different enterprise objects.

3 ‘Proof of Concept’ of the POP* Meta-model

Within the framework of the above-mentioned ATHENA Project, this paper describes the work performed in the ‘proof of concept’ of the POP* meta-model. The main objective of this research work is to demonstrate that the POP* meta-model is well defined, as it provides a common and standard language to exchange models among different Enterprise Modelling Tools.

3.1 Process Description

Our demonstration method includes two main steps, as shown in the diagram in Fig. 2. First, an existing model compliant with a specific Enterprise Modelling Tool (MO²GO) [14] is transformed by hand into a POP* model using a UML Profile 2.0. Second, the POP* model is imported into different Enterprise Modelling Tools (GraiTools [15] and Metis [16]).

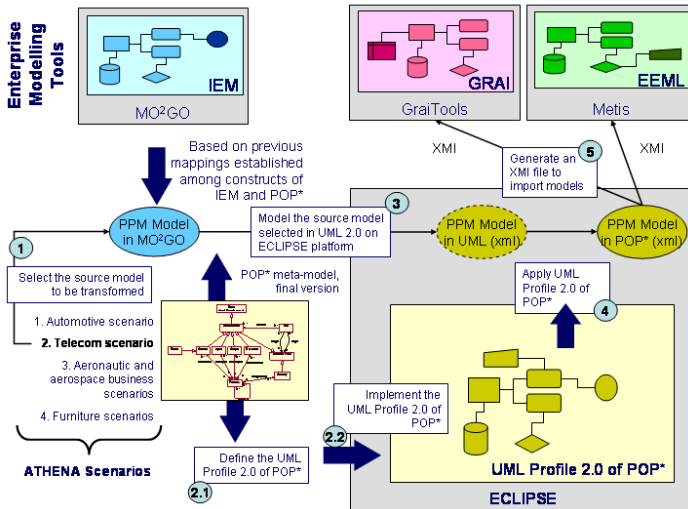


Fig. 2. Diagram showing tasks performed in the ‘proof of concept’ of the POP* meta-model

In order to achieve this goal, a UML Profile 2.0 of the POP* meta-model was implemented using the ECLIPSE platform. UML Profiles 2.0 is a mechanism that allows the metaclasses of an existing meta-model to be extended, in order to adapt it for different purposes. Therefore, this mechanism includes the ability

to tailor the UML meta-model to different platforms (such as J2EE or .NET) or domains (such as real-time or BPM) [17].

In our case, we will use this mechanism to define a UML Profile 2.0 of the POP* meta-model with the aim of carrying out a ‘proof of concept’ of POP*. The idea is to extend the UML meta-model within a specific domain by means of our profile. This profile can then be used to model collaborative enterprises according to the POP* meta-model.

Therefore, the first task to be carried out in this process is the definition of the UML Profile 2.0 of the POP* meta-model. Following the recommendations given in [18], the main steps involved in defining this profile are:

1. To include one stereotype for each element of the POP* meta-model in a ‘profile’ package.
2. To specify what elements of the UML meta-model are extended by the stereotypes.
3. To define the attributes of the POP* meta-model as tagged values.
4. To define the constraints of the domain.
5. To implement the profile defined by using the ECLIPSE UML 2.0 plug-in.

On the other hand, the remainder tasks shown in Fig. 2, which are needed to complete the ‘proof of concept’, are explained in more detail in the following section.

3.2 Work Performed

The ‘proof of concept’ of the POP* meta-model was performed in order to validate it and to demonstrate a real application of the POP* meta-model as an exchange format. Thus, the work performed and explained in this section can be useful to gain a better understanding of how the POP* meta-model could be used to exchange business process models. This work was carried out according to the steps proposed by the guidelines defined in [12] for applying and managing the POP* meta-model. In what follows, the main steps performed and illustrated in Fig. 2 are presented.

STEP 1. Select the source model to be transformed. For the ‘proof of concept’ we selected one of the ATHENA scenarios, from the Telecom sector. In particular, the scenario is related to the Product Portfolio Management Process (PPM). We used the PPM scenario modelled in MO²GO, and we chose only a part of this model in order to ensure that the work could be performed in a short amount of time.

The part of the PPM model selected was the ‘WIBAS¹ Project development’ process (see Fig. 3), because it illustrates some crucial POP* concepts. It includes almost all the elements that can be represented in a MO²GO model, and it is sufficiently complex to demonstrate the use of POP* as an exchange format.

¹ WIBAS is the name of a particular product development project.

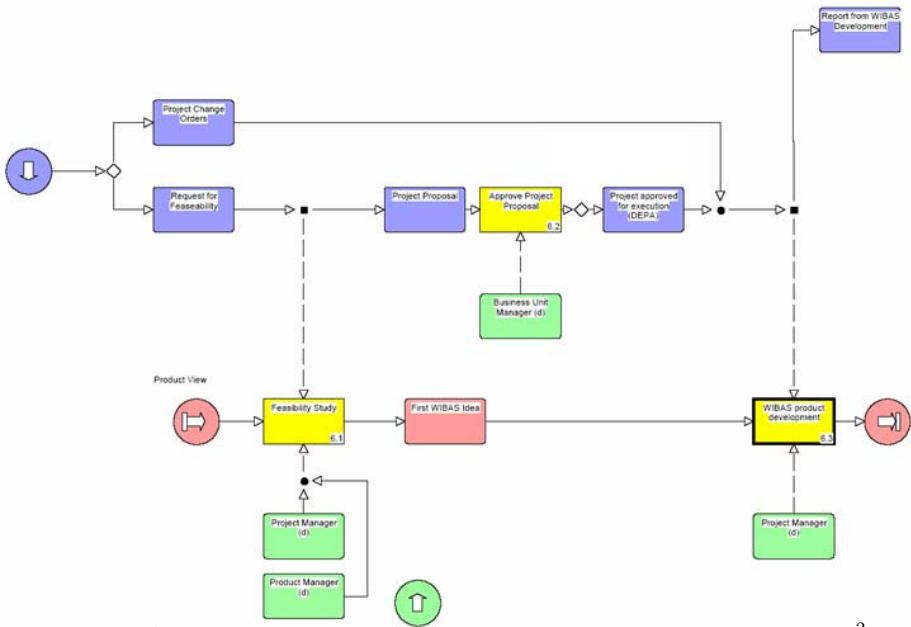


Fig. 3. ‘WIBAS project development’ process of PPM model developed in MO²GO

STEP 2. Define and implement the UML Profile 2.0 of POP*. The result of the tasks performed in this step is a description of the UML Profile 2.0 of the POP* meta-model. This profile could be used as a basis for further implementation of POP* as an Enterprise Modelling Language. The profile identifies a subset of the UML meta-model elements but does not remove any of the UML meta-model functionalities, and therefore all the utilities of UML remain available for the final users.

Three components are needed to create UML profiles: stereotypes, restrictions and tagged values. **Stereotypes** are defined by their names and the elements of the meta-model that are associated to them. They establish the features that designers assign to the elements that are extended by the profile. **Restrictions** are used to establish conditions over the stereotyped elements, and **tagged values** are additional meta-attributes that are associated to a meta-class in the extended meta-model. This profile specification was developed in accordance with the latest version of the Unified Modelling Language, UML 2.0 [17,19].

STEP 3. Model the source model selected in UML 2.0. Prior to modelling, it is necessary to select UML diagrams that are useful for our ‘proof of concept’. We focused on the most expressive UML diagrams that can be used for business processes modelling, which are class and activity diagrams. This step was carried out using the Rational Rose modeller from the Rational division of IBM on ECLIPSE platform. This tool was chosen in order to take advantage of the ECLIPSE UML 2.0 plug-in and to support advanced UML profile 2.0 management and XMI 2.0 interchange.

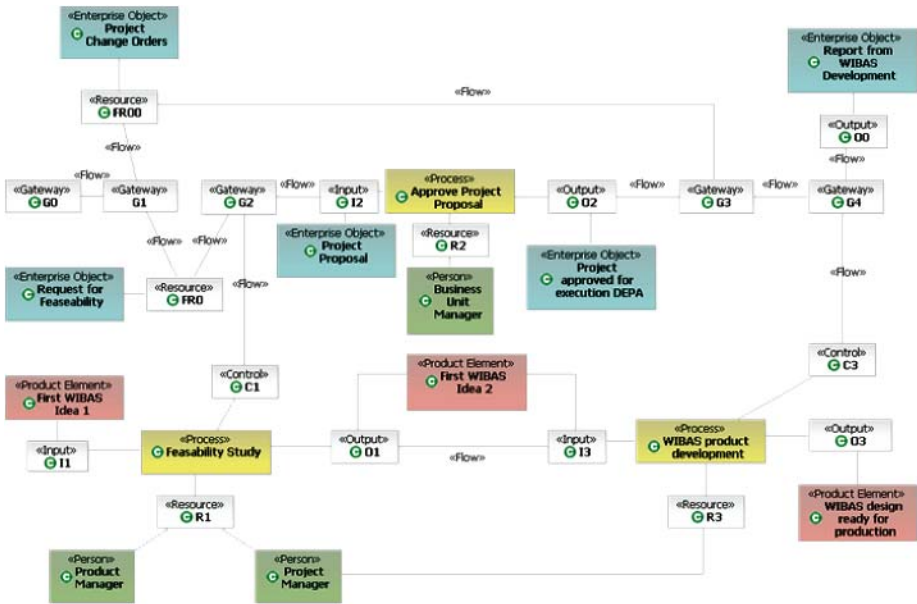


Fig. 4. ‘WIBAS project development’ process of PPM model developed in POP*

STEP 4. Stereotype the model developed in UML 2.0 with the UML Profile 2.0 of POP*. Using the UML Profile 2.0 of POP* thus implemented, all components of the model previously developed in UML 2.0 were extended using stereotypes (see Fig. 4). In this way we obtained a full, semantically equivalent model but which is now UML 2.0 compliant, that is, it is fully compliant with XMI 2.0 and therefore easily interchangeable.

Elements in the MO²GO diagram were replaced by POP* concepts, but obviously translating native models to POP* involves more than simply replacing each element in the native models by its corresponding element in POP*. The translated model should follow the rules that define how POP* concepts can be related (that is, the syntactic rules defined by the POP* meta-model). This could entail having to include new elements, as can be seen in Fig. 5. For example, in order to develop the class diagram of the translated POP* model:

- Processes were defined to include their interfaces with the outside world. These are specialisations of the Process Role: Input, Control, Output and Resource.
- ‘Split’ or ‘Join’ in the MO²GO diagram were transformed into Gateways in the POP* class diagram.
- Flows were stereotyped as associations in order to simplify the diagram and give it more expressiveness.
- According to the POP* meta-model, Flows can connect only Decision Points (this means Gateways or Process Roles). For example, we cannot connect two Processes (or an Object with a Process) directly by means of a Flow.

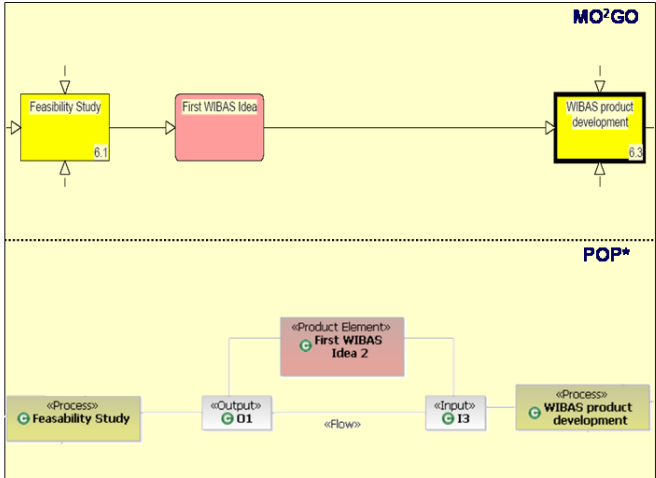


Fig. 5. Modelling of flows in POP*

STEP 5. Generate an XMI file to be imported. Finally, the objective is to generate an XMI file of the POP* model generated in the step 4 by means of the capabilities from the ECLIPSE platform, which will be imported into different Enterprise Modelling Tools, like GraiTools or Metis, for instance.

3.3 Results and Lessons Learned

The ‘proof of concept’ of the POP* meta-model fulfilled its initial purpose. It assisted in the final development of the POP* meta-model, clarifying some concepts of the meta-model and proving that it is possible to transform models developed in different Enterprise Modelling Tools by means of POP*. Moreover, the tangible results obtained in this research work are:

- The definition of the UML Profile 2.0 of the POP* meta-model, and its implementation in the ECLIPSE UML 2.0 plug-in.
- A real-use case modelled in POP*, based on a source model developed in a specific Enterprise Modelling Tool.
- The XMI files of the real-use case modelled in POP* that can be imported into other Enterprise Modelling Tools.

Finally, the main lessons learned in performing the ‘proof of concept’ of POP* can be summarised in the following points:

- Major problems were encountered in understanding the source model, especially because it was not developed by one of the team members. In spite of knowing the constructs of a specific Enterprise Modelling Language, the modelling process is sometimes subjective and hence it is hard to interpret a source model that is to be transformed into another model. To this regard, POP* can be useful since it establishes a mapping among the constructs of the most important Enterprise Modelling Languages.

- When transforming a source model into another one by means of POP*, it will sometimes be necessary to include some additional elements in the target model, as shown in Fig. 5. However, these new elements should not modify the semantics of the source model. Hence, it is possible to have some concepts in a specific Enterprise Modelling Language which do not have any correspondence with others. As a consequence, the transformation process must sometimes be performed in a semi-automatic way and with expertise human collaboration.

4 Conclusion

We can conclude that it is possible to use the POP* meta-model as an exchange format among enterprises that use different Enterprise Modelling Languages. Hence, it is a first step on the way to achieving interoperability in the context of collaborative enterprises at the modelling level, and a valid result to be taken into account in further works that are going to be developed in the ATHENA Project, such as the specification of the MPCE, for instance.

On the other hand, and even though it was not the initial objective of the ATHENA Project, the POP* meta-model is now sufficiently well defined to be able to use it as the basis for the further development of an Enterprise Modelling Language, which could be used by providers of tools with meta-modelling capabilities. However, the work within the ATHENA Project will continue to improve and refine the POP* meta-model, particularly the less mature dimensions like the decision dimension, and also to add new dimensions with the objective of providing an exchange format for Enterprise Modelling from a holistic point of view.

Finally, the ‘proof of concept’ of the POP* meta-model was useful as an aid to understanding how it is possible to exchange business process models among different partners in the context of collaborative enterprises using the POP*.

Acknowledgments

This work was funded by ATHENA IP (IST-2003-507849, [4]) and represents part of the European Software Institute’s contribution to the project. The authors are indebted to the A1 Project. In addition, it is partially supported by INTEROP NoE (IST-2003-508011, [6]) and by CICYT DPI2003-02515.

References

1. Vernadat, F.B.: Enterprise Modeling and Integration: Principles and Applications. Chapman and Hall (1996)
2. UEM: Deliverable D1.1. Report on the State of the Art in Enterprise Modelling. <http://www.ueml.org> (2002)
3. EXTERNAL: Extended Enterprise METHodology Project, Final version 1-12-d-2002-01-0 (IST-1999-10091). <http://research.dnv.com/external/default.htm> (2002)

4. ATHENA: Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications) Project (IST-2003-2004). <http://www.athena-ip.org> (2005)
5. IDEAS: IDEAS (Interoperability Development for Enterprise Application and Software) Project. <http://www.ideas-roadmap.net> (2005)
6. INTEROP: Interoperability Research for Networked Enterprises Applications and Software NoE (IST-2003-508011). <http://www.interop-noe.org> (2005)
7. UEML: UEML (Unified Enterprise Modelling Language) Project (IST-2001-34229). <http://www.ueml.org> (2005)
8. Grangel, R., Chalmeta, R., Campos, C., Coltell, O.: Enterprise Modelling, an overview focused on software generation. In Panetto, H., ed.: *Interoperability of Enterprise Software and Applications*, Hermes Science Publishing (2005)
9. Object Management Group, OMG: MDA Guide Version 1.0.1. Document number: omg/2003-06-01 edn. (2003)
10. Aguilar-Saven, R.S.: Business process modelling: Review and framework. *International Journal of Production Economics* **90** (2004) 129–149
11. Kalpic, B., Bernus, P.: Business process modelling in industry—the powerful tool in enterprise management. *Computers in Industry* **47** (2002) 299–318
12. ATHENA: Deliverable DA1.3.1. Report on Methodology description and guidelines definition. <http://www.athena-ip.org> (2005)
13. BPDM: Business Process Definition Metamodel. <http://www.omg.org> (2005)
14. IPK: MO²GO. <http://www.ipk.fhg.de> (2005)
15. LAP/GRAI: GraiTools. <http://www.graisoft.com> (2005)
16. Computas: METIS. <http://www.computas.com> (2005)
17. Object Management Group, OMG: Unified Modeling Language (UML) Specification: Superstructure, version 2.0. Document: ptc/04-10-02 (convenience document) edn. (2004)
18. Fuentes, L., Vallecillo, A.: Una introducción a los perfiles UML. *Novática marzo-abril* (2004) 6–11
19. Object Management Group, OMG: Unified Modeling Language (UML) Specification: Infrastructure, version 2.0. OMG Adopted specification ptc/03-09-15 edn. (2003)

UEML 1.0 and UEML 2.0: Benefits, Problems and Comparison

Giuseppe Berio

Dipartimento di Informatica, Università di Torino, C.so Svizzera 185,
10149 Torino, Italy
berio@di.unito.it

Abstract. Enterprise model integration, transformation, translation are today an essential issue for building complex systems that show high autonomy of constituents, and robustness to changes and evolution (especially for reducing the risk to make the “*next generation of legacy systems*”). Some of the problems raising in enterprise model integration, transformation, translation are related to the usage of distinct languages and to the distinct usage of languages. To address these problems, it is possible to try to characterise and to represent respectively, (i) relationships between constructs in several relevant languages, (ii) a unique language “UEML core language” (Unified Enterprise Modelling Language core language) and (iii) precise relationships between this unique language and these relevant languages. To this aim, this paper summarises and justifies the approaches undertaken in UEML 1.0 and UEML 2.0, tries to compare them alongside their strengths and benefits, and presents some related works.

1 Introduction

This paper presents the recent advances on UEML¹. Specifically, it presents UEML 1.0, i.e. a major result of the *UEML project* (www.ueml.org) [10], [14], and UEML 2.0, initial result of the current *INTEROP project* (www.interop-noe.org); then, the paper discusses benefits, problems and similarities between UEML 1.0 and UEML 2.0.

The main objective of UEML concerns the support of enterprise model integration, translation, transformation (shortly referred to as *model exchange* in the remainder) and to support the required global consistency between distinct enterprise models over their evolution [1]. These models are supposed to be represented in distinct enterprise modelling languages.

UEML should facilitate the model exchange and the required global consistency, by providing a set of *basic correspondences* between several enterprise modelling languages in which the various models are expressed. However, UEML alone is not the full solution to the model exchange and to consistency: indeed, the need to support

¹ This work is partially supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract N° 508011, <<http://www.interop-noe.org>>).

model integration, transformation and translation also appears whenever models are expressed in only one language; the language is always the same while the things the language is representing (in the models) are not necessarily the same ones. For instance, *class-diagram* allows to refer, in a context, to the set of employees in one enterprise by a class *Employee*; *class-diagram* allows to refer to a relational table *Employees* about employees, stored in a database. *Class-diagram* allows in both cases to represent the same phenomenon (let's say "*group of objects*") but models refer to distinct, maybe unrelated or partially related things (i.e. the employees). As a consequence, if these models should be integrated, it requires, additionally, to know that *class-diagram* is used to represent exactly to the same things (i.e. to the same set of employees).

Starting from this point of view, UEML would tend to describe possible basic correspondences between distinct modelling languages that are, in principle, context independent (in the *class-diagram* example above, the "*group of objects*").

Dealing with enterprise modelling raises two key issues that make the objectives of UEML very challenging:

1. which languages are enterprise modelling languages;
2. the informal nature of the underlying meaning (also called *semantics*) of many enterprise modelling languages.

According to point (1) (taking into account two recent states of the art [7],[16] and historical surveys [17]), several languages can be included (in principle, there is not limit), each of them concerning some aspects of enterprises. Some of these several languages often come from other disciplines, such as *software engineering* (e.g. UML, Petri Nets), *knowledge engineering* (e.g. OWL, PSL) and *information system engineering* (e.g. I*). As a consequence, languages for enterprise modelling are often very different in their nature, therefore difficult to be related by the advocated basic correspondences.

According to point (2), the meaning of a *construct* of an enterprise modelling language is often provided by a text in English, French, Italian etc. (that often represents how a specific language is used within a (methodological) context). Therefore, the phenomena that those languages are able to represent, are often unclear. However, problems do not really decrease even if some formalisation of the meaning underlying languages are available (Sect. 3.2). Indeed, a distinction to be carefully made is between, any formalisation and the understanding of the phenomena represented by a language (in the software engineering literature, this is often referred to as "*formal semantics*" (i.e. expressing the meaning in a formal language) and "*precise semantics*" (i.e. expressing the meaning without ambiguities) respectively [9]). Having any formalisation of the language meaning does not guarantee to understand the phenomena; moreover, for UEML, understanding the represented phenomena is the key point because formalisations of these phenomena may take several forms.

The paper is organised as follow. Section 2 provides few technical details. Sections 3 and 4 provide presentation of UEML 1.0 and UEML 2.0 approaches, and describe some benefits and problems. Section 5 contains the final remarks on UEML 1.0 and UEML 2.0 as well as some possible research directions.

2 Technical Definitions

The discussion needs some few technical details. Specifically, we need to introduce at least two layers: the *language layer*, often called the *meta-level*, and the *model layer* respectively.

In the literature concerning formal languages, three important concepts related to the concept of language are mentioned [3], [9]: the concept of *syntax*, the concept of *semantics* and the concept of *semantic domain*. While named distinctively, the concepts of syntax and semantic domain are closely related: as explained in [3], a semantic domain needs to be represented and therefore it corresponds to the syntax of some language. The same happens for semantics. What is usually called “formalisation of a language” means to represent the three concepts in, possibly several, formal languages. However, it often happens that the formalisation of a language is confused with the “formalisation of its syntax”.

The concept of semantics is often synonymous of *function* that associates to *syntactical-artefacts* belonging to the syntax (also called *model-artefacts* in the remainder), some *semantic-artefacts* that are represented in the semantic domain. Often, the semantics is fully independent from some conventions that can be applied to generate or to visualise the syntactical-artefacts: for instance, in IEM [10], the *resources* of an *activity* are visualised under the *activity box*. Sometimes, some meaning is hidden in these conventions (especially in *visual modelling languages*): this means that if conventions are not respected, the models may become not understandable (*visual meaning*). Nevertheless, we base our discussion on the fact that the semantics of a language is usually defined on what is called the *abstract syntax* i.e. a syntax that focuses on the constructs of a language and their structure (and not on the way the constructs are generated and visualised). This is why most of the formalisations of languages start from the definition of the abstract syntax. The abstract syntax is important to abstract from unimportant syntactical details of a language. However, these details can be represented by using a special syntax that is often referred to as *concrete syntax*.

3 From UEML 1.0 to UEML 2.0

3.1 UEML 1.0: The Approach

UEML 1.0 only described the abstract syntax without taking into account any specific method for making this abstract syntax. The reason was also to do not going inside of often difficult semantic aspects of languages (often not explicitly represented as said in the Introduction).

The idea developed in the UEML 1.0 was to state some basic correspondences between languages by using examples. That was inspired from the database design [15]. The undertaken way for approaching UEML 1.0 can be seen as organised in *four main iterative steps* [2]:

1) *The abstract syntax step*: i.e. to represent the abstract syntax of each language that should be taken into account; sometimes this is a difficult step because not all the enterprise modelling languages are equipped with an abstract syntax (or with a good

version of it or a commonly accepted version of it). Additionally, as pointed out in the Introduction, special attention should be devoted to which languages should be taken into account (i.e. are relevant for UEML).

2) *The common model step*: identifying, by using a *common model*, the constructs in the various languages used for representing the same sets of model-artefacts; in other words, the idea is to discover the relationships between distinct languages based on what distinct constructs in these distinct languages are able to represent; i.e. two constructs are made in *correspondence* iff they are able to model the same enterprise phenomena. This step reveals constructs, represented thorough the abstract syntaxes of the various languages, that refer to common phenomena (also called *common concepts*). Most of the complexity of this step is related to the manual procedure that has been undertaken.

Therefore, the underlying thinking of the UEML 1.0 approach can be summarised by the following definition:

there is a basic correspondence between two distinct constructs in two (distinct) languages iff these two constructs can (i.e. not necessarily must) always be used for modelling the same (real world or enterprise) phenomena. (1)

3) *The UEML constructs abstract syntax step*: according to what it has been performed in step (1), the revealed common concepts are represented throughout an abstract syntax; specifically, each common concept is represented with a corresponding *UEML common construct* (Fig.1). For instance, instead of saying <Action basically corresponds to Activity>, it is said <Action basically corresponds to UEML Activity> and <Activity basically corresponds to UEML Activity>.

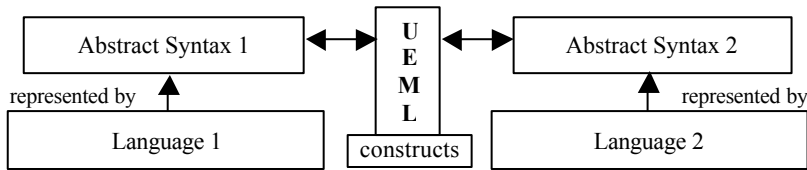


Fig. 1. Basic correspondences are represented in an effective way (reduced number of the correspondences to be represented) by using explicit UEML constructs

4) *The mapping step*: a common concept revealed by performing step (2) and represented throughout a UEML common construct according to step (3), is mapped on to the constructs belonging to the original languages, that contribute, in step (2), to reveal the common concepts themselves.

We can represent *two important axis* along which the UEML 1.0 approach is deployed and used (Fig. 2). The *vertical axis* provides the “Extent of Exchanged Information” (EEI): indeed, not all the information that can be represent in the various languages need to be taken into account for UEML. This fact has been summarised by the following formula associated to the UEML 1.0 [2]:

$$UEML = Common Concepts + (some\ of)\ non\ Common\ Concepts$$

The *horizontal axis* is associated to the effective exchange of models and it provides the “Extent of the Exchanged Information Under specific Objectives” (EEIUO). Existing model artefacts are represented according to UEML constructs according to the basic correspondences (i.e. existing model artefacts are exported in UEML). Afterwards, exported model artefacts can be compared for integration or can be used to generate new model artefacts, translating or transforming those exported artefacts. Then, the new model artefacts can be further imported in some of the languages taken into account in step (1). In other words, apart the basic correspondences, *specific mappings* (i.e. specific functions for transforming, integrating and translating models) can be additionally defined whenever models need to be exchanged, integrated, translated, under specific objectives stated in a given context of application.

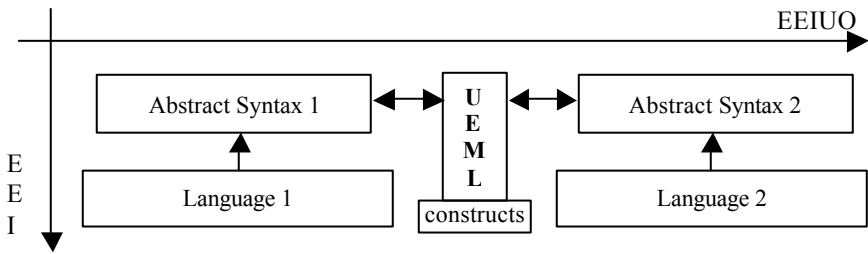


Fig. 2. Organisation of the UEML 1.0 approach

3.2 Discussion: Benefits and Problems of the UEML 1.0 Approach

Why introducing several and complex steps for developing a UEML? Where are the benefits? Where are the problems?

We can point out several underlying benefits that can be gained by the approach outlined in Section 3.1:

- *Practical*, because the method proposes at least one suitable way to map one language onto another one (throughout a set of correspondences and UEML constructs);
- *Conceptual*, because the model artefacts represented in a language and further represented by UEML constructs, are understandable in term of what they are intended to represent (because the correspondences found in step 2 tend to be basic correspondences);
- *Potential*, because, a part the simple exchanges of models that can be realised by using only the identified correspondences, more complex exchanges may also be realised iff a *mapping language* is available for representing mappings over *meta-model artefacts* (for instance, as in the *QVT initiative of OMG (Query View Transformation)*, and also in related initiatives as [5], [6]);
- *Architectural*, because the approach makes possible to implement an architecture in which it is possible to provide a *uniform interface* for accessing models represented in several languages;

- *Methodological*, because new relationships between UEMML constructs, not available between language constructs (because distinct languages are not related) can be identified (and new methods and methodologies can be developed).

Problems. The proposed approach seems difficult to be generalised, to be suitable for managing situations of potential inconsistencies of correspondences (e.g. a construct in a language can represent several constructs in another language), and to be independent on the models used to find out the correspondences and modellers building these models (step 2 described in Sect. 3.1).

The approach does not guarantee that the exported *model artefacts* according to the correspondences are “*formally semantically equivalent*” i.e. meaning preserving. This is however a “bad statement” for two reasons. First, the statement does not make sense in every case: despite formalised syntaxes, most of the enterprise modelling languages are not equipped with formalised semantics or even precise meaning. Second, the approach is very useful whenever languages are fully formalised in their semantics. Indeed, between two fully formalised languages (even between a language and itself) *there are several possible correspondences*: the problem is still how to identify *correspondences that might be basic correspondences*. To better illustrate the last reason, a good example is provided by the standardisation of High Level Petri nets [4]. In this case, implicit correspondences between nets are represented with four elements: *place*, *transition*, *token* and *arrow*. Despite the well-known formality of the several classes of Petri nets, it is explicitly stated that the proposed standard does not guarantee any formal equivalence of semantics between distinct Petri nets classes [4].

Finally, the advocated specific mappings that realise complex model exchanges can be represented if a specific *mapping language* is available (as for instance in [5], [6]): in this case, we can say that *models are exchanged by using this mapping language*. However, once time more, the UEMML 1.0 approach does not help to formally proof (correctness) properties of these exchanges.

Conclusions. On one hand, the approach undertaken in UEMML 1.0 allows:

- To identify correspondences between distinct constructs belonging to distinct languages;
- To partially gain evidence that the identified correspondences are also basic correspondences according to the definition (1) provided in Section 3.1;
- To represent these correspondences between distinct languages by using a set of specifically defined UEMML constructs;
- To further represent complex exchanges by adding languages for specifying mappings, eventually dependent on the objectives to be achieved.

On the other hand, the approach undertaken in UEMML 1.0 does not help:

- To formally proof, according to some meaning of models, properties of the identified basic correspondences; if languages taken into account in step (1) are fully formalised, these proofs should be provided elsewhere to eventually verify the identified basic correspondences;
- To formally proof, according to some meaning of models, properties of the exchanges; instead, these proofs should be provided elsewhere.

4 The UEML 2.0

4.1 The UEML 2.0: The Approach

The UEML 2.0 undertakes a very different, eventually complementary approach. Indeed, it requires to fully model the languages in their three conceptual components: abstract syntax, semantic domain and semantics. These three components are organised according to a *meta-meta-model* [12], [13] (also in [1]): any language is represented by constructs, in turn associated to some meaning provided by a semantic domain. Fig. 3 below provides a simplified version of the meta-meta-model as a UML class diagram.

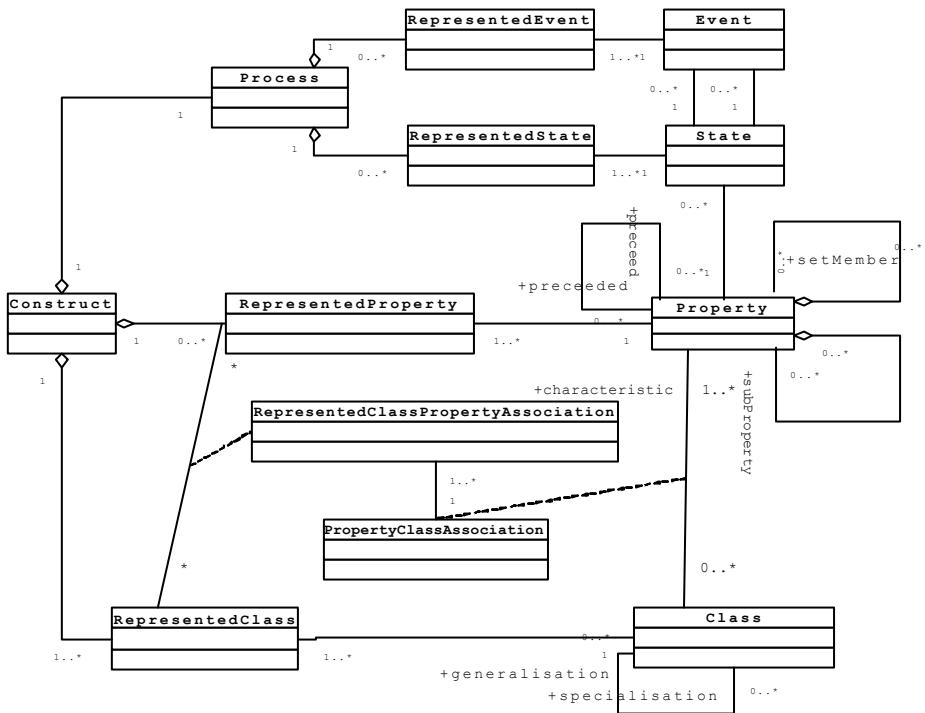


Fig. 3. A simplified version of the meta-meta model in the UEML 2.0 approach (UML class diagram)

Accordingly, the first approached problem is how to organise the semantic domain. Taking into account the literature, the INTEROP work and also the experience on UEML 1.0, we can observe that the semantic domain should describe as much as possible the underlying phenomena, in an explicit way. Generally speaking, the semantic domain represented according to the meta-meta-model is organised in two distinct parts: a *static part* and a *behavioural part*. In the static part, there are *classes* of things, the construct is intended to represent, and the relevant *properties* of these things. In the behavioural part, there are *events* and *statuses* concerning these things.

Now, the second approached problem is what classes, properties, events and statuses should be used for representing a language. In the first attempts performed in INTEROP, the idea has been to use an existing ontological theory, specifically the *BWW* [18], very general, that allows to distinguish between several real world phenomena (while we are not constrained by any ontological theory).

Basic correspondences are not statically defined as in step (3) of the UEML 1.0 approach (Sect. 3.1); instead, they should be inferred according to the represented semantics and the semantic domain. A further consequence is that the UEML 2.0 approach does not suggest UEML constructs to represent the basic correspondences (while this is possible once the represented semantic domain contains enough information). In the remainder, we develop a complete example on how basic correspondences can be inferred and how these basic correspondences can be used for supporting one model transformation.

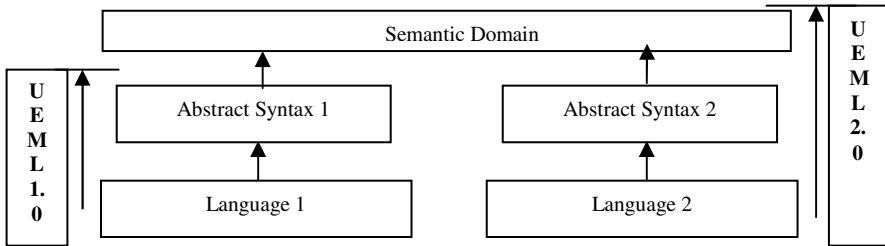


Fig. 4. Comparing UEML 2.0 approach and UEML 1.0 approach

The UEML 2.0 approach can be compared to the UEML 1.0 approach by Fig. 4 below. In Fig. 4, the most important similarities become evident: both ones requires to represent abstract syntaxes but in UEML 2.0 there is a standardised way to do the work, according to the meta-meta model. Additionally, the UEML 2.0 requires to explicitly represents the semantic domain and the semantics. Specifically, the semantic domain corresponds to objects of classes *Event*, *State*, *Property* and *Class* depicted in Fig. 3. The semantics corresponds to the links between *RepresentedClass*, *RepresentedProperty*, *RepresentedState* and *RepresentedEvent* (which objects nearly correspond to abstract syntaxes) and the associated classes of the semantic domain.

4.2 An Application of UEML 2.0 on Languages and Correspondences

In this section, we describe a complete example of how the proposed approach works by using two well-known enterprise modelling languages: IEM (that was analysed for the UEML 1.0) [10] and Coloured Petri Nets (CPN).

IEM comprises constructs such as *Activity*, *State* and *Flow* to represent enterprise processes; however, statuses are not mandatory. According to the *BWW* concepts (represented in bold in the remainder and implicitly referred by *objects* created from the class-diagram that represents the meta-meta model (Fig. 3)) and the IEM constructs definitions:

- Construct *Activity* represents the class *ActiveThings*;
- Construct *State* represents the class *AllThings* and the property *Regular property*;
- *InputArrowFromState* represented property from *State* to *Activity* represents any *MutualBinding property* describing, for instance, any pre-requisite of *Activity*;
- *OutputArrowToState* represented property can be any *MutualBinding property* between *Activity (ActiveThings)* and *State (AllThings)* describing, for instance, any post-condition of the *Activity*;
- *InputArrowFromActivity* represented property can be any *MutualBinding property* between *Activity (ActiveThings)* and itself;
- *OutputArrowToActivity* can be any *MutualBinding property* between *Activity (ActiveThings)* and itself;
- Construct *Connector* is “a combining arrow” but what does it describe really? May be the class *ActiveThings*, or the property *TransformationLaw property* (and some additional classes).

Similarly, a possible definition of CPN is as follow [1]:

- Construct *Transition* represents the class *ActiveThings*;
- *IncomingArrow* represented property between *Transition (ActiveThings)* and *Place (ActedOnThings)* represents any set of *MutualBinding property* because it essentially indicates the tokens outgoing from a place and resulting from some transition fire;
- *OutgoingArrow* represented property between *Transition (ActiveThings)* and *Place (ActedOnThings)* represents any set of *MutualBinding property* because it essentially indicates the tokens outgoing from a place and selected to fire some transition;
- Construct *Place* represents the class *ActedOnThings* (other than acting on tokens, transitions act on the places where tokens are located);
- *PlaceContent* represented property between *Place* and *Token* represents any set of *MutualBinding property* between *Token (ActedOnThings)* and *Place (ActedOnThings)*; because tokens in a place at a given step are what is entering minus what is exiting; therefore, *PlaceContent* represented property (i.e. *MutualBinding property*) is
 - sub-property of (the *MutualBinding property* linked to) *IncomingArrow*;
 - sub-property of (the *MutualBinding property* linked to) *OutgoingArrow*;
- Construct *Token* represents the class *ActedOnThings* because transitions also act on tokens, not only on places;
- *TokenColour* represented property is any *Regular property* of *Token*.

Inference of one basic correspondence. *Transition* and *Activity* both represent *ActiveThings*, therefore one activity and one transition can refer to the same active thing (element of *ActiveThings*). However, they probably represents distinct properties of these *ActiveThings*. *State* in IEM is used to represent any *Regular property* of *AllThings*. Given a thing (in *AllThings*) appearing as a status in a IEM model, if this thing can further be specialised into *ActedOnThings*, it is possible to establish one correspondence to CPN (it is interesting to note that this analysis reveals

that it is not implicitly forced by IEM that *AllThings* associated to *State*, are also things on which activities act on!). However, because in CPN both *Place* and *Token* represent *ActedOnThings*, there is the need to decide if the correspondence from *State* of IEM is to *ActedOnThings* linked to *Token* or *ActedOnThings* linked to *Place* of CPN. This is a very interesting issue: a thing represented by a status in IEM and specialised in *ActedOnThings* further linked to *Token* of CPN, seems reasonable. In fact, *State* essentially represents any *Regular property* associated to *AllThings*, describing statuses of things and *Token* does the same through *TokenColour*.

The conclusion is that, under the hypotheses that both in IEM and CPN is acceptable (correct) referring to *ActedOnThings*, it is possible to say that the construct *State* corresponds to the construct *Token*: additionally, in CPN, a place containing tokens is needed even if, according to the previous decision, *Place* has not counterpart in IEM.

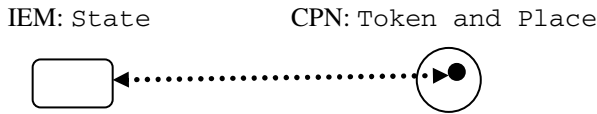


Fig. 5. A basic correspondence represented by a double-ended arrow

Even if it might seem quite strange, according to the BWW based definition of CPN, *Place* is a complex construct that indicates the set of things produced and consumed by transitions. That is not necessarily the case for IEM.

Supporting exchanges of models with inferred basic correspondences. Given a specific status in a IEM model, it is possible to build the corresponding thing in the class *AllThings*, then it might be possible to specialise this thing in the class *ActedOnThings* and, finally, to build, the corresponding token in a CPN model. However, in CPN, each token is involved in a *MutualBinding property* with a place and therefore it is required to additionally define a place (where locating the token). The required place, however, is not representing the specific status belonging to the IEM model but is representing an additional construct needed by CPN whenever tokens are introduced.

4.3 Discussion: Benefits and Problems of the UEML 2.0 Approach

The meta-meta model depicted in Fig. 3 introduces, formally, the notion of construct, and forces to become clear on the abstract syntax. This is a good point but it is also a strong point. For instance, in the example about CPN, arrows have been represented as properties of transitions; a natural question is: should it be possible to represent arrows as properties of places? The same might happen for IEM.

All the benefits concerning UEML 1.0 and mentioned in Section 3.2, are mostly valid for the UEML 2.0. Additionally, the UEML 2.0 approach improves the UEML 1.0 approach as follow:

- It guides, according to a meta-meta model, the representation of abstract syntax, semantics and semantic domain for any enterprise modelling language;

- It allows to infer basic correspondences between distinct languages, once their semantics and the semantic domain have been represented (i.e. to fully apply the definition (1) in Sect. 3.1);
- While not suggested, it allows to represent some UEML constructs and their semantics whenever the semantic domain contains enough information (i.e. whenever a significant number of relevant languages has been represented).

As for UEML 1.0, the undertaken approach does not allow

- To formally proof properties of basic correspondences and more complex exchanges; while the meta-meta model (Fig. 3) is represented with a UML class-diagram, it can be represented in some formal language enabling reasoning.

5 Conclusion

In this paper, we have reviewed the approaches undertaken in UEML 1.0 and UEML 2.0. We have also described benefits and problems of both approaches. These two approaches share similar ideas but the envisioned mechanisms to implement these ideas are different. Specifically, the paper presents how basic correspondences between languages are addressed in UEML 1.0 and UEML 2.0. While UEML 1.0 is pragmatics, UEML 2.0 explicitly requires the definition of a common semantic domain for languages, grounding the future work on correspondences on this domain. However, an important research work is nevertheless required to fully characterise the semantic domain and how basic correspondences can be characterised in term of properties on this domain.

The approach suggested by UEML 2.0 is surprisingly analogous to the *hybrid approach* for *ontology interoperability* [8]: the main difference is the layer (Sect. 2) where the approach is applied, i.e. *languages* in UEML 2.0 and *models* in ontologies. As pointed out several times in the paper, both layers are required for realising model exchanges. Therefore, further research work should focus on how to exploit the synergies between mechanisms suggested by UEML 2.0 (for instance the basic correspondences) and mechanisms suggested by ontologies for effectively realising complex exchanges of enterprise models.

References

1. Berio G. et al. (2005). UEML 2.0. Deliverable 5.1. INTEROP project UE-IST-508011 (www.interop-noe.org).
2. Berio G., Anaya V., Ortiz A (2004). Supporting Enterprise Integration through a Unified Enterprise Modeling Language. In Proceedings of EMOI 2004 (Janis Grundspenkis, Marite Kirikova Eds.), joint with CAiSE04, Riga, Latvia, June, vol. 3, pp. 165-176 (<http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/>).
3. Berio G., Petit M., "Enterprise Modelling and the UML: (sometimes) a conflict without a case". In Proceedings of 10th ISPE International Conference on Concurrent Engineering – Enhanced Interoperable Systems (R.Jardim-Golçalves, J.Cha, A Steiger-Garçao Eds), Madeira Island, 26-30 June, pp. 787-794. A.A. Balkema Publishers.

4. Billington J., Christensen S., van Hee K., Kindler E., Kummer O., Petrucci L., Post R., Stehno C., Weber M. (2003). The Petri Net Markup Language: Concepts, Technology, and Tools. In Proceedings of the ICATPN 2003, June, Eindhoven, Netherlands.
5. Clark T., Evans A., Sammut P., Willans J.S. (2004). Applied Metamodelling – A Foundation for Language Driven Development, Version 0.1, Xactium.
6. Evans A., Maskeri G., Sammut P., Willans J.S. (2003). Building families of languages for model-driven system development. In Proceedings of WiSME, joint with UML'2003 (<http://www.metamodel.com/wisme-2003/program.html>).
7. García Díez Belén Ana et al. (2004). State of the Art in Enterprise Modelling Techniques and Technologies to Support Enterprise Interoperability. Deliverable 1.1. ATHENA project UE-IST 507849 (www.athena-ip.org).
8. Goh C. (1997). Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources. PhD Thesis, MIT.
9. Harel D., Rumpe B. (2004). Meaningful Modeling: What's the Semantics of "Semantics"? In IEEE Computer, October, pp. 64-72.
10. Jochem R. (2002). Common representation through UEML – requirement and approach. In Proceedings of ICEIMT2002 (Kosanke K., Jochem R., Nell J., Ortiz Bas A. (Eds.)), April 24-26, Valencia, Spain, Kluwer.
11. Jochem, R. Mertins, K. (1999). Quality-Oriented Design of Business Processes. Kluwer, Boston.
12. Opdahl A.L. (2004). The UEML Template. Available at www.interop-noe.org.
13. Opdahl A.L., Henderson-Sellers, B. (2004). A Template for Defining Enterprise Modelling Constructs. In *Journal of Database Management* 15(2).
14. Panetto H., Berio G., Benali K., Boudjlida N., Petit M. (2004). A Unified Enterprise Modelling Language for enhanced interoperability of Enterprise Models. In Proceedings of the 11th IFAC INCOM2004 Symposium, April 5-7, Bahia, Brazil.
15. Petit, M. (2002). Some methodological clues for defining a UEML. In Proceedings of ICEIMT 2002 (Kosanke K., Jochem R., Nell J., Ortiz Bas A. (Eds.)), April 24-26, Valencia, Spain, Kluwer.
16. Petit M., et al. (2002). State of the Art in Enterprise Modelling. Deliverable 1.1 of the UEML Thematic Network UE-IST 2001 34229, (www.ueml.org).
17. Vernadat F. (1997). Enterprise Modelling Languages. In Proceedings of the ICEIMT97 Conference, October 28-30, Torino, Italy.
18. Wand, Y., Weber R. (1995). On the deep structure of information systems. In *Information Systems Journal*, vol. 5, pp. 203-223.

Facilitating Interoperability: A Cross-Analysis of the Language UEML and the Standard ISO/DIS 19440*

Petia Wohed¹, Birger Andersson², and Hervé Panetto¹

¹ Centre de Recherche en Automatique de Nancy,
Université Henri Poincaré, Nancy 1/CNRS,
BP239, 54506 Vandoeuvre les Nancy, France
{petia.wohed, herve.panetto}@cran.uhp-nancy.fr

² Department of Computer and Systems Sciences,
Stockholm University/Royal Institute of Technology,
Forum 100, SE-164 40 Kista, Sweden
ba@dsv.su.se

Abstract. During recent years Interoperability and Interoperable Enterprise Applications has gained a central place on the IS development arena. Presented in this paper is a cross-analysis of two languages for enterprise modelling and information systems development. The languages are the Unified Enterprise Modelling Language (UEML) and ISO/DIS 19440. The purpose of this cross-analysis is to make the languages more complete and well defined. The analysis includes a mapping between the languages. The results of the analysis can be used for further development of the languages which in the long run will be beneficial for the interoperability of enterprises modelled in them.

1 Introduction

In 1986 it was estimated [4] that during the previous years, hundreds, if not thousands of information systems development (ISD) methods had been introduced. Today we can observe that from 1986 up until now hundreds, if not thousands of ISD methods has been introduced. It can be argued as in [5,7] that what is needed is to understand existing ISD practices, rather than adding new methodologies, techniques and tools to an already large existing collection. On the other hand, this diversity may very well be beneficial for the finding of solutions for different problems in different domains.

During this period, with the emerging web in the background, interoperability between systems and organisations has become a focal point of interest where diversity has turned into a problem. Today the need for developing interoperable systems, as well as using compatible methods and languages for developing such systems, has become a vital interest for any organisation.

* This work is founded in part by Interop NoE, IST-508011.

To tackle the issues of interoperability, work on developing de facto standards like UML [12], and de jure standards like ISO/DIS 19440 [10] are ongoing. In addition to this, frameworks, e.g. UEML (Unified Enterprise Modelling Language) [3], are also developed to facilitate interoperability and integration. The main differences of the UML, ISO/DIS 19440, and the UEML framework developments lies in the approach and methodology applied in their development. While UML is developed by mainly identifying and integrating a number of necessary techniques for information systems development, ISO/DIS 19440 is developed through an analytical, top down approach and focused on a specific domain—enterprise modelling. Similarly to ISO/DIS 19440, the domain of UEML is enterprise modelling. In contrast to ISO/DIS 19440 the development approach is bottom up. In the development of UEML three different modelling techniques for enterprise process modelling, GRAI [6], IEM [11] and EEML [1], were cross-compared and a synthesized meta-model of them was developed.

In this paper we will compare the UEML meta-model to the one of ISO/DIS 19440. These frameworks are chosen for two reasons; 1) they are developed to cover the same domain. A result of a comparison should therefore uncover weaknesses or deficiencies in a model which could be amended with constructs from the other. 2) They are similar enough to apply the evaluation approach suggested by [8,13], where language meta-models are advocated for doing comparisons. For space reasons, we focus in this paper mainly on extensions of UEML. However, the analysis results may just as well be used on ISO/DIS 19440.

This paper proceeds by first presenting UEML and describing a number of open questions for further work on it. Then the ISO/DIS 19440 is presented and discussed. Finally a cross-analysis of these two notations is presented and some guidelines for their further development suggested.

2 UEML

In this section the UEML framework [12] is presented. The meta-model of the framework is reprinted in figure 1.

An *Activity* represents a generic description of a part of enterprise behaviour which produces outputs from a set of inputs. An activity may be decomposed into other activities and an activity may require one or several *Resource Role* played by *Resources* for its completion. A *Resource* may be a *Material Resource* or a *Human Resource*.

An *Activity* has at least one *Input Port* and at least one *Output Port*, where flows representing inputs or outputs of the activity are connected.

There are two ways to represent that some *Resource* is used by an *Activity*: 1) through the definition of a role (i.e. *Role Type*) which a resource plays in an *Activity*. This method is used when the origin of the resource is not explicitly given; and 2) through a flow, connected to the *Input Port* of the *Activity* carrying the resource. This method is used when the origin of the resource is explicitly given or if the resource to be used is the result of some decision, grouping or decomposition of some other resource(s) through a *Connection Operator*.

An *Object* is anything that can be attached to a *Flow*. In other words, it is anything that may be needed or produced by an *Activity*. It can be an *Information Object* or a *Resource*.

A *Flow* represents the flowing of an object from an origin to a target. The origin and target of a flow are called *Anchor* and can be either an *Input Port*, an *Output Port* or a *Connection Operator*. A *Flow* is either an *IO Flow*, a *Resource Flow* or a *Control Flow*.

- An *IO Flow* represents the flowing of an *Object* between two *Activities*. If the *Object* is an input object, then the *IO Flow* is connected to an *Input Port* of an *Activity* and this means that the *Object* is necessary for the execution of the *Activity*. The *Object* can possibly be consumed or modified by the *Activity*. If the *Object* is an output object of an *Activity* the *IO Flow* is connected to the *Output Port* of the activity indicating that the object has been produced by the *Activity*.
- A *Resource Flow* represents the flowing of a *Resource* between two *Activities*. The flow then connects an *Output Port* of an *Activity* that produces it and an *Input Port* of the *Activity* that requires it.
- A *Control Flow* connects two *Activities* and represents either: 1) a precedence relationship between *Activities* (a *Control Flow* carrying no *Object*); 2) a triggering of an *Activity* (a *Trigger Flow*, which carries an *Information Object* that triggers a second activity after the completion of a first one); 3) a constraining of an *Activity* (a *Constraint Flow* carrying a constraining *Information Object*. For instance, this could be a description of a procedure to be followed when executing the activity).

A *Connection Operator* represents the grouping or splitting (Join and Split) of flows between activities. A *Connection Operator* of type “Join” is target of at least two *Flows* and is the origin of exactly one *Flow*. A *Connection Operator* of type “Split” is origin of at least two *Flows* and is target of exactly one *Flow*. The attributes “AND”, “OR” and “XOR” are used to indicate parallelism, choice and synchronisation.

2.1 Open Questions

The UEML, as defined in [3] has been developed to facilitate the integration of different enterprise modelling languages and to demonstrate the feasibility of applying of a bottom up approach for doing so. UEML is under development and the current version, UEML 1.0, is to be considered as an intermediate result. Some outstanding issues and open questions in the current version are discussed below.

- A *Resource* is an *Object* that plays a specific *Role* for the performance of an *Activity*. The same *Resource* could play another *Role* for another *Activity*. The *Role* is qualified as an “object”, “organisation unit”, “person”, “tool”, or any other valid role. However, it is the responsibility of the modeler to

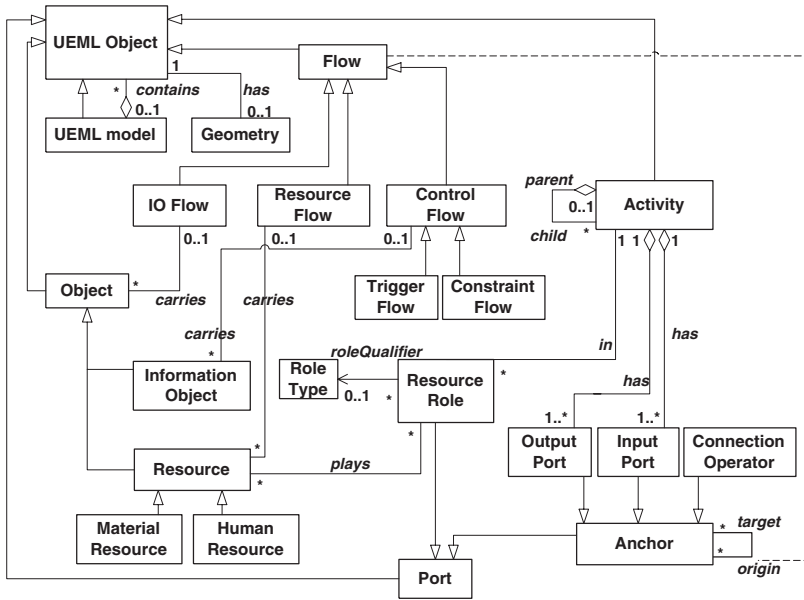


Fig. 1. UEML meta-model

define the qualified *Role* for every *Resource* performing an *Activity*. If this is not done, the relationship showing which *Resource* are carrying out what *Activity* is lost.

- An *Activity* modifies *Object*. This fact is currently not visible in the meta-model as there is no relationship included between the classes *Object* and *Activity*.
- The specialisation classes of *Object*, i.e., *Information Object* and *Resource* are not exhaustive. Additional useful subclasses could be *Product* or *Service*, but this should be determined after further investigations.
- The specialisation classes *Control Flow*, *Resource Flow* and *IO Flow* of the class *Flow* are not disjoint. In the definition of UEML, the type of flow (*IO Flow*, *Resource Flow*, *Control Flow*) is determined by the kind of *Object* the flow carries. However, because *Resource* and *Information Object* are kinds of *Object*, and as *IO Flow* is specified to carry objects, then *IO Flow* could also carry this kind of objects (and is thereby not explicitly distinguished from *Control Flow* and *Resource Flow*). *IO Flow* was introduced to specify the flow of objects that are consumed or produced by activities, while *Resource Flow* was introduced to capture the flow of *Resources* needed for the execution of an *Activity* and *Control Flow* the information (event or constraints) needed by the *Activity*. This subtle distinction cannot explicitly be derived from the model.
- The construct *Connection Operator*, was defined in order to split or join flows according to specific rules (AND, OR, XOR). It is a straight forward derivation to conclude that the construct of *Connection Operator* together

with the construct of *Action*, as origin and targets of flows, are enough for capturing the basic control flow patterns defined by WfMC [14], but a detailed analysis of whether they are sufficient for capturing more advanced control flow constructs like those defined within the Workflow Patterns Initiative¹ [2] remains to be done.

3 ISO/DIS 19440

Presented in this section is the ISO/DIS 19440 framework. The model in figure 2 is an integrated meta-model of the Informational, Functional, Resource, and Organizational views of ISO/DIS 19440. In contrast to these views, in figure 2 only the concepts described in “The modelling language constructs” section of the document [10] are shown, while the complementary concepts (as not currently included in the standard) are omitted.

A *Domain* represents the boundary and the content of an enterprise or a portion of an enterprise. A *Business Process* represents a certain part of enterprise behaviour. A *Business Process* is an aggregation of *Business Process* and/or *Enterprise Activity* together with information described by *Behaviour Rule*. An *Enterprise Activity* is the realisation of a transformation of inputs to outputs by a specific resource. *Enterprise Activity* and *Business Process* are collectively called *Enterprise Function*.

Behaviour Rules are used to define the behaviour of a business process. They define constraints on relationships e.g., sequencing between *Business Processes* and/or *Enterprise Activities*. A textual description of behaviour rules is given in the framework specification. Rules are, however, classified as complementary concept and are therefore not present in the graphical model in figure 2.

An *Event* initiates the execution of a *Business Process* or an *Enterprise Activity*. A special kind of *Event* is an *Order*. An *Order* is an instruction for the performance of an activity.

An *Enterprise Object* is the characteristics of the thing(s) being modelled during its(their) life-cycle(s). A subset of the characteristics comprises an *Enterprise Object View* (or *Object View*, for short). A special kind of an *Enterprise Object* is a *Product*. The production and sales of *Products* is the *raison-d’être* for the enterprise. Another kind of *Enterprise Object* is *Resource*. A *Resource* represents some or all of the capabilities required for the execution of an *Enterprise Activity*. A *Capability* is any device, tool or means at the disposal of the enterprise to produce goods or services. A capability (required for an *Activity*, and provided by a *Resource*) can be divided into *Capability Element*.

The organisational structure of an enterprise is captured through *Organisational Unit*, *Organisational Cell* and *Decision Center*. The *Organisational Unit* describes the roles and responsibilities in the hierarchical structure of an enterprise. It enables listing of the requirements in terms of skills profiles needed to perform an activity. The *Organisational Cell* construct provides identification of its contents and the contents positions in the hierarchy. A *Decision Center*

¹ <http://is.tm.tue.nl/research/patterns/patterns.htm>

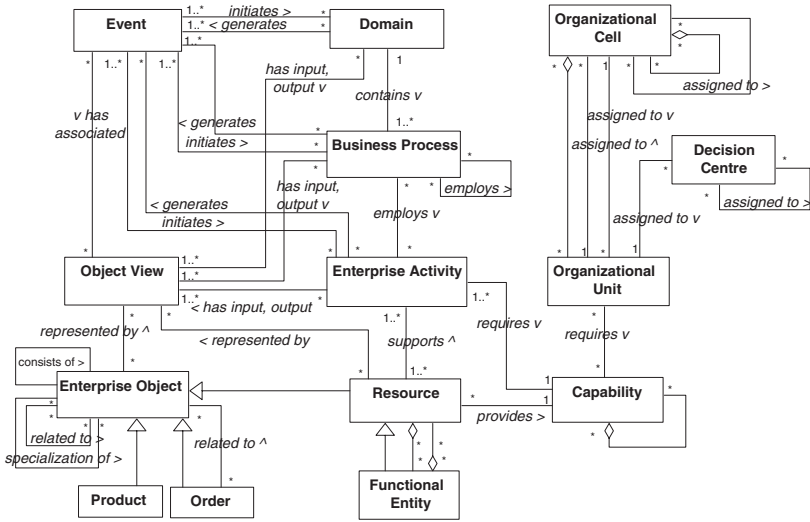


Fig. 2. ISO/DIS 19440 meta-model

is an elaborated concept to capture the modelling of a decision structure in a two-dimensional grid, where the rows represent the decision levels, the columns represent the decision functions and the intersections are the *Decision Centers*.

3.1 Comments on ISO/DIS 19440

Following is a list of issues in the ISO/DIS 19440 specification that we find valuable to address in order to enhance the framework².

- The concept *Domain* as representing the “boundary and the content of an enterprise” ([10],p.21) is a very general concept. For instance, *Domain* incorporates a number of important concepts, e.g., business objectives or performance indicators, each important enough to model explicitly as a class of its own.

Furthermore, it is somewhat unintuitive to talk about inputs and outputs for a domain. For example, in a “Health care” domain, what would the inputs and the outputs be? While a patient who has been fully treated in a hospital can be considered as input and output of a certain medical care process, he/she can surely not be considered as an input and output to the health care domain.

Moreover, concerning the goal of a domain: if a very high level goal for the health care domain is specified to be “To provide citizens with good health care services” it can well be argued that this is really a goal for the health care providers rather than a goal for the domain as such. The definitions of goals is a characteristic for enterprises and organisations, rather than the

² In cases of inconsistencies in the definitions of relationships we have followed the definitions according to the graphical models.

domain to which they belong. This raises the question whether the concept *Domain* was introduced as a generic/general term for enterprises and organisations. If it was, what then is its relationship to the other organisational aspect concepts in the model, i.e., *Organisational Unit*, *Organisation Cell* and *Decision Center*?

- The concepts *Organisational Unit*, *Organisation Cell* and *Decision Center* are introduced for capturing the organisational structure of an enterprise. However, the specification text and the graphical model do not provide an entirely clear semantics. To quote, “Organisational Cells describe the formal, hierarchical administrative structure of an enterprise” ([10],p.43) and “Organisational Units shall represent the [...] roles and responsibilities within a given hierarchical structure ..” ([10],p.47). A more intuitive naming would be *Organisational Unit* instead of *Organisational Cell* and *Role/Responsibility* instead of *Organisational Unit*.

This renaming also reveals some confusion concerning the concepts *Role* and *Responsibility*. It is possible to argue that indeed *Role* and *Responsibility* should be treated together and not separately but this will limit the expressive power of the model to a very coarse level of detail. For instance, the model would not be able to capture situations like when a person possessing different roles have different sets of responsibilities attached to each one of them.

Furthermore, the relationships *Organisational Cell - attached_to - Organisational Unit* and *Organisational Unit - attached_to - Organisational Cell* on top of the aggregate relationship from *Organisational Cell* to *Organisational Unit* are difficult to understand and needs further explanation.

Also, the concept *Decision Center* as well as its relation to *Organisational Unit* (or *Role/Responsibility* as suggested above) is not clear. It seems to have been introduced to capture the decision structure in an enterprise, but then it also should have been related to the *Organisational Cell* concept (or *Organisational Unit* according to the renaming above).

Finally, a relationship showing which *Organisational Cell* (i.e. *Organisational Unit* according to our renaming) is responsible for which *Business Process(es)* or *Enterprise Activity(ies)* is not present in the model. Why this supposedly useful and important relationship is absent should be explained.

- The concept *Behaviour Rule*, where the behaviour of a process is described, is considered as a complementary concept and is therefore not present in the figure 2. It is present in the Functional aspect model in the specification modelled as a single class and its complementary textual description is much longer than any other description of the concepts present in the model. The question is: why is this class considered as complementary? The behaviour aspect of a process is central in enterprise modelling.

4 Mapping the Frameworks

In this section we discuss and investigate the correspondence of concepts from both frameworks. The result of the investigation is presented in table 1.

The concept *Activity* in UEML corresponds to both *Enterprise Activity* and *Business Process* in ISO/DIS 19440. While ISO/DIS 19440 distinguishes between the leaves in a business process hierarchy as the smallest not further decomposable units of work, (i.e., *Enterprise Activity*) from those which are comprised of and orchestrate a number of smaller units of work (i.e. *Business Process*), UEML does not make this distinction and uses the term *Activity* for denoting both.

The concept of *Flow* in UEML which is elaborated and subdivided into *IO Flow*, *Resource Flow* and *Control Flow* (which is in turn divided in *Trigger Flow* and *Constraint Flow*) and where a class is introduced for each one of these concepts, is in ISO/DIS 19440 captured by relationships. So, the *Flow* is roughly captured by *Business Process - has_input,output - Object View* and *Enterprise Activity - has_input,output - Object View* relationships. This mapping is only very coarse because in UEML a flow can not only go between *Activity*, but also between *Activity* and *Connection Operator* such as AND/OR joins and splits. The *IO Flow* in UEML is then mapped as *Flow*, while for *Resource Flow* the restriction that the *Object View* shall represent a view for a *Resource* is applied. Furthermore, *Control Flow* corresponds to *Behaviour Rule*, *Trigger Flow* corresponds to *Event* and *Constraint Flow* to *Enterprise Activity/Business Process - has_input, output - Object View* relationship when the *Object View* is a view of an *Order*.

The concepts *Port* and *Anchor* with its *Input* and *Output* in UEML do not have any direct correspondence in ISO/DIS 19440. The *Connection Operator* is the only subclass of *Anchor* which can be partially mapped to *Behaviour Rule*.

The concept of *Resource* is far more elaborated in UEML than in ISO/DIS 19440. In UEML the distinction between *Human* and *Material Resource* is made and the roles different resources play in an activity are explicitly specified through the *Resource Role* and *Role Type* classes.

The concept of *Role* is not explicit in ISO/DIS 19440. The question whether it is partially covered in the *Organisational Unit* concept was already raised in the previous section. Instead of specifying the role of a resource in an *Enterprise Activity*, in ISO/DIS 19440 the capabilities which a *Resource* provides are specified through the *Capability* class. In this way capabilities can be matched to the *Capability* required for the execution of an *Enterprise Activity*, which then can be used for resource allocation of the *Resource(s)* capable for performing the *Activity*. The concept of *Capability* is not present in UEML.

The concepts aimed at capturing the organisational structure in ISO/DIS 19440, i.e. *Organisational Cell*, *Organisational Unit* and *Decision Center* has no correspondence in UEML. Even if, as discussed earlier, these concepts need to be further elaborated and clarified, their intention of capturing the organisational and decision making structure of an enterprise is clear.

Table 1 summarises the results from this initial coarse mapping between UEML and ISO/DIS 19440. As both frameworks are quite informal any precise mapping of them can not be provided at this stage. To be able to perform a precise analysis and comparison of two frameworks they must both necessarily be formalised.

Table 1. Mapping of the frameworks

UEML1.0	corr ISO/DIS19440	corr UEML1.0	ISO/DIS19440
Activity	Enterprise Activity, Business Process	Activity	Business Process
Flow	has_input,output relationship ⁵	Activity	Enterprise Activity
- IOFlow	has input, output relationship	TriggerFlow	Event
- ResourceFlow	has input, output relationship ⁶		
- ControlFlow	Behaviour Rule		
- TriggerFlow	Event		
- ConstraintFlow	has input, output relationship ⁷		
Port	-	-	Domain
Anchor	-	-	Organisational Cell
- InputPort	-	-	Organisational Unit
- OutputPort	-	-	Decision Center
- ConnectionOperator	Behaviour Rule		
Object	Enterprise Object		Object View
- InformationObject	Object View	Object	Enterprise Object
- Resource	Resource	Resource	Resource
- MaterialResource	Resource	MaterialResource	Product
- HumanResource	Resource	TriggerFlow	Order
ResourceRole	-	-	Capability
RoleType	-	-	Functional Entity

5 Usefulness of the Results

The analysis provided in the previous section can be used for further development of the frameworks. To exemplify how this can be realised we are suggesting few changes in UEML in accordance to the results from the comparison (see figure 3).

Inspired by the separation done in ISO/DIS 19440 between atomic tasks (*Enterprise Activity*) and composite tasks (*Business Process*), we introduce corresponding classes into the UEML meta-model. These classes are introduced as specialisations of the class *Activity*, which is also convenient to keep for expressing the common properties of the subclasses. This idea is also supported by the fact that there is a concept defined (i.e., *Enterprise Function*) in the textual description of ISO/DIS 19440 generalising the concepts *Enterprise Activity* and *Business Process*. Even if this concept is not graphically depicted in the meta-model, its correspondence to the concept *Activity* in UEML is straight forward. Furthermore, introducing the specialisation of a *Business Process* as an *Activity* implies that the aggregate relationship from the class *Activity* to itself can now more accurately be drawn from the class *Business Process* to the class *Activity*.

Furthermore, we consider the separation of the concept *Event* from the concepts *Business Process* and *Enterprise Activity* in ISO/DIS 19440 as an advantage as it captures the differentiation of external and internal activities. Based on this we suggest the introduction of the class *Event* into UEML.

As *Activity*, *Event*, and *Connection Operator* have the common property of being the origins/targets for flows the class *Flow Object* for generalising them is introduced. The *Flow Object* is naturally included as a specialisation of the *UEML Object* class.

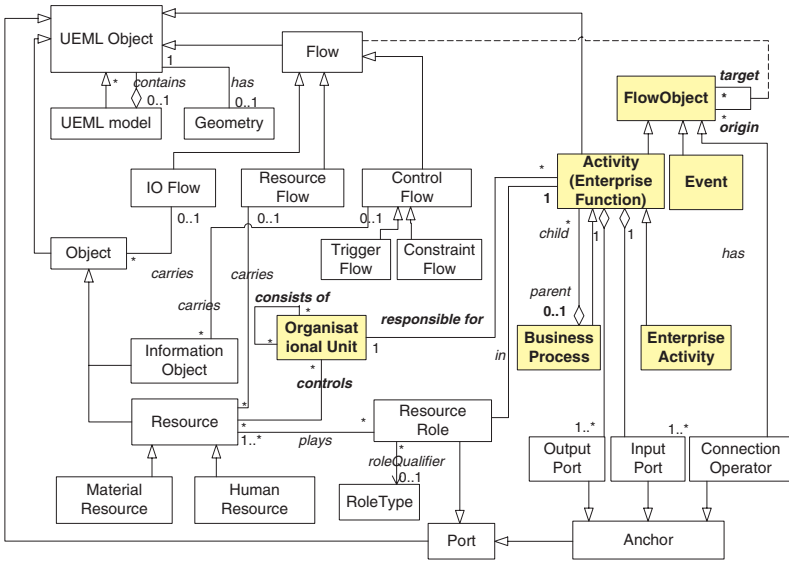


Fig. 3. Possible changes of UEML for aligning it with ISO/DIS 19440

Finally, the *Organisational Unit* class is introduced into UEML. As the definition of organisational structure as outlined in the ISO/DIS 19440 specification contains some unclear points, the minimal solution in figure 3 is only partially inspired by the one proposed in the ISO/DIS 19440. As an example, the organisational structure here is only captured by a single *Organisational Unit* class and the relationship *consists-of* having this class as domain and range. This construct, even if simple, is general enough for capturing both hierarchical as well as network structures. Finalising these extensions we are connecting *Organisational Unit* to *Resource* and *Activity* to capture the resources it controls and the activities it is responsible for.

Once again, the set of changes suggested above is not complete. But the collection is sufficient enough to exemplify how the development of the frameworks can benefit from the meta-model cross-analysis provided here.

6 Summary

In this paper, a presentation and a systematic analysis and comparison of the ISO/DIS 19440 and UEML enterprise modelling frameworks were reported. The main purpose of this work was to identify similarities and differences of the frameworks in order to further enhance them. The underlying assumption was that making them more complete and well defined would make them more attractive in contexts where interoperability is important.

The method used was in the form of a meta-model cross-analysis and the result is a mapping table between the frameworks. A number of extensions for one of the frameworks, UEML, was suggested in order to demonstrate the benefits of the work. These suggestions were presented in a graphical model and motivated in the text. The analysis was quite coarse depending on the low level of formalisation of the frameworks, but the results are nevertheless useful since both frameworks are under heavy development and the result may serve as an input to respective development teams.

Future research, which will be conducted within the on-going Interop NoE [9] project, involves a study where one and the same case will be modelled in both UEML and ISO/DIS 19440. Resulting models will be compared and the knowledge gained during this process will be used for fine adjustments of the analysis results provided this far.

Since we in this paper focused on suggesting improvements of UEML, completing this work and making the improvements compliant with the methodology used during the development of UEML, an additional enterprise modelling language needs to be included in the analysis. Only improvements supported by both ISO/DIS 19440 and this additional language will qualify to be taken into consideration when extending the UEML.

References

1. Extended Enterprise Resources, Networks and Learning, EC project. IST- 1999-10091, 2000.
2. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
3. G. Berio, et al. D3.1: Requirements analysis: initial core constructs and architecture, UEML TN IST 2001 34229. <http://www.ueml.org>, 2002.
4. J.A. Bubenko. Information system methodologies - a research view. In T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart, editors, *Information Systems Design Methodologies: Improving the Practice*, pages 289–318, North-Holland, Amsterdam, The Netherlands, 1986.
5. J.A. Bubenko and B. Wangler. *Research Directions in Conceptual Specification Development. Conceptual Modeling. Databases and CASE: An integrated view of Information Systems Development*. John Wiley, 1992.
6. G. Doumeingts, B. Vallespir, and D. Chen. Decision modelling GRAI grid. In P. Bernus, K. Mertins, and G. Schmidt, editors, *Handbook on architecture for Information Systems*. Springer-Verlag, 1998.

7. B.A. Fitzgerald. The use of systems development methodologies in practice: A field study. *The Information Systems Journal*, 7(3):201–212, 1997.
8. P. Green and M. Rosemann. Integrated Process Modeling: An Ontological Evaluation. *Information Systems*, 25(2):73–87, 2000.
9. INTEROP NoE: IST-508 011 Network of Excellence. Interoperability Research for Networked Enterprises Applications and Software,. Technical report, 2003. <http://www.interop-noe.org>.
10. ISO. EN ISO/DIS 19440, Enterprise integration - Constructs of enterprise modelling, Draft version. Accessed December 2004 from <http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=33834&scopelist=PROGRAMME>, 2004.
11. R. Jochem. Common Representation through UEMML-Requirements and Approach. In *International IFIP Conference on Enterprise Integration and Modelling Technology. Enterprise Inter- and Intra-Organizational Integration.*, Valencia, Spain, 2003. Kluwer.
12. OMG. UML 2.0 Superstructure Specification. OMG Draft Adopted Specification, ptc/03-08-02, Aug 2003. <http://www.omg.org/cgi-bin/doc?ptc/2003-08-02>.
13. M. Rosemann and P. Green. Developing a meta model for the Bunge-Wand-Weber ontological constructs. *Information Systems*, 27:75–91, 2002.
14. WfMC. Workflow Management Coalition Terminology & Glossary, Document Number WFMC-TC-1011, Document Status - Issue 3.0. Technical report, Workflow Management Coalition, Brussels, Belgium, February 1999.

Inter-agent Communications During the Virtual Enterprise Creation

Kamel Boukhelfa and Mahmoud Boufaida

LIRE Laboratory, Department of Computer Science,
Mentouri University of Constantine, 25000 Algeria,
Tel & Fax: (213) 31.63.90.10
{boukhelfakamel, boufaida_mahmoud}@yahoo.fr

Abstract. In the context of the virtual enterprise (VE), co-ordination is a process in which the agents share information about their activities. It can be realized through communication. Communication among the various actors of the VE is necessary for the successful realization of all their activities. The multi-agent approach for the development of VEs permits an effective management of the dynamic aspects of these ones. Thus, the use of a high level communication language preserves the entire richness of the interactions within a VE. So, it permits a first level of interoperability among the agents (partners companies). In this paper, we study different aspects related to the communication within a VE, during the creation phase.

1 Introduction

In a global marketplace, companies are compelled to adopt new forms of organisation that are mobile and reactive, and they are brought to focus on their skills and contract alliances in order to satisfy the needs of customers. These alliances must permit the constitution of teams with members coming from different companies to work together on a specific project, so as to accelerate production, while improving the quality and reducing substantially the costs. The following definition is one of several possible definitions of these alliances, usually called "Virtual Enterprises". A virtual enterprise (VE) is a temporary network of legally independent enterprises or individuals unifying their means, skills and other resources to work on a common project possibly transcending the capacities of each unit considered separately. This network aims at exploiting volatile opportunities, accessing new markets, and sharing costs and risks, with using new information and communication technologies.

The different actors of the VE should collaborate in an effective way to achieve their common purposes (project). So, they should supply competences, manage and produce knowledge, execute tasks and communicate permanently. The communication among the various actors that constitute the VE is necessary for the successful realization of all the activities of the VE. So, no activity of co-ordination and negotiation will take place without the definition of an effective mechanism of communication.

Recently, the MAS (for Multi-Agent System) became the dominant paradigm for developing complex systems, distributed information systems, and human-machine interfaces. The importance of the concepts of this paradigm lies on its aptitude to model complex knowledge and systems that are distributed, co-operative and intelligent. The multi-agent approach for the development of the VEs permits an effective management of the dynamic aspects of these VEs. Thus, the use of a high level communication language preserves the entire richness of the interactions within a VE.

In this paper, we study the different aspects related to the communication in the virtual enterprises developed with the agent approach. In this perspective, we use here the generic agent-based architecture proposed in [1].

2 Related Work

Most projects related to the development of VEs, such as NIIP [9], PRODNET [2], MASSYVE [12] are empirical in nature. They are meant to develop standards and reference architectures for the VE that apply to specific sectors of activity.

In terms of communication, PRODNET supplies an infrastructure of communication represented with a module responsible for the process of all the communications with the other nodes in the network (VE). It includes features such as selection of the protocol of communication, management of basic communications, mechanisms of privacy (cryptography) and secure channels for the exchange of messages between the various nodes [3].

The NIIP reference-model defines two interfaces of communication in the VE. The people to people interface that is available via conventional telephone or Internet and the agent to agent interface which is characterized by the use of KQML [9].

Some research works using the agent-based approach for modelling the VE assume that every individual enterprise is represented by an agent [6]. So, Camarinha et al. [2], Rocha et al. [13] proposed frameworks to develop VEs in which, the co-ordination mechanisms existing during the various phases of the VE's life cycle are developed through an inter-agent communication. Here, we can cite the *Opera's* work [10] concerning co-ordination in a distributed MAS modelling the VE, that of Petersen et al. [11] using intelligent agents to model and select partners in a VE, and that of Florea [7], which presents a MAS model to support business processes. In all of these works, communication is based on the exchange of messages among the different partners.

The various research works using the agent-approach aroused, do not present complete models for the VE. They study only some aspects of the VE (e.g. Co-ordination, Business Process modelling). Agent-based models adopted for the VE handle generally only of the first phase of VE life-cycle (partners selection and contracts negotiation). Some studies [10] claim that the operation phase is implicitly handled by the proposed model, without explaining how this phase is covered and without giving so much, the structures details of the agents in this model.

In this view, we use a generic agent-based architecture [1] as a base to study some aspects of communication during the creation phase of the VE.

3 The Global Architecture

The global architecture proposed for the development of a VE is generic and based on the notion of agent. It includes all the concepts necessary to ensure all the phases of the VE life-cycle [1]. We defined several types of agent, namely, the Enterprise Agent representing an individual enterprise, the Broker Agent, which is the initiator of the VE (creation phase), the VE Manager Agent (during the operation phase and the dissolution one) and the Electronic Market Manager Agent. The co-ordination and communication mechanisms recommended in the agent-based approach are also specified. The basic idea is to use the concepts of MAS to perform the different activities of the VE, and thus, to adapt the solutions provided by the MAS paradigm to solve the different problems encountered while establishing a VE.

The enterprises are considered to be within a sole electronic market. The Broker Agent reacts by seizing deal opportunities existing in the market, and proceeds thus to establish the corresponding VEs.

In the following, we give only the structure and the role of the principal agents that intervene during the creation phase: the broker agent and the enterprise one.

3.1 Description of the Enterprise Agent

We consider the Enterprise Agent as an entity including a communication module, a planning and co-ordination module, and an execution module. It has also an interface for the interaction with the user and two other knowledge bases: the individual knowledge base and the VE knowledge one (Fig. 1).

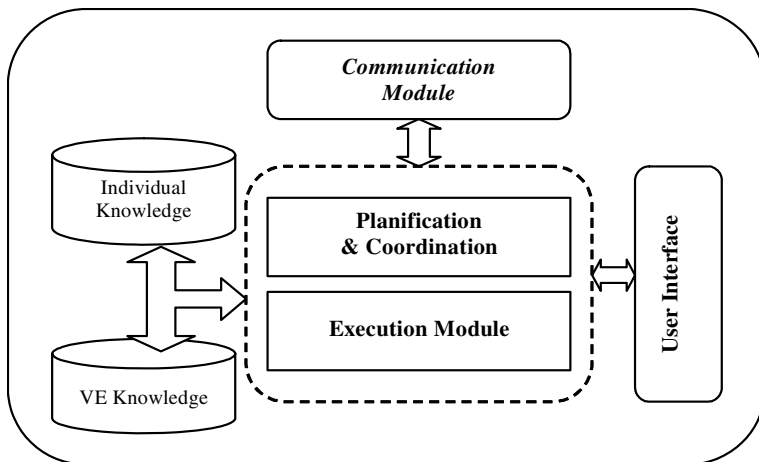


Fig. 1. Structure of an Enterprise Agent

- *The Communication Module*: contains all the processes required to handle the messages: reception, filtering, and translation of incoming messages, and formulation and sending of the outgoing messages.

- *The User Interface*: permits the interaction between the Enterprise Agent and the human agent (user). This latter assumes a decision role, whether to take part in a virtual enterprise by accepting the terms of the contract established with the broker, or in domains related to the exchange of confidential information.
- *The Planning and Co-ordination Module*: is responsible for managing the co-operation and formulating the offers for achieving sub-goals announced by the broker, it allows the agent to compete with other agents for membership in the VE.
- *The Execution Module*: this module contains the information about the internal resources of the individual enterprise, which makes possible the performance of local tasks that are assigned to the enterprise. It establishes the correspondence between the sub-goal assigned to the agent and the internal resources of the enterprise capable of achieving this sub-goal.
- *The VE-Knowledge Module*: contains information related to the organisational and operational rules defined by the VE. It contains a list of all other agents, member of the VE(s) corresponding to this agent. This module also possesses the information pertaining to the rights and obligations of the individual enterprise.
- *The Individual-Knowledge Module*: contains information about the agent itself: its capacities and skills, and the current state and workload, i.e. for each skill, indicators are assigned to determine availability, as well as the cost of such skill.

3.2 Description of the Broker Agent

The role of the Broker Agent (initiator) is to look for and identify the business opportunities in the market, to select appropriate partners to seize this opportunity, and to co-ordinate the activities of the VE. It is the primary contact with the customers of the VE. Once the Broker Agent finds out the description of the global goal from the information supplied by the customer, it proceeds to the decomposition of the global goal into sub-goals. Then, the Broker Agent allocates each sub-goal to the agent potentially capable of contributing to their achievement (Fig. 2).

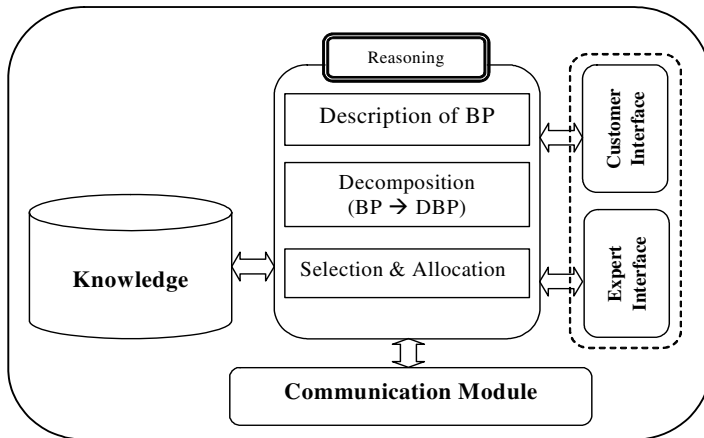


Fig. 2. Structure of the broker agent

The components of the Broker Agent are:

- *The Communication Module*: This module handles the whole communication process with the agents of the electronic market. It formulates the messages according to the language adopted (ACL or KQML), and assumes the function of translating messages received from various agents during the creation of the VE.
- *The User Interface*: the Broker provides a graphical interface for interaction with the customer and another one for interaction with the human broker (the expert).
- *The Reasoning Module*: contains the following parts:
 - *The "Global-Goal Description" Module*: the human broker (the expert), with the assistance of the broker agent, fetches the order of the customer and formulates a description of this order as being the global goal to be achieved, or, more precisely, the "Business Process".
 - *The "Global-Goal Decomposition" Module*: the broker agent proceeds to the decomposition of the global goal (described beforehand) on the basis of human expertise and/or learning from the cases encountered previously (use of knowledge acquired from past experience).
 - *The "Selection and Allocation" Module*: the function of this module is to formulate announcements corresponding to the sub-goals resulting from the decomposition.
- *Knowledge*: The knowledge of the Broker are of two sorts:
 - Knowledge acquired through learning, i.e., new knowledge that the broker acquires from past experiences with formerly requested products, about which it keeps information (description, the member agents of the corresponding VE, etc.)
 - A priori knowledge concerning its environment (electronic market).

4 Communications in the Agent-Based VE

The agents need to communicate among them, as well as with the services of their platform or environment. Several mechanisms of communication are possible: message exchanging and sending, methods invocation and the use of the "blackboard" mechanism. Consequently, standardized inter-agent communication languages should be supplied.

KQML (Knowledge Query Management Language) [4] was proposed to support inter-agent communication. This language defines a set of types of messages (called "performatives") and rules, which define the suggested behaviour for the agents that receive these messages. This language was developed in an ad-hoc way for the requirements of the developers of agent's software packages.

The ACL (Agent Communication Language) [5], successor of KQML, supplied a richer semantics. This language was proposed by FIPA, which aims to standardize communications among agents. ACL is also based on the language theory and is close to KQML at the level of the acts of the language, but not at the semantics level, which underwent a big improvement in ACL.

In our approach, we use the ACL to formulate messages and the XML to describe the contents of messages. The use of ACL-XML in inter-agent communications permits to achieve a first level of interoperability by surpassing the problem of heterogeneous exchanges among the different actors in the VEs.

4.1 The Exchanged Messages

FIPA [5] supplies a standard syntax for messages. These messages are based on the theory of the act of speech, which is the result of the linguistic analysis of human communication [5]. The basis of this theory is to produce an action from the language.

In the FIPA-ACL, no specific language for the description of the contents of messages is imposed. Several languages can be used for the description of the contents of the exchanged messages such as KIF (Knowledge Interchange Format), Semantic Language (SL), Prolog and XML (eXtensible Mark-up Language).

XML will be used for the description, the specification and the interpretation of the contents of messages exchanged within the VE. So, the messages exchanged among the agents are described in FIPA-ACL/XML. The use of XML for the contents of communications among agents permits the display of messages in a Web browser and facilitates the integration with existing Web-based applications.

The following example illustrates interaction between the Broker Agent and an Enterprise Agent, during the creation phase of the VE. The Broker announces the sub-goal "manufacturing of cylinder" as the following:

```
(request
:sender Broker-Agent
:receiver Enterprise-Agent(i)
:language XML
:content (
<Sub_Goal SB_id = 'manufacturing of cylinder'>
  <Delay> 100 </Delay>
  <Price> 1000 </Price>
  <List_Att>
    <Attribute Att_name = 'Dimension'>
      <Pref_Val> 100 </Pref_Val>
      <Dom Val_min='80' Val_max='120' />
      <Util> 75 </Util>
    </Attribute>
    <Attribute Att_name = 'Material'>
      <Pref_Val> Steel75 </Pref_Val>
      <Dom Val_min=' Steel65' Val_max='Steel80' />
      <Util> 25 </Util>
    </Attribute>
  </List_Att>
</Sub_Goal>
:reply-with propose)
```

The specification of the Sub_Goal (Price, Delay and attributes values) are described with XML in the "content" of the announcement message.

The Enterprise Agent answers by submitting a bid. The values of attributes proposed for this purpose ("manufacturing of cylinder") will be evaluated and compared with other values proposed in the bids of the other Enterprise Agents.

Similar to the case of Broker Agent, the details of the bid of the Enterprise Agent are expressed in XML, and the values of attributes are provided.

```

(propose
:sender Enterprise-Agent(i)
:receiver Broker-Agent
:language XML
:content (
<bid bid_id = 'bid_cyl_1'>
  <EAg_id> Enterprise Agent (i) </EAg_id>
  <Goal_id> manufacturing of cylinder </Goal_id>
  <Delay> 100 </Delay>
  <Price> 1050 </Price>
  <List_Att>
    <Attribute Att_name='Dimension'>
      <Val> 105 </Val>
    </Attribute>
    <Attribute Att_name='Material'>
      <Val> Steel75 </Val>
    </Attribute>
  </List_Att>
</bid>)
:reply-with inform)

```

4.2 The Communication to Support the Co-ordination and the Negotiation

In a VE, an agent acting through a well-defined negotiation protocol represents each member-enterprise. In an electronic market, the creation of a VE requires the initiation of competition among the various agents that send offers in answer to the Broker announcements, and this in view of being selected as partners in the VE.

Negotiation is the technique we use to be the co-ordination mechanism during the creation phase of the VE. We will detail the negotiation protocol associated with the proposed architecture in order to allow us the definition of a framework for co-ordination during this phase.

The “e-Market Manager Agent” keeps a register (Yellow Pages) of membership for the electronic market. Each enterprise member (agent) in the market has an entry in this register. This entry includes a list of all the skills offered by this enterprise. For each sub-goal, the Broker Agent, which is the initiator of the VE in this approach, asks the various potential agents to tender their offers, i.e., to give the necessary information concerning their activities (experience) in order that the Broker Agent checks them. Then, this latter decides (or not) to initiate negotiation for finding an agreement with the future partner. Such a negotiation can end upon withdrawal of either of the parts.

Based on the Contract-Net Protocol [5], the generic interaction protocol during the creation of the VE is the following:

1. The Broker announces the creation of the VE and the allocation of tasks through posting of an announcement (call for proposals) meant for the interested participants in the electronic market. For each task to be completed, the Broker announces a specification containing its description and a list of the requirements and constraints relating to this task;

2. The interested participants answer the announcement by making bids. According to well-determined criteria, the Broker evaluates the bids. Through this evaluation, the potential partners are determined;
3. The broker gets ready to negotiate with the potential partners (those which fulfil the requirements);
4. Finally, after negotiation, and rejection by the broker of certain uninteresting offers, certain potential partners will become members of the VE, each one with a signed contract to be executed.

Fig. 3 illustrates the process of negotiation between the Broker Agent and an Enterprise Agent. The activities that the Broker realizes in the "Selection and Allocation" module during the process of negotiation, in correspondence with those that the "Planning and Coordination" module achieves in the Enterprise Agent.

The negotiation process is supported by the "Selection and Allocation" module in the Broker Agent. On behalf of the Enterprise Agent, the "Planning and Coordination" module handles the process of negotiation. The FIPA-ACL messages are formulated and sent by the communication module of the Broker Agent and of the Enterprise Agents.

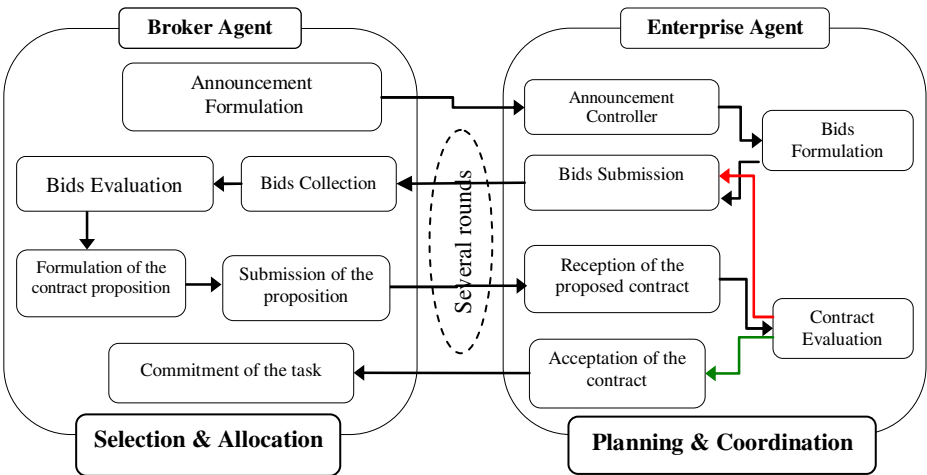


Fig. 3. Negotiation process between the Broker and an Enterprise Agent

5 Case Study and Simulation

We have simulated the creation of a VE in the domain of building of individual houses, named ENVICOL.

In this case study, we have to approach the problems of the co-ordination and so the communication during the phase of creation of the VE. The individual companies are clustered in a sole electronic market, dedicated to businesses related to the civil engineering (buildings, individual houses construction, etc.).

Form the implementation viewpoint, we would adopt the use of an agent-based platform to implement the different agents and concepts of the VE. Several platforms are supplied as software packages, such as JADE [8], ZEUS [16] for the cognitive agents or SWARMS [15] for the reactive agents.

5.1 Using JADE to Develop the Virtual Enterprises

JADE (Java Agent DEvelopment framework) is a multi-agent platform, developed in Java by CSELT (Research Group of Gruppo Telecom, Italy). It provides a FIPA-compliant environment for development and execution of MAS [8].

JADE includes two basic constituents: an agent platform and a software package for the development of the agents in Java. It supplies several facilities, among them, we can mention the following [8]:

- A distributed agent platform. The agent platform can be split among several hosts (provided they can be connected via RMI). Only one Java [14] application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as Java threads and live within Agent Containers that provide the runtime support to the agent execution.
- An efficient transport of ACL messages inside the same agent platform. In fact, messages are transferred encoded as Java objects, rather than strings. When crossing platform boundaries, the message is automatically converted to/from the FIPA compliant syntax, encoding, and transport protocol. This conversion is transparent to the agent implementers that only need to deal with Java objects.
- A library of FIPA interaction protocols that are ready to be used.

In JADE, an agent is an instance of the Java class defined by the programmer. This class itself is an extension of the basic Agent class (included in `jade.core`). It implies the inheritance of the set of basic methods to implement the personalized behaviour of the agent. The agent is implemented using multitasking, where the tasks (behaviours) are executed concurrently. Every functionality supplied by an agent must be implemented in one or several behaviours.

The following Java code represents the implementation of the Broker class and the EnterpriseAgent class. These classes are extensions of the basic Agent class defined in JADE. The Broker Agent and the Enterprise Agent are instances respectively of the Broker class and EnterpriseAgent class.

```
Public class Broker extends Agent {
    Protected void setup() {
        addBehaviour(new SimpleBehaviour(this){
            // Processing...
        })
    }
}

Public class EnterpriseAgent extends Agent {
    class reception extends SimpleBehaviour {
        // Processing...
    }
    Public reception(Agent a){ super(a); }
    protected void setup() {
        reception mybehaviour = new reception(this);
    }
}
```

```

    addBehaviour(mybehaviour);
}}

```

For the implementation of a VE, we consider that the use of the JADE platform is based on an additional layer on the Internet-Intranet network. Thus, every Enterprise Agent is launched in a separate host and in an appropriate container (Fig. 4). The Broker Agent, e-Market Agent and the VE-Manager Agent could be launched separately or in the same host.

5.2 Interaction Simulation

The interaction between the Broker Agent and the set of Enterprise Agents takes place as soon as the global goal is formulated and decomposed. The Broker produces a call for tender for this goal and determines for each sub-goal the corresponding Enterprise Agent that will be a VE-member. For each selected Enterprise Agent, the Broker sends the first version of the contract. If one or more clauses are not accepted by the Enterprise Agent, the Broker launches a round of negotiations to resolve the discord on the clauses of the contract. It is possible, that the Broker proceeds to several rounds of negotiations before arriving at the establishment of the final contract with the selected Enterprise Agent.

In the case illustrated in Fig. 5, the Enterprise Agent “EAg2” refuses to be a member in the VE to be created, whereas, each of the Enterprise Agents: EAg1, EAg3 and EAg4 answers by the proposition (PROPOSE) of an offer to the announced sub-goal.

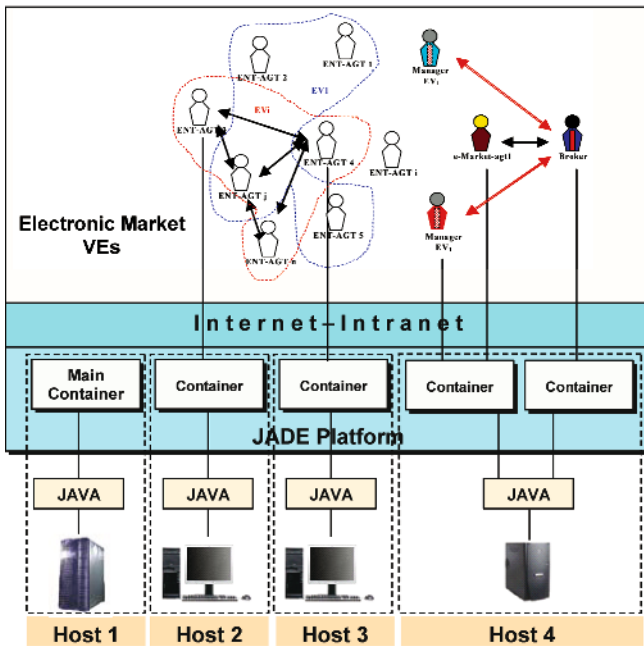


Fig. 4. Implementation of a virtual enterprise in JADE

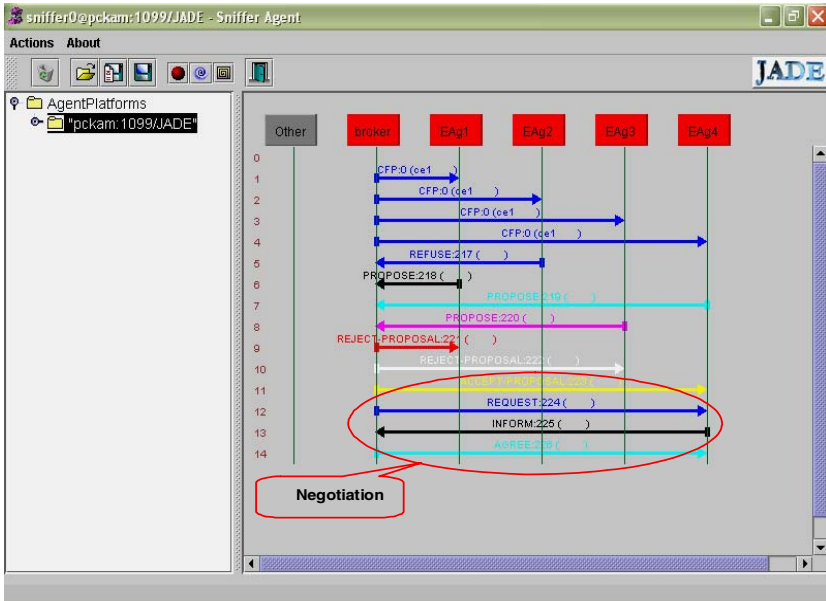


Fig. 5. Simulation of interactions between the Broker Agent and a set of Enterprise Agents

The Broker Agent evaluates the different offers and rejects those of the agents EAg1 and EAg3 (REJECT-PROPOSAL) and accepts that of the agent EAg4 (ACCEPT-PROPOSAL).

6 Conclusion

The communication among the various actors that constitute the VE is necessary for the successful realization of all the activities of the VE. In order to study the different aspects related to the communication in the VEs during the creation phase, we have used the generic agent-based architecture proposed in a previous work [1].

The multi-agent approach for the development of the VEs provides high level communication languages that preserve the entire richness of the communications within a VE. Thus, the FIPA-ACL is used in our approach to formulate messages exchanged between the different agents constituting the VE. The content of these messages is described with XML.

The use of the JADE platform has allowed us to consider an implementation of a VE, to implement the Broker Agent and the Enterprise Agents and thus, to simulate the interaction between these entities, which is based on sending of ACL/XML messages.

However, this study is subject to a future extension concerning the other aspects of communication in the operation and dissolution phases of the VE life cycle. During these phases, there is a need to several mechanisms to permit information exchange (commercial, technical, product/service, etc.) and interoperability between the various used standards.

Acknowledgment

We would like to thank the INTEROP NoE (Network of Excellence, IST 508011, www.interop-noe.org) for its partial financial support covering the accommodation to attend the BPM-ENEI'05 workshop

References

1. Boukhelfa, K., Boufaida, M.: A Generic Multi-Agent Architecture for the Virtual Enterprise. In: EMISA'2004 Information Systems in E-Business and E-Government, LNI Edition Vol 56. Luxemburg (2004) 199-210
2. Camarinha-Matos L. M., Afsarmanesh, H., Rabelo, J.: Infrastructure developments for agile virtual enterprises. In: J.Computer Integrated Manufacturing (2002)
3. Camarinha-Matos L. M., Lima, C. P.: A Framework for Cooperation in Virtual Enterprises. Proceedings of DIISM'98, Design of Information Infrastructures Systems for Manufacturing. Fort Worth, USA (1998)
4. Ferber, J.: Les systèmes multi-agent: Vers une intelligence collective. InterEdition (1997) 239-245.
5. FIPA, (1999). <http://www.fipa.org/spec/FIPA98.html>.
6. Fischer, K., Müller J. P., Heimig, I., Scheer, A. W.: Intelligent Agents in Virtual Enterprises. In: Proceedings of PAAM96 (First International Conference on "The Practical Applications of Intelligent Agents and Multi-Agent Technology"), London, 22nd-24th April, (1996) 205-223.
7. Florea, A. M.: Intelligent Agents Technology in Virtual Enterprise Design. Department of Computer Science "Politechnica", University of Bucharest (1998) http://cs.pub.ro/cs-dept/ai_mas/publications/F-ISOCE98.zip
8. Fabio, B., Giovanni, C., Tiziana, T., Giovanni, R.: Jade Programmer's Guide. University of Parma, (2000-2003). <http://jade.cselt.it/>
9. NIIP Reference Architecture Concepts and Guidelines. Report No. NTR95-01. Jan (1995). <http://niip01.npo.org/>
10. Oprea, M.: Coordination in an Agent-Based Virtual Enterprise. In: Studies in Informatics and control. Vol. 12, No.3, September (2003) 215-225.
11. Petersen, S. A., Matskin, M.: Agent Interaction Protocols for the Selection of Partners for Virtual Enterprises. In: Multi-agent Systems and Applications III, 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic. V. Maik, J. Müller, M. P chou ek (Eds.), LNAI 2691, Springer-Verlag, (2003)
12. Rabelo, R. J., Afsarmanesh, H., Camarinha-Matos, L. M.: Federated Multi-Agent Scheduling in Virtual Enterprises. In: E-business and Virtual Enterprises—Managing Business-to-Business Cooperation. Kluwer Academic Publishers, (2001) 145-156
13. Rocha, A., Oliveira, E.: An electronic market architecture for the formation of virtual enterprises. (1999). <http://www.fe.up.pt/~eol/PUBLICATIONS/espamecoo.ps.zip>
14. SUN - JINI Technology Architectural Overview, Jan (1999) <http://www.sun.com/jini/whitepapers/architecture.html>
15. SWARMS (1996). <http://www.swarm.org/index.html>
16. ZEUS (1999). <http://www.labs.bt.com/projects/agents/zeus>

Applying Enterprise Models to Design Cooperative Scientific Environments

Andrea Bosin¹, Nicoletta Dessi¹, Maria Grazia Fugini²,
Diego Liberati³, and Barbara Pes¹

¹ Università degli Studi di Cagliari, Dipartimento di Matematica e Informatica,
Via Ospedale 72, 09124 Cagliari
andrea.bosin@dsf.unica.it,
{dessi, pes}@unica.it

² Politecnico di Milano, Dipartimento di Elettronica e Informazione,
Piazza da Vinci 32, I-20133 Milano
fugini@elet.polimi.it

³ IEIIT CNR c/o Politecnico di Milano, Piazza da Vinci 32, I-20133 Milano
liberati@elet.polimi.it

Abstract. Scientific experiments are supported by activities that create, use, communicate and distribute information whose organizational dynamics is similar to processes performed by distributed cooperative enterprise units. On this premise, the aim of this paper is to apply existing enterprise models and processes for designing cooperative scientific experiments. The presented approach assumes the Service Oriented Architecture as the enacting paradigm to formalize experiments as cooperative services on various computational nodes of a network. Specifically, a framework is proposed that defines the responsibility of e-nodes in offering services, and the set of rules under which each service can be accessed by e-nodes through service invocation. By discussing a representative case study, the paper details how specific classes of experiments can be mapped into a service-oriented model whose implementation is carried out in a prototypical scientific environment.

1 Introduction

A key success factor to promote research intensive products is the vision of a large scale scientific exploration carried out in a networked environment, with a high performance computing infrastructure, e.g., of grid type, that supports flexible collaborations and computation on a global scale. The availability of such a virtual cooperative environment should lower barriers among researchers taking advantage of individual innovation and allowing the development of collaborative scientific experiments.

Up to now, however, rarely are technologies developed specifically for the research community, and ICT developments are harnessed to support scientific applications varying in scale and purpose and encompassing a full range of engagement points, from single-purpose-built experiments to complex support environments.

The range of accessible technologies and services useful to scientific experiments can be classified broadly into three categories:

- Toolkits specifically aimed at supporting experiments.
- Software tools that are not specifically on-purpose for support to experiments, but that are still essential in enabling them (e.g., Matlab, data mining tools, data warehousing).
- More widely deployed infrastructures that may be useful in scientific experiments, such as Web services, or Grid computing.

Hence, a problem is the *integrated* use of heterogeneous applications and software tools that were not designed specifically to promote interaction and cooperation, but still are inherently suitable for cooperation support. This scenario is similar to that of enterprise environments, whose progress requires large-scale collaboration and efficient access to very large data collections and computing resources [1]. Although sustainable interoperability models are emerging for market players (such as service providers, stakeholders, policy makers, and market regulators), they are currently deployed mostly in areas where high computing power and storage capabilities, usually needed by scientific environments, are not mission-critical.

Recently, emerging technologies such as Web Services and the Grid [2] [3] [4] have enabled new types of scientific applications consisting in a set of services to be invoked by researchers. Assuming the Service Oriented Architecture (SOA) as enacting paradigm [5], the purpose of this paper is to explore the application of existing enterprise models and processes for supporting distributed scientific services. To this aim, we address how experiments can be formalized as workflows that express a sequence of cooperative tasks, each performed by a set of e-services. The “enterprise” environment supporting distributed scientific “processes” is a network of cooperative e-nodes (e.g., the research laboratories, the hospitals, the analysis centers) having a local configuration and a set of shared resources. Since an experiment involves multiple e-nodes interacting with one another in order to offer or to ask for services, the paper proposes a framework that defines the responsibilities of e-nodes in offering services, and the set of rules under which each service can be accessed by e-nodes through service invocation.

To illustrate how a methodology for executing scientific experiments can be decomposed into simpler tasks, the paper considers four main classes of methodological approaches directly pertaining the experimental environment, whose application is mainly thought for data (and possibly also computationally) intensive processing. To identify how an experiment can be mapped onto a service oriented model, we discuss a representative case study related to one of the considered classes. In more detail, we present a model of the experiment in a 4-level UML Use-Case diagram. Finally, a prototype environment based upon emerging Web service technology is described and applied.

The paper is organized as follows. Section 2 reviews some related works on cooperative systems and e-services. The cooperative framework is described in Section 3. Section 4 outlines four main classes of experiments and Section 5 shows how these experiments can be modelled using UML diagrams. An implementation prototype is presented in Section 6. Finally, conclusions as well as future works are outlined in Section 7.

2 Related Works

In general, the term cooperative systems is used to denote distributed information systems that are employed by users of different organizations under a common goal [6][7]. An extension of the cooperative paradigm, referred to as *e-applications*, is becoming more and more frequent: e-applications allow the dynamic composition of services, referred to as *e-services* in the following, provided by different organizations on the net. In addition to geographical distribution and inter-organization cooperation, in e-applications (i) cooperating organizations may not know each other in advance and (ii) e-services can be composed both at design and run-time. Moreover, the availability of complex platforms for e-services [8] allows open cooperation among different organizations.

A classical approach to e-applications is based on UDDI registry, allowing publication and discovery of services on the Web. In UDDI, offered e-services are described, based on free text, and consumers of the e-services interact with the offering organization on the basis of interfaces. Other proposals for architectures for e-services and workflow-based environments have been recently presented in the literature [8][9]. The starting point is the concept of *cooperative process* [10][11], defined as a complex process involving different organizations.

Activities that involve multiple different organizations using e-service technology to exchange information need to be coordinated. WSDL is one of the currently most popular linguistic means providing a mechanism by which the format and structure of the exchanged messages is defined. Methods are not yet completely defined in the literature to specify the sequence and conditions, or *choreography*, of message exchange. A proposal is under study in the W3C [12]. To solve this problem, a shared common or global definition of the sequence and conditions where messages are exchanged is produced that describes the behavior of all the involved participants.

In general, high-performance computing and communication technologies are enabling computational scientists, or e-scientists, to study and better understand complex systems, allowing for new forms of collaboration with the ability to process, disseminate, and share information [13]. Global-scale experimental networking initiatives have been developed in the last years: the aim is to advance cyberinfrastructure for e-scientists through the collaborative development of networking tools and advanced Grid services [14] [15].

3 The Cooperative Framework

The concept of “what an experiment is” is rapidly changing in an ICT oriented environment, moving from the idea of a local laboratory activity towards a computer and network supported application including the integration of:

- a variety of information and data sources;
- the interaction with physical devices;
- the use of heterogeneous software systems.

In our approach, scientific experiments are modeled analogously to cooperative enterprise processes, as *e-processes* that operate on, and manipulate, data sources and

physical devices. Each e-process is attributed to and associated with a set of specific experimental tasks and workflows are employed to control these tasks.

This modeling approach is assumed as an enactment paradigm to conceive an experiment, at least for certain application domains; an experiment is regarded as an application whose tasks can be decomposed and made executable as (granular) services individually. The decomposition is based on appropriate modeling of the experiment as a set of components that need to be mapped to distinct services. The decomposition offers good opportunities for achieving an open, scalable, and cooperative environment for scientific experiments.

Let us now introduce the framework modeling the experiments on a network of cooperative e-nodes having a local configuration and a set of shared resources. Services correspond to different functionalities across several research domains, and encapsulate problem solving and simulation capabilities. All services have an e-node that is responsible for offering the service and which sets the rules under which the service can be accessed by other e-nodes through service invocation. An experiment involves multiple e-nodes interacting with one another in order to offer or to ask for services.

A user is a researcher who has the following possibilities:

- 1) selection of the experiment of interest and of the information sources he/she wants the experiment be carried on;
- 2) acquisition and collection of local data;
- 3) surveillance/monitoring of local experiments, which are part of a cooperative experiment;
- 4) definition of new experiments; this implies that the global workflow of the experiment must be designed;
- 5) inspection of remote data sources and experiment results, e.g., by mining in a data warehouse;
- 6) cooperation with users of other e-nodes, for example to co-design experiments and jointly evaluate results.

Fig.1 shows a conceptual framework for e-nodes cooperation whose basic components are Pool of Services (PS) and Smart Scientific Spaces (S^3). A **Pool of Services (PS)** is an e-node described by:

- 1) A set of local information sources;
- 2) A set of local services.

The *local information sources* collect the experimental data related to the activities assigned to the e-node. Their granularity can vary from the file, to the database, or data warehouse level. Parts of local information sources can be declared as public by the e-node and hence become part of the network (i.e., remotely accessible).

The *local services* map granular experimental tasks performed by the e-node or other local control activities. The services organize their activity on the basis of both local and network information sources, and are related to a particular experimental context by a workflow describing the tasks to be executed and the context knowledge applied to solve a problem, to enact a decision or to achieve a goal.

The *design of an experiment* is accomplished by a *Chief Scientist* user who is in charge of breaking down the experiment into a sequence of tasks to be executed in a distributed way. The granularity of these tasks is decided by the Scientist, and their

assignment to the e-nodes is negotiated in terms of resources, time, costs and other project management parameters by an *Experiment Manager* user. This user ensures that each task is realizable by some e-node according to fixed time and cost parameters, in order to avoid unrealizable and ambiguous tasks. The specified tasks express also the life cycle of the experiment and defines the set of PSs participating in the experiment.

In the case of complex experiments requiring the collaboration of various services spread over different e-nodes, the selected PSs interact in a particular environmental context called *Smart Scientific Space* (S^3) and established by a selected e-node that act as *Experiment Master Pool* (EMP). The EMP is responsible for setting up, controlling, and splitting the experiment. A representation schema enables the EMP to identify the PS allowed to interact with one another in the context of the (S^3), and a workflow describes the experiment plan, i.e., the sequence of activities to be carried on to execute the cooperative experiment.

Each interacting PS locally has the knowledge of the portion of workflow activities, called *component services*, that are assigned to the node. Making a component service available in the appropriate smart space requires registration facilities to be available at the EMP. Besides local knowledge, the local workflow includes: points of access to remote information sources, and points of cooperation with other e-nodes' workflows, for example requests of parallel processing services available on a specialized e-node, as well as possible destinations of processed information.

The EMP is also in charge of designing the cooperative experiment and of configuring a monitoring activity (project management) to ensure that the experiment plan is adhered to. A cooperative experiment is represented by a single S^3 or by a series of distinct smart spaces, each responsible for different facets of a single cooperative experiment. This mechanism enables different PSs to retain the ownership of their own experiments, but to enable the cooperation of PSs under appropriate conditions.

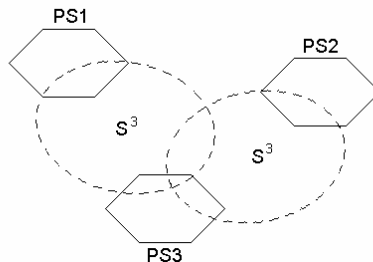


Fig. 1. Conceptual framework for e-nodes cooperation: Pool of Services (PS) and Smart Scientific Spaces (S^3)

The global experimental environment is viewed as a number of smart scientific spaces where the physical distribution of resources and access problems are masked via the service interface and the workflow ensures that appropriate PSs cooperate. As new information sources and processing techniques become available, they are simply represented as new services thus ensuring the environment scalability.

4 Classes of Experiments

In our approach, an experiment is defined as the application of a subset of the available methodological approaches to the selected data set on an execution platform composed of: 1) a workflow; 2) the distribution thereof; 3) the involved nodes and their relative roles in the experiment; 4) the set of involved resources, such as data areas, data repositories, and e-services.

Four main classes of methodological approaches to the experiments can be identified:

1. *Process Simulation and Visualization* on the already available information sources.
2. *Supervised or Unsupervised Classification* of observed events without inferring any correlation nor causality, such as in clustering, and neural networks.
3. *Machine Learning: Rule Generation and Bayesian Networks* able to select and to link salient involved variables in order to understand relationships and to extract knowledge on the reliability and possibly causal relationships among related co-factors via tools like logical networks and Cart-models.
4. *Identification of the Process Dynamics*.

Such classes, listed in increasing order of logical complexity, might have an impact on the design of the experiment and of its execution modality in terms of execution resources. For example the first class does not seem to require more than mediating possibly different repositories, while the other classes may require recursive approaches and hence intense computation either on a single specialized node or in a grid structure.

In more detail, in the following we are going to identify how an experiment can be mapped onto a service-oriented model. As a case study, we consider an experiment related to the identification of process dynamics: the piecewise affine identification problem [16].

Piecewise affine identification associates temporal data points in the multivariable space in such a way to determine both a sequence of linear sub-models and their respective regions of operation. It does not even impose any continuity at each change in the derivative. It exploits a clustering algorithm. The three following steps are executed:

Step 1 – Local linear identification: small sets of data points close to each other are like to belong to the same sub-model. Thus, for each data point, a local set is tentatively built, collecting the selected point together with a given number of its neighbors. For each local data set, a linear model is identified through least squares procedure.

Step 2 – Clustering. The algorithm clusters the parameter vectors of the identified models and thus also the corresponding data points.

Step 3 – Piecewise affine identification. Both the linear sub-models and their regions are estimated from the data in each subset. The coefficients are estimated via weighted least squares, taking into account the confidence measures. The shape of the polyhedral region characterizing the domain of each model may be obtained via Linear Support Vector Machines [17].

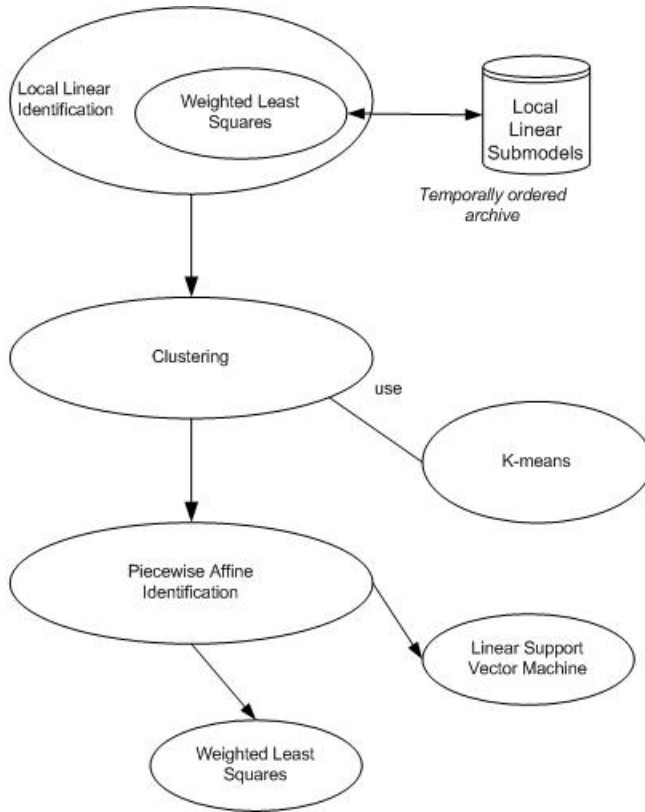


Fig. 2. Steps of the identification of the process dynamics experiment

The steps are depicted in Fig.2, where the circles represent activities and the arrows represent dependencies. The K-means algorithm is employed (*use* line) and can be (re)used several times. As an independent step, the Clustering activity can be assigned to a dedicated node. Also the algorithm deputed to define the regions in which every linear model is valid may be one of the many general purpose available for such task (here a simple Support Vector Machine is chosen). As such, it is also indicated as a separate re-usable module, useful in many other problems. Moreover, the Weighted Least Square module is another possible re-usable e-service, being used twice within this kind of experiment.

The illustrated experiment shows that some portions, both of processes and of data or knowledge, can be shared in a collaborative environment. One of the reasons for executing an experiment in a distributed way might be that one organization needs to process data under a specific costly product available on a node; rather than acquiring the product (e.g. SAS, Oracle, or Matlab), the organization invokes a remote service on the remote node endowed with the license. Another reason is that some cooperating organizations might want to inspect data dispersed on their databases, with no changes to their local computational environment.

5 Modeling Scientific Experiments

In the current laboratory practice, a researcher begins with the assertion of a high level goal needed to test a scientific hypothesis or to obtain some additional knowledge on a previous experiment.

According to the presented approach, this goal has to be decomposed into a set of tasks (the experiment life cycle) each accomplished by an appropriate class of services. From a methodological point of view, we observe that heterogeneous services can provide similar capabilities, but the Chief Scientist is in charge of choosing the most suitable methods to accomplish each task, that is, he is in charge of designing the workflow of the scientific experiment. In particular, if the researcher wants to rerun an experiment, the workflow must take into account the changes in the choice of methods as well as in the service availability.

In the presented approach, the Chief Scientist interacts and chooses services, workflows, and data within an experimental environment whose cooperative framework has been defined to extend the integration of scientific experiments to a level of scenario-based interaction. This scenario is profitable for many reasons, like exchanging scientific data and processing tools which results in a reduced number of software acquisitions, or load balancing work between specialized researchers.

The modeling process is organized in three steps.

- 1) The experiment is decomposed into a set of basic tasks.
- 2) A model is provided that structures basic tasks according to the template of the experiment.
- 3) The experiment flow is stated by the definition of a public workflow for all pools involved in the experiment.

In some more detail, we again focus on *Identification of the Process Dynamics* experiments since they are representative for the considered classes of experiment. The experiment decomposition has been described in the 4th class of experiments, and Fig. 2 shows the flow of basic tasks. This flow may be considered identical for all the first-glance experiments on *Identification of the Process Dynamics*, except for the choice of the clustering methodology, that can be different from the k-means algorithm.

We then consider the experiment to be performed in the following scenario. Two researchers at two different PSs (namely the pools PA and PB) carry on a cooperative experiment belonging to the class *Identification of the Process Dynamics*. The pool PB acts as EMP and the pool PA as a cooperative pool. Since we suppose that a Support Vector Machine knowledge is not available at either pools, they need the cooperation of a third pool PC, where the Support Vector Machine service is available. Also, it may be substituted by a more complex but also more refined technique, should the specifications of the experiment require such effort, which would probably need in turn a still different domain knowledge.

Fig. 3 shows the model of the experiment in a 4-level UML Use-Case diagram. Pools are modeled by participating actors invoking services offered at different levels. The core idea is that the scientist designs an abstract workflow out of the available services by hiding the low-levels details.

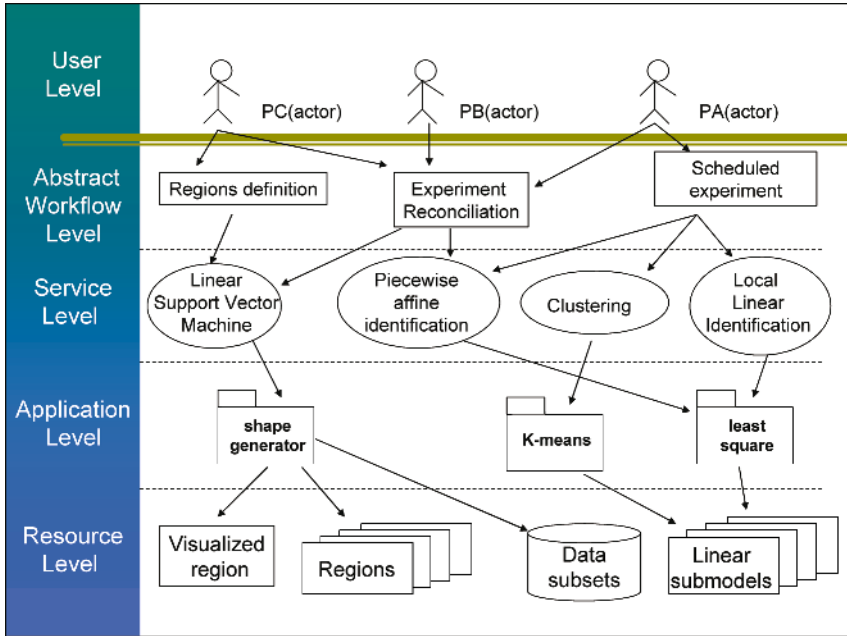


Fig. 3. The model of the experiment in a 4-level UML Use-Case diagram

It is important to outline that our approach takes into account the structure of the whole experiment rather than only the workflow layers as conventional workflow systems do.

6 Addressing Implementation Challenges

The proposed approach views a scientific experiment as a process and a flow of relationships rather than a compartmentalised event. It assumes that users are similar in that ready access to various resources will empower collaborative experimentation for novice users as well as for experts. What remains open, however, are many questions about the evaluation of such an approach. Since the realisation of a distributed general purpose scientific environment is not immediate, the evaluation effort described here involves a prototypical environment based upon emerging web service technology and applied to the above mentioned four classes of experiments.

The prototype is intended to help scientists to extract increasingly complex knowledge from data, promoting information reuse in well defined experimental patterns. In this section, we detail the general architecture of the prototype that implements a set of basic data mining tasks as widely accessible Web services. What we intend to demonstrate is how to realize in practice data mining e-services that can be used in a distributed cooperative experiment, according to the layered model shown in Fig. 3. In particular, the abstract workflow layer, dealing with planning and composition of all the tasks that make up an experiment, has not been implemented yet. We plan to

do it making use of a workflow engine based on WfMC [18] and OMG [19] specifications, which uses the WfMC XPDL definition format, such as Enhydra Shark [20].

The implementation of lower levels is based on the J2EE [21] and Oracle [22] platforms; however, the use of standard technologies (HTTP, XML, SOAP, WSDL) and languages (Java, SQL) makes it flexible and easily expandable.

The prototype maps the layered model into the multi-tier architecture shown in Fig. 4. While the tier separation can be purely logical, our prototype allows the physical separation of tiers, where each one is located on a separated and networked hardware resource.

The user tier represents the consumers of data mining services. Clients located across the Web (i.e. on the e-nodes of the experiment which requires the data mining service) invoke the data mining services provided by the service layer. Clients are implemented by standalone Java applications that make use of existing libraries (J2EE application client container) in charge of the low-level data preparation and communication (HTTP, SOAP, WSDL).

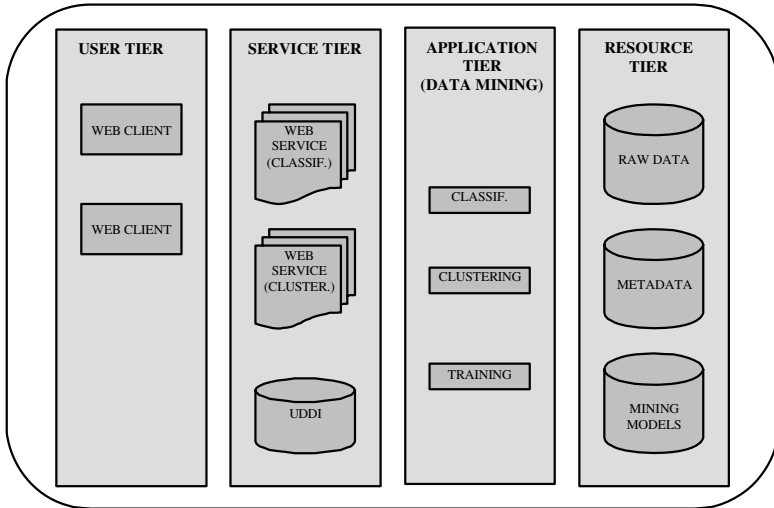


Fig. 4. The multi-tier architecture

The service tier exposes the data mining services as web services to its clients. Web services are implemented as Java web components along with their WSDL description, and are deployed in a web container managed by a J2EE Application Server. The application server and the web services it exposes can run anywhere on the web. The clients through the use of an UDDI registry can query web service endpoint locations. At this level, the implementation-independent web service invocation performed by clients, is mapped to the specific data mining implementations (i.e. the Java classes, specific to Oracle or to other data mining platforms, are activated).

The core of the data mining activities is implemented by the application tier, that consists in proprietary Java libraries and PL/SQL procedures (Oracle data mining

server). This level can be distributed across the Web (different mining activities can take place on different e-nodes), and can be physically separated from the previous level, as it is in our implementation.

The resource tier is composed by the data repositories (raw data, metadata, mining models, etc.) accessed by the single data mining activities. In our implementation all the data repositories are managed by a single DBMS [23] which also acts as a data mining server (at the data mining level), although in general this is not required since data repositories can be distributed. Actually, the prototype supports access to flat files and structured data including relational tables stored by a DBMS, that can be combined to build highly personalised data-sets. Remote raw data can be pre-processed for data mining tasks and added to the database so that they can serve for future experiments. The actions performed by the prototype include the access to user procedure as well as to a DBMS data-mining component [23].

Although we have implemented a prototypical version, aimed at carrying out data mining experiments, our approach is extendible to different cooperative experiments.

7 Conclusions and Future Work

We have illustrated how enterprise models and processes can be applied for modeling scientific cooperative services. The proposed cooperative framework for distributed experiments is quite general and flexible, being adaptable to different contexts.

Given the challenge of evaluating the effects of applying emerging web service technology to the scientific community, the evaluation performed up to now takes a flexible and multi-faceted approach: it aims at assessing task-user-system functionality and can be extended incrementally according to the continuous evolution of scientific cooperative environment.

In future works, we are going to extend the classes of experiments of interest, to implement the proposed structure in specific application fields where the need looks of primary relevance, and of course to detail the sequences of interaction among actors in the specific use cases.

References

1. Pollock J.T., Hodgson R., *Adaptive Information: Improving Business Through Semantic Interoperability, Grid Computing, and Enterprise Integration*, Wiley Series in Systems Engineering and Management, Wiley-Interscience, 2004.
2. De Roure D., Baker M.A., Jennings N. R., Shadbolt N., *The Evolution of the Grid*, in Berman et al. (eds), 2003.
3. Berman F., Fox G., Hey T. (eds), *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley and Sons, Inc., New York, 2003.
4. Foster I., Kesselman C., Nick J.M., Tuecke S., *The Physiology of the Grid: An Open Grid Service Architecture for Distributed System Integration*, The Globus Project, 2003.
5. Pilioura T., Tsalgatidou A., *e-Services: Current Technologies and Open Issues*, Proc. of VLDB-TES, 2001.
6. Brodie M.L., *The Cooperative Computing Initiative. A Contribution to the Middleware and Software Technologies*, GTE Laboratories Technical Publication, 1998.

7. Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS), 25 - 29 October 2004, Larnaca, Cyprus, Springer Verlag, 2004.
8. Mecella M., Parisi Presicce F., Pernici B., Modeling e-Services Orchestration through Petri Nets, Proc. VLDB-TES Workshop, 2002.
9. Baresi L., Bianchini D., De Antonellis V., Fugini M.G., Pernici B., Plebani P., Context-aware Composition of E-Services, Proc. VLDB-TES Workshop, Berlin, September 2003.
10. Schuster H., Georgakopoulos D., Cichocki A., Baker D., Modeling and Composing Service-based and Reference Process-based Multi-enterprise Processes, Proc. 12th International Conference on Advanced Information Systems Engineering (CAISE 2000), Stockholm, Sweden, 2000.
11. Hsieh Y., Shi M., Yang G., CovaModeler: A multi-user tool for modelling cooperative processes, International Journal of Computer Applications in Technology 2003 - Vol. 16, No.2/3 pp. 67-72.
12. W3C Working Draft 24 March 2004, <http://www.w3.org/TR/2004/WD-ws-chor-model-20040324/>.
13. Brown, M. (2003). "Blueprint for the Future of High-performance Networking", Communications of the ACM, vol. 46, no. 11.
14. De Fanti, T. et. al. (2003). "Translight: A Global-scale LambdaGrid for e-science", Communications of the ACM, vol. 46, no. 11.
15. Newman, H. et al. (2003). "Data-intensive for e-Science, Communications of the ACM, vol. 46, no. 11.
16. Ferrari Trecate G., Muselli M., Liberati D., Morari M., A clustering technique for the identification of piecewise affine systems, Automatica 39: 205-217, 2003.
17. Vapnik V., Statistical Learning Theory, New York: Wiley, 1998.
18. <http://www.wfmc.org>
19. <http://www.omg.org>
20. <http://shark.objectweb.org>
21. Armstrong E., Ball J., Bodoff S., Bode Carson D., et al., The J2EE 1.4 Tutorial, December 16, 2004.
22. <http://www.oracle.com>
23. http://www.oracle.com/database/Enterprise_Edition.html

Architecture-Based Interoperability Evaluation in Evolutions of Networked Enterprises

Pin Chen

DSTO, Department of Defence,
Canberra, ACT 2600, Australia
Pin.Chen@dsto.defence.gov.au

Abstract. Interoperability resulted from various evolutions occurring in a modern networked enterprise, or Systems-of-Systems (SoS) is a goal for an organisation to maintain a sustained, sustainable and controlled SoS evolution. A common issue facing networked enterprises in these evolutions is how to systematically manage and effectively achieve desired interoperability meeting new requirements of business and technologies. This paper presents an understanding of challenges in dealing with interoperability in evolutions of networked enterprises or SoS, and discusses an architecture-based approach to interoperability evaluation in various evolutions of networked enterprises.

1 Introduction

The concept of System-of-Systems (SoS) is a fundamental feature of modern networked enterprises and has been extensively examined [Maier, 1996; Arnold, 2001; Kaffenberger 2001; Chen, 2003], and is broadly acknowledged as a challenging issue due to its high complexity in interoperability and architectures. The interoperability within or between enterprises is largely determined by their systems interoperability. Traditional disciplines, such as Systems Engineering (SE) [SE, 2000; Blanchard, 1998; Grady, 1995], Software Engineering and Information Systems have been challenged on their approaches to and solutions in dealing with interoperability and evolution management in many organisations - especially defence organisations. System evolutions resulting from various developments in networked enterprises require the organisation to be capable in maintaining a sustained, sustainable and controlled SoS evolution with desired sound interoperability and responding quickly and cost-effectively to various requirements of systems and organisation evolutions.

Relevant communities of research and development have started to address many interoperability and evolution issues and difficulties caused by the emerging SoS concept [IDABC, 2004, EICTA, 2004]. The C4ISR Architecture Framework [C4ISR Architecture Working Group, 1997] was developed by the US DoD to specifically address challenges and issues of interoperability facing defence organizations. Maier [1996] discussed architecting principles for systems-of-systems. Although it is well known that architecture and interoperability are strongly related, the benefits through using architecture to address interoperability engineering and management problems are limited. Despite these efforts made by relevant disciplines, interoperability analy-

sis and evaluation remains as an undisciplined and expensive practice that largely relies on ability and experience of individual architects or systems integrators.

The merging of individual system life cycles into a SoS life cycle presents a range of challenges for information systems management, in particular in interoperability evaluation and management, which is a key to success of networked enterprises. Interoperability is no longer a purely technical concept, and now widely used by even business and management communities in discussions on organisational interoperability and business interoperability. Interoperability is not only a design or implementation issue, more importantly also an engineering and management issue for the whole organisation. This paper presents an architecture-based approach to interoperability evaluation in different aspects for SoS evolutions and networked enterprises.

2 Interoperability Reference and Maturity Models

The well-known Open Systems Interconnection (OSI) Reference Model defines levels of interoperability for communication systems at 7 layers where various communications protocols are used. The users application sits above the top layer and uses facilities provided. More interoperability reference models, such as European Interoperability Framework (EIF) [IDABC, 2004], have been developed recently to help conceptually address interoperability definition and management issues. Examining and planning interoperability between legacy systems or systems to be built, however, is not straightforward and sometimes even very difficult to determine because, firstly, there is no maturity construct that recognizes incremental levels of sophistication regarding system-to-system information exchanges and no comprehensive understanding of the interoperability attributes of information systems, secondly, it is difficult to identify the suite of methods and tools needed and associated implementation options available to enable each level of interoperability to be achieved; and, finally, there is no adequate support for assessing the interoperability “metric” of a given system or system pair, and for enabling the community to work collaboratively toward achieving more mature levels of interoperability.

In order to address these issues, an interoperability maturity model, Levels of Information Systems Interoperability (LISI) shown in Fig. 1, was developed by the US DoD [C4ISR AWG, 1998]. LISI provides a common basis for requirements definition and for incremental system improvement by defining five levels of interoperability maturity.

- *Level 4 Enterprise*, which provides cross-domain information and applications sharing, and advanced collaboration with features like interactive manipulation plus shared data and application;
- *Level 3 Domain*, which provides shared databases and sophisticated collaboration with features like only shared data plus “separate” applications;
- *Level 2 Functional*, which implements information exchange between heterogeneous products and basic collaboration based on minimal common functions and separate data and applications, such as web-based information publication and browsing;

- *Level 1 Connected*, which has electronic or network connections between homogeneous products performing information exchange with completely separate data and applications, such as Telnet, FTP, e-mail and tactical data links; and
- *Level 0 Isolated*, which carries out interoperability through manual gateways.

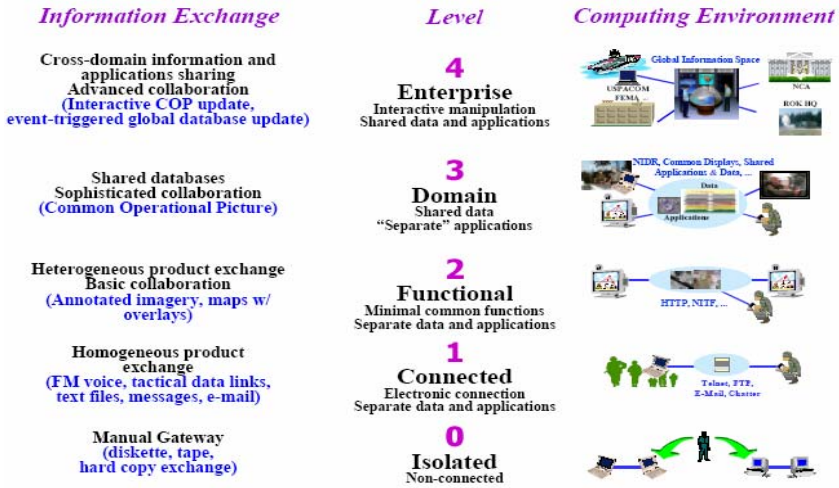


Fig. 1. LISI model

In addition to the definition of levels of interoperability maturity, LISI has also proposed a paradigm with four elements, Procedures, Applications, Infrastructure and Data (PAID), to define the interoperability attributes of information systems.

- *Procedure*, specifying what policies and procedures or operations enable systems to exchange information, capabilities, and services;
- *Applications*, specifying what set of applications enable information exchange, processing, or manipulation;
- *Data*, specifying what information formats, data protocols, or databases enable the exchange of data and information;
- *Infrastructure* (software platforms (SP) and hardware platforms (HP)), specifying what environment (hardware, communications and network, system services, and security) enables system interaction.

These interoperability attributes are usually presented in different architecture views/products. Comparing these four elements (PAID) of LISI with the seven-layer OSI architecture, we found that the infrastructure element in LISI can be further detailed by OSI architecture-based specifications; and however, other three elements are not represented properly or sufficiently within the OSI architecture. Despite the availability of LISI-like products, however, the use of them in engineering practice is not clearly defined by either the developers of the products or SE practitioners. The interoperability is thus a concept cross all levels from very technical one to high levels of

processes and policy within an organisation or between organisations. The agility of networked enterprises requires both static and dynamic features of interoperability, which suggests new and higher requirements for the ability of the organisation in evaluation and management of interoperability.

3 Interoperability in SoS Evolutions

In modern networked enterprises, systems are aggregations composed of entities (components). In a recursive manner, following integration, aggregates become entities that can be further integrated into higher-level systems (or SoS). Apart from the aggregation feature observed in the SoS composition, there are also other formation features of SoS, which are often resulted by changing interoperability requirements between relevant systems. For example, a system is not considered as a component of another system since these two systems work as partners in either a “supplier-consumer” relation or achieving a joint function of business. In other words, a system can be part of a number of SoS contexts because of its interoperability with multiple systems that belong to different SoS. In addition, a system can be a component system of a number of SoS contexts where the system plays different roles and has different interactions with other component systems. These SoS formation features greatly complicate life-cycle management and interoperability management of systems and future changes and development of systems.

After development of various systems in last three decades, large organisations are facing the reality of a SoS context, or a networked enterprise, for their future development rather than a choice or an option. A system evolution occurs, as illustrated in Fig. 2., when facing any of the following situations:

- *Self-Evolution*: a change is introduced into a system as a consequence of redesign, redevelopment, modification or improvement i.e. the evolution of a single system in a SoS context (an example in Fig. 2 is the redevelopment or improvement of the Military Operations Centre without changing interfaces to other components of SoS);
- *Joint Evolution*: two or more systems are to be integrated for improved interoperability and business support (two joint-evolution examples are shown in Fig. 2: 1) improvement of interoperability between the surveillance plan and the air defence missile, and 2) integration of the warship and the existing SoS); or
- *Emergent Evolution*: a new system is to be developed on the basis of or in relation to existing systems with new functionalities or capabilities (an example of the emergent evolution in Fig. 2 is to develop an integrated air picture based on the existing defence capability systems).

Complex system evolutions can involve a combination of the aforementioned situations, and the related systems in a SoS context or in different SoS contexts thus may evolve simultaneously. The ability to effectively and efficiently evaluate interoperability in a context of SoS must be developed on a basis of proper strategies and methods for different evolution scenarios.

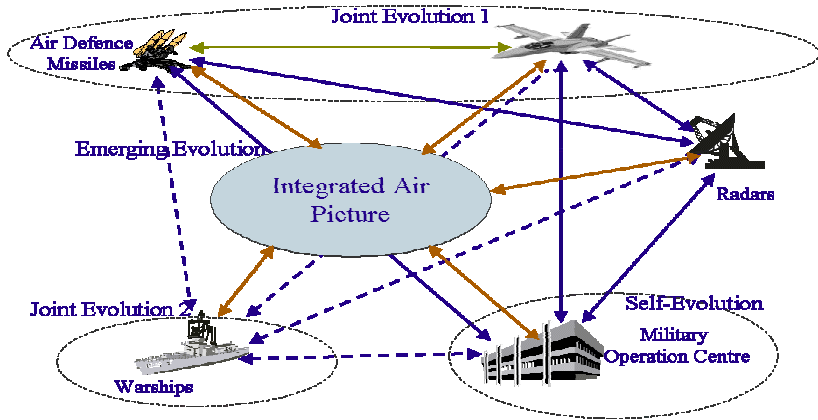


Fig. 2. System evolutions in a SoS context

4 Architectures for Interoperability Evaluation

Interoperability can be studied, according to LISI and those features discussed earlier, for different purposes from different perspectives. Various architectures or architecture views (not only graphic representations of system design or structure solutions) suggested by various architecture frameworks or approaches provide indeed an information-rich basis for interoperability evaluation. Such evaluation can be carried out in a relatively straightforward manner if evaluation requirements are not complicated and proper architecture and system information is available. The situation with SoS evolutions, however, is quite different and sometimes evaluations can be very difficult and expensive due to high complexity and concurrent activities of SoS evolutions. The increasing appreciations of architecture values and gradually improved architecture practices, on the other hand, have presented both requirements and opportunities for developing systematic interoperability evaluation methods.

4.1 Architecture Levels, Views and Environments

Modern architecture practice in information systems [Bass, 1998; CAWG, 1997; Chen, 2003; AWG, 1998; ISO, 1996; Meta, 1999; OIC, 1997; Shin, 2000; and Zachman, 1996] produces a range of architecture products or descriptions for various purposes from different perspectives or viewpoints. The architecture concept means different things to different people and no longer is only the representation of hardware and software design solutions. More architecture activities are considered by organisational and business management communities, such enterprise architectures and Business Process Modelling /Management (BPM). Architecture frameworks for architecture development and systems design often provide guidance in strategies, structures and formats for construction or generation of defined sets of related architecture products or descriptions. In order to ensure interoperability between systems, for example, the DoD Architecture Framework defines around 27 essential and support-

ing views (in 4 groups, namely, AVs, OV, SVs, and TV) for capability and systems architecture development in defence. Such a set of architecture views can jointly present knowledge and information about features of systems interoperability. By themselves, however, these views are still not sufficient for evaluating and analysing interoperability in the context of systems involved not only because of complex relations between these views but also the high complexity of the relations between the systems that can be either existing or future ones.

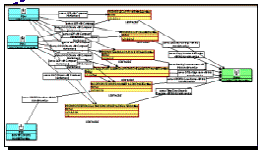
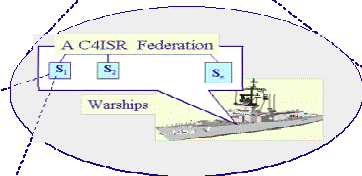
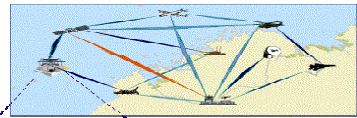
A defence operational scenario shown in Fig. 3 presents a context of a networked enterprise, where architectures (views) and systems relations are considered at different levels with different information related to interoperability. At the top level, the requirements of connectivity and information exchanges between nodes are first conceptually defined. Technical solutions of connectivity and infrastructures integration are generated at the middle (node) level where relevant system and technical views (SVs and TV) of architecture provides details on technical interoperability. Moving further down to the bottom level, we can see how component systems interact each other and how applications are integrated.

Operation Scenario (capability) Level:

- “N” nodes involved with different features;
- Shared by all nodes involved;
- Relations between scenarios.

Node (platform) Level:

- Each node contains one or more systems.
- Each system may have sub-systems.
- Interfaces:
 - Between nodes;
 - Between systems;
 - Between subsystems;
 - Between components.



System Level:

- Internal system /subsystems views.
- Relations to other systems (7 possible types)

Fig. 3. Leveled architecture views and relations

Overall SoS evolution is constituted by individual evolutions occurring in various parts of a SoS. The diversity of system evolution scenarios requires different strategies to effectively manage and present evolution environments for each evolution. The architecture evolution environment (AEE) [Chen, 2001] is a concept that must be addressed before an evolution (architecture) solution can be reached. An evolution environment needs to cover at least the following aspects:

- Business/operation requirements;
- Architecture evolution environment (AEE); and
- Technology options.

The issue concerning AEE is critical for SoS evolution as shown in Fig. 4. However, there is no apparent satisfactory solution from either an information system viewpoint or other relevant disciplines such as software engineering or architecture methodologies (including the US DoD Architecture Framework). The improved approaches should thus be developed as comprehensive solutions that include elements of well-discussed concepts, defined processes and their associated enablers addressing identified technical and management issues in an engineering fashion.

Maier discussed [Maier, 1996] the importance of interfaces between systems for SoS architecting. Due to the complexity of architecture and interoperability, the concept of AEE [Chen, 2001] is introduced to further define the architecture interface in a layered structure and aspects of interfaces at different layers. It must be realized that architecture interfaces for legacy systems are usually unavailable or not accessible in a satisfied and consistent manner. There may be a need for additional engineering efforts to construct architecture interfaces not only for legacy systems but also those under development and to be developed.

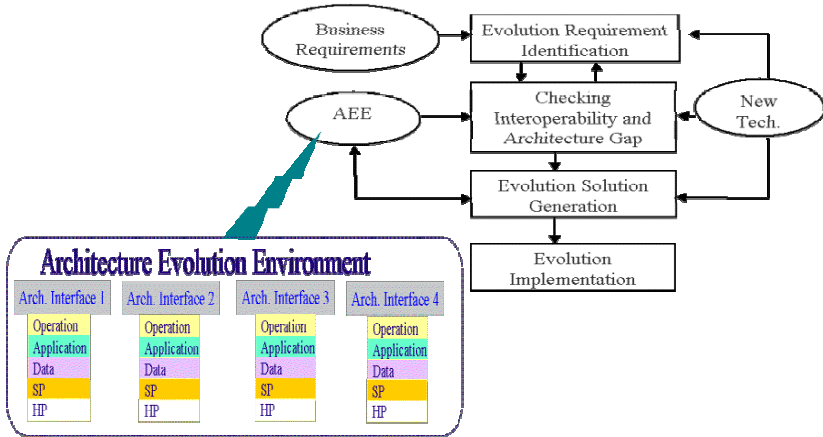


Fig. 4. AEE applications in SoS evolution activities

To respond quickly and cost-effectively to evolution requirements and achieve desired interoperability, establishing systematic solutions for AEE development and management from both technical and management viewpoints in an organization context is necessary. There is now an interesting but also frustrating situation surrounding the concepts of architecture and interoperability that they are obviously the main issues to be addressed by the four main processes shown in the Fig. 4, but, on the other hand, are actually complicated and difficult in searching systematic and integrated systems engineering solutions for their processes. The main cause of the situation is due to, first, fast changing technologies and system design concepts in information systems; and secondly, immaturity of the architecture practice that involving several professional communities related to information systems. The change

to or improvement of this situation will be mainly determined by the ability and solutions of architecture management in an organization [Chen, 2004].

4.2 Architecture Information Model (AIM)

Increasing interests and development in architecture shows a general trend that many organisations are gradually building up a body of architecture knowledge that covers all aspects related to interoperability. Despite the promises from enterprise architecture initiatives/frameworks [Zachman, 1996; Meta, 1999] and their associated tools and efforts made by organisations in developing enterprise architectures, complexity and challenges of SoS issues and interoperability evaluation and management for large organisations, like Defence, remain unsolved.

In order to address issues on architecture relations and systems relations and systematically and effectively manage and use architectures in a large organisation, the concepts, Architecture Information Model (AIM) and Architecture Repository are introduced [Chen, 2004]. AIM is a holistic and model-based description of classes and relationships concerning architectures, serving as one of key components of the architecture management solution and presenting a sound knowledge schema for developing an architecture repository environment. The AIM is developed using an object-oriented model and aimed at, through class definitions of attributes and relationship, modelling and managing not only various architectures descriptions and their meta data but also relevant concepts, such as systems and projects, and their meta data. More importantly, relations among those entities defined in the AIM are all defined to enable the relation management that is critical for the goals of effective interoperability analysis and evaluation. One of important outcomes of using AIM and its associated repository is the context management for architectures and systems. Being managed by AIM and the repository, each system or architecture description (product or view) is managed with its context or meta data [Chen, 2004]. Through the context information or meta data, one can easily identify and trace relations and dependency between systems and architectures. There is a SoS context class defined in AIM to manage various SoS context objects (special entities of SoS), such as an operational scenario. It is these SoS context objects (or SoS) in the repository that can help define various evolution scenarios and relations between systems.

Through using AIM and its associated repository, the architecture management can be achieved in a fashion with the following features:

- Each system or its like concept is seen as an object of a defined class in AIM and is recorded and stored in the repository where all relations from this object to other objects are also captured and managed;
- Each system has relations (or links) to its own architectures managed through an architecture set of its own (concerning mainly its own design, functions, structure and interfaces);
- Each system may be defined as part, a participating node or systems, of a number of SoS-context objects that are from its own perspective and have their related architecture sets (for example a warship operates in different regions with different capabilities or allies);

- Each system may be defined as part of a number of SoS- context objects that are from perspectives of other systems and have their related architecture sets (for example, an aircraft is considered as a participating node in operational scenarios of other capability platforms or allies); and
- A SoS-context object works as a bridging object that allows architecture traceability crossing systems and manages common (architecture) interests of participating systems.

5 Interoperability Evaluation and Analysis

The purpose of the architecture management is to enable systematic and effective interoperability analysis and evaluation. Using AIM and its associated repository, an organisation can systematically and effectively building up a body of architecture knowledge which interoperability analysis and evaluation can greatly benefit from.

One of common activities carried out in system planning and analysis is to analyse an information environment that involves a number of information-based systems, and can be seen as a SoS context. The analysis is usually to address two main issues:

- **Information technical environment.** The information technical environment analysis requires a detailed description of all technical architectures (technical components including platforms, networks, protocols, messaging, and system software) of involved systems in a form like a technical profile of SoS. Such a description is required for many planning and development activities related to the systems. Given the holding technical architecture information for all systems, a task for the information technical environment analysis can be carried out easily, efficiently and cost-effectively through performing a pre-developed architecture application (function or program) of the repository. The information exchange requirements for technical interoperability can be presented in different forms. A common approach to presenting it is to use a matrix with “traffic lights”.
- **Information exchange requirements.** An information exchange requirement study is to generate the requirement specifications of information exchange for a selected environment for specific purposes. The generated specifications define the level of interoperability between systems. Information (content) interoperability, based on the technical interoperability, is interactions between two systems and delivered by applications or functions of the systems. To describe or define the information interoperability, applications (or software) architecture and data architecture (models) information are required.

Comparing with the technical environment specifications, the information (content) interoperability specification is more complicated and more detailed. The repository can provide both system and architecture information of all systems and their applications and data architectures to effectively support the informa-

tion interoperability study through mapping interface descriptions through identification and construction of an AEE for a given evaluation or analysis context.

Example

Technical information environment analysis is a task to analyse an information environment that is of interest to stakeholders from a particular perspective. Such an environment is often viewed as a SoS (context). As observed in the current architecture practice, such an analysis usually starts with the identification of the SoS (context) concerned and often described in an architecture view, that is, OV1. Based on the OV1, systems analysts need to collect all architecture information and artefacts that are interested and useful from various architecture products associated with all systems or nodes involved. The algorithm presented here is to describe a process that can be carried out through using the repository to analyse a selected information environment and generate a report in a desired format illustrated in Fig. 5.

An algorithm presented below, as an example, can perform required analysis and generate an analysis report with an information interoperability matrix. The analysis activities or functions performed in the Step 9 can be carried out in different manners depending on the information captured in those SVs and TVs and their formats.

1. *Select an instance from SoS context class;*
2. *Get the associated OV1;*
3. *List all nodes or systems involved in OV1;*
4. *Create a system-to-system matrix;*
5. *For sys (or node) i from 1 to N do*
 6. *Get its SVs and TVs;*
 7. *For system (or node) j from 1 to N (j ≠ i) do*
 8. *Get SVs and TVs of sys j;*
 9. *Check SVs and TVs of sys i and sys j and time attributes;*
 10. *Decide a value (or colour) as an indicator;*
 11. *Set the indicator in a proper cell in the matrix;*
 12. *End of Step 7;*
13. *End of Step 5;*
14. *End.*

The algorithm demonstrates the method that can be manually or, if possible, automatically carried out through walking through the process, based the concept of AEE and the body of architecture knowledge built on AIM and the repository, of identifying relevant entities of systems and architecture descriptions/views based on their meta data captured, collecting architecture information and evaluating them for given requirements. To automate the process, in addition to using AIM and the repository, formats and notations of certain architecture descriptions/views must be standardised and machine-readable, such as template/table-based, XML or UML.

The interoperability evaluation should not be limited to technical aspects, in particular, when architecture descriptions/views are fully established in other aspects of the organisation, such as business process modeling, policy management, technical standards, rules and reference models/architectures. The interoperability evaluation can be performed not only within an organisation but also across organisations or enterprises in a similar manner as far as their systems and architecture information is available.

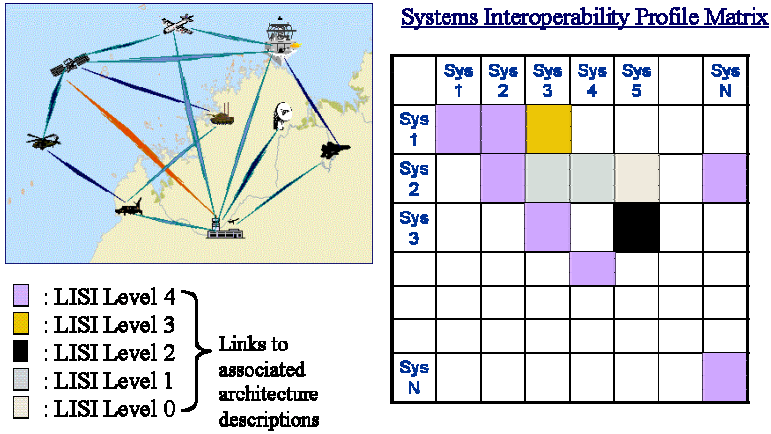


Fig. 5. An example of interoperability evaluation

6 Conclusions

The paper presents a systematic approach to analysing and evaluating interoperability in evolutions of networked enterprises or SoS, which effectively uses well-developed and well-managed architecture resources and benefits from architecture modelling and context management concepts. The discussions of the approach explore the potentials of automating the interoperability evaluation process with the support from AIM and a properly developed architecture repository if architecture description formats and notations can be standardised and adequate reasoning methods and algorithms can be developed. The study on the interoperability analysis and evaluation, on the other hand, can provide advice and requirements for the architecture community to consider how to improve development, management and applications of architectures. The approach also suggests a shift of the enterprise interoperability study from the traditional system development viewpoint to considering it as part of systems planning and management such that networked enterprises can be more confident in dealing with interoperability in their future evolutions.

References

1. S. Arnold and P. Brook, *Managing the Wood not the Trees — The Smart Acquisition Approach to Systems of Systems*, In Proceedings of the 11th annual Symposium of INCOSE, July 2001.
2. L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice*, Addison-Wesley Longman, 1998.
3. B. S. Blanchard, *System Engineering Management*, 2nd ed. John Wiley & Sons, Inc. 1998.
4. B. H. Boar, *Constructing Blueprints for Enterprise IT Architectures*, Wiley Computer Publishing, 1999.

5. P. Chen and J. Han, Facilitating Systems-of-Systems Evolution with Architecture Support, Proceedings of International Workshop of Principles of Software Evolution (IWPSE), Vienna, Austria, 2001. 130-133.
6. P. Chen and J. Clothier, Advancing Systems Engineering for Systems-of-Systems Challenges, Sys. Eng. Journal, Vol. 6, No. 3, 2003.
7. P. Chen, R. Gori and A. Pozgay, Systems and Capabilities Relation Management in Defence Systems-of-Systems Context, the Proceedings of the 10th International Command, Control Research and Technology Symposium (ICCRTS2004), California, USA, June, 2004.
8. The Council on IDABC, European Interoperability Framework for Pan-European eGovernment Services, published by European Commission in 2004, url: <http://europa.eu.in/idabc>.
9. C4ISR Architecture Working Group (AWG), *C4ISR Architecture Framework (Version 2.0)*, December 1997.
10. C4ISR Architecture Working Group, Levels of Information Systems Interoperability (LISI), US DoD, 1998.
11. EICTA, 2004, EICTA Interoperability White Paper, url: <http://www.eicta.org/files/WhitePaper-103753A.pdf>.
12. J. Grady, System Engineering Planning and Enterprise Identity, CRC Press, 1995.
13. IEEE Architecture Working Group (AWG), IEEE Recommended Practice for Architectural Description, IEEE Std 1471, Draft Version 3.0, 3 July 1998.
14. Information Standards Organization (ISO), Open Distributed Processing – Reference Model (ODP-RM), ISO/IEC DIS 10746-2, and ISO/IEC DIS 10746-3, 1996
15. R. Kaffenberger and J. Fischer, Designing Systems of Systems Without Getting Trapped in the Subsystem Maze, In Proceedings of the 11th annual Symposium of INCOSE, July 2001.
16. A. H. Levis and L. W. Wagenhals, Developing a Process for C4ISR Architecture Design, Sys. Eng 3(4), 2000, pp. 314-321.
17. M. W. Maier, Architecting Principles for Systems-of-Systems, Proceedings of the 6th Annual Symposium of INCOSE, pp. 567-574, 1996.
18. Meta Group, *Enterprise Architecture Strategies (EAS)*, Meta Delta, 31 March 1999
19. Object Ideas Corporation (OIC), *Enterprise Architecture Modelling*, October 1997, <http://www.object-ideas.com/eamd1/tsld001.htm>.
20. SE Handbook Working Group, Systems Engineering Handbook, Version 2.0, International Council on Systems Engineering (INCOSE), July 2000, Web site at <http://www.incose.org>.
21. S. S. Y. Shin, V. Pendyala, M. Sundaram and J. Z. Gao, Business-to-Business E-commerce Frameworks, IEEE *Computer*, pp40-47, October, 2000.
22. J. Zachman, *Enterprise Architecture: The Issue of the Century*, <http://www.zifa.com/zifajz01.htm>, 1996.

Enabling Interoperability of Networked Enterprises Through an Integrative Information System Architecture for CRM and SCM

Bernhard Selk¹, Sebastian Kloeckner¹, and Antonia Albani²

¹ University of Augsburg,
Chair of Business Informatics and Systems Engineering,
86159 Augsburg, Germany
{bernhard.selk, sebastian.kloeckner}@wiwi.uni-augsburg.de
² Delft University of Technology,
Chair of Software Engineering,
PO Box 5031, 2600 GA Delft, The Netherlands
a.albani@ewi.tudelft.nl

Abstract. With the deployment of specialized application systems e.g. customer relationship management (CRM) or supply chain management (SCM), the interconnection of enterprises becomes more and more complex. At the same time the need for exchanging information between enterprises, being part of different value networks, increases. This interconnection of enterprises is often hampered as data structures as well as functionality is difficult to integrate. The deployment of an integrated information system architecture (ISA) would facilitate inter-organizational integration and reduce data management problems at the same time. This paper illustrates on basis of two examples how an integrated information system architecture for CRM and SCM can strengthen inter-organizational integration and enable a continuous exchange of information between all enterprises, which are involved in the value network.

1 Introduction

Innovations in information and communication technologies, primarily the emergence of the Internet, combined with drastic changes in the competitive landscape (e.g. globalization of sales- and sourcing-markets, shortened product lifecycles, innovative pressure on processes), shifted managerial attention towards the use of information technologies to increase flexibility of the business system and to improve inter-company collaboration in value networks, often referred to as inter-organizational systems (IOS), e-collaboration and collaborative commerce [1, 2].

In order to support not only intra- but also inter-organizational business processes along the value network, the systems used in the single network nodes need to be integrated. This implies that the IT application systems of customers, partners and suppliers need to be integrated into an inter-organizational system in order to allow automated data interchange in the value network. The integration includes different functional areas like service, marketing, sales, procurement, production, distribution and waste disposal. Only the integration of all these functional areas, which are

directly involved in the value creation, allows a realization of a transparent and continuous supply network. Thanks to the deployment of enterprise resource planning (ERP) system internal integration is already on a high level and at the same time precondition for the realization of inter-organizational integration, which contains a potential by far larger than that of intra-organizational integration [3-5]. Inter-organizational integration in this context is defined as “(...) the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [6].

Nonetheless, the realization of inter-organizational integration in the business world is not as far as science has explored it. This is mainly caused by the lack of standards and absence of adequate information system architectures, which contain the construction plan of the information system – in terms of a specification and documentation of its components as well as their relationships – as well as the rules of construction for the creation of such a construction plan [7].

As an inter-organizational integration has an enormous influence on the design of information systems and as preliminary and downstream enterprises in the supply network have to be integrated, the creation of an inter-organizational information system architecture is necessary in order to ensure a continuous exchange of information between the involved partners throughout the whole value network. But before such an inter-organizational integration becomes possible, a corresponding intra-organizational integration has to be achieved.

Inter-organizational integration is supported by systems like Customer Relationship Management (CRM) and Supply Chain Management (SCM) [8]. These two concepts cover most of the functional areas involved directly in value creation and are therefore adequate as basis for the development of an integrative information system architecture [9]. Additionally, SCM and CRM systems are usually the endpoints of the internal value chain. CRM creates the connection with the customer within the functional areas of service, sales and marketing, while SCM creates the connection with the suppliers, partners and customers through the functional areas procurement, production, logistics and waste disposal.

This paper aims to illustrate the adequacy of an integrative Information Systems Architecture (ISA) for CRM and SCM. In doing so the following research question will be answered: Why is an intra-organizational integration necessary in order to achieve inter-organizational integration? In what way does a business components-based information system architecture contribute to the interoperability in a value network? What types of business components are relevant for the interoperability and how do the components need to be composed in an inter-organizational system?

Therefore, in section 2 the state of the art of inter-organizational systems is illustrated, showing the necessity of integrating CRM and SCM in an enterprise in order to provide an adequate basis for the development of inter-organizational systems. In section 3 an integrative information systems architecture for CRM and SCM is derived in order to ensure interoperability. Section 4 explains the integrative architecture by means of two examples. Concluding remarks and future work can be found in section 5.

2 Inter-organizational Information Systems

By interconnecting the CRM-systems of the suppliers with the SCM-systems of the customers – allowing business partners to streamline and optimize for example their production processes, inventory management as well as their customer service – the collaboration between enterprises can be improved and the exchange of information between customer and supplier accelerated. As far as only the relationship between the supplier and the customer is taken into account this kind of e-collaboration is adequate. But as actual real-world examples show the limitation of focus to only one relationship in the whole supply network as area of interest is not sufficient. While the considered relationship between supplier and customer performs well, problems at preliminary stages of the supply network might cause extensive interruptions of production processes downstream.

Demand driven value networks [10, 11] do not just only take the relationship from the OEM to the subsequent tier into account, but the whole supply network with several tiers. As shown in Fig. 1, each supplier in the supply network is viewed as a node which knows its preliminary supplier and where each node checks his own preliminary supplier for its ability to deliver in case that a downstream customer request for quotation arrives. The resulting information allows each supplier to judge if he and his preliminary suppliers are able to accomplish the potential customer order. But the assumption that every node is able to process an incoming request for quotation and convert it into outgoing requests for quotation for his own preliminary suppliers is not valid in most cases.

In fact, most often the internal systems of a company are coupled by standard interfaces or customized adapters. As the interfaces and adapters do not fit exactly with the coupled systems, the systems cannot communicate with each other correctly resulting

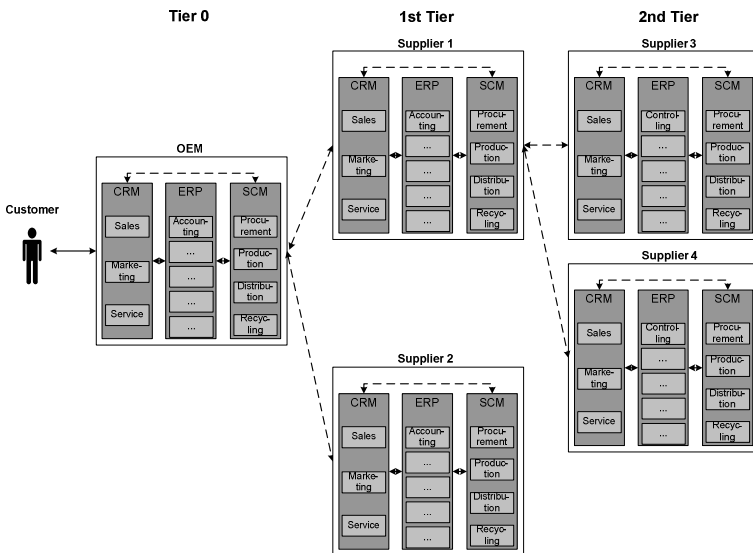


Fig. 1. Actual state of interconnection of internal and external systems

in a loss of information or functionality. The loss of information can be caused by several reasons: first, as two or more independent systems exist, data is stored redundantly and concurrent updates are not guaranteed. As a result, the reliability and actuality of data cannot be assured making it impossible to determine if an order or request can be fulfilled or not. Second, as structures and semantics of data can vary between the systems, a matching of the ontologies can be difficult or sometimes even impossible [12-15]. The loss of functionality can also be caused by several reasons: first, a system does not offer a public interface or API for a certain function. Consequently, this function cannot be called by an external system. Second, even if a public interface or API is offered, the signature of the interface may not be suitable for another system as additional information would be needed. Accordingly, even if the function could be called by an external system, it would not operate correctly as not all input parameters may be available [16, 17].

The problems mentioned become even worse as some systems – especially SCM and CRM systems – are interconnected only through a third system – usually the ERP system – multiplying the problems of lost information and functionality and, in the worst case scenario, not allowing any exchange of information or utilization of functionality of the connected systems at all [18-20]. While a direct connection between a CRM system and a SCM system would be favorable, in most cases such a direct interconnection does not exist. Therefore, the data exchange between those systems inside an enterprise is very limited, while it would be the precondition for successful interconnection of the whole supply network. In order to solve this problem, direct connections between these systems have to be established. Direct interconnection does allow to reciprocally access the data and to use the functionality of the independent systems.

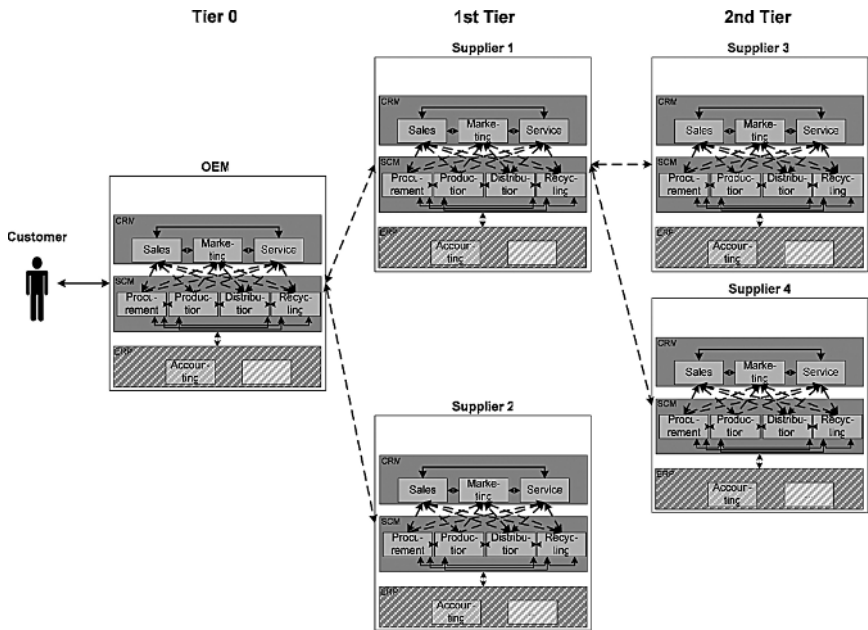


Fig. 2. Direct interconnection of CRM and SCM systems

As shown in Fig. 2, the precondition for such a direct interconnection is the availability of all important interfaces of an independent system to external systems.

The publication of interfaces of subsystems of a concerned system allows additional enhancement of the interconnection of the systems. Consequently, each subsystem could be interconnected making it possible to make use of further functionality. But even if all interfaces are published and available to external systems, the problem of redundant data, concurrent updates and different data structures still exists as both systems contain independent data management functionality. For solving this problem an integrated information systems architecture for CRM and SCM, as the endpoints of the internal value chain, has to be developed [21].

3 Integrative Information System Architecture for CRM and SCM for Ensuring Interoperability

One possibility to ensure inter-organizational integration is, as described in section 2, the usage of an integrated information system architecture for CRM and SCM. Fig. 3 shows the information relationships between seven functional areas in terms of an integrated ISA. Subsequent to Fig. 3 the methodology used for the development of the integrated ISA will be illustrated.

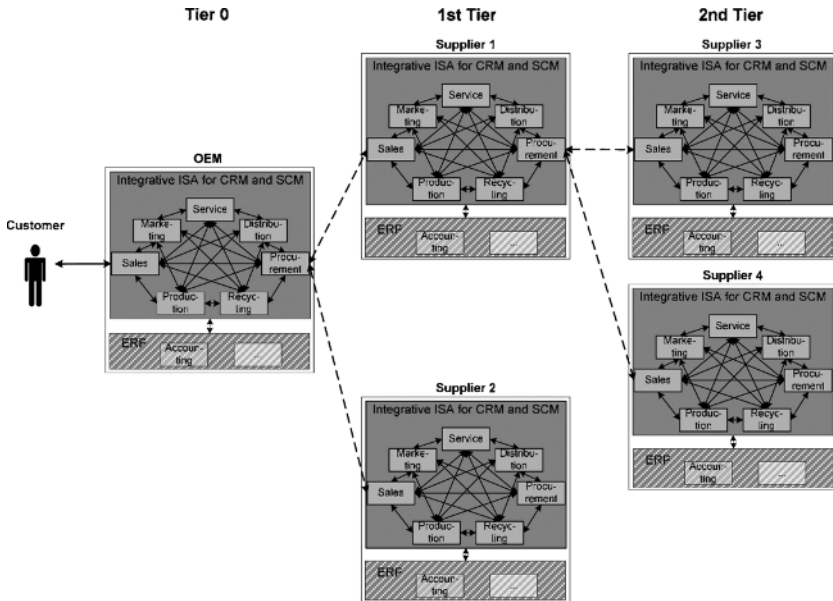


Fig. 3. Integrative information systems architecture for CRM and SCM

The integration of SCM and CRM systems, as shown in Fig. 3, allows direct interaction of the functional areas and usage of a coordinated data management, reducing the described problems of data consistency. Within the integration context some

functions of one business area are assigned to a business component [22, 23] containing additional functions of different business areas and providing the functionality to the outside world over well defined services [23] following the idea of a service oriented architecture. Under an integrative perspective business components are no longer strictly predetermined by membership in a certain business area, but rather composed in a way that relationships of information objects are optimized. The goal of the composition is to maximize the exchange of information within a business component and minimize the exchange of information between the business components while simultaneously avoiding that the technical purpose of the business component is affected negatively. The rearrangement of functionality does not only allow direct access to functions and information objects of other business areas. In addition, due to the incorporation in one integrative ISA, other problems like data redundancy or data matching issues are solved, because of a coordinated data management. Consequently, this integrative information system architecture allows the inter-organizational integration of organizations involved in the supply network and therefore a continuous exchange of information throughout the whole supply network.

Due to complexity reduction reasons the interconnection of the integrated ISA with other application systems, like an ERP-System as shown in Fig. 3, will not be explained here in depth. Nonetheless, an interconnection or integration of such a system can be achieved in the same way as it was shown for SCM and CRM. The ERP-Systems illustrated in Fig. 3 are rather for pointing out that additional interfaces for other systems are necessary.

For the development of the integrative information system architecture the Business Component Identification (BCI)-Method [24, 25] has been used, which is based upon the Business System Planning (BSP) [26] method and which has been modified for the field of business components identification. The basis for the BCI method is a well elaborated domain analysis. The BCI method takes as input the business tasks of a specific domain, as e.g. defined in the functional-decomposition diagram, and the domain based data model, both obtained from the domain analysis. In a first step a matrix is built defining the relationships between the single business tasks and the informational data. The relationships are visualized inserting “C” and “U” in the matrix. “C” denotes that the data is created by the specific business task, and “U” denotes the usage of informational data by a given task. In changing the order of data and of business tasks according to some metrics defined – e.g. minimal communication between and maximal compactness of components – groups of relationships can be recognized [25]. These groups identify potential business components. If some “U”’s are outside of the groups, arrows are used to identify the data flow from one group to the other. The result of the BCI is an abstract business component model with some already defined dependencies between components [24, 27, 28].

Fig. 4 illustrates a subset of the identified business components of the integrated ISA for SCM and CRM. The components shown are all member of the “customer request” process. Due to display reasons an illustration of all identified business components is not feasible, whereas the relevant components for the examples illustrated in the next section are described in detail in section 4. Additionally, one further component is included: the orchestration component. Its purpose is to coordinate the communicating between the other components. This allows simple adjustments of the communication channels if the services of one or more components are altered.

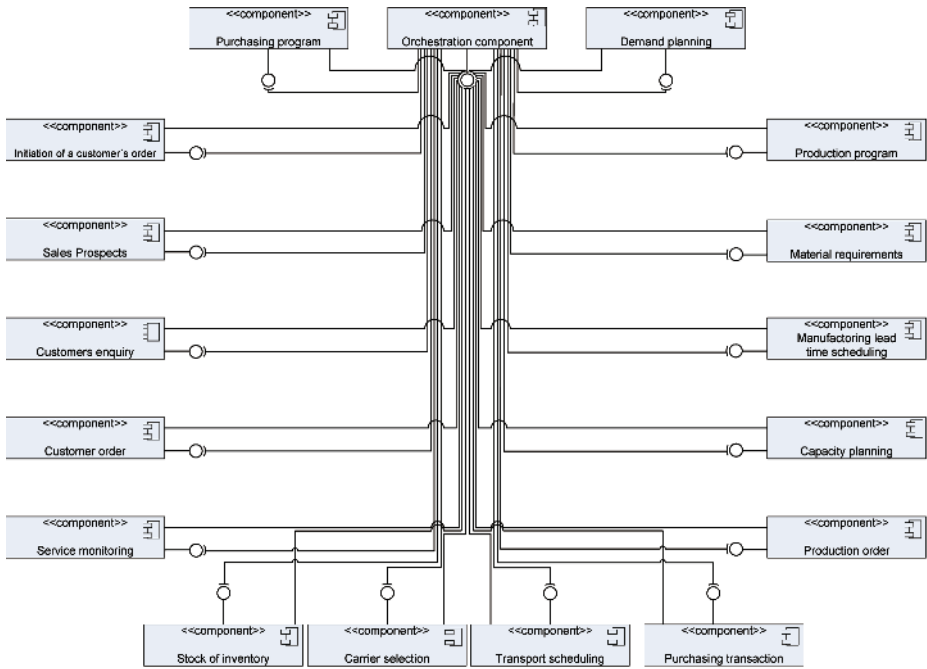


Fig. 4. Identified business components for the integrative ISA for CRM and SCM for the customer process

4 “Request for Quotation” Processes Within the Integrative ISA

In order to illustrate the advantages of the integrated information system architecture for CRM and SCM, which has been introduced in section 3, two examples of the “request for quotation” process are described in this section. The first example process shows a request for quotation, which can be satisfied without additional requests to preliminary suppliers. The second example extends the first one by assuming that the organization does not have all required parts on stock. Consequently, preliminary suppliers have to be involved in order to answer the request of the customer correctly. The preliminary suppliers themselves also have to contact their preliminary suppliers for evaluating the availability of the required parts. Due to complexity reasons only three of the seven examined functional areas of the integrated ISA are included. Both examples show the informational relationships of the business components involved. As already mentioned above, the orchestration component is also included. Additionally, a communication component, which is responsible for the communication with external systems, is incorporated for completeness reasons. Its functionality is not explained further in this paper as it does not have a direct influence on the ISA, but on the inter-organizational communication.

Already the simple request for quotation illustrates the need for an integration of SCM and CRM and demonstrates that company-internal integration is a precondition for an inter-organizational integration.

Prior to the illustration of the example processes, the business components of the example processes are explained. From all the business components of the component model presented in section 3 only the 6 components, which are relevant for the example processes, are described next.

Customer Order

The business component *customer order* contains all necessary functionality for further processing of a customer quote. With regard to the example process the order execution planning and the delivery date confirmation are of particular importance. Additionally, all basic agreements of the customer order are arranged and submitted to the customer.

Material Requirements

The business component *material requirements* prepares the on-time allocation of all required materials in regard to type, quality and amount for the production process.

Capacity Planning

The functions of the business component *capacity planning* include design of capacity capability, determination of the capacity demand, deployment of staff, comparison of capacity demand and availability, scheduling of capacity and planning of machine allocation sequence.

Stock of Inventory

The business component *stock of inventory* contains the functionality which is associated with stock movements. Within others, this affects the inventory management, which is the link between demand and order planning.

Purchasing Transaction

The business component *purchasing transaction* includes all functions which are needed for the transaction-oriented part of the procurement initiation. This involves acceptance and transmission of requirement requests, requests for offers, which precede the registration and evaluation of requirements.

Manufacturing lead time scheduling

The business component *manufacturing lead time scheduling* contains all relevant functionality for successful scheduling like determination of process steps and operations or definitions of process and administrative times. By creation of task schedules, determination of lead times and evaluation of possible lead time reductions preliminary starting and ending times for the different process can be generated.

Scenario for the “Request for Quotation” examples

Both examples are based on the following assumptions: A customer asks the OEM for a customized product. Besides the price and the configuration of the product, the sales employee wants to tell the customer the delivery date as it is one of the crucial factors for his purchase decision. The determination of the delivery date requires access to all functional areas involved. For example the capacity of production, the delivery dates of required materials in procurement as well as the shipment slots in the distribution department. The resulting delivery date should already be available during the customer meeting since alternatives might be necessary. This information would allow discussing possible alternative product configurations fitting with the requirements of the customer in regard to the delivery date.

While example one assumes that all required material is available from stock, example two requires communication with the preliminary suppliers.

Example 1: “Request for Quotation” process within a company

After the definition of the basic agreements and the planning of the order processing, a date of delivery for confirming the customer order is required. The data of the expected customer order is passed to the lead time determination (*manufacturing lead time scheduling* component) in order to execute the required scheduling. For this determination, information of the business component *material requirement*, like net primary demand, is needed (see Fig. 5). The check if the required materials are available from stock is only feasible if the *stock of inventory* component delivers the required information. The business component *purchasing transaction* is not needed in this scenario, as it is assumed that all required materials are available from stock.

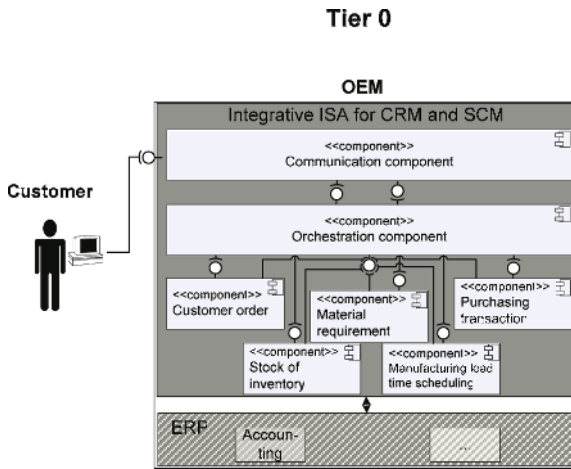


Fig. 5. Information relationships within the OEM for the request for quotation process

Example 2: “Request for Quotation” process within the supply network

In contrast to the first example, the OEM now does not have all required parts on stock. Consequently, he has to obtain the parts needed from his suppliers. Supplier 2 can fulfill the order of the OEM from stock, while supplier 1 has to obtain the parts for his (sub-) product from supplier 3 and 4 in order to satisfy the order of the OEM. Supplier 4 can serve the request of supplier 1 out of his inventory, while supplier 3 again has to contact his preliminary suppliers. Fig. 6 shows the relationships of information within as well as between the concerned companies due to the fact that a customer’s request not only affects the OEM, but also the preliminary suppliers in the supply network. These preliminary suppliers of supplier 3, the business components which are not part of the process shown as well as the unimportant relationships of the included business components are not illustrated in Fig. 6 due complexity and display reasons.

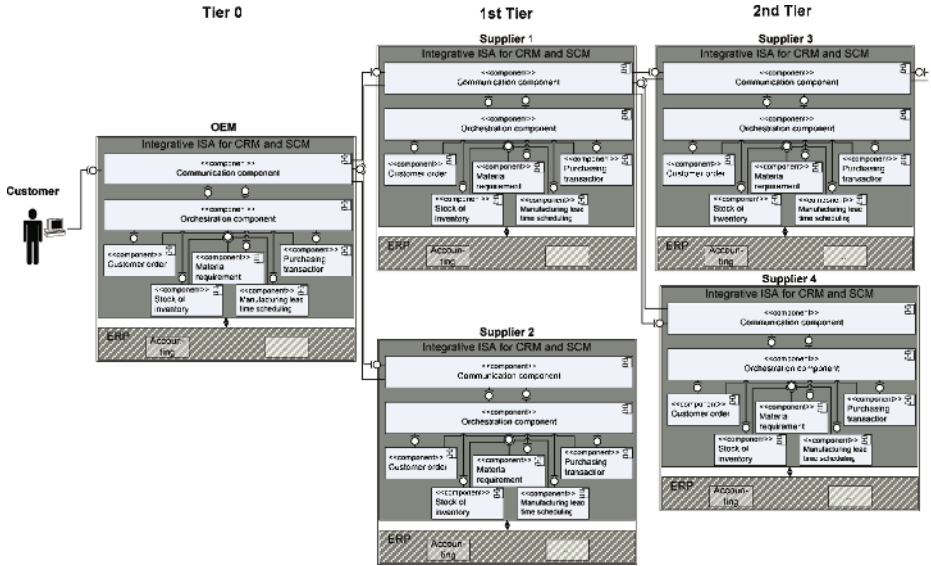


Fig. 6. Information relationships within the supply network for the request for quotation process

As it is assumed that required parts are not available from stock, a procurement transaction has to be invoked. After arrival of the estimated delivery date of the required parts, this information is passed back to the business component *stock of inventory*. From there the information is given to the *material requirement* component and further to the *manufacturing lead time scheduling* component. Thereby all information is available for determining the delivery date for the customer. This delivery date is then passed back to the *customer order* component, which transfers the information to the customer.

The request of the OEM to his suppliers invokes the same process also at supplier 1 and then at supplier 3. As Supplier 2 and 4 do have the required material available from stock no further procurement transaction has to be executed.

These two suppliers can determine the date of delivery without using preliminary suppliers. The information about the date of delivery is then, as described above, passed back to the business component *customer order* in order to inform the customer, the OEM in the case of supplier 2 and supplier 1 in case of supplier 4.

5 Conclusion

This paper showed that enterprises have to implement inter-organizational integration due to increasing competition and globalization. But an inter-organizational integration is not feasible without prior intra-organizational integration. The presented integrated information system architecture (ISA) for CRM and SCM allows the coordination of data management and functionality of all functional areas (sales, service, marketing, procurement, production, logistics and waste disposal), which are directly

involved in the value creation process. Consequently, the problems which arise if application systems are only interconnected – e.g. redundant data or different data structures – do not occur anymore. This integration is then used as basis for a transparent inter-organizational integration of all members of the whole supply network allowing a continuous exchange of information between the members. Additional investigations are needed in improving and refining the presented architecture for CRM and SCM and in integrating it with existing ERP systems.

References

1. Kopanaki, E., et al. *The Impact of Inter-organizational Information Systems on the Flexibility of Organizations*. In *Proceedings of the Sixth Americas Conference on Information Systems (AMCIS)*. 2000. Long Beach, CA.
2. Picot, A., R. Reichwald, and R. Wiegand, *Die grenzenlose Unternehmung - Information, Organisation und Management*. 4. Auflage ed. 2001, Wiesbaden. 2-6.
3. Warnecke, H.-J., *Vom Fraktal zum Produktionsnetzwerk. Unternehmenskooperationen erfolgreich gestalten*, ed. J.H. Braun. 1999, Berlin.
4. Sawhney, M. and J. Zabin, *The Seven Steps to Nirvana: Strategic Insights into eBusiness Transformation*. 2001, New York.
5. Malone, T.W. and R.J. Lautbacher, *The Dawn of the E-Lance Economy*. Harvard Business Review, 1998(September-October): p. 145 - 152.
6. Engineers, I.o.E.a.E., *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. 1990.
7. Sinz, E.J., *Architektur von Informationssystemen*, in *Informatikhandbuch*, P. Rechenberger and G. Pomberger, Editors. 1999, Hanser Verlag: München/Wien.
8. Fleisch, E., *Das Netzwerkunternehmen: Strategien und Prozesse zur Steigerung der Wettbewerbsfähigkeit in der "Networked economy"*. 2001, Berlin et al.: Springer.
9. Selk, B., K. Turowski, and C. Winnewisser. *Information System Design for Demand-Driven Supply Networks - Integrating CRM & SCM*. In *Fourth International ICSC Symposium on Engineering of Intelligent Systems EIS 2004, University of Madeira, Funchal, Portugal, February 29th - March 2, 2004*. 2004. University of Madeira, Funchal: ICSC Academic Press.
10. Albani, A., et al. *Dynamic Modelling of Strategic Supply Chains*. In *E-Commerce and Web Technologies: 4th International Conference, EC-Web 2003 Prague, Czech Republic, September 2003, LNCS 2738*. 2003. Prague, Czech Republic: Springer-Verlag.
11. Albani, A., C. Winnewisser, and K. Turowski. *Dynamic Modelling of Demand Driven Value Networks*. In *On The Move to Meaningful Internet Systems and Ubiquitous Computing 2004: CoopIS, DOA and ODBASE, LNCS*. 2004. Larnaca, Cyprus: Springer Verlag.
12. Fingar, P., H. Kumar, and T. Sharma, *Enterprise E-Commerce - The Software Component Breakthrough for Business-to-Business Commerce*. 2000, Tampa: Meghan-Kiffer Press.
13. Kalakota, R. and M. Robinson, *e-Business 2.0 Roadmap for Success*. 2. ed. 2001, Boston: Addison Wesley Verlag.
14. Hagel, J., *Out of the Box: Strategies for Achieving Profits Today and Growth Tomorrow through Web Services*. 2002, Boston: Harvard Business School Press.
15. Puschmann, T., *Collaboration Portale. Architektur, Integration, Umsetzung und Beispiele*. 2003, Universität St. Gallen: St. Gallen.

16. Bazijanec, B., et al. *Component-based Architecture for Protocol Vector Conversion in Inter-organizational Systems*. In *International Workshop on Modeling Inter-Organizational Systems (MIOS'04)*. 2004. Larnaca, Cyprus: Springer, LNCS 3292.
17. Bazijanec, B., et al. *Establishing Interoperability of Coordination Protocols in ad-hoc Inter-Organizational Collaborations*. In *Interop-Esa 2005 - First International Conference on Interoperability of Enterprise Software and Applications*. 2005. Geneva, Switzerland: Springer, science + business media.
18. Vandermerve, S., *How Increasing Value to Customer Improves Business Results*. MIT Sloan Management Review, 2000. 42: p. 27-37.
19. Harmon, P., M. Rosen, and M. Guttman, *Developing E-Business Systems and Architectures: A Manager's Guide*. 2001, San Francisco: Morgan Kaufmann.
20. Österle, H., *Geschäftsmodell des Informationszeitalters*, in *Business Networking in der Praxis: Beispiele und Strategien zur Vernetzung mit Kunden und Lieferanten*, H. Österle, E. Fleisch, and R. Alt, Editors. 2002, Springer: Berlin et al. p. 17-38.
21. Mertens, P., *Integrierte Informationsverarbeitung 1 - Operative Systeme in der Industrie*. 14. ed. 2004, Wiesbaden: Gabler Verlag.
22. Turowski, K., *Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme*. 2003, Aachen: Shaker Verlag.
23. Turowski, K. and J.M. Zaha, *Methodological Standard for Service Specification*. International Journal of Services and Standards, 2004.
24. Albani, A., et al. *Domain Based Identification and Modelling of Business Component Applications*. In *7th East-European Conference on Advances in Databases and Informations Systems (ADBIS-03)*, LNCS 2798. 2003. Dresden, Deutschland: Springer Verlag.
25. Albani, A., J.L.G. Dietz, and J.M. Zaha. *Identifying Business Components on the basis of an Enterprise Ontology*. In *Interop-Esa 2005 - First International Conference on Interoperability of Enterprise Software and Applications*. 2005. Geneva, Switzerland.
26. IBM, *Business Systems Planning-Information Systems Planning Guide*, ed. I.D. (Ed.). 1984, Atlanta: International Business Machines.
27. Albani, A., et al. *Component Framework for Strategic Supply Network Development*. In *8th East-European Conference on Advances in Databases and Information Systems (ADBIS-04)*, LNCS 3255. 2004. Budapest, Hungary: Springer Verlag.
28. Albani, A., et al. *Komponentenmodell für die Strategische Lieferkettenentwicklung*. In *6. Internationale Tagung Wirtschaftsinformatik (WI-03) Medien - Märkte - Mobilität*. 2003. Dresden, Deutschland: Physica-Verlag.

Interoperability in Service-Based Communities

Toni Ruokolainen and Lea Kutvonen

Department of Computer Science,
P.O. Box 68 (Gustaf Hällströmin katu 2b),
FI-00014 University of Helsinki, Finland

Toni.Ruokolainen@cs.Helsinki.FI, Lea.Kutvonen@cs.Helsinki.FI

Abstract. Interoperability is a multifaceted problem caused by issues surpassing those of technological incompatibilities. The real interoperability challenges are stemming from various sources, such as organisational incompatibilities buried deeply into the structures of collaborating enterprises, architectural mismatches and defective assumptions about business application behaviour, or from the inherent properties of business collaboration models.

To achieve interoperability in enterprise computing environments, the aspects of interoperability must be identified and their properties analysed. This paper studies interoperability issues in enterprise computing environments. Enterprise computing environments under analysis are based on Service Oriented Computing paradigm and enhanced with necessary infrastructure facilities. Several classes of causes for interoperability problems are identified and the mechanisms for overcoming the problems in these classes are briefly discussed.

1 Introduction

Interoperability of business applications in enterprise computing environments is a challenging problem. Work addressed by such projects as INTEROP [15] and Athena [3] try to grasp the different angles of interoperability issues in enterprise applications and systems. Issues related to purely technological interoperability can nowadays be usually handled with use of a common distributed object computing platforms such as CORBA, or by exploiting service oriented architectures and Web Services.

Interoperability is nevertheless a multifaceted problem caused by issues surpassing those of technological incompatibilities. The real interoperability challenges are stemming from various sources, such as organisational incompatibilities buried deeply into the structures of collaborating enterprises, architectural mismatches and defective assumptions about business application behaviour, or from the inherent properties of business collaboration models.

The notion of interoperability has been left quite vague both in the academia and industry. However, for achieving interoperability in enterprise computing environments, the aspects of interoperability must be identified and their properties analysed. This paper studies interoperability issues in enterprise computing environments. Several classes of causes for interoperability problems are identified

and the mechanisms for overcoming interoperability problems in these classes are briefly discussed.

Section 2 discusses interoperability in general and describes different models of collaboration and typical means of achieving interoperability in these models. Section 3 introduces a taxonomy for interoperability in context of federated service communities. After introducing the taxonomy, methods and mechanism for establishing interoperation with respect to the identified aspects are briefly discussed in Sections 4, 5 and 6.

2 Interoperability Methods

Interoperability has been defined as the ability of two or more entities to communicate and operate together in a meaningful way, such that information gets exchanged between collaborating parties and it is used in a meaningful way despite differences in language, interface or operation environments [14, 18, 45]. Interoperability, or capability to collaborate, means effective capability of mutual communication of information. Interoperability covers technical, semantic and pragmatic interoperability. Technical interoperability means that messages can be transported from one application to another. Semantic interoperability means that the message content becomes understood in the same way by the senders and the receivers. This may mean both information representation or messaging sequences. Finally, the pragmatic interoperability captures the willingness of partners for the actions necessary for the collaboration. The willingness to participate involves both capability of performing a requested action, and policies dictating whether the potential action is preferable for the enterprise to be involved in.

Three different forms of model interoperability have been identified in [16]. The identified forms of interoperation, namely integrated, unified and federated interoperability, can also be identified from enterprise computing environments as different models for collaboration. These models of collaboration are distinguished from each other on the grounds of where information needed for achieving interoperability is found.

In the integrated model of collaboration the knowledge for ensuring interoperability is implicitly injected into different software components of the computation system. The integrated collaboration model can be furthermore classified into three different solution methods based on the deployment point of the interoperation knowledge. The solution methods are 1) tightly coupled integration, 2) software adaption, and 3) use of common computing environments (middleware).

In tightly coupled integration the knowledge needed for interoperation is weaved inside the business applications implicitly by software designers and implementors. This has been a very popular approach to achieve interoperability of intra- and inter-enterprise business applications in the past. The implicit assumptions about other components and operating environment however make this tightly coupled integration and interoperation model an impermanent

solution in general. Tight coupling is however very cost-effective and easy to implement if there is an absolute certainty that there will never be any changes in any part of the system.

In software adaption approach interoperation knowledge and functionality related to interoperation is partly isolated from the application components. Interoperation knowledge is injected into intermediary software components such as software adapters and wrappers. Adapters are used to mediate incompatibility of software entities [45] and they can perform mappings between data values and schema structures or even adapt the behaviour of services [5]. Wrappers are used for introducing completely new behaviour that is executed usually before or after the actual functionality [6].

Third form of integrated collaboration is based on use of a common computing environment. In this method the interoperability knowledge is located in computation and communication facilities of the system. A common computing environment provides a homogeneous technology and communication platform as well as computation model to be used in an enterprise computing environment. Interoperability is mediated between business applications and the computing platform via an intermediary language, an interface description language (IDL) [22]. Middleware platforms such as CORBA [28] or J2EE [41] and interface description languages have been used successfully to bridge technological differences between operation environments.

Second model of collaboration in enterprise computing environments is the unified model. In unified collaboration model a shared meta-information entity describes the functionality and responsibilities of each community participant. Two kinds of meta-information entities can be identified: standards and explicitly shared meta-information.

Interoperation has been achieved in traditional forms of industry through standardisation. However, in software engineering, standardisation of software entities such as components or even communication technology has not been as successful. First of all, software components and computing systems themselves are usually highly dynamic entities whereas standardisation processes are slow with respect to the advances in ICT technology.

Secondly, not even standardisation guarantees interoperation if either standards are too ambiguous or developers do not comply to the standards. Interoperability problems between implementations of the same CORBA service from different vendors were studied in [4]. The result was that the formal specification for OMG's CORBA Event Service middleware function was too ambiguous and underspecified for guaranteeing interoperable and substitutable implementations of the same standard. Interoperability problems stemming from different interpretations of the same standard or too loose standardisation are found also from Web Services technology [36].

Unified collaboration can be achieved by use of explicitly shared meta-information. Meta-information defined using an appropriate modeling language, such as UML, describe component functionality, their properties and interrelationships using computation platform independent notations. Typically the

meta-information, or a model is used for generating the actual business application components. As the components implemented by generative methods are based on the same platform independent conceptual model, interoperation between components generated by different vendors should be possible, given appropriate code generation tools. The most renown representative of this approach is OMG's Model-Driven Architecture initiative [11].

Federated collaboration means that no shared, native meta-information describing the operation of a collaboration is presupposed or needed. Each participant may have their own models describing their business services. To achieve interoperability a *shared meta-model* is exploited. Interoperation is established by negotiation mechanisms, model verification and monitoring of service behaviour with respect to the interoperability contract. Meta-information needed for ensuring interoperability must be explicitly available, especially during operation of service communities. Federated collaboration model needs additional infrastructure facilities for publication and management of meta-information, and for controlling and monitoring the communities.

The model of integrated application collaboration provides solutions for establishing technical interoperability. Heterogeneity in technical level is supported but usually there are a very strict bindings between the collaborating business applications and underlying computation and communication platform. Therefore heterogeneity in higher abstraction levels, such as service behaviour or information representation levels, is not usually tolerated. Integrated collaboration model does neither tolerate dynamism or autonomy of participants. As the information about interoperation prerequisites is hidden inside business application and infrastructure components, dynamic changes in the system can not be coped with.

Unified collaboration model provides support for both technical and semantic interoperability. Heterogeneity of computation and communication platforms is also supported, assuming that the meta-information is platform independent. Unified collaboration based on shared models is however inflexible due to the fact that although the design entities in the models are reusable, the actual service components are typically specialised for the specific architecture and use-case described in the model. Model evolution is effectively supported but real dynamism, that is awareness and adaption to runtime changes is not supported by the unified collaboration model, unless explicitly modeled in the meta-information.

Interoperability is an issue which can not be fully realised by homogenising the execution environment through distribution middleware, or by using unified meta-languages or some other means of mediation. Object and component interoperability even in a homogeneous computing platform is a multi-faceted issue with syntactic, semantic and behavioural aspects [44]. When considering enterprise computing systems with heterogeneous implementation platforms and autonomous participants, interoperability of software components becomes even more complicated, since these kind of computing environments are characterised by their heterogeneity (freedom of design), autonomy (freedom of action)

and dynamism (freedom of configuration) [38, 39]. Both integrated and unified collaboration models support at least partially technological heterogeneity in enterprise computing environments. They however fail to address the autonomy and dynamism aspects, and do not provide pragmatic interoperability.

To establish pragmatic interoperability needed in enterprise computing environments, the federated collaboration model should be used. Federated collaboration provides support for heterogeneity in technology, computation, communication and information levels via loose coupling of business applications and contract based co-operation. Negotiation mechanisms and monitoring facilities provide support for runtime dynamism and autonomy over service activity. Model evolution is also supported, as interoperability between business applications is achieved via shared meta-model and interoperability validation facilities. Service Oriented Computing approach [30] is especially suitable framework for federated collaboration model as it promotes use of self-descriptive, independent and composable software entities and loosely coupled collaborations based on the notion of contracts.

3 A Taxonomy of Service Interoperability

To establish and support an open model of interoperability, the different aspects of interoperation must be identified and analysed. The notion of interoperability must be separated into independent aspects, each aspect grasping a different need or viewpoint of enterprise computing. A clean separation of aspects is important because otherwise it would be very difficult to identify the requirements interoperability imposes on modeling concepts and infrastructure facilities.

In this section a taxonomy of interoperability aspects for service based communities using a federated collaboration model is presented. Interoperability is classified into different abstraction layers, each layer grasping more abstract interoperation concepts than the previous one. Classification is based on previous studies of interoperability (see for example [9, 44]) and on conceptualisation of enterprise computing environments made in web-Pilarcos project [21].

Interoperability in federated communities is divided into five abstraction levels: 1) technology, 2) service, 3) community, 4) organisation, and 5) business level. This division is based on identification of the subjects responsible for deciding if interoperation can be achieved. Each level is further divided into different aspects; the classification as a whole is illustrated in Figure 1.

At the technology level, technical interoperability must be achieved between communication and computation platforms. Interoperation is established by selecting and configuring appropriate middleware services and their parameters. When we consider only technical interoperability, that is the connectivity, communication and encoding related aspects, incompatibilities between languages, interfaces or operational environments can be solved quite efficiently. Methods and techniques like interface description languages [22, 27], adaptors [34, 46], wrappers [26], middleware [28, 41] and middleware bridges [10] have quite success-

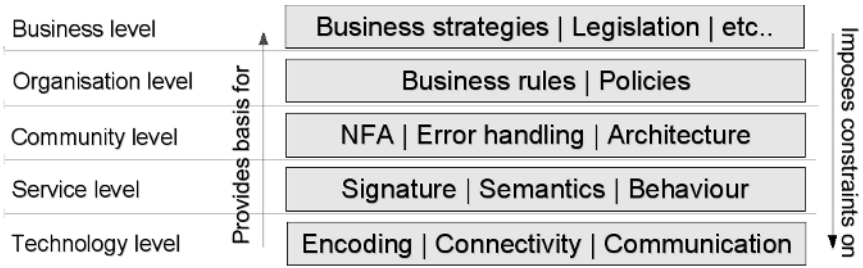


Fig. 1. Aspects of interoperability

fully been applied for enterprise integration. However, while providing the necessary means for collaboration, technology level interoperability and the methods for achieving it are only the basis of the “interoperability stack”.

At the service level, both technical (compatibility between service signatures) and semantic interoperability (semantics and behaviour of services) between service end-points must be established. Service discovery mechanisms are used for this purpose and the decision making procedures are bilateral. Service level interoperability means capability of interoperation between electronic services with well-defined, self-descriptive interfaces.

Interoperation between distinct services does not guarantee that functionality of the whole system is consistent and flawless. Requirements and constraints for application interoperation are induced by the global properties of the community in question. For establishing community level interoperability the aspects of architectural properties, failure handling procedures and compensation processes, and non-functional aspects of communities must be addressed. Decision making at the community level is multi-lateral since the properties of all the participants must be taken into consideration. Negotiation mechanisms are used for populating communities with compatible services. Both technical (non-functional aspects) and semantic interoperability (failure handling and community architecture) is addressed at the community level.

Pragmatic interoperability is addressed at the organisation and business levels. Business rules and policies must be agreed upon at the organisation level. Organisation level interoperability deals with issues related to the needs of autonomous enterprises. Policies and business rules are business knowledge which must be explicitly represented if inter-organisational collaboration should be achieved. Policies are used to constraint community behaviour such that the common objective of the community can be achieved [40]. Business rules are declarative rules that constraint or define some aspects of business [12]. Both policies and business rules are organisational entities that are independent of community or service life-cycles; thus it is necessary to separate these aspects from the aspects related to community and service level interoperability.

Organisation level is the last level of abstraction that is embodied as explicitly available meta-information. Corporation business strategies and legislation con-

cerning for example geographic regions or business domains are typically available as implicit regulations and constraints at the lower levels. In the following sections we will discuss three most important levels of the presented taxonomy in more detail, namely the service level, community level and organisation level interoperability.

4 Service Level Interoperability

Interoperability at the service level is characterised by three aspects, namely syntactic, semantic and behavioural properties of service interfaces [43, 44]. An interface is an abstraction of application functionality which decouples the internal implementation details from the externally provided service. A service interface description provides definitions of the service syntax (interface signature and document structures), semantics and its behaviour.

Service level interoperability has been studied mainly among object oriented and component based approaches [8, 18, 43, 44]. Object oriented interoperability was first addressed in [18]. This work recognised that interoperability conflicts in object oriented platforms can not be solved by simple adaptation or procedure parameters between heterogeneous objects with use of unifying type systems. It is the overall functionality and semantics of an object which is important [18].

Substitutability and compatibility of software entities can be considered as the most relevant concepts in this level. When considering syntactic and semantic aspects we are interested if two entities can be substituted by each other. The concept of compatibility is relevant only when behavioural aspects are taken into consideration.

Validation of syntactic interoperability, that is substitutability of syntactic structures, reduces to type matching. Type matching problem is about finding and defining bindings and transformations between the interface a client wants to use and the interface provided by a service [18]. Type matching problem in general is impossible, since identification of operation semantics and information contents used in the operations or attributes can not be fully automated. However, if two interface signatures are described using the same language (type system) or the interface descriptions can be unified, and only syntactic properties of services are considered, efficient type matching methods and algorithms can be used [17, 29].

When considering type matching in Web Services based environments, the notion of schema matching emerges (see for example [32, 33]). Schemas define document structures used for information descriptions. When considering Web Services based environments, XML-Schemas are used for describing document structures. The type system behind XML-Schema mixes both structural and name-based features [37]. This makes XML-Schema matching a bit complicated and the type system less elegant, since purely structural matching methods cannot be used.

Semantic aspect of service level interoperability is concerned with the meaning of service operations and documents. Matching of service interfaces based on

their operational semantics have been addressed for example in [47]. Operational semantics are usually attached to a service as operation-specific pre- and post-conditions (or effects). These conditions are definitions given in appropriate logic describing the assumptions and results of the operations.

Semantics are use also for attaching meaning for information contents exchanged between services. In tightly coupled and closed systems interpretation of semantics is implicitly coded into the applications, since the operational environment is known during development of the application. Exploiting explicit shared ontologies for description of operation and information semantics provides a more loosely coupled approach. Attaching semantics to service operations and messages for establishing interoperation of services sharing a common ontology is the approach taken for Semantic Web services [25, 31]

Attaching behavioural descriptions to interface signatures provides stricter guarantees of service interoperability. When only syntactic and semantic aspects are considered, we cannot clearly specify how the service should be used. If a formal specification of behaviour is attached to service interface, compatibility or equivalence of services behaviours can be verified using formal methods [7, 46].

5 Community Level Interoperability

Interoperability at the community level must be guaranteed with respect to non-functional aspects, failure handling mechanisms and architectural properties. Interoperation is established by multi-lateral negotiations during community breeding process [20]. Interoperation at the community level is a mutual agreement between all the participants. Community level interoperability grasps rest of the semantic interoperability aspects in enterprise computing environment in addition to the semantic and behavioural aspects described at the service level.

Agreement of non-functional properties, such as quality of service, security, trust, location or availability is an important aspect in community level interoperation. Mutually agreed values for non-functional properties are used for configuring communication channels and middleware services, and are supervised during community operation by the monitoring facilities.

Simple error handling, such as service exception handling, is usually provided and agreed in the technology and service levels. There are also more abstract errors related to enterprise computing which manifest themselves as contract breaches. Failure handling is a community specific activity grasping both kinds of the previous failure types. Failure handling mechanisms and compensation processes have to be agreed upon between all the participants of a community.

Architectural aspects contain such properties as topology of community, composition of services into business roles and coordination of services across the community. Mismatches in architectural properties of communities can be caused by faulty assumptions about other components, connections between components or topology of the community [13]. Architecture description languages such as Wright [1], Darwin [24] or Rapide [23] have been developed for defining software architectures. These languages formalise architectural properties, thus making it possible to automate validation of architectural interoperability.

Standardisation of business community architectures and business cases has also been used for providing architectural interoperability. This is the approach taken for example in ebXML [19] or RosettaNet [35].

6 Organisational Level Interoperability

Pragmatic interoperability is established at the organisational level. Properties stemming from the business level aspects such as strategies, legislation and intentions of different organisations manifest themselves at the organisational level as business rules and organisational policies. Organisational level interoperability is established by negotiation and monitoring facilities during community breeding and operation.

Business rules are declarative statements that define or constraint some aspect of a business [12]. They are part of the organisation's business knowledge which direct and influence the behaviour of an organisation [2]. Typical examples of business rules are different kinds of service pricing policies or regulations on service availability based on customer classifications. A business rule may affect the non-functional or behavioural properties of services by constraining the possible values of attributes or by introducing new kind of behaviour during service operation. To achieve automated validation of business rule interoperation, the business rules should be expressed using a feasible logical framework. For example conceptual graphs [42] and defeasible logic [2] have been used for modeling of business rules.

Organisational policies declare autonomic intentions of organisations and they are specified through the concepts of obligation, permission and prohibitions [40]. An obligation expresses that certain behaviour is required whereas permissions and prohibitions express allowable behaviour. Policies may thus modify behaviour of services by requiring certain actions to be taken instead of the others, or by prohibiting certain actions.

When organisational policies of collaborating participants are known beforehand, policy conflicts can be identified before community operation. If behavioural descriptions are given using an appropriate logic, interoperation of organisation policies with respect to the behavioural descriptions can be verified for example with model checking. However, organisational policies are inherently dynamic entities and not even necessarily published outside the organisations. Organisational policies are one of the primary causes for the dynamism in enterprise computing environments.

7 Conclusion

This paper analysed and identified different aspects of interoperability in service oriented enterprise computing environments. Different collaboration models, namely integrated, unified and federated, have been used for implementing distributed computation systems. Each of these collaboration models possess characteristic solution methods, such as common computing environments or shared

meta-information. Federated community model was identified as the most appropriate collaboration model, as this model supports the heterogeneity, autonomy and dynamism requirements inherent for this kind of environment. This support is provided by additional infrastructure services such as meta-information repositories and monitoring facilities [21], as well as negotiation mechanisms and collaboration contracts.

Interoperability was analysed using five different levels of abstraction. Division into different levels was based on the abstraction level of the concepts to be agreed upon, as well as on the subjects of interoperation. Abstraction levels were named as technology, service, community, organisation and business levels. Each of these levels contain several aspects which must be considered when establishing interoperability. For example when establishing interoperability in service level, the syntactic, semantic and behavioural properties of services must be examined. After identification of interoperability aspects, a discussion about the interoperability aspects in service, community and organisational level was given. Methods and mechanisms for establishing interoperation with respect to each aspect were briefly described.

Acknowledgement

This article is based on work performed in the web-Pilarcos project at the Department of Computer Science at the University of Helsinki. Project was funded by the National Technology Agency TEKES in Finland and Tellabs with active partners VTT, Elisa and SysOpen. The work much integrates with RM-ODP standards work, and recently has found an interesting context in INTEROP NoE collaboration.

References

1. R. Allen and D. Garlan. Formalizing architectural connection. In *ICSE '94*, pages 71–80, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.
2. G. Antoniou and M. Arief. Executable declarative business rules and their use in electronic commerce. In *SAC '02: ACM Symposium on Applied computing*, pages 6–10, New York, NY, USA, 2002. ACM Press.
3. Athena Integrated Project, EU FP6. <http://www.athena-ip.org/index.php>.
4. R. Bastide, P. Palanque, O. Sy, and D. Navarre. Formal specification of CORBA services: experience and lessons learned. In *Proceedings of OOPSLA '00*, pages 105–117, New York, NY, USA, 2000. ACM Press.
5. A. Bracciali, A. Brogi, and C. Canal. Dynamically Adapting the Behaviour of Software Components. In F. Arhab and C. Talcott, editors, *COORDINATION 2002*, volume 2315 of *LNCS*, pages 88–95. Springer-Verlag Heidelberg, 2002.
6. J. Brant, B. Foote, R. E. Johnson, and D. Roberts. Wrappers to the rescue. In *ECOOP'98*, volume 1445 of *LNCS*, pages 396–417. Springer-Verlag, 1998.
7. C. Canal, L. Fuentes, E. Pimentel, J. M. Troya, and A. Vallecillo. Extending CORBA Interfaces with Protocols. *The Computer Journal*, 44(5):448–462, Oct. 2001.

8. New issues in object interoperability. In A. M. J. Malenfant, S. Moisan, editor, *Object-Oriented Technology: ECOOP 2000 Workshops, Panels, and Posters*, volume 1964 of *LNCS*. Springer-Verlag GmbH, 2000.
9. J. Fang, S. Hu, and Y. Han. A service interoperability assessment model for service composition. In *IEEE International Conference on Services Computing (SCC 2004)*, pages 153–158. IEEE, 2004.
10. R. Fatoohi, V. Gunwani, Q. Wang, and C. Zheng. Performance evaluation of middleware bridging technologies. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 34–39. IEEE, 2000.
11. D. S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. OMG Press, 2003.
12. G. Fu, J. Shao, S. Embury, W. Gray, and X. Liu. A framework for business rule presentation. In *12th International Workshop on Database and Expert Systems Applications*, pages 922–926, 2001.
13. D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch or why it's hard to build systems out of existing parts. In *ICSE '95*, pages 179–185, New York, NY, USA, 1995. ACM Press.
14. IEC. *TC65/290/DC: Device Profile Guideline*, Mar. 2002.
15. INTEROP NoE, EU FP6. <http://tmitwww.tm.tue.nl/research/Interop-NoE.html>.
16. ISO. *ISO 14258 – Concepts and rules for enterprise models*. ISO TC184 SC5 WG1, Apr. 1999.
17. S. Jha, J. Palsberg, and T. Zhao. Efficient Type Matching. *Lecture Notes in Computer Science*, 2303:187–206, 2002.
18. D. Konstantas. Object oriented interoperability. In *ECOOP '93 - Object-Oriented Programming: 7th European Conference*, volume 707 of *LNCS*, pages 80–102. Springer-Verlag GmbH, 1993.
19. A. Kotok and D. R. R. Webber. *ebXML: The New Global Standard for Doing Business Over the Internet*. New Riders, Boston, 2001.
20. L. Kutvonen, J. Metso, and T. Ruokolainen. Inter-enterprise collaboration management in dynamic business networks. In *CoopIS 2005 conference*, Agya Napa, Cyprus, Oct. 2005. To be published.
21. L. Kutvonen, T. Ruokolainen, J. Metso, and J. Haataja. Interoperability middleware for federated enterprise applications in web-Pilarcos. In *INTEROP-ESA '05*, 2005.
22. D. A. Lamb. IDL: sharing intermediate representations. *ACM Trans. Program. Lang. Syst.*, 9(3):297–318, 1987.
23. D. Luckham and J. Vera. An event-based architecture definition language. *IEEE Transactions on Software Engineering*, 21(9):717–734, sep 1995.
24. J. Magee, N. Dulay, and J. Kramer. Structuring parallel and distributed programs. *Software Engineering*, 8(2):73–82, Mar. 1993.
25. S. McIlraith, T. Son, and H. Zeng. Semantic Web services. *Intelligent Systems*, 16(2):46–53, 2001.
26. M. Mecella and B. Pernici. Designing wrapper components for e-services in integrating heterogeneous systems. *The VLDB Journal*, 10(1):2–15, 2001.
27. Object Management Group. *CORBA 3.0 - OMG IDL Syntax and Semantics chapter*, jul 2002.
28. R. Orfali, D. Harkey, and J. Edwards. *Instant CORBA*. John Wiley & Sons, Inc., 1997.
29. J. Palsberg and T. Zhao. Efficient and flexible matching of recursive types. In *Logic in Computer Science*, pages 388–398, 2000.

30. M. P. Papazoglou and D. Georgakopoulos. Introduction. *Commun. ACM*, 46(10):24–28, 2003. Special issue on Service-Oriented Computing.
31. J. Peer. Bringing together semantic web and web services. In *The Semantic Web - ISWC 2002: First International Semantic Web Conference*, volume 2342 of *Lecture Notes in Computer Science*, pages 279–291. Springer-Verlag Heidelberg, 2002.
32. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.
33. E. Rahm, H.-H. Do, and S. Massmann. Matching large XML schemas. *SIGMOD Rec.*, 33(4):26–31, 2004.
34. D. Rine, N. Nada, and K. Jaber. Using adapters to reduce interaction complexity in reusable component-based software development. In *SSR '99: Symposium on Software Reusability*, pages 37–43, New York, NY, USA, 1999. ACM Press.
35. RosettaNet Consortium. Rosettanet implementation framework: Core specification v02.00.00, 2004. <http://www.rosettanet.org/>.
36. P. Siddhartha and S. Sengupta. Web services interoperability: A practitioner's experience. In *CoopIS/DOA/ODBASE*, pages 587–601, 2002.
37. J. Siméon and P. Wadler. The essence of XML. In *POPL '03: Proceedings of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 1–13, New York, NY, USA, 2003. ACM Press.
38. M. P. Singh, A. K. Chopra, N. Desai, and A. U. Mallya. Protocols for processes: programming in the large for open systems. *SIGPLAN Not.*, 39(12):73–83, 2004.
39. M. P. Singh and M. N. Huhns. *Service-Oriented Computing: Semantic, Processes, Agents*. John Wiley & Sons, Ltd., 2005.
40. M. Steen and J. Derrick. Formalising ODP enterprise policies. In *EDOC'99*, pages 84–93. IEEE, 1999.
41. Sun Microsystems. *Java 2 Platform, Enterprise Edition (J2EE), 1.4 Specification*, 2002.
42. I. Valatkaite and O. Vasilecas. A conceptual graphs approach for business rules modeling. In *Advances in Databases and Information Systems*, volume 2798 of *LNCS*, pages 178–189, Sept. 2003.
43. A. Vallecillo, J. Hernandez, and J. M. Troya. Component Interoperability. Technical Report ITI-2000-37, University of Malaga, July 2000.
44. A. Vallecillo, J. Hernández, and J. M. Troya. Object Interoperability. In *Object-Oriented Technology. ECOOP'99 Workshop*, volume 1743 of *LNCS*, pages 1–21. Springer-Verlag Heidelberg, 1999.
45. P. Wegner. Interoperability. *ACM Computing Surveys (CSUR)*, 28(1):285–287, 1996.
46. D. M. Yellin and R. E. Strom. Protocol specifications and component adaptors. *ACM Trans. Program. Lang. Syst.*, 19(2):292–333, 1997.
47. A. M. Zaremski and J. M. Wing. Specification matching of software components. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(4):333–369, 1997.

CoSeRT: A Framework for Composing Service-Based Real-Time Applications

Marisol Garcia-Valls, Iria Estevez-Ayres, Pablo Basanta, and Carlos Delgado-Kloos

Telematics Engineering Department, Universidad Carlos III de Madrid, Spain
{mvalls, ayres, pbasanta, cdk}@it.uc3m.es

Abstract. This paper describes a generic framework for the composition of real-time applications based on multiple services. The framework allows that services specify their functionality, as well as their QoS requirements in the form of real-time characteristics. The framework supports static composition; all services required to create an application have to be discovered before launching the whole application. To show the validity and feasibility of this framework, it has been implemented using Java technology.

1 Introduction

The appearance of distributed ubiquitous computing environments has introduced the possibility of building service-based applications ad-hoc. Services that reside in the pervasive atmosphere are downloaded in the machine of the user; then, they can be appropriately composed to create a whole application. This is a powerful tool to build service-based real-time applications, such as multimedia systems. Since multimedia requires that end-to-end timeliness be fulfilled, the chosen service set will have to be such that the applications end-to-end time requirements are met. For instance, let us imagine the case of a person carrying a palm that walks into a VIP room at an airport. The room contains a pervasive environment with various services (encoders, decoders, image scaling services, etc.). Notification of these services is received in the palm device, so that different applications can be composed (games, TV players, pay-TV decoders, etc.). Most of these applications have time requirements, since they are multimedia. Depending on the resource availability of the target platform, a given set of services will be downloaded. For instance, if the user knows that s/he is almost out of battery power, services with low processing demands will be downloaded. It might even be necessary to execute only a specific profile for some services, so that hardware resources are saved for the device to operate for as much time as possible. Of course, the lowest profile of a service will also deliver the lowest output quality to the user; however, this is the reasonable and inevitable trade-off.

To provide acceptable functionality to the user, it is important to support robustness, reliability, and timeliness in the execution of these applications. The general user is more and more concerned about reliable and timely execution on all platforms, ranging from the robust TV set to the smallest personal device. So,

technologies that formerly were only in the sphere of the critical real-time systems are now progressively reaching the multimedia scene as a means of developing robust systems. Real-time technology (such as resource-aware scheduling, execution platforms, etc.) can be applied to multimedia service-based applications, since they have time requirements. Enabling real-time technology requires that services and applications are appropriately characterized according to their real-time requirements. This way, we say that the applications and services can provide their required Quality of Service (QoS), in terms of their real-time requirements and required resources. From such requirements, the environment must provide the guarantees to the services (and application as a whole), so that they will receive the resources they need to execute appropriately.

In this paper, we present a framework to aid the development of real-time applications based on a set of ubiquitous services. A client or user that wants to create such an application can specify the functionality of the services s/he looks for, and the overall QoS requirements of the whole application. For this, our framework allows: (1) service implementers to express the QoS requirements of services upon service announcement, (2) clients to express the functionality of the services they require and the overall applications QoS requirements, and (3) clients to execute composing algorithms to obtain the appropriate service-set, so that the real-time properties of the whole application are fulfilled. As a whole, this framework allows to specify the QoS requirements of a multimedia service based on its resource needs and time requirements. Also, it is allowed to search for a set of services to create a specific application based on its functionality and QoS needs. An architectural view of the framework and a prototype of it based on Jini are also presented.

The paper is structured as follows. In section 2, the related work is stated. Section 3 presents a functional description of the framework. Section 4 describes the architecture. Section 4 describes how the characterization of the real-time properties of services is done. In section 5, an overview of the architecture of the framework is given. Section 6 explains how the composition phase can be performed, and more precisely the static composition (the one of our framework) is described. Section 7 gives the implementation details of the prototype framework we have developed. Eventually, section 8 presents some conclusions.

2 Related Work

Though the high difficulty of providing output timeliness in distributed computing systems is known [1], this handicap remains also for centralized systems, when services that have been developed remotely are connected or composed. Multimedia service-based applications that are developed ad-hoc can execute in two ways: in a centralized environment or in a distributed one. In a centralized service-based application, services are downloaded from a remote location, but they execute locally, i.e., with no remote connections. However, in a distributed service-based application, services are also remotely downloaded and they can communicate with other remote systems to carry out their operation.

The execution environment is an important issue when determining timeliness of services. However, with the appearance of the write-once-run-anywhere paradigm (where translation to intermediate code is possible) these effects can be somehow mitigated. This way, the Java language, and its real-time extension [6], are a good contribution towards providing a real-time execution platform for platform-independent code and services. Code analysis techniques [2] may allow us to determine timeliness over a given platform from a platform-independent code.

Future middleware platforms are expected to provide QoS support that is needed by these new service-based multimedia applications. However, so far, only some general architectures for QoS sensitive ubiquitous applications are appearing such as [3], that covers all aspects of a pervasive environment with no specific coverage of real-time requirements. Other more specific contributions in the development of real-time middle-ware architectures, such as [4], address the architecture internals to provide predictable operation such as memory management and timeliness on Real-Time Java.

There are few research efforts in the direction of integrating real-time awareness into ubiquitous environments. Composition of real-time applications has been addressed, among others in [7,8], but not for pervasive resource-constrained fields. In this work, we present an object-oriented framework aimed at the composition of ubiquitous services to develop applications for multimedia environments in a centralised fashion.

3 Functional Description of the Framework

CoSeRT (Composition of Services with Real Time requirements) offers a generic framework for custom composition of applications with QoS requirements based on ubiquitous services. QoS requirements refer to the resource needs and timeliness of services and applications. As shown in figure 1, the framework allows: (1) specifying time requirements and, in general, QoS requirements of services, (2) announcing services with QoS requirements, (3) discovering services that match a specific functionality,

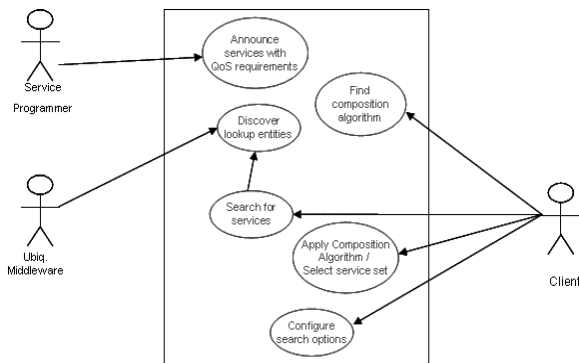


Fig. 1. Functionality of the framework

and (4) integrating and applying composition algorithms for selecting service sets that match the required functionality and meet the applications end-to-end QoS requirements.

3.1 External Interaction of the Framework

The external actors of the framework, as shown in figure 1, are the service programmer, the ubiquitous middleware, and the client.

The *service programmer* is the person in charge of developing ubiquitous services. S/he will attach the parameters that express the QoS requirements of the service. Also, s/he will perform the maintenance of the service (issuing periodic requests for leasing to the lookup service, code updates, etc.).

The basic *ubiquitous middleware* is the entity that supports the execution of the framework. Basically, it provides the framework with:

- ¥ Distributed computing environment functionality: it provides service announcement and discovery facilities.
- ¥ Communication facilities: it provides the basic operations of message passing, remote invocations, serialization, etc., in order to allow remote communication among clients and lookup entities.

The *client* is the actor that makes use of the framework to compose an application ad-hoc. The user will specify the functionality s/he desires, and the requested services to make such an application. Also, s/he will have to specify the QoS requirements that the application as a whole has. The framework needs the information on these requirements to select an appropriate set of services so that they all meet the overall application's end-to-end requirements.

3.2 Processes

The main objective of the framework is the ad-hoc development of real-time service-based applications. For this, the framework supports the main three processes or phases: service announcement, service discovery, and composition of services. Execution and service launching is not covered in this paper.

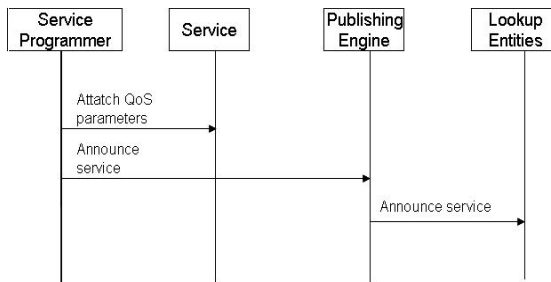


Fig. 2. Steps of the announcement phase

Announcement. Individual services are published in lookup services specifying their functionality and their QoS requirements in terms of their resource needs (processor and memory usage). The functionality of a service is based on a unique identifier or tag. Figure 2 shows an overview of this phase.

Discovery. To compose an application, services have to be previously discovered. The discovery is previously configured by the client, specifying the lookup services to poll (if any) and the functionality of the required services. Services will be discovered based on their unique tag. Figure 3 presents the discovery process in combination with the subsequent composition phase.

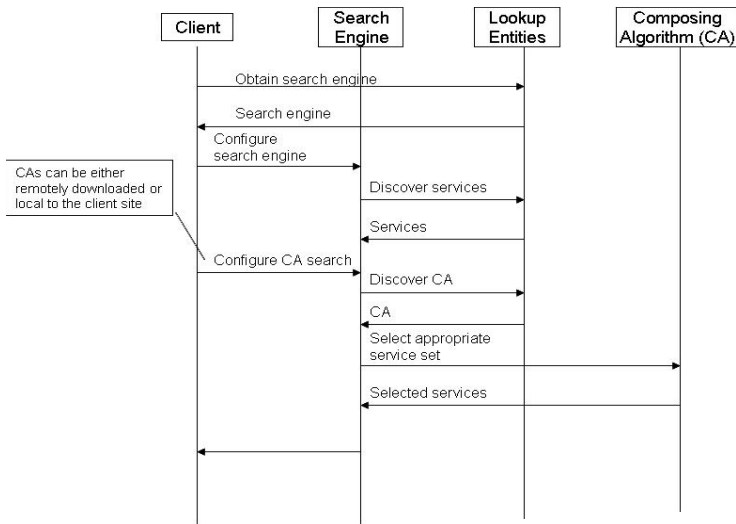


Fig. 3. Discovery and composition processes

Composition. This phase begins after discovery. The composition phase is the selection of a suitable service set that will be part of the desired application. Election is based on the functionality of services and their QoS requirements, so that the overall QoS requirements of the application are met.

4 Real-Time Properties of Services

An application is made of services as shown in figure 4. For the framework to determine the set of services that have to be part of such application so that it can meet its overall QoS requirements, services must carry a QoS characterization with them. Such characterization is based on the expression of their real-time properties as an attribute of type `TimeService`.

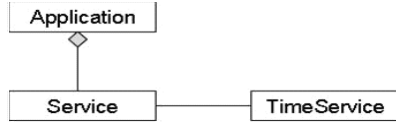


Fig. 4. Relation among application and services

The characterization of the real-time properties of ubiquitous services is presented as a structural view in UML in figure 5. The `TimeService` class is an attribute that contains all real-time properties; therefore, it implements the `Entry` interface, which marks a class as an attribute. Figure 5 also identifies the main entities or parts of a service.

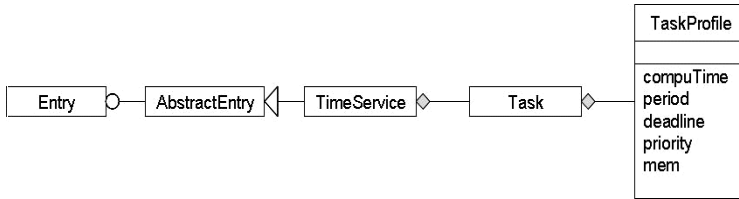


Fig. 5. QoS characterization of services

In the same way that services are the subparts of a whole application, also services may be decomposed into various parts or *tasks*, also known as threads. A task (in the real-time sense) is the concurrency unit of a service, i.e., a service can be made of more than one thread of control. In our framework, a service should be structured as a set of tasks, each specifying its real-time characteristics in the form of resource requirements, mainly w.r.t.:

- ¥ *time* (computation time, activation period, execution deadline, and priority), and
- ¥ *memory*, that is the average required memory also specified by the service programmer.

A task profile also contains other attributes specifying the platform for which it was developed and tested. However, these data (and the real-time values) are overwritten by the composing algorithm if a platform-independent bytecode analysis is performed for the target platform.

5 Architecture of CoSeRT

CoSeRT is based on a centralized distributed computing environment, such as Jini, as shown in figure 6.

CoSeRT relies on the existence of a basic communications middleware. This can be the case of Java RMI or CORBA. On top of that, the framework defines the following main entities to perform the operations described before:

Services. These are the ubiquitous entities that perform a certain functionality. They are created by service programmers. When they are announced, the framework supports the statement of their functionality and QoS requirements.

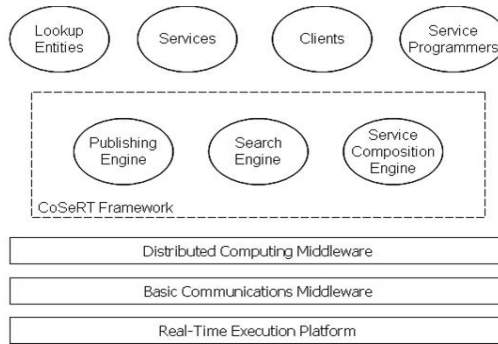


Fig. 6. CoSeRT

Composition Algorithm (CA). These are entities or algorithms that contain the logic for selecting the service set to create an application, i.e., they execute the composition phase. The service set is selected to meet the QoS requirements and timeliness of the application as a whole. CAs can be either ubiquitous services downloaded from a remote server or local entities created by clients to customize the creation of applications. As a result of their operation, CAs deliver the selected service set to clients that will be part of the final application.

Time Service Proxy (TSP). This is the core part of the framework. It has two main parts: the core control flow of the framework (TSPI) and a graphical user interface (GUI). The TSPI performs the basic functionality of the announcement phase (to announce services specifying their QoS requirements), discovery phase (delivers services specified by clients), and composition phase (looking for CAs or applying them directly to select service sets that make up the final application). The GUI allows interaction with clients and service programmers to perform the former activities. The TSPI has the three main engines shown in figure 6.

The ubiquitous environment is based on centralized lookup entities. The TSPI will contact lookup entities to search for the desired services. Services specify their QoS requirements as attributes in their announcement phase. The framework allows that lookup entities be contacted in unicast or multicast mode. Requests to them indicate the service type that has been searched for. Therefore, matching parameters are mainly the service type and the execution parameters of the service. The service type is based on a unique functionality identifier. This is enough since there are restrictions: the number of services that there are, their data input/output format, and the source of the services is known; the service federation is limited and centrally managed.

Answers of lookup services are handled by means of discovering entities and by individual threads of control; this separates the process of collecting answers from the process of obtaining the proxies of lookup services for downloading matching services. It increases efficiency.

6 Composing Services

In our framework, composition of services refers to the selection of the appropriate service set that will be part of the application. Service composition can be either static or dynamic.

Static composition. All services are available a priori, i.e., before the application is launched. Once the application is started, no service replacement is possible at run-time. Also, no run-time re-configuration of individual services or of the pipeline (connection) of services is done.

Dynamic composition. Services may be replaced and/or reconfigured at run-time, i.e., during application execution. Dynamic composition has some strong implications mainly related to (1) support from the underlying platform (either real-time operating system and/or resource manager) to launch, stop, reconfigure the buffer connections among services, restart services, etc., and (2) extra delay time to look for other services that may cause to enter a new discovery phase.

Dynamic composition allows to implement dynamic QoS management; if an application is not functioning correctly with the current resources it has been assigned, its services may be reconfigured to a lower profile so that they will need less resources.

The CoSeRT framework supports static composition. The composing algorithm applies WCET techniques to extract the WCET based on the analysis of the intermediate code. Based on the QoS requirements of the services, the calculated WCET, and the applications QoS requirements, the composing algorithm selects the appropriate service set.

7 Prototype Implementation

CoSeRT has been implemented on top of Jini. Services are programmed in Java language and serialized in the lookup services.

7.1 Service Announcement

Service announcement and discovery is centralized in lookup services. Unicast or multicast requests are sent to one or more lookup services. When a suitable one is discovered, the service will be registered in it by sending its serialized implementation.

Service announcement in CoSeRT is very similar to the process of announcing any service in Jini. However, CoSeRT allows services to describe their QoS requirements by means of an additional attribute consisting of introducing entries. The class that describes these requirements is `TimeService`, as shown in figure 7.

`TimeService` has the method `setTimeServiceParameters`, used by the service programmer to implement and register a service. They are the QoS requirements, its arguments:

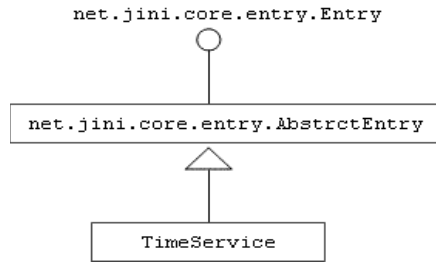


Fig. 7. QoS specification in the TimeService Jini entry

- ¥ `taskvector`: a list of tasks or threads that are part of a service. Each position contains an object of class `Task`. Just as an application is made of services, a service can be made of a set of tasks or threads.
- ¥ `label`: a unique identifier expressing the functionality of the service.
- ¥ `serveBefore`: the precedence list, in case that the service requires the execution of another service before it.
- ¥ `mem`: the memory that the service requires, as average, for its execution.
- ¥ `proc`: a list of pair processor requirements and processor platform for which it was measured.

7.2 Service Discovery

The framework defines two main classes for performing service discovery:

- ¥ `TimeServiceProxy`. This is the fundamental entity for performing service discovery. It contains the methods for clients to configure the options to search for services that will potentially be part of an application. It also contains the methods to find composition algorithms.

```

public interface TimeServiceProxy{
    void setSearchOptions(Vector stateDiscovery);
    void setParametersHost(ParametersHost paramHost);
    void setServiceOptions(ServiceTemplate[] services, int[] maxMatches), long searchTime);
    void setAlgorithmOptions(ServiceTemplate algorithmTemplate, Algorithm algorithmLocal);
    Vector getTimeServices();
}
  
```

- ¥ `TimeServiceProxyImpl` (TSPI). This class is the main entity of the CoSeRT framework that contains the execution phase of it. It is an instance of `TimeServiceProxy`, that clients download to perform all operations of the framework.

The search engine is made of the `TimeServiceProxy` and the Discovery Entity; the later comes from the Jini environment. To start the search, clients must configure the TSPI object setting the search options as shown in figure 8. Service discovery is carried out using the TSPI entity. This entity allows to initiate the search for objects with (1) a certain profile of QoS requirements and (2) a certain functionality.

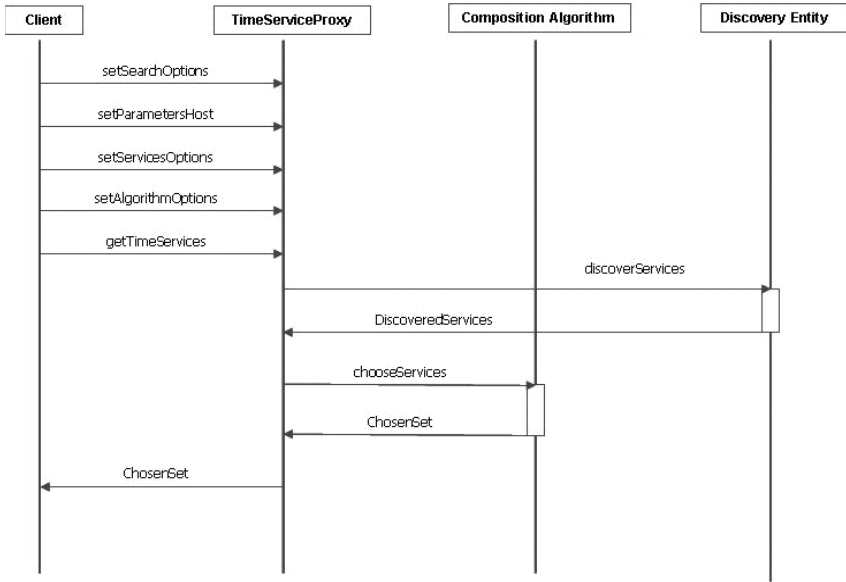


Fig. 8. Discovery and composition processes

The call to `getTimeServices` starts a *search thread* for each type of service that has been specified by the client in the process of configuration of the TSPI object. Each search thread makes discovery requests for lookup services in unicast or multicast mode, depending on the client settings. Each thread will give lookup services an object of class `ServiceTemplate`, that is the ubiquitous service that is being searched for. Once a set of service implementations have been discovered for each service, the QoS requirements for each service implementation can be obtained. These requirements are contained in the attribute `servicesVector` (an array of `TimeService` objects).

7.3 Service Composition/Selection

The last three steps of figure 8 show the composition phase, where the CA selects the set of services that will be given to the client to create his/her application. After the discovery phase, a composing algorithm can be applied to obtain the service set that matches the client specification to compose an application. Composition algorithms of CoSeRT currently implement three policies: (1) FIFO, so that composition time is minimized, (2) WCET analysis based on the code of services, and (3) shorter end-to-end response time.

To be part of CoSeRT, a CA must implement the interface `Algorithm`. This is a markup interface that contains a method which represents the service-selection logic, `chooseServices(TimeService[] services)`. Therefore, a CA must implement its logic inside this method. It receives the set of QoS requirements of all

discovered services. After, applying the composition algorithm to them, a vector containing the selected service-set is returned to the search engine (to the TimeServiceProxy).

The framework supports that composition algorithms may be ubiquitous services used by clients or they can pre-exist at the client site. If an implementer of a composition algorithm desires to announce it, the procedure is very similar. The attribute TimeService will not be included. The following classes are provided for its announcement:

- ¥ **Algorithm.** Any composition algorithm must implement this interface which contains the method for selecting the service set depending on its logic. Composition algorithms receive a vector with the QoS requirements of services and the relevant information about the client execution environment. They apply their logic to these data and, as a result, they obtain the set of services that are more appropriate for the application.
- ¥ **AlgorithmType.** When a composing algorithm is announced, it must include an attribute that describes the algorithm. This attribute is an instance of `AlgorithmType`. The description of the type of algorithm is based on two character sequences which are unique: the name and the description, where unique keywords are contained.

7.4 GUI and Operation

The CoSeRT framework provides a Graphical User Interface that allows both service programmers and clients to interact with the framework in a friendly environment. Figures 5 and 6 give an overview of this part.

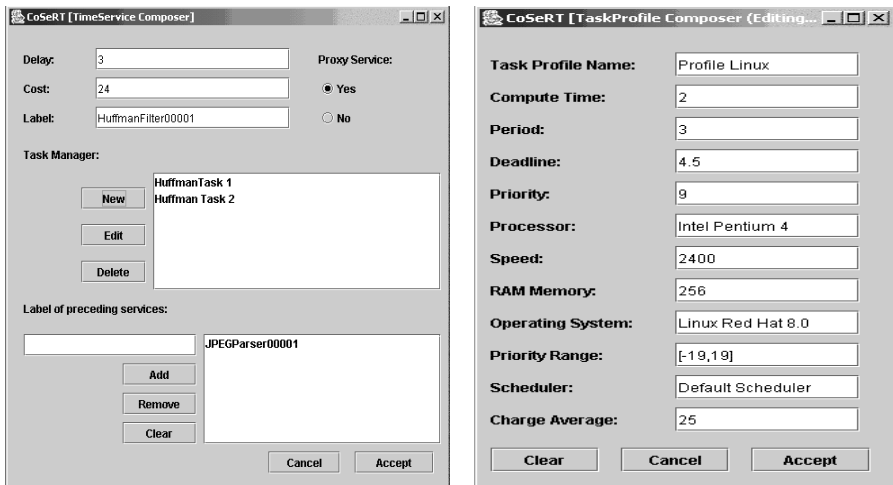


Fig. 9. (a) Specification of potential composing services, and (b) Editor of QoS requirements

To show the feasibility of the framework, the operation of the framework has been performed on simulated applications. Following, a multimedia processing application that contains four services, each with three different QoS profiles, is shown. Figures 9 and 10 show the GUI part of the framework that interacts with the core TSPI to perform the configuration of the discovery of the services, their discovery and composition.

Figure 10 shows the result of the operation of the CoSeRT framework, where four services have been selected among the discovered services to compose a multimedia application. The application required a JPEG Data Stream Parser, a Huffman Decoder, a Zig Zag Run Length Expansion, and a Coefficient Dequantization service. All services resided in at least two lookup services in the environment, and each had two profiles. The environment was made of four distributed PCs each running Java RMI, Java Jini, and CoSeRT. The used CA was local to the client and used a logic based on the shorter end-to-end response time.

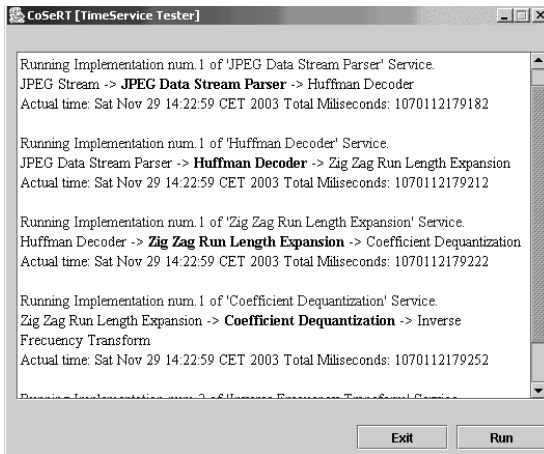


Fig. 10. Operation of the selected service set

8 Conclusions

Most of ubiquitous service infrastructures focus on the development of efficient discovery protocols, communication protocols, agent platforms, and mobility aspects. However, integrating parameters to capture the needs of real-time applications (in the form of QoS requirements) in these environments has not been sufficiently addressed in the literature. In the field of QoS, most of the research has been applied to other application domains, for instance, the distributed domain with no special ubiquitous concerns or component frameworks for application composition. In this paper, a framework for integrating real-time properties of services is presented. This framework allows to announce services with their QoS requirements as attributes, to search for services with a given functionality and QoS requirements, and to compose these services selecting the most appropriate ones to develop a real-time application.

A prototype of the framework has been developed to show the feasibility and validity of this idea.

References

- [1] Kim, K. H. Towards QoS Certification of Real-Time Distributed Computing Systems. In *Proc. of the 7th IEEE International Symposium on High-Assurance Systems Engineering (HASE 2002)*. Tokyo, Japan. October 2002.
- [2] Pushner, P. and Schedl, A. Computing Maximum Task Execution Times – A Graph-Based Approach. *Journal of Real-Time Systems*, 13 (1): 67-91. 1997.
- [3] Nahrstedt, K., Xu, D., Wichadakul D., and Li. B. QoS-aware Middleware for Ubiquitous and Heterogeneous Environments. *IEEE Communications Magazine*, 29(2). 140-148. November 2001.
- [4] Basanta-Val, P., García-Valls, M., and Estévez-Ayres, I. Towards the Integration of Scoped Memory in Distributed Real-Time Java. In *Proc. of the 8th IEEE International Symposium on Object-Oriented Real-Time Systems (ISORC 2005)*. Seattle (WA), USA. May 2005. Accepted for publication.
- [5] Garcia-Valls, M., Alonso-Munoz, A., Ruiz, J., and Groba, A. an Architecture of a QoS Resource Manager for Flexible Multimedia Embedded Systems. In *Proc. of the 3rd International Workshop on Software Engineering and Middleware (SEM 2002)*. In *Lecture Notes in Computer Science*, vol 2596. Orlando (Florida), USA. November 2002.
- [6] G. Bolella, B. Brosgol, P. Dibble, S. Furr, J. Gosling, D. Hardin, and M. Turnbull. *The Real-Time Specification for Java*. Addison Wesley, 2000.
- [7] I. Crnkovic and M. Larsson. A Case Study: Demands on Component-Based Development. In *Proc. Of the 22nd International Conference on Software Engineering*. Cannes (France), May 2002.
- [8] D. Isovich and C. Norström. Components in Real-Time Systems. In *Proc. Of the 8th Conference on Real-Time Computing Systems and Applications*. Tokyo (Japan) 2002.

Supporting Business Experts in the Design of B2B Transactions Through Interactive Process Simulation

Michael Schmitt, Christophe Incoul, and Eric Dubois

Centre d'Innovation par les Technologies de l'Information(CITI),
Centre de Recherche Public Henri Tudor, 29 Avenue John F. Kennedy L-1885,
Luxembourg-Kirchberg, Luxembourg
{Christophe.Incoul, Michael.Schmitt}@tudor.lu

Abstract. The paper presents a toolkit for the design and the interactive validation of message-based document exchange within the context of multi-partner electronic business transactions. The business case used in this paper is a simplified version of the purchasing process used in the electronics industry, based on the EDIFICE standard.

1 Introduction

Electronic business transactions are more and more replacing paper-based business processes, the objective being to offer better and more convenient services to customers and trading partners, at lower cost, more efficiently and with less manual and time-consuming human intervention.

The activity of designing or customizing an electronic business process involves experts from different companies and with different educational backgrounds. First, there are domain experts from the concerned business areas, such as purchasing or sales experts. Then, there are EDI or messaging specialists who will set up the data mappings for the conversion from the messaging standard used to the in-house application systems that will need to process the data given. Finally, there are business process experts that liaise with the trading partners as regards the structure and content of the messages exchanged and that lead the integration tests.

Due to the different backgrounds and areas of expertise of these experts, effective communication and a shared understanding of the targeted business objectives are a critical factor of success. Without a shared understanding, there's the risk that the IT application will not or only partially match the business requirements, which leads to re-work and hence to increased efforts and cost.

This paper illustrates the benefits of formal modeling with regards to the design and the customization of electronic transactions based on messaging standards in the domain of electronic purchasing. Moreover, it emphasizes the benefits of simulation of the transaction models in order to support the business experts in the validation of the transaction already at the analysis stage and not when it is deployed.

After a brief introduction of the EFFICIENT modeling framework in section 2, we detail in section 3 the overall approach in a real case study associated with an e-purchasing business case. Section 4 describes the EFFICIENT simulation toolset. Finally, section 5 wraps up with conclusions.

2 The EFFICIENT Modelling Framework

The EFFICIENT R&D project (E-business Framework For an efficient Capture and Implementation of ENd-to-end Transactions) aims at enhancing the quality and the effort of system development associated with multi-partner B2B transactions while reducing the time to market and augmenting the STP (Straight Through Processing) level. Efficient proposes a two-phased development process for new B2B transactions:

- A modeling phase that turns the business requirements identified by the different experts into computer readable models. The modeling language used to capture the business requirements is UML [11] and the methodology employed to structure the activities in this phase is based on UN/CEFACT Unified Modeling Methodology (UMM) [10].
- An interactive validation phase that translates the models created into code, which can be processed by a simulation environment. The simulation allows the domain experts to validate the correctness of the data models without having to formally understand and be able to modify the computer models themselves.

In EFFICIENT, an electronic transaction is described in terms of the following information layers: (see Figure 1):

- The *business layer* [3] provides a top-level view on the business scenario that governs the transaction. It depicts the transaction configuration and allows the trading partners to develop a common understanding about the business goals, the vocabulary used and about the roles and responsibilities of each participant.

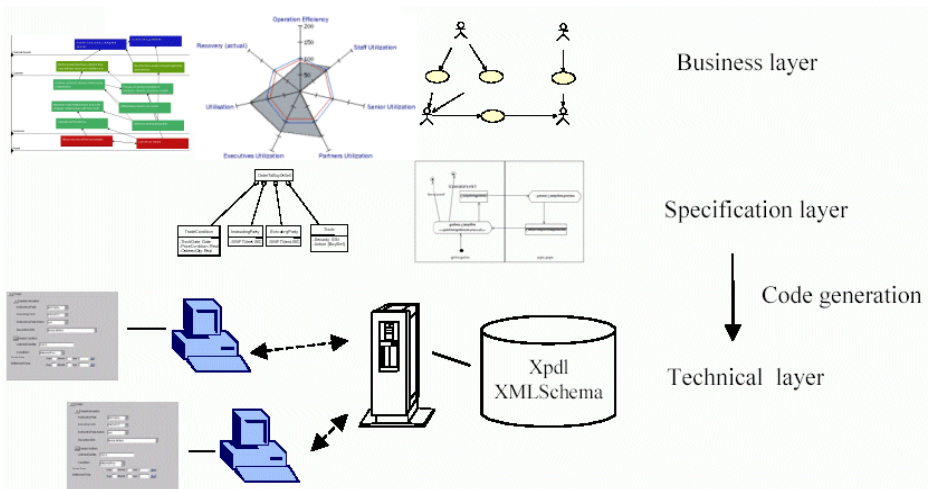


Fig. 1. The modeling layers of the EFFICIENT toolset

- The *specification layer* details the flow and the content of the business documents (messages) that the trading partners exchange in the transaction, as well as the rules governing the exchange.
- At the *technical layer* the business transaction is simulated creating a shared understanding between the participants and facilitating its verification and validation.

2.1 Technical Details

Efficient uses MagigDraw [17], a commercial UML CASE tool for the representation of the static and dynamic aspects of a B2B transaction. A plug-in to MagicDraw was developed to implement the functionality required for model verification and code generation.

Once the transaction has been designed and formally verified, the infrastructure needed for the orchestration (animation) at the technical layer is automatically generated from the UML models developed at the specification layer. The flow of the business transaction, modeled in an UML activity diagram, is translated into an XPDL [14] file that initializes the workflow engine, the core component of the Efficient server. Each message, specified in a restricted UML class diagram, translates into an XMLSchema [16]; the business rules that govern the transaction are transformed into an xlinkit [20] representation. These files are stored in an xml database used by the workflow engine.

The Efficient client component is internet-based. The messages that the business partners exchange in a transaction are displayed using the XMLSchema definitions and XForms [21] recommendations.

The technological choice we have taken in the Efficient project follow two principles: to choose open-source tools wherever possible and adopt standards when they exist. These principles allow us a maximum of flexibility and facilitate the interoperability with existing products and frameworks. The list of standards we use are the following:

- WMFOpen [13], an open-source workflow engine, which is fed with the XML Process Definition Language (XPDL) standard definitions for process orchestration (animation). Note that unlike BPEL [19], XPDL supports the concept of "subflow" with a clearly defined interface between a calling and a called process. Efficient uses this feature of process composition in that it allows a transaction designer to re-use previously defined transaction models as well as to import industry specific standard sets of transactions (such as Rosettanet PIPs, EDIFICE [23], SWIFT [22] and others). A re-use of existing components and industry standards reduces the effort involved in transaction development and increases the time-to-market.
- SOAP [15] as the core messaging protocol.
- XML:DB API [24], to access the eXist [18] XML database.
- W3C XMLSchema, for the description of messages Chiba [25], based upon the XForms recommendation of the W3C.
- XLINKIT for a representation of first-order logic business rules that constraint the transaction at both the content and the flow level.

3 The E-Purchasing Case Study

We have used the EFFICIENT toolset to model the electronic purchasing process as proposed by the EDIFICE standardization [1] organization for the electronics industry. A case study was performed in collaboration with Avnet Electronics Marketing [2], a global distributor for electronic components, for tailoring and customizing a subset of the messaging standards associated with electronic purchasing to their particular needs. The case study spans the complete purchasing process, from the purchase order to a supplier to the settlement of obligations created by the purchase act.

The following sections detail the different static and dynamic properties of the e-purchasing transaction and outline the process of validating their correctness and feasibility both from an IT (soundness, performance, completeness) and from an usage perspective (appropriateness, legal considerations, particular needs). It will be argued that only a joint effort of the IT experts and business experts leads to a sound specification of the electronic transaction. The EFFICIENT toolset is used to create a shared understanding between both groups and to guide them through the process of building and agreeing on the different aspects of the transaction.

3.1 Business Domain

When designing a new transaction or customizing an existing one, first the actors involved need to be identified. In an electronic purchasing scenario, the following actors exist:

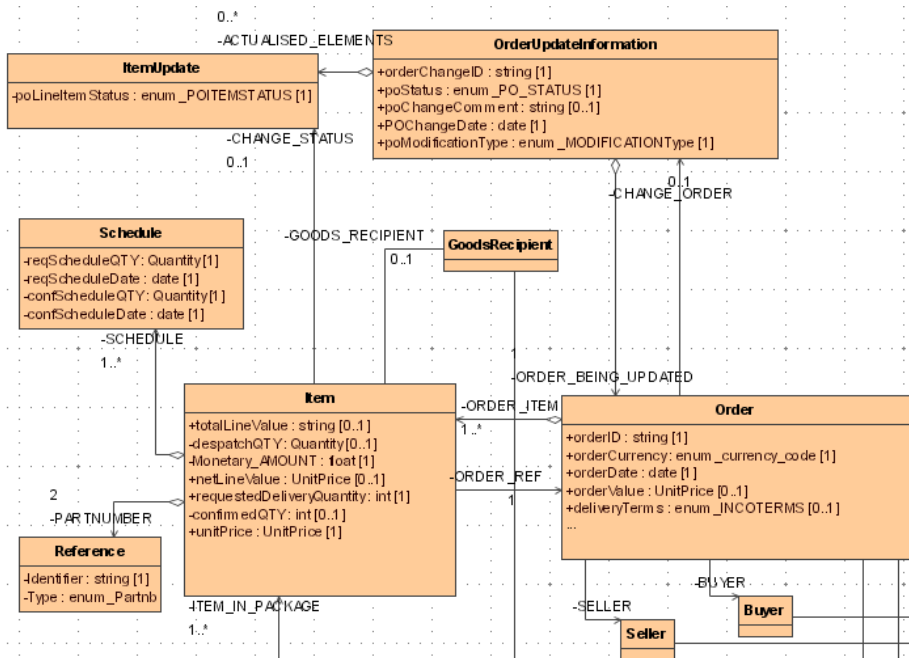


Fig. 2. A subset of the ePurchasing Business Domain

- Avnet: a distributor of electronic components that issues a request to buy goods, services or works in response to a customer stimulus (sales order, minimum stock quantities etc.) or to satisfy the requirements identified by their logistics service portfolio.
- Supplier: a supplier or service provider who aims at proposing his services, goods or works in response to the request of his customers. A supplier may be a component manufacturer or a wholesaler.

Next, the different parties or actors need to have a common understanding of what we call the business domain, that is, that they “speak the same language”. The business domain is made up of the key concepts of the business sector and details the relationships that hold between them. In EFFICIENT, the business domain is modeled using a specific UML class diagram, which consists of classes with their properties (attributes) and the relationships (associations) that hold between the classes. The business domain of our transaction is shown in Figure 2. In our business case, the business domain contains such concepts as an ORDER, an ORDER ITEM, the SCHEDULE information for an ITEM, the PARTNUMBERS that identify the product component both from a vendor and a buyer’s part of view as well as PRICING information and some miscellaneous information about what a business ORGANIZATION is. Compared to the EDIFICE standard the above figure already incorporates some company specific tailoring such as that the GOODSRECIPIENT is stated at the line item level whereas on the order level there’s no delivery related information. Another example is that Avnet does not use SCHEDULE information on the order level. The business domain depicted in Figure 2 is a simplification of the actual business domain used by Avnet.

3.2 Dynamics of the Transaction

Once the basic vocabulary and a common understanding of the business domain have been achieved, we can then describe the business process underlying the transaction. We use a UML activity diagram to detail the flow of business documents among the participating actors, and the activities (responsibilities) each participant will need to carry out at what stage of the transaction. The activity diagram is composed of swimlanes representing the different roles that occur in the transaction and activities performed. Each business document the actors exchange is defined by a UML class diagram detailing its structure and information content see section 3.3.

Figure 3 depicts the beginning of the activity diagram of the ePurchasing transaction: Avnet initiates the process by sending a purchase order request to a part supplier. The supplier can then verify and validate the purchase order and responds either by rejecting or by accepting Avnet’s request. In the acceptance case, the supplier will confirm partly or partially each of the line items of the order.

Upon reception of the POResponse message from the supplier, Avnet’s purchasing manager can then decide whether the confirmed delivery schedules do correspond to their needs or whether he or she wants to place a change order (POChange). Last but not least, Avnet may still cancel their purchasing request at this stage of the

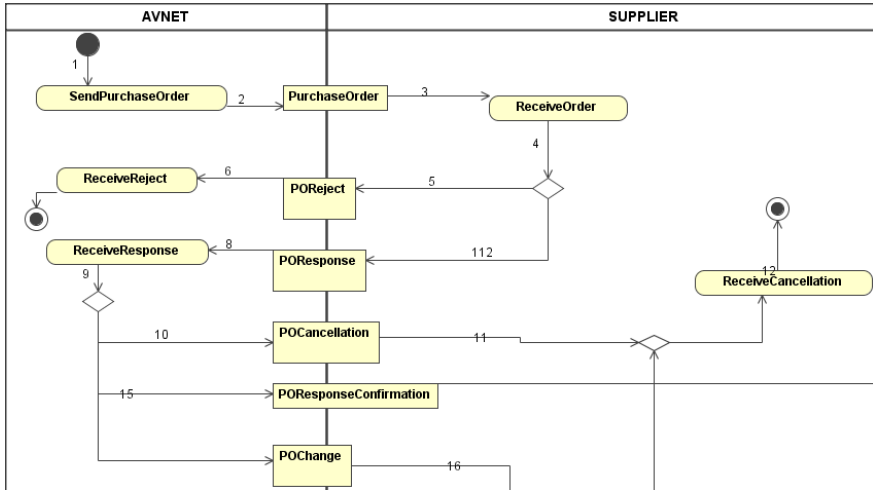


Fig. 3. Part of the activity diagram of the ePurchasing transaction

transaction using the **POCancellation** message. Note that the messages in this activity diagram do not match one to one with the corresponding messages defined by EDIFICE. As for instance, in standard EDI terms both of these messages, the purchase order cancellation and the purchase order response are implemented using the same message type, an Order Response message.

3.3 Structure of Messages

In the next phase, we define the information content for each of the business documents used in the activity diagram. The decision on what needs to be in and what doesn't for a particular document is based on the information needs of the document recipient in order to perform the business activities he's requested to. In EFFICIENT we model business documents by restricted UML class diagrams, which are built by selecting a subset of the classes and relationships from the business domain.

The class diagram for the message "PurchaseOrder" is given in Figure 4. It contains some header information such as the ORDER number and the order date, one or more ORDER_ITEMS each of which is characterized by the PARTNUMBER information, the requested delivery quantity of the part as well as the required delivery SCHEDULES.

There are three parties listed in the orders message, a buyer, a seller and a delivery party. Whenever feasible, Avnet refers to a party by a previously agreed party identification code and does not send the complete address information, unless otherwise requested by the trading partner. This way, a single contractual agreement may be used for a business partner where all corresponding manufacturing sites are subsumed, and also this prevent typing mistakes in the postal addresses to be uploaded into the internal application systems. For any kind of manual intervention such as open questions, a contact person may be included. Note that this class

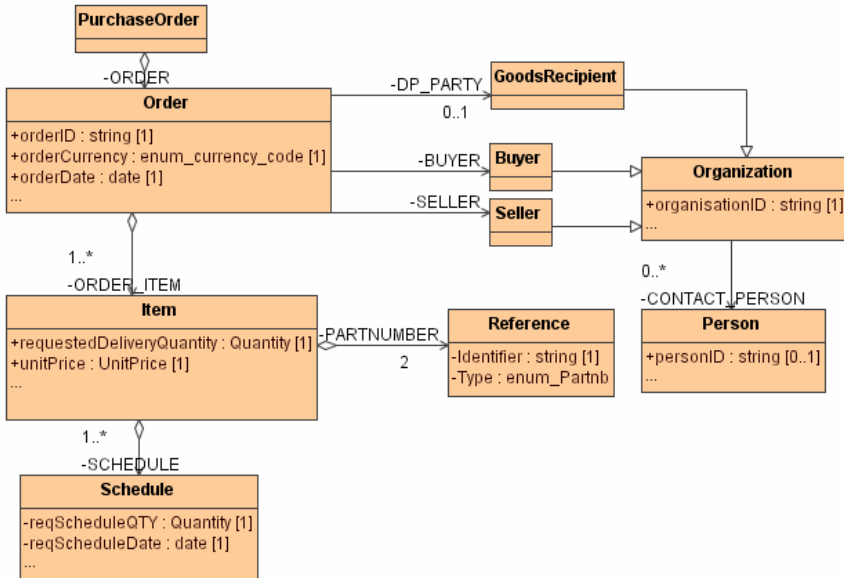


Fig. 4. The class diagram of the "PurchaseOrder" message

diagram is a subset of the concepts of the business domain. For each concept of the business domain, it must be decided whether or not it is considered as relevant for the recipient of a message and hence needs to be part of the message content.

3.4 Business Rules

Rules may be added to constrain the information content that may be input in a business document. Figure 5 shows a rule that states that the values of a POResponse message do relate and will be taken from the previous PurchaseOrder message. So for example the OrderID, currency and the line item information can be copied over from the previous PurchaseOrder message. Only where there is new information in the POResponse message that are not part of the purchase order request the sales person at the supplier needs to fill in respective values: In our sample message these are the confirmed delivery quantities for the parts requested and the confirmed delivery schedules. These may or may not be identical to the requested value from the purchase order request. Another type of business rule that we have implemented in the EFFICIENT toolset is a logical constraint based on the business facts that can be imposed on a message. So for example the confirmed delivery date for an item should not be in the past of the requested delivery date. Another example is that whether or not the supplier can confirm the requested delivery quantities and schedules for a specific component, the confirmed quantity must never exceed the quantity that was requested. The business rule that verifies this constraint is given in figure 5, in the POResponse class. More about business rules, their structure and the way they are implemented is given in [4].

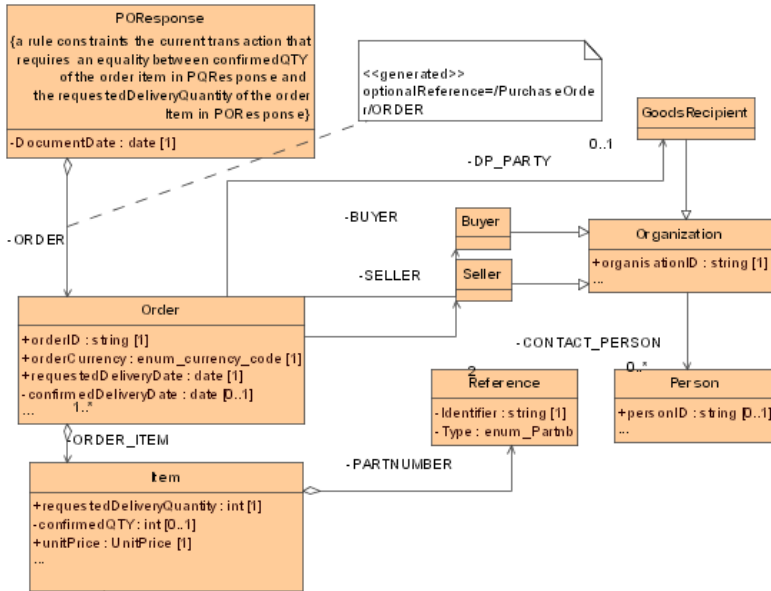


Fig. 5. An inter-message rule and a simple business rules

4 Validation and Customizing of the E-Purchasing Transaction

The EFFICIENT toolset supports the generation of code from the static and dynamic UML models as well as from the business rules that are the input into a workflow-based simulation environment. The transaction simulator, which we refer to as the EFFICIENT animator, allows the participants of the transaction to simulate the business process and hence to validate the correctness of the data models. The code generated for the animator includes the code generated for the transaction flow monitor, the code associated with the business rules as well as with the interfaces for reading and writing messages. More about the animator and the code generation can be found in [6].

The animator consists of an Internet based client component that connects with a server workflow (WF) engine that coordinates the execution of the business process. Figure 6 illustrates the behavior of the workflow engine for the initial part of the message exchange between Avnet and its supplier.

Avnet sends a “PurchaseOrder” message (see the generated use interface on Figure 7) to its part supplier. The animator receives the message and verifies whether it satisfies the requirements specified by the UML data models. If it finds no error, it forwards the message to the recipients together with the set of possible answers as defined in the UML activity diagram. The supplier can then either reject the purchasing request by sending a PORreject message or respond to Avnet’s request by an POResponse.

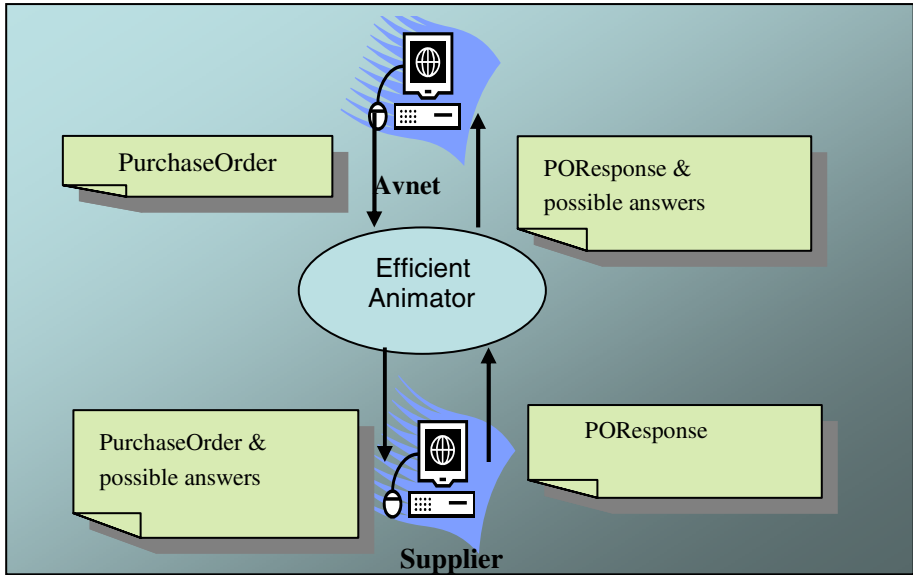


Fig. 6. Exchange of messages in the Efficient Animator



Fig. 7. The generated user interface for the PurchaseOrder message

5 Conclusion

Efficient is a tool that allows business experts to design and validate a B2B transaction before its implementation. The modeling of the business domain renders the concepts and the structure of complex business scenarios more transparent. It allows business experts to select a subset of the information that is provided with a messaging standard such as EDIFICE or Rosettanet [12] and to customize and tailor

this subset of a standard to their business needs. EFFICIENT generates a user-friendly interface (web-forms) that allows the business expert to validate whether his data models are sufficient and whether all the information required is available in the internal application systems. Furthermore, the transaction animation (simulation) helps to detect incoherencies at the message exchange level. More information can be found at our website, <http://efficient.citi.tudor.lu>.

Future work is in two directions. On the hand there is a need for improving our understanding of a business scenario and the link between business and process models [5, 7]. Another topic is the economic value associated with an e-transaction [8, 9]. Moreover, another effort consists in the development of advanced verification tools for the analysis of the consistency and completeness of the models created. Last but not least, a transaction monitor will be developed that facilitates the search and the tracking of business information associated with an end-to-end transaction involving multiple message exchanges and a trading partners.

References

1. Edifice, European B2B Forum for the Electronics industry, <http://www.edifice.org>
2. Avnet Electronics Marketing, <http://www.avnet.com>
3. M. Schmitt, B. Grégoire, C. Incoul, S. Ramel, P. Brimont and E. Dubois, *If business models could speak! Efficient: a framework for appraisal, design and simulation of electronic business transactions*, ICEIMT 04, 2004, <http://efficient.citi.tudor.lu>.
4. Amel Mammam, Sophie Ramel, Bertrand Grégoire, Michael Schmitt, Nicolas Guelfi. *Efficient: A Toolset for Building Trusted B2B Transactions*. Accepted for 17th Conference on Advanced Information Systems Engineering (CAISE'05), Porto, June 2005.
5. Maria Bergholtz, Prasad Jayaweera, Paul Johannesson and Petia Wohed, "Process Models and Business Models - A Unified Approach", *International WorkShop on Conceptual Modeling Approaches for e-Business*, ER 2002 Tampere, Springer LNCS.
6. R. Eshuis, P. Brimont, E. Dubois, B. Grégoire, S. Ramel. *Animating ebXML Transactions with a Workflow Engine*, In Proc. CoopIS 2003, volume 2888 of LNCS, Springer, 2003.
7. Prasad M. Jayaweera, "A Unified Framework for e-Commerce Systems Development", Kista : Dept. of Computer and Systems Sciences, Univ./KTH, PhD thesis, 2004.
8. J. Gordijn, J.M. Akkermans and J.C. van Vliet, "Business Modelling is not Process Modelling". in: *Conceptual Modeling for E-Business and the Web*, LNCS 1921, pg 40-51, ECOMO 2000, October 9-12, 2000 Salt Lake City, USA, Springer-Verlag.
9. Alexander Osterwalder Yves Pigneur, *An e-Business Model Ontology for Modeling e-Business*, Proc. of the Bled Electronic Commerce Conference 2002, Bled.
10. http://www.unece.org/cefact/umm/umm_index.htm
11. Unified Modeling Language (UML), <http://www.uml.org>
12. Rosettanet, eBusiness standards for the global supply chain, www.rosettanet.org
13. <http://wfmopen.sourceforge.net/>
14. <http://www.wfmc.org/standards/XPDL.htm>
15. <http://www.w3.org/TR/soap/>
16. <http://www.w3.org/XML/Schema>
17. <http://www.magicdraw.com/>
18. <http://exist.sourceforge.net/>
19. <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>
20. <http://www.xlinkit.com>

21. <http://www.w3.org/MarkUp/Forms/>
22. <http://www.swift.com/>
23. <http://www.edifice.org/>
24. <http://www.xmldb.org/>
25. <http://chiba.sourceforge.net/>

Requirements to Improve the Synchronisation of Inter-enterprise Models

Cristina Campos, Reyes Grangel, Ricardo Chalmeta, and Òscar Coltell

Grupo de Investigación en Integración y Re-Ingeniería de Sistemas (IRIS),
Dept. de Llenguatges i Sistemes Informàtics, Universitat Jaume I,
Campus del Riu Sec s/n, 12071 Castelló, Spain
{camposc, grangel, rchalmet, coltell}@uji.es

Abstract. Virtual Enterprises have become a good organisational solution to cope with the current economic environment. A number of methodologies have been developed to assist in the creation and management of a Virtual Enterprise, using Enterprise Modelling as a useful way to enhance its performance. However, it is necessary to develop new mechanisms and methodologies to improve the interoperability and to synchronise changes among different inter-enterprise models.

In this paper, we present a definition of a set of requirements needed to synchronise enterprise models in order to improve Virtual Enterprise interoperability. The work is based on previous projects dealing with interoperability and Enterprise Modelling, like UEML or INTEROP. The requirements described in this paper were selected and analysed with the aim of adapting them to the necessities of the synchronisation of enterprise models in a Virtual Enterprise.

1 Introduction

A **Virtual Enterprise** is a temporary alliance of independent enterprises that come together to share resources, skills and costs, with the support of the Information and Communication Technologies, in order to better attend market opportunities. To design an efficient and flexible Virtual Enterprise that gives the appearance of being a single enterprise to customers is a very complex task [1].

In order to help the creation and management of a Virtual Enterprise, partners develop models using different Enterprise Modelling Languages and different background knowledge. These **inter-enterprise models** need to be interchangeable and understandable for people involved in each enterprise. In addition, Virtual Enterprises need to update their models due to the natural evolution of business, new legal requirements, changes in the strategy of the partners, and so forth. This kind of changes can affect concepts, business, results and other aspects in enterprise models that are needed to work correctly in real time. These issues are really important in the process of creation of a Virtual Enterprise but they become more critical when a Virtual Enterprise is actually running.

This paper is organised as follows. Section 2 gives a brief description of the framework in which this research work carried out. Section 3 reviews the main

concepts related to Enterprise Modelling and synchronisation. In section 4, the main results obtained in the analysis of requirements are presented and, finally, section 5 outlines the main conclusions.

2 Framework of the Work

INTEROP (Interoperability Research for Networked Enterprises Applications and Software) is a Network of Excellence supported by the European Commission for a three-year-period. **INTEROP** aims to create the conditions for innovative and competitive research in the domain of Interoperability for Enterprise Applications and Software [2].

Interoperability is, from a **system-oriented point of view**, the ability of two or more systems or components to exchange information and then use that information without any special effort in either system. Moreover, from a **user-oriented point of view**, interoperability is the user's ability to successfully search for and retrieve information in a meaningful way and have confidence in the results [2].

The work presented in this paper was carried out within this framework and focused on the '**Common Enterprise Modelling Framework (CEMF)**'. One of the main objectives related to **CEMF** is to develop a new version of **UEML** (Unified Enterprise Modelling Language) [3] and to complete it with other facilities like the templates for mapping Enterprise Modelling Languages and mechanisms to allow Collaborative Enterprises to synchronise their enterprise models.

On the other hand, the **IRIS Group** of the Universitat Jaume I in Castelló (Spain) has been working on several projects related to Virtual Enterprise in different sectors (transport, tile industry, textile, and so forth) since 1999 [4,5,6,7]. The main aim of these projects has been to define and apply an architecture capable of supporting the design and creation of a Virtual Enterprise, as a particular case of Collaborative Enterprise. Some of the most useful results obtained have been a methodology and a set of techniques, reference models, and software applications that enable all the elements (organisational, technological, human resources, and so forth) of a Virtual Enterprise to be coordinated and integrated.

The creation and management of a Virtual Enterprise is an extremely complex process that involves different technological, human and organisational elements. Indeed, there is an extensive array of approaches and methodologies which describe the management processes for enterprise integration (Purdue Guide for Master Planning, GRAI-GIM, and so forth), Enterprise Modelling Languages (IEM, EEML, GRAI, IDEF, and so forth), and supporting Enterprise Modelling Tools (MO²GO, METIS, GraiTools, BONAPART, and so forth) for the design of enterprise models mainly for individual enterprises.

Nowadays, the group's activity is centred on extending the methodologies developed in its previous projects to include issues related to interoperability among partners of a Virtual Enterprise. Hence, this kind of enterprises present huge difficulties to interoperate and a cultural transformation is needed in order

to achieve a correct sharing of information and knowledge. The IRIS Group is also involved in INTEROP Network, especially in the work related to CEMF. In this work group one of the main contributions provided by IRIS group's members has been to define a set of requirements that can help the synchronisation of enterprise models developed by different enterprises that collaborate in order to achieve a common objective.

3 Synchronisation of Enterprise Models in Virtual Enterprises

Enterprise Modelling is defined in [8] as the art of 'externalising' enterprise knowledge, which adds value to the enterprise or needs to be shared, i.e., representing the enterprise in terms of its organisation and operations (processes, behaviour, activities, information, objects and material flows, resources and organisation units, and system infrastructure and architectures). Therefore, this art consists in obtaining enterprise models that are a computational representation of the structure, activities, information, resources, and so forth of an enterprise, government or any other type of business or organisation. Models can be descriptive and/or definitional and they can show what is and what should be. Its role should be to support the design, analysis and operation of the enterprise according to the model, in a 'model-driven' mode [9].

Enterprise Modelling can be used to better understand and improve business processes, to select and develop computer systems and so on, but the most important benefit of enterprise models is their capacity to add value to the enterprise [8]. Such models are able to generate explicit facts and knowledge which can be shared by users and different enterprise applications in order to improve enterprise performance. However, these models are developed with different Enterprise Modelling Languages and integrating them is a complicated task, since tools for exchanging models created with different languages do not exist [2,3,10,11].

UEML [3] can help to achieve the exchange of enterprise models among several organisations, but a synchronisation among models is also necessary in order to deal with evolution and different views in these models. Synchronisation can be defined as the adequate temporal coordination of two or more systems or organisations that have a business relation, in order to optimise the integration of their processes.

Synchronisation of enterprise models is needed when these models represent activities which offer results that affect and condition the performance and results of other enterprises or systems. This is a critical aspect when models represent enterprise processes, information, organisational structures, products, decisions and knowledge that are closely connected, as is the case for example in a Virtual Enterprise.

Moreover, enterprise models must change and evolve if they are to be useful throughout the whole life cycle of the Virtual Enterprise. Model evolution and changes must be performed in accurate time and taking into account all the

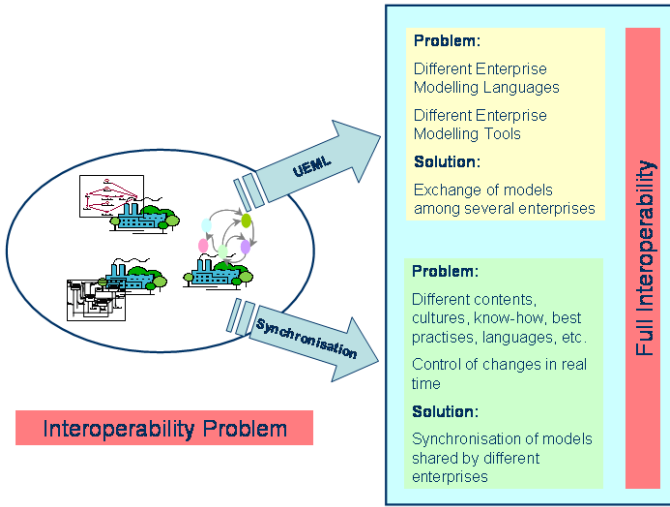


Fig. 1. Interoperability problem among different enterprises at the modelling level

aspects and possible partners affected by the modifications. This solution must be supported by an adequate synchronisation (see Fig.1).

In this paper, we have defined and analysed a set of requirements that are necessary for the synchronisation of enterprise models in a Virtual Enterprise. In order to classify these requirements we have taken into account:

1. The work developed in the group '**Synchronisation of Different Distributed Enterprise Models**' in INTEROP [12], where the requirements were classify as: (a) consistence of models, (b) model maintenance and flexibility adaptation in a distributed environment, (c) security of model management, (d) decisional and social aspects, (e) extraction of knowledge represented in different models, (f) federated analysing, (g) evaluation and simulation of enterprise models across the company borders, (h) model design procedures, and (i) model evaluation procedures.
2. The characteristics, structures, and objectives of **Virtual Enterprises**, different companies with different department structures and different policies but with a common objective.
3. The results and experience obtained from the different **projects developed on Virtual Enterprises** by our research group.

The classification proposed in order to analyse a first set of requirements has been simplified to four main categories: organisational and decisional category, ontological category to deal with concepts and contents, technological category, and modelling language category. The aim of defining this classification has been to organise the fields to study and to better analyse the main necessities in a Virtual Enterprise.

4 Requirements for Synchronisation of Virtual Enterprise Models

In defining and analysing the necessities for a complete synchronisation of enterprise models in Virtual Enterprises, no requirements about **Modelling language category** have been considered, taking into account that UEML solves transformations and mappings among Enterprise Modelling Languages. The following tasks were developed in order to define a set of requirements to synchronisation of enterprise models in the Virtual Enterprise:

- **Review of requirements defined for UEML** that were considered to be outside the scope of the UEML Project [3]. Some of them are related to the capacity of enterprises to collaborate and to synchronise their models.
- **Review of other approaches and projects** [13,14,15] that deal with problems that can be solved by means of synchronisation of models or that use this issue as part of the solution proposed.
- **Addition of new requirements** considering particular aspects and characteristics of a Virtual Enterprise.

Regarding the issues considered and defined in the previous section, the requirements were organised in the following categories (see Fig. 2):

- **Organisational and Decisional category:** partners that make up a Virtual Enterprise have common objectives and share enterprise models that are connected to each others. These connexions must be identified and controlled by establishing a suitable form of synchronisation. When a partner needs to modify a process that affects other processes from other partners

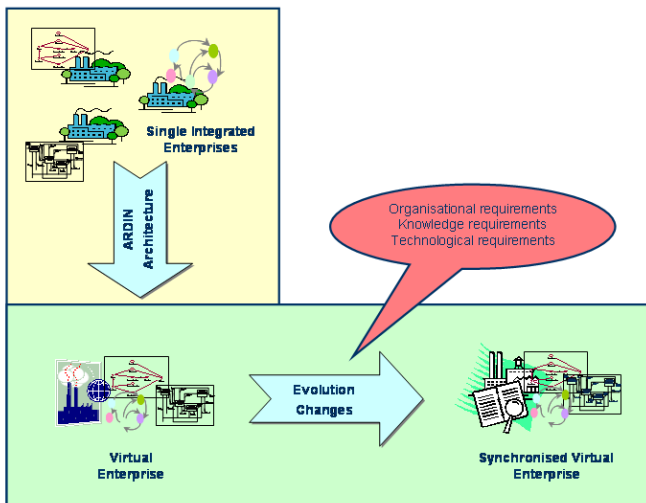


Fig. 2. Synchronisation of enterprise models in Virtual Enterprises

in a Virtual Enterprise, some criteria about how to deal with these changes and when are needed. Changes can affect different organisations in a Virtual Enterprise and all of them must agree to update these models, it could therefore become a decisional problem. Thus, requirements about organisational and decisional aspects have to be defined (see Table 1).

- **Ontological category:** distinct enterprises have different notations, concepts and levels for naming any aspect that is modelled. Each of the partners can be from a different industrial sector, or can be in a different position (i.e. supplier or customer) in the Virtual Enterprise; people from each company have diverse backgrounds and knowledge and what should be more frequent, a different notation or vocabulary. Therefore, it is essential to define requirements that will help to define a common conceptual framework that allows models to interoperate (see Table 2).
- **Technological category:** each partner in the Virtual Enterprise, even if using the same Enterprise Modelling Tools, could represent its processes using different levels of detail, precision, and so forth. To establish what is needed to deal with correct models from a technological point of view will be needed (see Table 3).
- **Modelling language category:** this aspect will not be taken into account in this paper. Although it is an important issue to establish a good synchronisation of inter-enterprise models, the requirements needed to solve this problem are considered in the UEML Project [3].

The requirements for each of these categories are presented in Tables 1, 2, and 3. These requirements are based on a Virtual Enterprise where partners need to share enterprise models in order to establish flexible and effective cooperation through the synchronisation of these models. A basic situation could be as follows: an enterprise A needs to update part of its models that are connected to other models in enterprise B, and the changes produced in models in enterprise A affect models in enterprise B.

4.1 Organisational and Decisional Requirements

Enterprises involved in a Virtual Enterprise must develop procedures to define different levels of models. The main goal is to establish a correct organisational policy for introducing new versions of enterprise models. This category should define whether the changes can be assumed automatically by all the members of the Virtual Enterprise or if they need previous discussions and agreements. In this case, semiautomatic or manual procedures must be developed to guarantee the correctness of the synchronisation process. The requirements related to establishing this kind of procedures are shown in Table 1.

On the other hand, employees' participation in the synchronisation process is required to assure its success. Therefore, setting up a work group with people from all the partners in the Virtual Enterprise can be useful to evaluate and make decisions about the evolution of enterprise models in a synchronised way. This group will work together and be in contact throughout the entire life cycle

Table 1. Requirements in organisational and decisional category

Requirement	Description
To define procedures and mechanisms to allow sharing of models between enterprises	Procedures for managing common parts or parts of models in a Virtual Enterprise are necessary to be in a position to make better decisions about changes
To detect model changes that affect models of other enterprises	Any change in a model of enterprise A that affects models or processes modelled in enterprise B has to be detected
To generate the description about required model changes	When a change in a model of enterprise A is produced, information about models affected in enterprise B also has to be generated
To define procedures and mechanisms for enterprise B to confirm or reject model changes. Any change in one of the components of a Virtual Enterprise has to be notified	Changes can be accepted or not, depending on whether the change is compulsory for all partners of the Virtual Enterprise. For instance, strategic aspects must be analysed before implementing the change
To define security levels	It is necessary to define different security levels for users who can maintain models and make decisions about them

of the Virtual Enterprise, and one of the main benefits will be the ongoing communication that will help to make decisions.

4.2 Ontological Requirements

One of the main aspects that must be supported by synchronisation of enterprise models is the sharing of concepts, notations and meanings. Therefore, the definition of one ontology that gives support to all these aspects is really important in Virtual Enterprise in order to achieve a real synchronisation at the modelling level. The requirements related to ontological aspects are described in Table 2.

Synchronisation of models deals mainly with the content of a model and not with modelling languages. Therefore, it is related to ontological issues i.e. semantics of the models. Results obtained in recent works on ontologies and semantic enrichment of enterprise models should be analysed in order to be studied and included in this category of requirements.

4.3 Technological Requirements

As regards the technological level, the easiest solution would be for all the partners in a Virtual Enterprise to adopt the same Enterprise Modelling Tool to develop their models. This is the solution proposed in the ARDIN Architecture [6], where a common platform is defined for using by the partners, but this is not always possible due to several reasons.

Table 2. Requirements in ontological category

Requirement	Description
To maintain information consistence between models	One of the main issues to be dealt with is the information shared between models. Concepts and names must be consistent and must always keep the same meaning
To define common concepts in order to adapt and connect models in a Virtual Enterprise	When some enterprises decide to cooperate in a Virtual Enterprise, it is necessary to define a common framework in order to define concepts and knowledge that can be shared and used
To define mappings	It is necessary to establish correspondences between concepts that can be explained and understood in different ways

Table 3. Requirements in technological category

Requirement	Description
To provide different levels of automation for model updates	When some enterprises decide to cooperate by exchanging their enterprise models, they need a mechanism to define a priori different levels of model updates: manual, semiautomatic or automatic. Some changes can be obviously considered in any model but others would need analysis and decisions
To provide control of versions and changes	Regarding models that can change and affect different elements of the Virtual Enterprise, controlling different versions correctly is fundamental
To provide a standard exchange format with which to exchange models	For example, the use of XML to enable the exchange of models
To promote the use of homogeneous tools to improve and help synchronisation in a Virtual Enterprise	Tools can also help in version control, the use of homogeneous tools is an important decision that will improve synchronisation and model changes
To make easy concurrent access to models	To enable sharing of the same model between different users in real time

In any case, if distinct partners use different Enterprise Modelling Tools, these tools need to have export and import utilities to enable the exchange of enterprise models. Although, this kind of utilities is not enough, and more facilities and functionality are required in this kind of tools in order to achieve a complete synchronisation of enterprise models. The set of requirements related to this technological aspect are detailed in Table 3.

5 Conclusion

Virtual Enterprises need to develop mechanisms in order to interoperate throughout their entire life cycle. This interoperability can be reached thanks to the implementation of a good methodology for the design and creation of the Virtual Enterprise and well defined procedures that allow the collaboration to be maintained. One of the main questions to help these methodologies and procedures is the use of inter-enterprise models. But enterprise models must be updated in order to show any change or modification carried out in the Virtual Enterprise or in any of its partners. Synchronisation of enterprise models is, therefore, an important issue for the success of Virtual Enterprises and developing well defined procedures to achieve this model synchronisation will be an important aspect to be considered in the evolution of this kind of enterprises.

How changes in models or in the processes represented by these models, can be detected in order to synchronise them with other models in the same Virtual Enterprise, and how the models can be updated taking into account decisional aspects from all the partners involved in the Virtual Enterprise are some of the questions answered by the set of requirements proposed in this paper. This is a first step to achieve a good synchronisation of inter-enterprise models in a Virtual Enterprise and the future work is going to improve and to enlarge the definition of the requirements in order to identify more necessities and to better evaluate them.

Acknowledgments

This work was funded by CICYT DPI2003-02515. It is also partially supported by the European Commission within the 6th Framework Programme by INTEROP NoE (IST-2003-508011, [2]). The authors are indebted to WP5.

References

1. Bernus, P., Nemes, L.: Organisational Design: Dynamically Creating and Sustaining Integrated Virtual Enterprises. In Chen, H.F., Cheng, D.Z., Zhang, J.F., eds.: Proceedings of IFAC World Congress. Volume A., Elsevier (1999) 189–194
2. INTEROP: Interoperability Research for Networked Enterprises Applications and Software NoE (IST-2003-508011). <http://www.interop-noe.org> (2005)
3. UEML: Unified Enterprise Modelling Language Project (IST-2001-34229). <http://www.ueml.org> (2005)
4. Chalmeta, R.: Virtual Transport Enterprise Integration. *Journal of Integrated Design and Process Science* **4** (2000) 45–55 Publisher IOS Press publishes.
5. Chalmeta, R., Campos, C., Grangel, R.: References architectures for enterprises integration. *The Journal of Systems and Software* **57** (2001) 175–191 Elsevier.
6. Chalmeta, R., Grangel, R.: ARDIN extension for virtual enterprise integration. *The Journal of Systems and Software* **67** (2003) 141–152 Elsevier.
7. Chalmeta, R., Grangel, R.: Performance measurement systems for virtual enterprise integration. *International Journal of Computer Integrated Manufacturing* **18** (2005) 73–84 Taylor & Francis.

8. Vernadat, F.B.: Enterprise Modeling and Integration: Principles and Applications. Chapman and Hall (1996)
9. Fox, M.S., Gruninger, M.: Enterprise Modelling. *AI Magazine* **19** (1998) 109–121
10. ATHENA: Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications) Project (IST-2003-2004). <http://www.athena-ip.org> (2005)
11. IDEAS: IDEAS (Interoperability Development for Enterprise Application and Software) Project. <http://www.ideas-roadmap.net> (2005)
12. Jaekel, F.W., Perry, N., Campos, C., Dassisti, M.: Deliverable D5.2: Description of Objectives, Requirements and Scenarios for Synchronisation of Different Distributed Enterprise Models Including Benefits and Gaps. <http://www.interop-noe.org> (2005)
13. MISSION: Modelling and Simulation Environments for Design, Planning and Operation of Globally Distributed Enterprises. <http://www.ims-mission.de/> (2005)
14. Opdahl, A.L., Sindre, G.: Facet Modelling: An Approach to Flexible and Integrated Conceptual Modelling. *Information Systems* **22** (1997) 291–323
15. SPIDER-WIN: SuPply Information Dynamic Exchange and contRol by Web-based Interaction Network. <http://www.spider-win.de/spider-win.htm> (2005)

On Automating Networked Enterprise Management

Ustun Yildiz, Olivier Perrin, and Claude Godart

LORIA-INRIA,
BP.239 Campus Scientifique,
54500 Vandœuvre-lès-Nancy, France
{yildiz, operrin, godart}@loria.fr

Abstract. With the new middleware IT technologies such as Web Services and peer-to-peer computing facilities, a Virtual Enterprise can be built easier achieving some problems of interoperability. Although existing standards deal with syntactic issues they are primitive for the maintenance of VE while the processes are fulfilled in the untrusted and dynamic environment of the Web. In this paper, we investigate the problems of maintenance and propose a generic model that can be used for the automation purposes of monitoring and management. The mechanism we propose models the process of the Virtual Enterprise and the perspective of process manager. Thus, it automates the maintenance according to predefined configurations.

1 Introduction

The Web services standards such as UDDI [8], WSDL [3] and SOAP [10] make inter-organizational interactions easier than in the past. Like previous generations of middleware supports, they aim to facilitate application integration. Although they can achieve some of important interoperability problems and make loosely-coupled collaborations easier, they are too primitive for the automated invocation and composition of services and for the maintenance of built compositions. To deal with these issues, existing WS standards need to be supported with additional languages, architectures, and related approaches. We notice various efforts in the research literature such as BPEL4WS [7], OWL-S [5], WSLA [4] that aim to complement existing standards with additional features providing formal specification for automated discovery, composition, and execution of WS based virtual collaborations.

However, the use of a set of services in tandem to achieve a precise goal with a number of constraints, goes beyond the use of common data formats and exchange mechanisms. It requires complex, dynamic, adaptive mechanisms that permit to monitor and manage WS collaborative processes with minimal problems of reliability, performance and cost. In this paper, we review some difficulties of the automated monitoring which is the core part of process management frameworks and we explain our initial ideas about our monitoring approach and framework.

The rest of paper is organized as follows, the next section details the problem statement while Section 3 explains the overview of our approach. Next, we detail the mechanism of our approach. Section 5 gives a case study where our approach can be deployed. Section 6 reviews similar works, then we conclude and make a discussion about our future work.

2 Problem Statement

A Web service based Virtual Enterprise(VE) gathers autonomous organizations that provide particular capabilities to enact a collaborative business process. As in traditional business activities, the virtual collaboration has a number of goals such as the respect of mutual commitments, the increase of benefits, the decrease of risks and the protection of privacy. In the highly dynamic and untrusted environment of the Web, it is unrealistic to expect the outcome of a complex process to be as planned initially. From a process management automation point of view, the execution of collaborative process should be supported by additional mechanisms that permit the monitoring of its behavior. The observation of process activities helps process managers to analyze the performance of contributors, to detect anomalous behavior that can cause lately unwanted situations or to review management alternatives that can increase benefits. The challenge of an *ideal* monitoring agent -human or software- is to fulfill its task in a real-time, accurate, reliable and autonomous manner. The limitations of existing middleware technologies and antagonist nature of different business features make the setup of an ideal monitoring support very difficult. For example, if a monitoring agent requires a key data which characterizes the state of an observed service and if this data is private to the service that holds it then the outcome of the monitoring can not be efficient, or there is always a network delay between the occurrence and detection time of an event in the observed system. The second important issue is indirectly related to similar limitations, the process monitor decides what characterizes an undesired or wanted behavior of an activity while there are no certain and precise facts. An execution can be considered as a normal behavior by a monitoring agent and as abnormal behavior by another. Most of the time, the monitoring analysis can have fuzzy nature. Classical monitoring solutions that we detailed some in the Related Work section, try to compute conventional process properties such as deadlines or costs with exception based analysis. First, this approach that does not provide any predictive indication concerning the future states of activities can not be an efficient support for the ad-hoc nature of Web services context where there are numerous management alternatives at the disposal. It lacks for rich semantics and support that can permit process monitor to express its interested features.

We focus on pro-active and predictive business management which goes one step further comparing to classical management approaches. We try to provide monitoring analysis as close as possible to realistic facts. Our aim is to provide a generic support for human process managers permitting to define, compute their interested process measures.

3 Overview of Our Approach

Our approach aims to provide a support composed by structures, concepts and algorithms to facilitate the automated monitoring of WS based virtual enterprises. In this context, we consider a VE as an organization that crosses the boundary between the virtual and physical world. Because an information system that provides a service is a part of an overall process that makes effects in the physical world.

3.1 Virtual Enterprise Process Model

This section provides a brief introduction to process models of virtual enterprises. The first step of the process management is the selection of the services to compose. The process manager chooses services according to their functionalities they provide and process constraints, it defines the dependencies that exist among them. The process can be initially scheduled or the services can be dynamically chosen during the enactment. The process manager can compose services using computation patterns (e.g. Workflow Patterns [9]). Although there is a flexibility, the process has a number of global constraints that must be respected such as the global cost or a deadline. Another important point that concerns the composition is the dependencies among services. Among dependencies, there are different degrees that can tightly couple one service with another. For example, a service is supposed to begin its execution at a precise date and it needs the output of another service at this date. If the former fails then it will cause cascading impacts on service that follow it. As a matter of monitoring, a system designer puts emphasis on critical dependencies.

3.2 Service Behavior

The service behavior is the characterization of the instantiated service operation such as the way it produces and consumes events, it responds to invocations or it operates on business metrics. In the previous section, we mentioned that the monitoring outcome can not be the result of an *ideal* analyze. To model the monitoring analysis of a monitoring agent, we consider two kinds of monitoring analyze:

1. *Exact output (EO)*: The exact output is the result of an justified analyze done by a monitoring mechanism. In such cases, the output of monitoring describes a situation that really happened or will certainly happen.
2. *Indicational output (IO)*: The second type consists of analysis that give precise information about process states and behavior.

The *EO* is specified using predicates in first order logic and do not depend on any subjective analyze. The predicates depict relationships among involving entities. If the predicates that illustrate the *healthy* execution hold then the process enacts or will enact as planned. If the predicates that illustrate *unwanted* executions hold then the service can fail or can not be enacted as planned. For example, a service that uploads a certain amount data over a network will precisely fail if the rest of data can not be uploaded before the deadline using the maximum bandwidth of the service consumer.

The *IO* consists of sophisticated analysis that takes as input broadest view of the involving entities and gives indications about run-time process sevaluation as output. The indications can be behavior analysis such as compliance, deviation or violation of expected behavior, performance analysis such as low, normal or outstanding performance, or any analyze required by process manager.

4 Putting Automation into Practice

The key of our approach is to configure the monitoring policies of a VE at high level abstract processes. The configuration consists of the mining of metrics and events involv-

ing in the process. Precisely, we choose measurable illustrations to express the analysis in order to provide a good support for the comparison of uncomplete or undetermined behavior.

4.1 Definition of Entities to Compute

The process that will be enacted by different services has a number of constraints and has an initial execution plan. Each service guarantees a number of commitments such as the beginning date, end date, their capabilities and associated qualities to provide. The process manager has the unambiguous knowledge of the process constraints and the agreed behavior of services. The second set of information to compute is the run-time behavior of the services. The run-time behavior expresses the properties of the operations done by the service related to its agreement with service consumer. The behavior of the service can be examined in a single instantiation or in loops. Thus, its behavior is stored in process logs and it is compared to its agreed execution. Besides these two observable and objective phenomena, the process manager defines an expected(or desired) execution of the used service. The expected behavior of a service can be defined using the critical relationships, dependencies, previous use of the service or with the expertise of the process manager. For example, there is a critical dependency between two services such as a service needs the output of its preceding before a precise date to begin its execution, otherwise it fails and causes a damage to process manager. If the preceding service has the habit to fulfill its task far before the deadline, the process manager can be concerned when it does not as previously or to make a reliable composition the process manager may want the service to fulfill its task as soon as possible. Contrary to classical approaches, we do not rely the expected behavior of the service only to its former execution results. It can be calculated by the run-time environment of that time also. For example, if there are circulation problems, a delivery service can be expected to be later than usual. The goal of monitoring framework is to check the conformity of the run-time behavior of the service and its expected behavior.

Process description. We call \mathcal{P} , the description of the process. It consists of process constraints, actors, objects, services to compose and the qualitative and quantitative properties of the process. The Process Manager plans the initial execution of the process defining the relationships among the entities involved in the process. Briefly, the set \mathcal{P} describes what a process is supposed to provide when it is started.

Process behavior. We call \mathcal{C} , the current state of the process. The process behavior consists of features that illustrate the current state of services such as received QoS, fulfilled steps, underlying network activities or any information that can illustrate the current state of observed services. The \mathcal{C} characterizes what the process has been doing since its start.

Process expected behavior. We call \mathcal{Q} , is the description of the expected execution of process described by \mathcal{P} . For example, let's suppose there is service that consists of a good delivery before a deadline. The good can be delivered at any time between the activation date of the service and its deadline. If the service requester is concerned by

the failure of this service, it can consider to take precautions before the deadline. In case where there are no alternatives to execute before x hours before the deadline then this precise date can be a critical point of the service that the service requester expects the fulfillment of the service. The expected execution of a service can be defined by using various features such as the reputation of the service. For example, let's suppose the service delivered the good always y hours after its invocation in the previous invocations then its behavior can be its expected execution.

4.2 Monitoring Engine

We introduce a monitoring function \mathcal{M} that takes a process (or a party of a process) as input, uses disposed information to analyze its state and returns EO or IO .

Due to lack of space and facilitation of the lecture, we consider a set of sub-services that compose a complex service that can be observed independently. The technique that we use for sub-processes can be easily thought for the process in general. A sub-process, itself, can be considered as a basic service. For example, a service that delivers several goods with different properties in different contexts after the reception of an order. We can consider the deliverance of each good as a sub-process. The motivation of our approach is to not detail complex interactions or various outputs that can have place but to show how the behavior of a service can be modeled.

Definition 1 (Monitoring function). *The \mathcal{M} is a function that takes as input a running service and returns either an information that depicts a precise and justified state (EO) or analysis (IO) about this service.*

Let \mathcal{S} the set of monitored services that compose the process described by \mathcal{P} ,

$$\mathcal{M}: \mathcal{S} \longrightarrow [0, 1]$$

The return value of \mathcal{M} has different meanings, the two return values of \mathcal{M} correspond to EO :

Let $s \in \mathcal{S}$, s is sub-process (or a service) of an overall \mathcal{P} ,

if $\mathcal{M}(s) = 1$ then the service is provided or will be certainly provided as planned,

if $\mathcal{M}(s) = 0$ then the service is failed or will not be provided as planned,

The intermediate values in $]0, 1[$ characterize the run-time behavior of monitored service, the convergence toward 1 can illustrate the completion of a running service as planned, or an outstanding performance. The convergence toward 0 can illustrate the deviation of a service from its expected behavior or an irregularity.

The algorithm below depicts partially the intern mechanism of \mathcal{M} function while it illustrates IO .

- Γ is the set of predicates that illustrate the planned fulfillment of observed service, they consist of relationships between \mathcal{P} and \mathcal{C} . In case they hold, the service is fulfilled or will be certainly fulfilled as planned,
- Λ is the set of predicates that illustrate the failure of a service contrary to its planned fulfillment, in case they hold the service is failed or will certainly fail,

- $s_i \in \mathcal{P}$, is an output, effect, or step of a service, it is described by quantitative and qualitative features such as a service that uploads data.
- $c_i \in \mathcal{C}$, is a set of information associated with run-time s_i . It can be the amount transferred data, current time, underlying network activities related to transfer etc.
- c_i^j is one particular aspect that characterizes the context. For example, c_i^j can be the temporal context of c_i , and c_i^{j+1} can the fact that concerns only the received amount of data,
- $\Delta_{c_i}^{s_i}$ depicts the relation of s_i and its run-time state c_i , $\Delta_{c_i}^{s_i}$ is composed by $\{\Delta_{c_i^0}^{s_i}, \dots, \Delta_{c_i^j}^{s_i}, \Delta_{c_i^{j+1}}^{s_i} \dots\}$, the states that describe the current state depending on single aspects. For example, the completion of the service can be related to received amount of data then the $\Delta_{c_i^{j+1}}^{s_i}$ will characterize the current state of service using the rate or difference of received and total amount of data to transfer. Normally, $\Delta_{c_i^{j+1}}^{s_i}$ will converge to 1 while the received amount of data increases, or will be constant while there is no reception of data.
- Each $\Delta_{c_i^j}^{s_i}$ has a weight in the composition of $\Delta_{c_i}^{s_i}$, we call a weight v_i^j , and v_i , the sum all weights,

Algorithm 1: Monitoring function computation algorithm

$\forall \lambda \in \Lambda, \forall \gamma \in \Gamma, \Delta_{c_i}^{s_i} \in]0, 1[$,

- 1: **while**($\neg \lambda$ **and** $\neg \gamma$),
 - 2: **for**(all running s_i)
 - 3: **for**(all associated c_i^j)
 - 4: $\Delta_{c_i}^{s_i} \leftarrow \Delta_{c_i}^{s_i} + \frac{v_i^j}{v_i} \Delta_{c_i^j}^{s_i}$
 - 5: **end for**
 - 6: **end for**
 - 7: **end while**
-

The above algorithm depicts partially the monitoring function that computes a service s . We chose the part that concerns the analyze of a running or invoked s_i . We do not depict the predicates that illustrate EO as they consist of conditional relationships. This algorithm iterates(1-7) as long as the predicates that illustrate the EO do not hold. For each observed s_i , a $\Delta_{c_i}^{s_i}$ is calculated(4) using all of the different $\Delta_{c_i^j}^{s_i}$ and their weights(3-5). As the algorithm iterates $\Delta_{c_i}^{s_i}$ can have different values over observation interval. Continuous values of $\Delta_{c_i}^{s_i}$ can show the changes of the service since its invocation, its completion, its irregular behavior etc. The analysis made by single s_i s can be gathered to have the IO of the service s in order to make more general analysis.

The operation of the monitoring function is the subject of the process manager. It leaves much room for configuration, the process manager can rely the state of a service to the properties it desires. If the outcome of a service consists of an instantaneous feature, it is hard to model its behavior because one property that can be interpreted is the

response time to an invocation. But its behavior can be modeled in multiple instantiations using the quality of the its operation after each invocation.

4.3 Process Decision Engine

The decision engine uses the output of the monitoring function, it consists of comparing the run-time behavior of the service to its expected behavior. If the running service is not likely to be fulfilled or gives a bad performance, it can be aborted and a better service can be chosen. If the running service gives bad signs and the alternatives are not desirable because of their cost or risk, the running service can be kept. In this paper, we are not concerned by how one can define the expected behavior of a service using dependencies or previous executions, we will study it in the next work. To facilitate the selection of relevant execution plans, we define quality properties for each management decision that can be done.

The set of alternatives . The set $\mathcal{W}_i = \{w_i^1, w_i^2, \dots, w_i^n, \dots\}$ is a finite set of actions that can be done when s_i of a service is enacting. Each element w_i^n of \mathcal{W}_i has: an associated cost ct_i^n , an income g_i^n , a risk r_i^n , and can have an associated set fc_i^n that includes its forward choices. We do not mention the set of \mathcal{W} that includes all sets \mathcal{W}_i for all the whole process that uses the observed service. The management decision consists of choosing one of the alternatives of \mathcal{W}_i including the kept of the existing enactment. As the new services are discovered continuously, or when the number of possible actions decreases over the execution, the cardinality of \mathcal{W}_i can change. The critical points that we have mentioned in 4.1 are actually the points that the elements of \mathcal{W}_i are very limited. In the critical points, the elements of \mathcal{W}_i can be only canceling or keeping the execution. We define a decision function \mathcal{D} that takes a service and the set of management alternatives that can be done during its execution as input, and returns a alternative to execute as output.

Definition 2 (Decision function). *The decision function \mathcal{D} , is a function that takes as input a service and alternatives that can be performed. It returns the best decision that corresponds to its inputs.*

$$\mathcal{D} : \mathcal{S} \times \mathcal{W} \longrightarrow \mathcal{W}$$

As we did in the definition of monitoring function, we define \mathcal{D} partially. The algorithm 2 computes the monitoring result of a service s_i with its alternatives within \mathcal{W}_i and gives the best decision corresponding to the state of s_i .

- $\Delta_{c_i}^{s_i}$ is the result of monitoring analyze done for s_i
- $q_{i c_i}$ is the expected state of s_i in the context c_i . For example, if c_i^j consists of data to transfer then $q_{i c_i^j}$ is the expected amount of data that is supposed to be transferred,
- $q_{i c_i}$ is calculated like $\Delta_{c_i}^{s_i}$, different weights are taken into account, we call ϑ_i^j , the weight of each $q_{i c_i^j}$ and ϑ_i , the sum of all weights,
- m_{s_i} is the depicts the deviation from expected execution,
- \ominus is an operator that returns the difference of an expected execution and current execution,

Algorithm 2 : Management function computation algorithm

```

 $\forall \lambda \in \Lambda, \forall \gamma \in \Gamma, q_{i_{c_i}} \in ]0, 1[$ 
1: while( $\neg \lambda$  and  $\neg \gamma$ )
2:   for(all running  $p_i$ )
3:     for(all associated  $c_i^j$ )
4:        $q_{i_{c_i}} \leftarrow q_{i_{c_i}} + \frac{\vartheta_i^j}{\vartheta_i} q_{i_{c_i}^j}$ 
5:     end for
6:      $m_{s_i} \leftarrow \Delta_{c_i}^{s_i} \ominus q_{i_{c_i}}$ 
7:     if  $m_{s_i} \geq 0$  then keep execution else
8:       review alternatives end if
9:   end for
10: end while

```

In the above algorithm, for each s_i an expected execution is calculated(3-5). The algorithm calls the monitoring function(6) and uses its return value $\Delta_{c_i}^{s_i}$ in order to compare to its expected value. The management policy we used in this algorithm is very basic, if the expected and current state are equal or there is an outstanding performance then the running service is kept(7), else the management alternatives(besides keeping the execution) are reviewed(8). This process can be resulted by the choice of an alternative which is better than keeping the running service. The alternatives are taken into account basic properties such as income, cost, risk and the alternatives that can follow them.

To compare the expected and current state of a running service, we use a special operator of comparison (\ominus) that expresses only concerned differences. The algorithm we used, can be extended easily to express the general behavior of a service, for example the record of differences between expected and current state of all s_i s can be illustrated as its general quality over its use of its sub-processes.

5 A Case Study: Distributed Software Production

This section presents a simple example of collaborative software production process. According to [1], it has been estimated that the software development industry has an 85% failure rate in the development of software products. This means that most of the time, the outcome of a software production does not accomplish its planned results. When the development occurs over the Web, the production becomes more difficult. The processes and entities involving in them should be rigorously managed.

The scenario we use in this paper can be resumed as follows, the software house (SH), a software development agent that produces software for end-users, has customers that place orders including the complete and unambiguous description of the software they require. The SH makes a production planning and dispatches different components and steps of production among its partners(service(s)) that provide corresponding services(functional core development, user interface development, delivery to end-user, payment service ...). The agreement of SH and its customer depicts

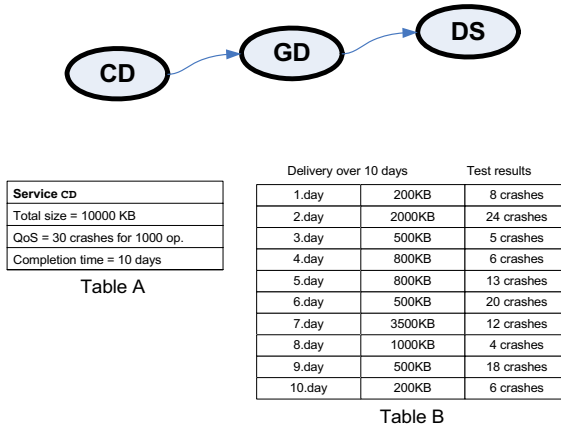


Fig. 1. An example of service composition

the aspects related to quality of required software such as the number of time that the software can crash in a precise period or the execution time of a operation and the features related to classical production process such as product delivery date or payment process. The SH plans an initial production and makes best effort to respect its agreement. During the production process, SH has initiatives. Basically, it can abort a service that does not show its expected performance, it can choose a cheaper service instead of an activated service canceling the latter, or for a risky dependency it can choose providers having good reputation. Due to lack of space, we explain one step of the complete production scenario. The figure 1 depicts three services used in the production, Core Development (CD) service, GUI Development (GD) service, and Delivery (DS) service. There are serial dependencies among services, GD and DS use the outcome of the preceding services. The s_i to observe consists of the production of a software functional core composed by independent components(which has 10MB size), an expected quality(less than 30 crashes for 1000 operations invoked), all of the components must be delivered at most in 10 days. In this example, we suppose the separate components are delivered when their production are completed. The elements we mentioned above, describe the service outcome to observe. The system designer defines a monitoring and management policy for this service. The corresponding IO of this service can be defined by various ways. The completion of service can be associated to the amount received components proportionally to the total expected amount. In this case, the IO of the service converges toward 1 according to the relation of received and total expected amount of components to transfer. In the figure 2, graphic(1) illustrates the IO associated with this relation. As example of the delivery, we used the data of Table B in the figure 1. The implementation of corresponding \mathcal{M} function is the proportion of received and total amount of data. If the system designer lies the IO to this relation then the default output of \mathcal{M} is a certain value close to 0. We can not consider 0 because, it corresponds a service which is not fulfilled as planned and also we can consider 1 as a return value if the service is fulfilled or certain to be fulfilled as planned. In the graphic(3), the system designer that is interested in temporal context

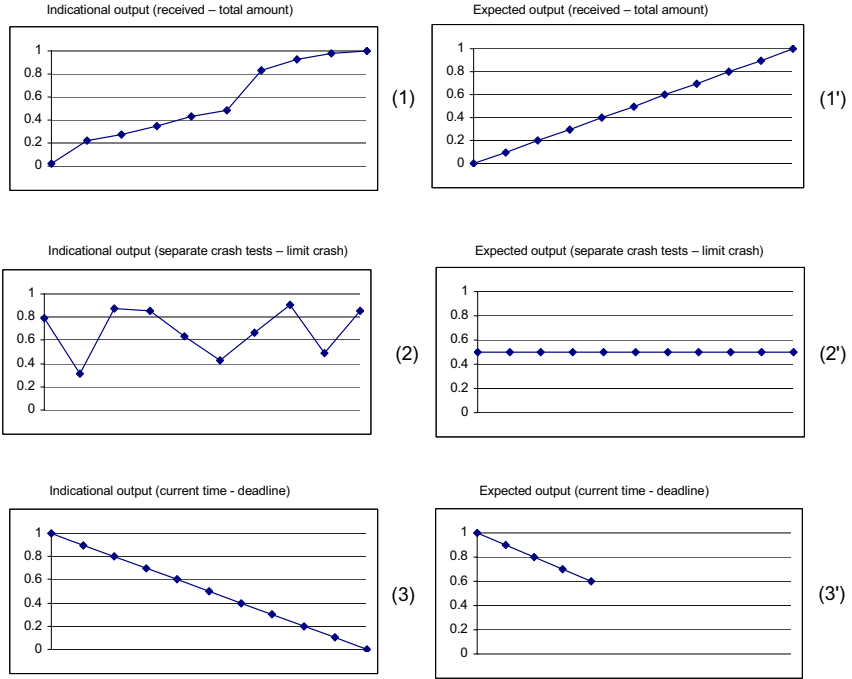


Fig. 2. Examples of monitoring output and corresponding expected states

associates the output to the relation of current time and deadline. Thus, he does not take the other metrics into account, he considers the service approaches to its failure as long as its is not fulfilled completely. In this interpretation, logically, the default state of the service is very close to 1 but its not equal either to 1 or to 0 as long as there is certain information about the fulfillment or failure of the service. In the same figure, the graphic(2) illustrates the state of service according to the number of crashes detected for each delivered component. As 30 crashes characterizes the failure of the service, the output is close to 1 when the number of crashes is not high. In the same figure, the graphic(1') depicts the expected state of the service according to the relation of (1). The intention of system designer consists of the expectation of regular component delivery that will begin with the invocation and end before the deadline. If we compare visually or arithmetically two illustrations, there is no important deviations. In the graphic(2'), the expectation of the system designer consists of a constant quality for each delivered product. In this case, the comparison of the service behavior to expected behavior gives the signs of deviation from expectations. The expectation depicted in graphic(3') can be interpreted as follows, the service is expected to complete its fulfillment before the deadline. In our example, the service we considered ends its fulfillment just before its deadline. As result, at a certain date, the management function considers this service deviates from its expected execution.

The most efficient way of computation is to gather all of separate illustrations in a single illustration. Their importance in the composition can be characterized with the

weights. Actually, the graphics(1, 2, 3) correspond to three $\Delta_{c_i^j}^{s_i}$, the graphic that will fusion them with their weights will be $\Delta_{c_i^j}^{s_i}$. The graphics(1', 2', 3') correspond to three $q_{i c_i^j}$ and respectively their fusion will be the $q_{i c_i}$ of the service.

In the graphics, we do not label the x line, the observation can occur over the time or single events.

6 Related Work

The work of Xu [11] is one of the closest works to ours. The authors consider the collaborative process as an e-contract. The parties have predefined mutual commitments composed by predefined actions series. Within the commitments, the actions are interconnected with temporal relationships that called constraints. The commitments are observed during the enactment, the monitoring party puts guards on constraints, the guards capture how far the commitments have progressed. If there is a non-compliance to prescribed scheme of actions, the concerned parties are notified.

Sayal et al. present in [6] a set of integrated tools that support business and IT users in managing process execution quality. The authors provide a data model and a generic architecture(HP Process Manager) to model and compute the execution of processes. Although the approach is interesting, it suffers heavy analyze of workflow log data.

Cardoso et al. [2] propose a model to manage workflow components from a QoS perspective. The QoS presented includes three dimensions: time, cost, and reliability. The QoS measures are applied separately to each workflow task and then they are automatically computed for the overall QoS of the workflow.

In the literature, there are many examples of Web service composition propositions that rely the behavior of a Web service to its network level activities such as response time to any invocation, availability, the probability of response to any invocation etc. In these approaches, the metrics and events that parameterize the behavior a service do not characterize the process manager perspective. In our work, we are mainly concerned by high level business properties of the processes.

7 Conclusion and Outlook

In this paper, we examined the problem of VE maintenance within the context of WS. In the dynamic context of the Web, each process manager has an individual policy that governs the processes. We proposed a generic mechanism that can be used to express monitoring and management policies computing basic semantic features of services. Our proposition includes a monitoring function that can compute basic relationships among process metrics, events, and objects. The monitoring is processed with measurable and continuous indications of how likely the process is enacted. The management mechanism we propose, uses a similar mechanism to deal with different alternatives. It permits the automated comparison of different alternatives according to their basic properties.

This paper depicts the preliminary concepts our future work. Our aim is to provide a framework that can monitor and verify the run-time behavior of the Web processes and

also manage the composition of Web services according to the management policies. The model we proposed requires a complex data model that will express basic properties of processes and the features that characterize their key performance indicators. The technologies that can support our effort are data warehousing and mining techniques, complex event processing, workflow and business policy management approaches.

References

1. Jones Caper. Minimizing the risks of software development. *Cutter IT Journal*, 11(6), June 1998.
2. Jorge Cardoso, Amit P. Sheth, John A. Miller, Jonathan Arnold, and Krysz Kochut. Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281–308, 2004.
3. Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. *Web Services Description Language (WSDL) 1.1*. W3C, 1.1 edition, March 2001. URL: <http://www.w3c.org/TR/wsdl>.
4. Alexander Keller and Heiko Ludwig. The ws-la framework: Specifying and monitoring service level agreements for web services. *J. Network Syst. Manage.*, 11(1), 2003.
5. David Martin(editor), Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srinu Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. *OWL-S: Semantic Markup for Web Services*, 2004. URL: <http://www.daml.org/services/owl-s/1.1/>.
6. Mehmet Sayal, Akhil Sahai, Vijay Machiraju, and Fabio Casati. Semantic analysis of e-business operations. *J. Network Syst. Manage.*, 11(1), 2003.
7. T.Andrews, F.Curbera, H.Dholakia, Y.Goland, J.Klein, F.Leymann, K.Liu, D.Roller, D.Smith, S.Thatte, I.Trickovic, and S.Weerawarana. Business process execution language for web services. BEA, IBM, Microsoft, SAP, Siebel, 2003.
8. UDDI. *Universal Description, Discovery, and Integration of Business for the Web*, October 2001. URL: <http://www.uddi.org>.
9. Wil M. P. van der Aalst, Boudewijn F. van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.
10. W3C. *Simple Object Access Protocol (SOAP) 1.1*, 2000. URL: <http://www.w3c.org/TR/SOAP>
11. Lai Xu and Manfred A. Jeusfeld. Detecting violators of multi-party contracts. In *CoopIS/DOA/ODBASE (1)*, pages 526–543, 2004.

A Methodology for Process Evaluation and Activity Based Costing in Health Care Supply Chain

Michelle Chabrol¹, Julie Chauvet², Pierre Féniès², and Michel Gourgand¹

LIMOS CNRS UMR 6158,

¹ ISIMA Institut Supérieur d'Informatique, de Modélisation et de leurs Applications,
Campus des Cézeaux, 63173 Aubière, France

² IUP Management et Gestion des Entreprises,
Pôle Tertiaire, 63008 Clermont-Ferrand, France

{chabrol, chauvet, fenies, gourgand}@isima.fr

Abstract. This paper proposes a methodological approach for process evaluation in health care system. This methodology allows conceiving a software environment which is an integrated set of tools and methods organized in order to model and evaluate complex health care system as a Supply Chain. The proposed methodology is applied in New Hospital of Estaing (NHE).

1 Introduction

Health Care Systems can be seen as a Health care Supply Chain (HSC). We define HSC as an opened set, crossed by human, material, informational and financial flows. HSC is composed of autonomous entities (suppliers, hospital departments, logistic services and external medical services...) which use restricted resources (time, material, capital...) and coordinate their actions thanks to an integrated logistic process to improve firstly their collective performance (patient satisfaction) and secondly their individual performance [1]. Because of changes in financing modalities of public hospital, HCS manager has to possess a set of tools and methods capable of helping him in design problems as in piloting problems. It is worth noting that there is a need for a general approach for both HCS modeling and its evaluation which combines data from physical flow, informational flow, and financial flow in one type of software which is more than a global Advanced Planning and Scheduling (APS) [2]. We call Advanced Budgeting and Scheduling (ABS) this type of APS which combines all the flows (physical, financial, informational) and integrates data from the information system. Because of the number of entities, applications, and flows, connections between information system and decision making tools are difficult. Data integration in HSC has a double aspect: interoperability and standardization of data. This integration in HSC consists in sharing and communicating data coming from heterogeneous databases (patient care databases, managerial heterogeneous databases...). In this context, decision models are going to play a crucial role. Their characteristic is that they require data representing several types of flow, like patient, human (other than patients), materials, financial and information. These flows interact between them in various entities of the system at different time and space scales. The purpose of this article is

to propose an approach to evaluate performance in a HSC by discussing connections among all the Supply Chain flows concerned with management science and computer science and to connect decision making software with HSC information system. These flows are mainly physical and financial flows. Judging from the literature, these flows do not always overlap in a Supply Chain. Nevertheless, we believe there are important synergies worth exploring. This paper is organized as follows: in next section, a state of the art about HSC modeling and evaluation is given. A modeling methodology which explains how linking physical and financial flows is presented in section 3. Section 4 is dedicated to a real case study, the New Hospital of Estaing. The case study is introduced to illustrate the modeling methodology processes. Finally, conclusion and future research work are presented.

2 State of the Art

In this section, we present a brief state of the art on the two domains that influence our works: the evaluation systems which combine financial and physical flow for Supply Chain and HSC, and a typology of modeling problem for HSC.

2.1 Supply Chain Physical and Financial Flow Evaluation

In a literature review, [3] show that Activity Based Costing (ABC) system is the best type of cost model for complex system because of its connections with Supply Chain management. The ABC method considers the organization like a set of activities. Each activity represents an interface between used resources and objects of costs. As shown by [4], integrating financial flow and physical flow in Supply Chain management is essential to optimize financial flow. Nevertheless, almost all scientific literature [4] concerning physical flow impact on financial flow deals with Supply Chain network configuration which is a strategic problem.

2.2 Modeling Typologies in Health Care Supply Chain

The HSC manager has to possess a set of tools and methods able to help him in design and in piloting problems. In a generic way, whatever the Supply Chain under study, these problems can be classified according to three temporal levels [5]: *(i)* strategic level, which correspond to design problem and HSC's network's construction; *(ii)* tactical level which corresponds to network's utilization, adequacy means/needs; *(iii)* operational level which relates to HSC's piloting with short-term. These various temporal horizons need different modeling levels for any modeling study realized to bring decision-making tools. It is interesting to couple temporal sight with various possible types of modeling and simulation on HSC. Three approaches [6] allow characterizing a modeling approach by flows thanks to 3 types of modeling: macroscopic, mesoscopic, and microscopic modeling. Macroscopic modeling considers the flow in a complex system as an aggregated phenomenon, whereas microscopic modeling considers individual interactions. Mesoscopic approach incorporates entities in package forms and constitutes an intermediate level between the macroscopic one and the microscopic one. Table 1 shows the coupling between these three approaches with various temporal horizons.

Table 1. The coupling temporal horizons and surrounding areas of modeling

	MACROSCOPIC	MESOSCOPIC	MICROSCOPIC
STRATEGIC	Global design Ex: HSC conception [7]	Process design Ex: HSC operating process conception [8]	Activity design Ex: Redesigning the supply of gasses in a hospital [9]
TACTIC	Network flows configuration Ex: Resources planning for the whole HSC [10]	Process configuration Ex: Resources planning for operating process [11]	Activity configuration Ex: Resources planning for supply of gasses [12]
OPERATIONAL	Network controlling Ex.: Interaction management between entities in HSC [13]	Process controlling Ex: Operating process modification according to emergency situation [14]	Activity controlling Ex: Resources daily adjustment in nursing staff [15]

Various types of problems for HSC modeling are thus characterized in table 1 and are clarified using an example from the literature. These types of problems correspond thus to dedicated problems. A study of approaches presented in table 1 by [16] shows that they are not easily re-usable. In a literature review about HSC modeling, [17] give a state of the art about global methodology for HSC and that many models are only a conceptual network without implementation, therefore very abstract to realize or only design and simulate for a specific problem and hence lack general applicability. To conclude this section, note that there is a need for a general approach for both Supply Chain modeling and its evaluation which combines physical and financial flows thanks to ABC.

3 A Modeling Methodology for Supply Chain Evaluation

[18] propose a methodology called ASDI (Analysis-Specification-Design-Implementation) used for the design and the implementation of modeling, simulation and piloting software environments dedicated to a domain. In this section we propose to use ASDI in order to conceive a modeling methodology for Supply Chain evaluation. In the first paragraph, we present the methodology process. In paragraph two, we give a generic knowledge model for Supply Chain which integrates physical and financial flows. The principle of a model for PProcess EVALuation and Analysis (PREVA) is given in paragraph three.

3.1 Methodology Process

Supply Chain behaviours study may be a complex task. That's why we propose an approach dedicated to the study of such systems. The first part of the methodology (figure 1) is dedicated to the Supply Chain class system. Thanks to the integration of a paradigm view, ontology and epistemology for the modeling expert are integrated in the methodology process. The Generic Domain Knowledge Model is dedicated to the domain. This model is built during Analysis and Specification steps, by identifying the common points of all the systems of the domain according to the entities that they contain and their functioning. During the Design step, the Generic Design Model is built by discovering and integrating, thanks to the Generic Knowledge Model, mathe-

mational or simulation models. The component coding is done during Implementation step in order to build the Software Libraries. This modeling approach constructs a model of a system according to an iterative process (on the right side of Figure 1). This process is divided into four phases (Figure 2):

- (i) analysis and formalizing of data in order to design the knowledge model (the knowledge model of a system is a formalization in a natural or graphic language of the system operation and structure);
- (ii) translation of the knowledge model into an action model (Action model is a descriptive or prescriptive model derived from the software components library for the system under study) using a formalism allowing its exploitation to provide performance criteria;
- (iii) exploitation of the action model to provide performance criteria;
- (iv) interpretation of results and decisions about actions to perform on the system.

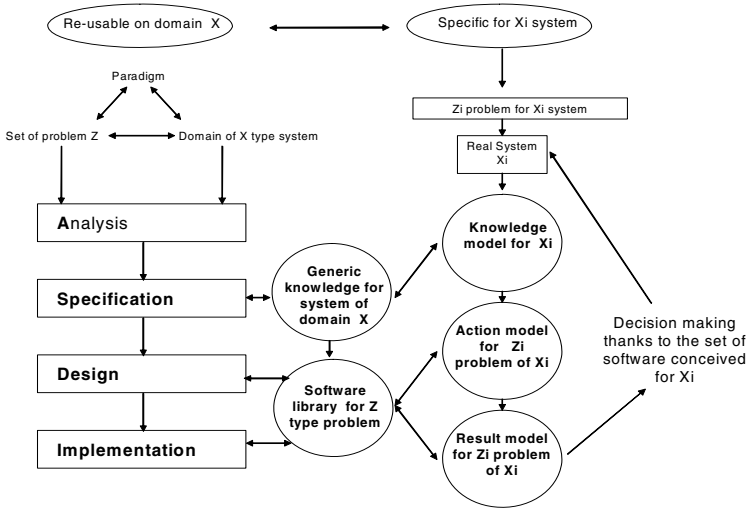


Fig. 1. The proposed methodology

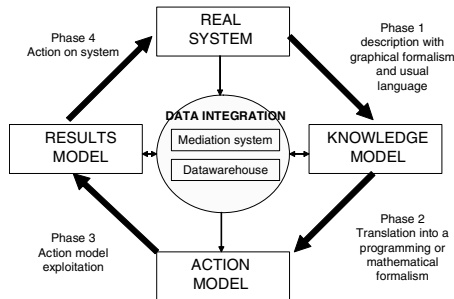


Fig. 2. Towards the integration of data warehouse in the modeling process of a complex system: a crucial link for decision making for a complex system

The experience of modeling process used shows that data collect from many types of applications and many types of information systems is an important need in the modeling process of the system. Therefore, integrating data collect and interoperability between applications put this problem as an important point of the modeling process, as described in figure 2.

3.2 Analysis and Specification for a Generic Supply Chain

In order to formalize knowledge on a Supply Chain, ARIS (which means Architecture of Integrated Information systems) suggested by [19], and UML are used to model processes [20].The global model of a Supply Chain, presented in Figure 3 characterizes the links between the three sub systems and gives an operational and a functional view.

The development of a global knowledge model (Figure 3) proposing a holistic view of a studied domain is essential before building a model of a system, whatever

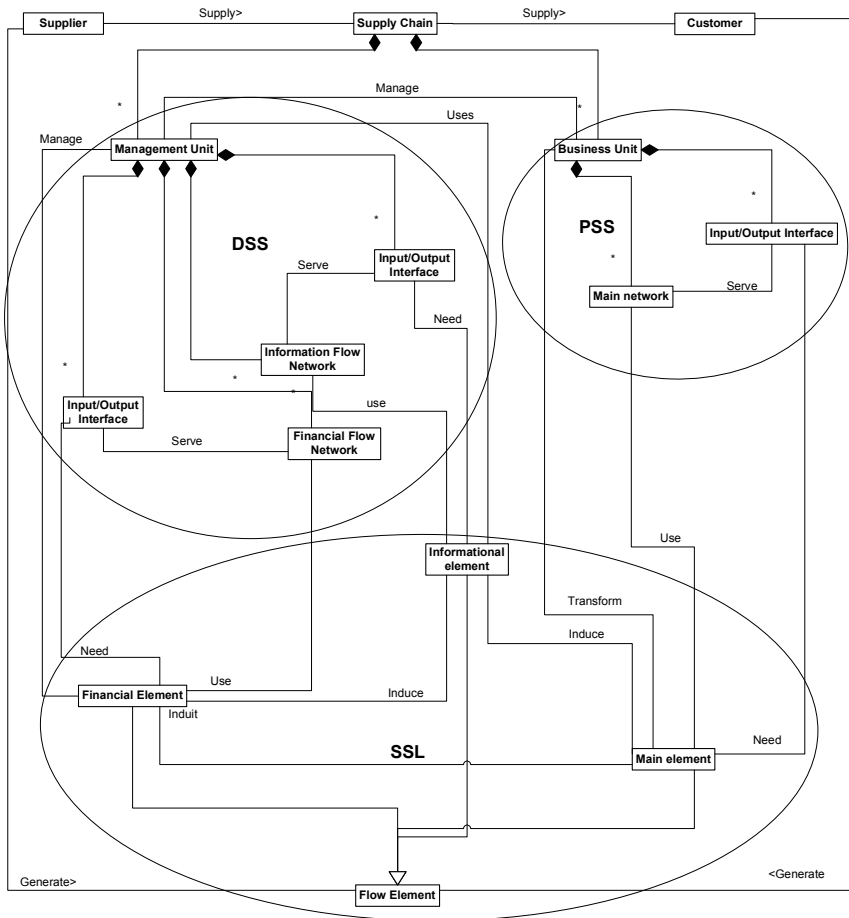


Fig. 3. Generic domain knowledge model for a Supply Chain (Extract)

its type. The used systemic decomposition arises in the following form, for a complex system which is integrated in the Supply Chain, as for the Supply Chain system:

- (i) the Logical Sub System (LSS) contains the transactions (flows) treated by the system;
- (ii) the Physical Sub System (PSS), which is structured in small units (Business Unit) containing physical entities needed for elementary operations;
- (iii) the Decision Sub System (DSS), which is structured in decision center.

The three sub systems are thus complementary and communicating two to two. Supply Chain is considered as a finished number of complex systems which are structured with the systemic decomposition suggested by ASDI.

The system domain analysis enables us to distinguish three families of flows in LSS: the principal flow or main flow (customer flow, patient flow...), the information flow which includes any type of flows providing with the information on the entities circulating in the system, and the financial flow. Supply Chain is represented like a set of entities or nodes which are interrelated and its activities consist to treat, transform and store the flow elements. The PSS includes Business unit (a sort of business unit), which are consisted of the main network, (Logistic network...) while DSS includes the decision-making centers and the management units which are consisted of the financial and informational networks.

3.3 PREVA: A Generic Approach for Process Evaluation

An evaluation model is conceived thanks to the generic knowledge model and the modeling objectives. Our approach gives to the Supply Chain manager the possibility to evaluate the impact on Supply Chain financial flow of different Supply Chain physical flow decisions, and to choose between strategies the best one. We call it PProcess EVALuation (PREVA). Action model is derived from the knowledge model. Mathematical approach, simulation or controlling tools can be used. Heuristics with financial and physical aspects are built in order to improve Supply Chain working thanks to knowledge model and Supply Chain objectives. The evaluation is done in three steps:

- **Step 1 Physical process evaluation:** Action models give physical data about the physical flow. From these data a global evaluation is built.
- **Step 2 ABC (Activity Based Costing) evaluation:** Our generic approach must be able to determine indirect costs consumption thanks to process cost evaluation on each business unit and for the global complex system. Between each business unit, transactions are evaluated in transfer price or in market price. Differences between Activited Based Costing and transfer price /or market price give managers the possibility to evaluate value creation (a sort of profit level) in each system entity.
- **Step 3 Financial process evaluation:** Thanks to ABC evaluation, direct and indirect resources consumptions and net sales are determined. Because of the nature of the cost (calculated cost and real cost) and because of the payment term which is different between each type of resources and each type of customers (patients...), there is a difference, in medium term, or in short term between profit level and cash flow in the same period. Therefore, preceding periods have an impact on actual period in cash flow evaluation. Our model proposes to integrate cash flow thanks to resources consumption evaluation in ABC. We are able to evaluate cash position and cash flow in each entity of the chain and for the global system.

A formalization of PREVA for Supply Chain evaluation is given in [3]. Thanks to resources consumption in ABC and resources payment, a bridge is done between physical flows decision impact on financial flows (Figure 4).

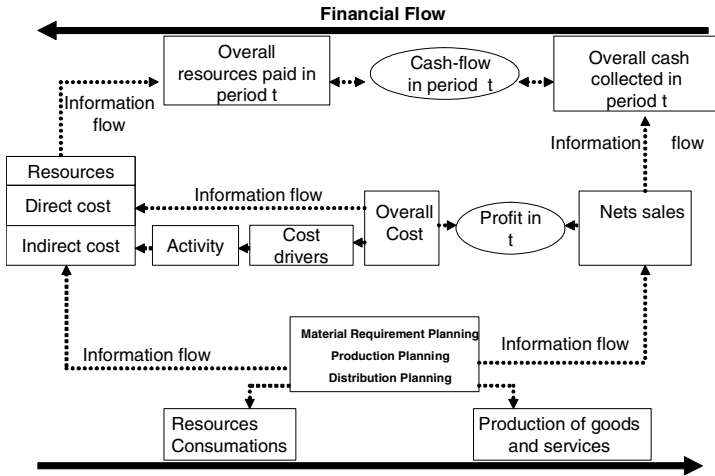


Fig. 4. PREVA: A bridge between financial and physical flow

In next section, we give an example of the methodology implementation for a Supply Chain in a real case study.

4 Methodology Improvement: Knowledge Model and Case Study in New Hospital of Estaing

The reengineering of Clermont-Ferrand public health care system implies to transfer an old hospital (Hôtel Dieu - HD) on a new site: Estaing’s site. This project needs the construction of a new building called New Hospital of Estaing (NHE). This project mobilizes all hospital’s actors. All these professionals of health think about a new organization which will be more adapted to their needs. The objective of this change is the passage of a structure divided up to an independent units’ system registered in a pool with average outbuildings to reach objectives centered on patient. By the same way, hospital managers want to conceive a new information system to help them in operational, tactical and strategic decision making. In order to build this software tool, ASDI is used. The time horizon of this project is about 4 years. In this section, we only present the first part of the methodology, which consist in Knowledge Model Formalization. Knowledge formalization built with ASDI is given in paragraph 1 and Knowledge model using is given in paragraph 2.

4.1 Knowledge Formalization with ASDI

Inquiries are used to collect information and elaborate statistical data in order to build the knowledge model. This problem needs the interrogation of 60 persons (of medical and paramedical staff). This set of persons is representative. This interrogation allows

collecting knowledge following various sights of structure: medical team has a strategic approach while paramedical has an operational vision. Thanks to this collect, ASDI process uses ARIS for knowledge model's building. From macroscopic, mesoscopic and microscopic approaches organizing structure, we ordered current system's structure according to 8 descriptive levels (figure 5).

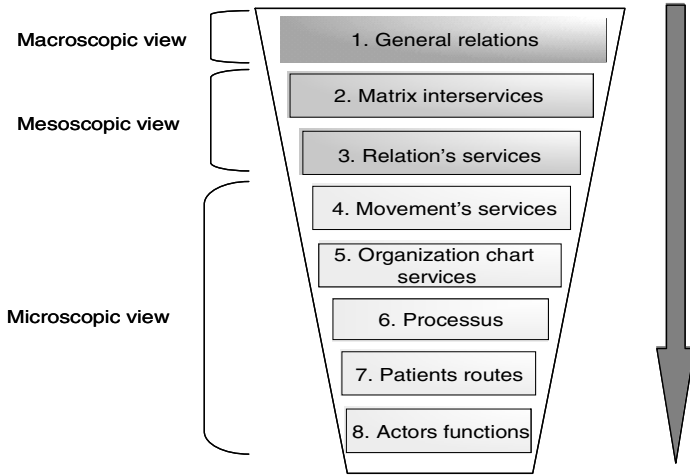


Fig. 5. Descriptive levels of modeling in NHE

These levels give a simplified vision of the system's structure. It is important to choose models types affected to levels. Every type of model is composed of its own objects types and relations types. These models give diagrams of internal, external flows, horizontal, vertical views of system. They constitute a strategic tool for a hospital organization's description. Note that, how if they've been defined thanks to a query in HD hospital, these level are generic for each HSC. For each type of level, a different ARIS formalism and scheme is used. For example, for level 6 and 7, Process Chain is used. In the construction of knowledge model, knowledge collection was done according to competences pole to obtain an analysis basis to help hospital management. Managers, in using knowledge model constructed from the HD, can modify it directly from the perspective of a projection in NHE. Consequently, knowledge model allows having a NHE management vision before construction begins. Thus, an anticipation of different problems such as material and human flows can be approached with this knowledge model. For the time being, a very detailed analysis on three different flows (physical, financial and physical flows) has been done from existing data sources of hospitals to be integrated into decision making tools (ABS). 35 different software applications are connected with 35 heterogeneous database, and needs to be connected in a datawarehouse in order to transfer data for the future ABS.

4.2 Knowledge Model Using with Process Evaluation in Public Health Care

Thanks to the knowledge model of the HD, we are able to evaluate each process of the NHE LSS in order to help hospital managers in the new hospital configuration. An

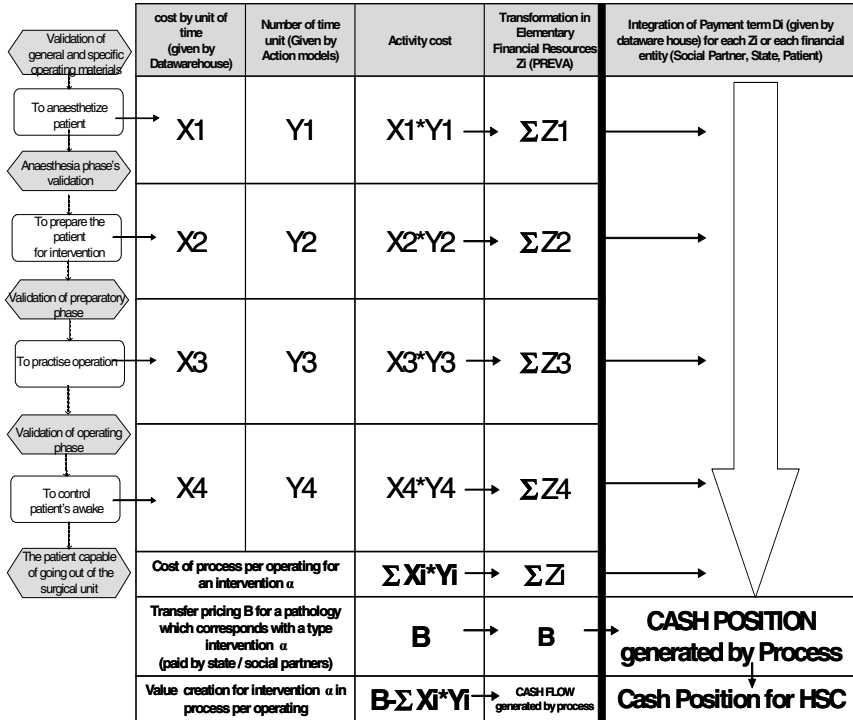


Fig. 6. Towards evaluation of Per-Operatory process with PREVA in NHE LSS

example of the value evaluation in NHE per-operative process thanks to the proposed methodology is given in figure 6.

Figure 6 shows the links between future performance of financial flow because of physical flow impact. Macroscopic dataflow interaction between physical and financial flow are presented; connections between action model and data flow are also shown.

5 Conclusion

In this paper, we have proposed a methodology for Supply Chain modeling and process evaluation. This paper has described how object-oriented modeling can provide a new way of managing and designing complex systems with financial, physical and informational flow. The proposed methodology, which is a bridge between computer sciences, management sciences and operational research, gives a solution to conceive information system. The proposed approach provides more than an ABC system. Of course, it improves the visibility of cost but the most important point in our approach is to show how physical flow impact is passed down to financial flow. In further research we want to integrate ABC and cash flow optimization in action model in order to improve Health Care Supply Chain working and to apply it in NHE.

References

1. Fénies, P., Gourgand, M., Tchernev, N.: Une contribution à la mesure de la performance dans la supply chain hospitalière : L'exemple du processus opératoire. In 2ème conférence francophone en Gestion et Ingénierie de Systèmes Hospitaliers (GISEH), Mons (2004).
2. Stadler H., Kilger C.: Supply chain Management and Advanced Planning, Springer, (2001)
3. Comelli, M., Fenies, P., Gourgand, M., Tchernev, N: A generic evaluation model for cash flow and activity based costing in a company supply chain. accepted at International Conference on Industrial Engineering and Systems Management IESM (2005)
4. Vidal, C.J., Goetschackx, M.: A global Supply Chain model with transfer pricing and transportation cost allocation, *European Journal of Operational Research*, Vol. 129. (2001)
5. Ballou, R.: *Business Logistics Management*, Prentice-Hall Inc Englewood Cliffs, New Jersey (1992)
6. Chabrol, M., Sarramia, D. : Modélisation orientée objets et multi agents du système d'information des systèmes de trafic urbain. INFORSID, Genève (2001).
7. Abouïssa, H., Nicolas, J.C., Benasser, A., Cherkouk, N. : Systèmes Multi Agent et réseaux de Petri pour la modélisation et l'Evaluation des Performance des Systèmes hospitaliers, In 1ère conférence francophone en Gestion et Ingénierie de Systèmes Hospitaliers (GISEH). Lyon (2003)
8. Rossetti, M.D., Selandari, F.: Multi-objective analysis of hospital delivery systems. *Computers and Industrial Engineering* 41 (2001)
9. Van Donk, P.D.: Redesigning the supply of gasses in a hospital. *Journal of purchasing and supply chain management* 9 (2003)
10. Brigl, B., Ammenwerth, E., Dujat, C. Gräber, S., Grosse, A., Häber, A., Jostes, C., Winter, A.: Preparing strategic information management plans for hospitals: a practical guideline SIM plans for hospital: a guideline. *International Journal of Medical Informatics*. (2004)
11. Artiba, A., Briquet, M., Colin, J., Dontaine, A., Gourc, D., Pourcel, C., Stock, R.: Modélisation d'établissement de santé. In 2ème conférence francophone en Gestion et Ingénierie de Systèmes Hospitaliers (GISEH). Mons (2004)
12. Syi Su ScD., Chung Liang Shih M.D Resource Reallocation in an Emergency Medical Service System Using Computer Simulation. *American Journal of Emergency Medicine*, Vol 20. (2002)
13. Lanzola, G., Gatti, L., Falasconi, S., Stefanelli, M.: A framework for building cooperative software agents in medical application». *Artificial Intelligence in Medicine* (1999)
14. Doheny, J.G., and J.L., Fraser: MOBEDIC – A decision modeling tool for emergency situation. *Expert system with applications*, vol 10. (1996)
15. Bard, J.F., Purnomo, H.D.: Preference scheduling for nurses using column generation. *EJOR* 164 (2005)
16. Chauvet, J., Fénies, P., Chabrol, M., Gourgand, M: The New Hospital of Estaing Knowledge model: an operational tool for strategic management and flow modelling in a hospital supply chain, 4th International Conference on the Management of Healthcare & Medical Technology. Aalborg Danemark (2005)
17. Chabrol M., Fenies P., Gourgand M., and Tchernev N. : Un environnement de modélisation pour la Supply Chain : application au Nouvel Hôpital d'Estaing. Grenoble INFORSID (2005)
18. Gourgand, M., Kellert, P.: Conception d'un environnement de modélisation des systèmes de production. 3ème congrès international de génie industriel, Tours (1991)
19. Scheer, A.W.: *ARIS– Business Process Modelling*, Springer (2002)
20. Green, P., Roseman, M.: Integrated Process Modelling: an ontological evaluation. In *Information systems*, Vol. 25. (2000)

Web Services Experiment Using TravelXML: Standard XML for Electronic Commerce in the Travel Industry

Michiko Oba¹, Norikazu Matsuyama², Kojiro Nakayama³, and Norihisa Komoda⁴

¹ Software Division, Hitachi Ltd. Omori Bellport A Bldg. 26-2,
Minamioi 6-chome, Shinagawa-ku, Tokyo 140-8573, Japan
michiko.oba.cq@hitachi.com

² Product Development Division, PFU Active Labs Limited, Nu-98-2,
Uno-ke, Kahoku-shi, Ishikawa 929-1192, Japan
matsuyama.nori@pfu.fujitsu.com

³ Systems Development Laboratory, Hitachi Ltd. Ozenji 1099,
Asao-ku, Kawasaki 215-0013, Japan
kojiro@sdl.hitachi.co.jp

⁴ Dept. of Multimedia Engineering, Graduate School of Information Science
and Technologies, Osaka University, 2-1, Yamada-oka, Suita 565-0871, Japan
komoda@ist.eng.osaka-u.ac.jp

Abstract. The travel industry requires computerized standardization of the information exchanged between travel agents, transportation systems, and accommodation facilities. Consequently, the Japan Association of Travel Agents and the XML Consortium are currently engaged in efforts to standardize an XML format (called TravelXML) for electronic commerce in the travel industry. A new business model for the travel industry was recently proposed and tested by the XML Consortium, using TravelXML and web services. 15 XML Consortium company members participated in the experiment. In this paper, we show the results of applying TravelXML and web services to a real business model through the experiment.

1 Introduction

XML is the standard format for business documentation and data exchange. Moreover, XML is becoming an indispensable technology for electronic commerce between enterprises and electronic applications, in conjunction with web services. It is hoped that information relating to trade between travel agents, transportation companies, hotels, and various service establishments can be put into an electronic form shareable throughout the travel industry.

The Japan Association of Travel Agents (JATA)[1] and XML Consortium [2] are developing an XML standard, TravelXML[3], for electronic commerce relating to the travel industry. We performed a large-scale experiment using TravelXML and web services technology [4], and we developed a model for electronic commerce in the travel industry. 15 companies participated in the experiment.

This report describes the results of applying TravelXML and web services to a real business model.

2 A Business Model for the Travel Industry

In the travel industry, electronic data interchange (EDI) has been used from the early stages of electronic communication. However, since each party uses a data format that is unique to its system, a standard data format has been desired for a long time. In Europe and in the United States, the Open Travel Alliance (OTA) [5] has developed an XML-based specification for individual travel bookings. However, in Japan, package tour travel is more popular than individual travel, making the OTA specification not so suitable for the local industry. TravelXML was developed by JATA and the XML Consortium, which is an organization set up to make XML-related technology widespread in Japan. It defines data formats for package tour bookings.

A business model for the travel industry in Japan is shown in Figure 1. Several entities are involved in the process of a package tour booking. Wholesalers have resources in stock, such as rooms from hotels, airplane seats from airline companies, and train tickets from railway companies. Wholesalers plan package tours combining these resources, and then commission retailers sell the tours. Travelers make travel arrangements through retailers.

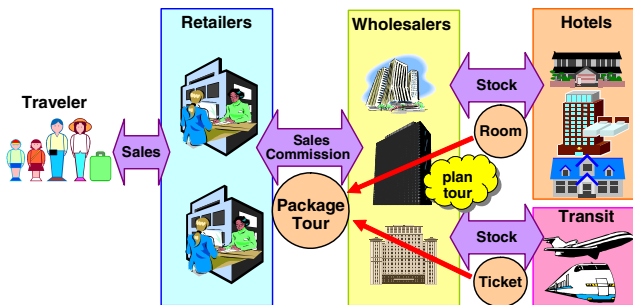


Fig. 1. Business model for the travel industry

3 Details of the Experiment

3.1 Demonstration Scenario

The demonstration scenario is as follows:

1. A prospective traveler visits a travel retailer. A sales clerk selects a suitable package tour according to the traveler’s requirements.
2. The retailer sends a “Booking Request” message to a wholesaler.
3. The wholesaler receives the “Booking Request” message and updates the number of rooms in stock.
4. The wholesaler sends a “Booking Report” message to a hotel.

- The hotel receives the “Booking Report” message and updates the booking status.

3.2 Method of Processing

In the experiment, the business tasks for reserving the package tours were executed by the following processing methods:

- Synchronous type: This processing includes all automatic forms of processing.
- Asynchronous type: This processing requires human decision-making.

3.2.1 Synchronous Type Sequence

The following is an overview of a synchronous type sequence.

After receiving a request from a retailer, the wholesaler A checks whether it has a hotel room in stock. If a room is in stock, a reservation is automatically made and notified to the retailer. If there is no room in stock, a request for an extra room is sent to a hotel X. When the request for an extra room is received from the wholesaler, the hotel X checks for availability. In the experiment, scenarios for room availability and non-availability were tested. Figure 2 shows the processing sequence when a room is not available.

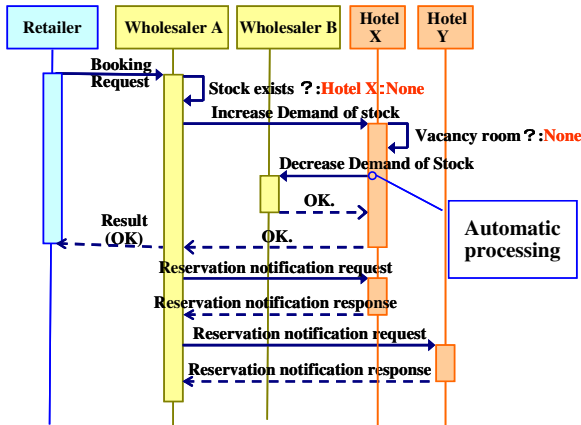


Fig. 2. Sequence of the synchronous type

3.2.2 Achievement of Security

In an actual travel reservation, personal information like credit card details is often processed. Achieving security for the information in high volume transactions is a key goal. This was achieved by partial encryption/signature technology that used WS-Security. In the experiment, WS-Security was processed with products that included hardware (XML Security Gateway (DataPower XS40[6])) and software (WS-Security handler implemented by Apache XML Security[7]).

4 Experiment Results and Considerations

4.1 Proposed Model

In this experiment, both asynchronous and synchronous modes of processing were tested. The asynchronous mode is similar to the conventional human-based processing. Answering takes a lot of time and flexible stock adjustment is also difficult. On the other hand, synchronous processing is real-time and does not need human input. The result is business efficiency. The method also achieves a dynamic adjustment of stocked resources so that inventories can be minimized.

For web services, there are disadvantages for both processing types. Asynchronous systems feature lengthy transactions, while synchronous systems need policy definition to automate stock adjustment. However, these problems will be solved as standardization related to web services progresses.

4.2 Implementation of Security

In the experiment, end-to-end security was achieved through WS-Security. By mounting WS-Security in all the systems, interconnectivity was managed successfully even though various software and hardware platforms were used.

There were a few difficulties however. In this experiment, interoperability could not be secured when WS-Security versions were different because the name space URIs used were also different. This problem was resolved by executing processing that could absorb the various specification differences among the versions.

4.3 Development

Table 1 shows our development profile. It took three months of planning and examination of specifications, one month for implementation, and two months for execution of the connection tests (executed nine times). In this experiment, worker-hours were required for the normal tasks of examining the specifications and connection tests. However, in this experiment two extra factors were added to the required time:

- (1) Extra time was needed to adjust and match the specifications due to the high number of participating companies.
- (2) The time needed to verify items and execute connection tests was longer than usual because of the high number of participating company combinations.

Ten kinds of tools related to web services were used. Interconnectivity between different platforms was confirmed in the experiment.

We think that the reason for the achievement of such short implementation cycles is the use of a standard data format (TravelXML) and a standard communications protocol (Web services). Using these two techniques enabled each development task to be isolated and implemented effectively.

At last, we found 5 TravelXML specification bugs through the experiment. The reasons are vocabulary luck, unmatched vocabulary and so on.

Table 1. Development Profile

Implementation	6 months		
Developer	37 persons		
Joint Company	15 Companies		
Person power	Total 232 person days	Retailers(3 Clients)	41 person days
		Wholesalers(5 Servers)	61 person days
		Hotels(5 clients)	94 person days
		UDDI,Security,Monitor	36 person days
Program Steps	Total 90 K Steps	Retailers(3 Clients)	20.7 K Steps
		Wholesalers(4 Servers)	26.8 K Steps
		Hotels(5 clients)	29.8 K Steps
		UDDI,Security,Monitor	12.8 K Steps
Total Test	9 times, 2 months		
Products	Total 20 Products	Application Server & Web Services	10 Products
		Relational Database	5 Products
		Application Package	1 Product
		Security	4 Products

5 Conclusion

Using TravelXML, we conducted a demonstration experiment of a web service system. We succeeded in achieving standardization, improving efficiency in the travel business, and providing end-to-end security. The experiment confirmed the effectiveness of XML, web services, and WS-Security.

TravelXML format is applied to several actual hotel reservation systems. JATA and XML consortium are now intending to cover the international travel business in cooperation with OTA (open travel alliance). Web service is still a candidate of implementation techniques for electronic commerce in the travel industry.

References

1. Japan Association of Travel Agents (JATA), <http://www.jata-net.or.jp/english/>
2. XML Consortium, <http://www.xmlconsortium.org/introduce/index-e.html>
3. XML Consortium, TravelXML specification (in Japanese), http://www.xmlconsortium.org/wg/TravelXML/data/TravelXML1_1_1-Rec.pdf
4. XML Consortium, Project of Web Services Demonstration Experiment using TravelXML, Result Data (in Japanese)
5. Open Travel Alliance, <http://www.opentravel.org/>
6. DataPower XS40 XML Security Gateway, <http://www.datapower.com/products/xs40.html>
7. Apache XML Security, <http://xml.apache.org/security/>

Data Level Enterprise Applications Integration

Marko Vujasinovic¹ and Zoran Marjanovic²

¹ Breza Software Engineering Company,
Kraljice Natalije 23a, 11000 Belgrade, Serbia & Montenegro
marko.vujasinovic@brezasoftware.com
www.brezasoftware.com

² University of Belgrade, Faculty of Organizational Sciences, IT Department,
Jove Ilica 154, 11000 Belgrade, Serbia & Montenegro
zoran.marjanovic@fon.bg.ac.yu

Abstract. Information integration across heterogeneous systems is a key issue for successful enterprise application systems development. Particularly challenging is integration of different applications deployed on different platforms into one integrated and stable business information system. This paper summarizes results of resolving a real enterprise applications integration problem during an ERP system implementation. The solution resolves the integration problem by using the data level integration approach. The solution is general but only its core functionality is presented in this paper.

1 Introduction

In a modern business environment, cooperating companies need to integrate different enterprise applications into one stable and integrated information system. In a business-to-business (B2B) relationship there are at least two independent systems that need to work together and communicate in order to achieve a business goal. The main challenge is how to achieve interoperable data exchange between such two systems.

Interaction between two systems can take place at least at two levels: data level and business process level. For data level integration, systems have to be able to read data from each other or they must be able to receive data in some known format, to interpret data, and to call an appropriate business service. The most popular approach for data exchange is to exchange textual data structures using Extensible Markup Language (XML). Although independent syntax of data exchange is available, the problem of recognizing and validating data semantics still needs to be resolved. A domain ontology provides semantic description that is used for formal definition of concepts in a given domain. The Web Ontology Language (OWL) [1] is a language that enables representation of data semantics in a formal way as well as automated reasoning that follows from the data semantics. A business process level of interaction

between systems can be achieved using Service-oriented architecture (SOA) [3] and Web Services technologies that use several related technologies (i.e., SOAP, XSDL, UDDI) [4].

At both levels of interaction, in use are platform independent and free technologies because an interoperable solution requires technology independence and portability. The data level integration is the strongest way of systems integration, but it can be platform depended solution. The crucial requirements for integration solutions are platform independence, easy way of interaction between systems, and mediatory role between different vendor applications that are realized using different technologies and that work on different platforms.

We experienced real data level integration problem and the goal was to make good solution that has required features.

In section 2, we present a real integration problem and the reasons for choosing data level integration. Section 3 presents our solution for the EAI problem which is based on free and platform independent technologies, XML, and Java programming language. Finally, section 4 presents concluding remarks and directions for our future work in the EAI field.

2 Integration Challenge

Breza ERP is a well-positioned local integrated business software solution for enterprise resource planning. Like the well known global international ERP systems, such as SAP, Navision or Oracle eBS [5], Breza ERP contains modules to support all business activities of a company: financial, inventory management, purchasing, payroll, sales, financial accountancy, human resource, and other modules. The system is realized using the most recent Oracle Java XML n-tier development environment using J2EE technologies. In the system, there are clearly defined data integration structures and interfaces for each module. Thanks to the well-defined module interfaces and modern J2EE technology, the system is open for integration with other business environment subjects (e.g., banks, others ERP software, etc).

Recently, we received a request from a large international company with local presence to implement and integrate the Payroll and HR (human resources) Breza ERP modules. The company employs locally more than 10.000 employees located in three different cities. The head office of the company is in the USA and the main European office is in Slovakia. The two selected modules need to take into account the local legal system and for that reason were acquired to be integrated with the company's own ERP system. During implementation of the modules (Payroll and HR), it was necessary to acquire licences of other three software systems that are made by different producers:

- Application for employees in/out time evidence (application uses Microsoft SQL Server database)
- Oracle Financial application for earnings accountancy (application is deployed in Slovakia)
- Banking applications for payment transfer orders processing (there were several banks involved with different data formats).

Figure below depicts the system architecture;

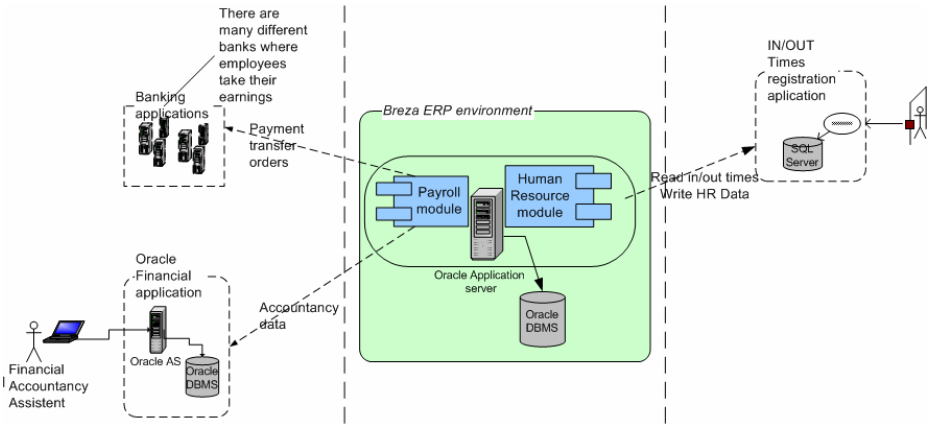


Fig. 1. System architecture

There were several reasons why we decided to make data level integration: (1) existence of many relationships with non Breza ERP environment; As mentioned before, one of the main request was that Breza ERP need to work together and communicate with other software systems which was already in use in company. Each of them had already defined data integration structures (2) each bank had its own data format for payment transfer order; All what banking applications needed was textual files. There wasn't requirement for XML data exchange or for using OWL and automated reasoning. (3) there was not a Web service for accountancy or for payment transfer orders processing; According to this there was no need for making SOA integration approach. (4) in/out times registration application wasn't able to save data in a single Oracle database; For using in/out time evidence it was necessary to make application which is able to read and write data from one datasource to other. Not to make application for calling business operations owned by other software system, just application which is able to read, transfer, and write in/out time evidence, periodically at defined interval of time. And the one of main reason is (5) data integration is stable way for enterprise application integration.

3 Our Solution

During the problem analysis stage, we identified a number of requirements for our solution: to resolve data level integration through either the database or through the file system approach and to ensure platform independence, easy maintenance, simplicity of changing rules and data sources, greater formality of the model, and greater declarative management of the roles.. There need to be two elements of the solution;

- Declarative definition of data source type, data destination type, mappings between fields, field's data types, and formats.

- A component which will use this definition to read data from data source (database, memory, etc) and write data to appropriate destination (database, textual file, XML file).

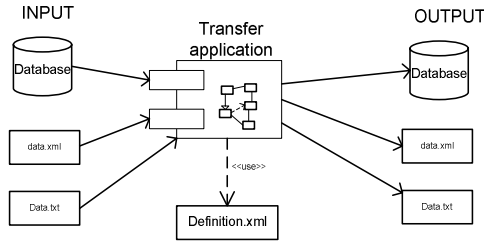


Fig. 2. Component model

We decided to use Extensible Markup Language (XML) to achieve declarative description for the component (application) behavior during the transfer of data from one data source to the other (i.e. from one database to the other, or from database to adequate file format). In further text we will use term “definition file” for this declarative description. First of all, definition file needs to be strictly and formally described because transfer application that uses the definition file expects an exactly defined format. The meta-model for the definition file is shown in figure below: that is the object model that transfer application created after definition file had been loaded. After creation of the object model, the rest of the classes have the responsibility to accomplish data transfer.

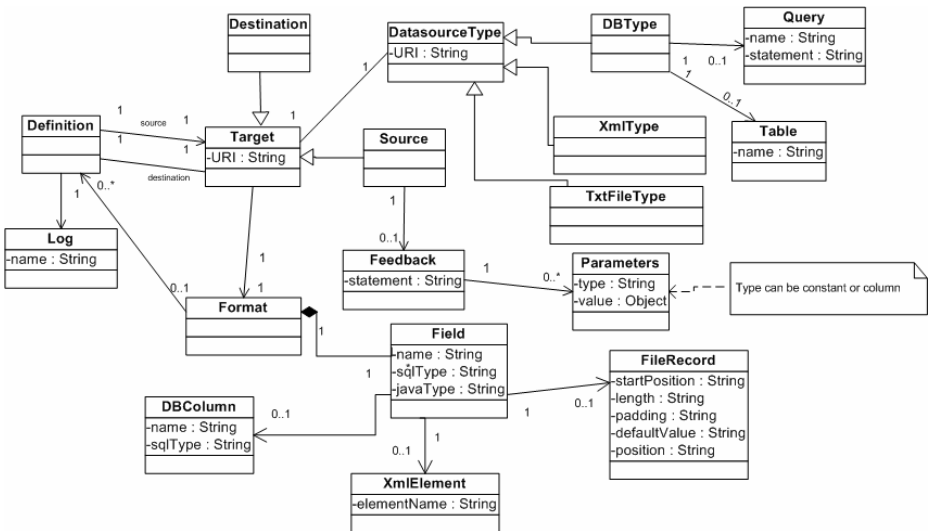


Fig. 3. Definition file meta-model

The *Definition* class is the basic point and that class contains two targets: source and destination. *Source* class defines data source from which system will read the data. If we use a database table for the source, then *Source* have association with *DBType* class which have defined an SQL statement. *Destination* class also has a data source type that represents the destination type. It can be a table in the second database (*DBType*), a textual file (*TxtFileType*), or an xml file (*XmlType*). In this case, *DBType* contains *Table* class that can give information about the specific table where the component inserts the data. *Destination* class has *Format* which describes mappings between retrieved columns from Source to appropriate destination table column or textual file or even an element. If we want to make master-details relationship for export data in an XML file, then *Format* class can be associated with a new *Definition* class.

Using this model, we can derive Definition.xml file. In tables below there are two examples of definition.xml files.

After the definition file is completed, the transfer application parses this file by using SAX and DOM technology and creates an object model.

<i>Describe In/out data transfer</i>	<i>Describe payment transfer orders export</i>
<pre> definition> <source URI="jdbc:microsoft:sqlserver://10.21.46.200\ITPR:1433; DatabaseName=InfoTime Transfer;User=sa;Password=br123"> <dataSourceType>DBType</dataSourceType> <SQLQuery name="IN/OUT time data"> select id,regtime,direction, regloc, typeid, READSTATUS from it_reg where readStatus=0 and typeid='E' order by regtime </SQLQuery> <feedback statement="update it_reg set READSTATUS=? where ID=? and REGTIME=? and DIRECTION=? and REGLOC=? and TYPEID=?"> <parameter type="constant" value="1"> </parameter> <parameter type="column" value="id"> </parameter> <parameter type="column" value="typeid"> </parameter> </feedback> </source> <destination URI="jdbc:oracle:thin:@dbhost:1521:dbsid;username;password"> <dataSourceType>DBType</dataSourceType> <table name="GATE"> </table> <format> <field name="id" SQLType="NUMERIC" JavaType="java.math.BigDecimal"> <dbcolumn name="employeeid" sqltype="NUMERIC"> </dbcolumn> </field> <field name="regtime" SQLType="DATE" JavaType="java.sql.Date"> <dbcolumn name="time" sqltype="DATE"> </dbcolumn> </field> <field name="direction" SQLType="VARCHAR" JavaType="java.lang.String"> <dbcolumn name="direction" sqltype="VARCHAR"> </dbcolumn> </field> <field name="typeid" SQLType="VARCHAR" JavaType="java.lang.String"> <dbcolumn name="typeid" sqltype="VARCHAR"> </dbcolumn> </field> </format> </destination> <log filename="c:\log\transfer.log"> </log> </definition> </pre>	<pre> <definition> <source URI="jdbc:oracle:thin:@dbhost:1521:dbsid;usernam e; password"> <dataSourceType>DBType</dataSourceType> <SQLQuery name="PAYMENT_TRANSFER_ORDER"> select ppid, date,place,nameN,accountN,callNoN, nameM,accountM,calNom, payCode,purpose,amount from payment_t_order where ppid='A' </SQLQuery> </source> <destination URI="c:\paymentTransferOrders\YUBank\pto.txt"> <dataSourceType>TxtFileType</dataSourceType> <format> <field name="ppid" SQLType="NUMERIC" JavaType="java.math.BigDecimal"> <filerecord startPosition="1" length="3" defaultValue="X" position="10" padding=" "> </filerecord> </field> <field name="date" SQLType="DATE" JavaType="java.sql.Date"> <filerecord startPosition="4" length="6" position="10"> </filerecord> </field> <field name="place" SQLType="VARCHAR" JavaType="java.lang.String"> <filerecord startPosition="10" length="12" position="10" padding=" "> </filerecord> </field> </format> </destination> <log filename="c:\log\dataexport.log"> </log> </definition> </pre>

We have not implemented functionality for the transfer data from textual file or XML file to database because there are already several tools and technologies that integrators can use for this purpose (Oracle SQL loader, XSLT transformation, etc). However, this functionality can be added to the model and does not require much effort.

4 Conclusion

This paper has provided an experience report in resolving an enterprise application integration problem. The presented solution does not cover all possible combination of data exchange or data sources; however, we are working on creation of a full generic solution that does not require any adjustments at all. Our current solution provides basics for the most frequent data level integration approach: through a database and thru an ASCII file system.

If we assume that interoperability is “ability of two or more systems or components to exchange information and to use information that has been exchanged [2]“ then we can give conclusion that our solution complies with the definition.

Our future work will focus on defining specific data level integration model for ERP software owners to achieve communication between their product and external environment applications. The main work area will be adding more elements to the previously defined model to achieve more functionality and for a greater number of data level integration combinations. We hope that our final result will be a formal language for describing data level integration, and component/application execution based on the language.

References

1. OWL Web Ontology Language Guide, 2004, <http://www.w3.org/TR/owl-guide/>, August 2004
2. IEEE (1990) Institute of Electrical and Electronics Engineering; Standard computer Dictionary – A Compilation of IEEE Standard Computer Glossaries
3. Hao He; What is Service-oriented architecture <http://www.xml.com/pub/a/ws/2003/09/30/soa.html>
4. IBM Web Services Architecture Team: Web Services Architecture Overview. <http://www-106.ibm.com/developerworks/library/w-ovr/>, September 2000
5. <http://www.oracle.com/applications/>, <http://www.sap.com>
6. Abraham Kang; Enterprise Application Integration using J2EE, <http://www.javaworld.com>, August 2005.
7. Steve McClure; Oracle's Solution for Heterogeneous Data Integration, August 2003, Oracle Corporation
8. Robert Worden, Four Levels of Data Integration, March 2005
9. Nenad Anicic, Zoran Marjanovic; Integration of Business Applications using Semantic Web Technologies; INTEROP-ESA'2005 Conference, Geneva, 2005

Preface

(BPD 2005)

The conscious (re)design of business processes as a powerful means of improving performance and raising customer satisfaction is nearing its 15th birthday. Since the breakthrough publications on successful, systematic corporate design initiatives in 1990, streamlining business processes using advanced IT has been on the agenda of almost every organization.

Despite its respectable age and its obvious pay-offs, process design is still more art than science. Many handbooks on the subject remain vague about how to actually derive superior process designs. Consultancies put much emphasis on stimulating the creativity of business professionals to come up with new process lay-outs, but the outcomes of such efforts are hard to predict. Scientific approaches so far have focused on small, well-understood business domains. Overall, much more attention is devoted to process modeling techniques and standards. In a way, this is similar to agreeing on the language, without knowing what to say.

The aim of this workshop, which took place on September 5 in Nancy, was to bring people together who have an interest in advancing the state of the art in process design (as in contrast to mere process modeling). Researchers and practitioners from Belgium, the USA, the Netherlands, the UAE, France, Sweden, and Italy presented their work and actively exchanged their ideas. Each paper in this section was presented at the workshop and includes the insights that followed from the workshop discussions.

September 2005
Eindhoven

Hajo A. Reijers

Organization

Organization Committee

Tom Davenport

School of Executive Education
Babson College

Hajo A. Reijers

Department of Technology Management
Eindhoven University of Technology

Michael Rosemann

Faculty of Information Technology
Queensland University of Technology

Program Committee

Wil van der Aalst (Eindhoven University of Technology, Netherlands)
Leonid Churilov (Monash University, Australia)
Marlon Dumas (Queensland University of Technology, Australia)
Peter Green (University of Queensland, Australia)
Kees van Hee (Eindhoven University of Technology, Netherlands)
Monique Jansen-Vullers (Eindhoven University of Technology, Netherlands)
Peter Kueng (Credit Suisse, Switzerland)
Selma Limam Mansar (Zayed University, UAE)
Trevor Naidoo (IDS America, USA)
Stephan Poelmans (VLEKHO Business School, Belgium)
Brad Power (Babson College, USA)
Andrew Spanyi (Spanyi International, USA)
Robert van der Toorn (ING Investment Management, Netherlands)
Roger Tregear (Leonardo Consulting, Australia)
Kees Voorhoeve (Deloitte, Netherlands)
Michael zur Muehlen (Stevens Institute of Technology, USA)

Design Processes for Sustainable Performances: A Model and a Method

Antonella Longo and Gianmario Motta

Department of Innovation Engineering, Università di Lecce
antonella.longo@unile.it
Department of Industrial Engineering, Politecnico di Milano
gianmario.motta@polimi.it

Abstract. In this paper we present a model that defines performances of a business process. The process is conceived as a service chain that delivers services to customers by a glow that links a set of organizations. This concept implies a wider view of performance. The model includes all the performances that increase overall competitiveness (i.e. classic cost and efficiency indicators are integrated by effectiveness indicators, that measure service level and quality). Second, the model considers the different stakeholders involved in a process: management, customers and operators. The model maps the performance indicators on these perspectives. This multiple perspective can be used in benchmarking and diagnosis to evaluate an existing process or to design a new process.

1 Business Process as a Service Chain

We define a business process as a service chain, by which a network of organizations processes a service request, made by a customer, and delivers a product or service to the customer. This simple schema can depict a variety of real life cases, like buying a book from Amazon, processing a building permit in Government, responding to a customer order in a machinery vendor.

Actually, the concepts of business process, of flow of activities and process-oriented organizations are well-established in classic authors of business process engineering such as Hammer (5) and Davenport (2).

In the concept of “service chain” a process exists inasmuch it delivers a service to a customer. The importance of the service concept is testified by the common business practice. Actually, transportation authorities, government authorities, health authorities and utilities publish on their Web sites service statements that define the service promise to the customer. Moreover, many organizations, when outsource a service, set service level agreements, that define the service scope and service levels expected from the outsourcer. In a service-oriented environment, with the customer focus as a cornerstone (21), measuring service chains becomes a must. These “qualitative” and non-financial measures complement the financial measures necessary to management to control the efficiency, as it happens with Activity Based Costing (8). In short, service chains are increasingly important, and their measurement becomes important too. The performance of business processes is a quadrant of the Balanced Score Card (7) and it can be seen as a standard framework of performance measurement. Additionally service and quality

performance indicators are incorporated in a reference model of the supply chain (SCOR), a major research field in logistic management (1). In recent years the quest of process performance is boosting the success of continuous improvement methods, such as Six Sigma (4).

In short, the measurement of the business process should encompass financial, service and quality measures. This wide measurement framework can be used (i) to control the performances of an existing process, thus defining the measures of its control panel, (ii) to benchmark the performances of a given process (iii) to set the design objectives of a new process. Here we focus on point (iii).

The objective of this paper is to define a set of performance measures that help the analyst to design good and sustainable business processes. In our view, a process design is good if it allows good performance on the whole range of performance measures. Further, the process design is sustainable if it allows good performance for the diverse actors who are involved in the process, who are regarded as process stakeholders. They include the management, who controls the process, the customer, who makes request and receives the output and the operators, who are people who actually work on the process.

In the following sections we present the key points of our model and method. In section 2 we present the key performance measures and map the performance indicators within the different stakeholders perspectives. Section 3 illustrates the method steps and relations between performance modeling and the design of business process. The major novelty is in section 2. The model and the method are illustrated by a case study in the e-government in section 3.

2 Identifying the Indicators and Relating Them with Stakeholders

Based on empiric experience we have defined a general framework to be specialized by the analyst when dealing with an individual process (Figure 1). The general framework reflects the process model as service chain we have introduced above and includes four major classes of indicators: overall, cost, quality, service and time.

Overall indicators have the objective of describing the process context by quantifying the size of requests made by customers (i.e. input of the service process), outputs produced and resources used. Resources include both physical (human resources, equipment, inventory) and virtual resources such as information. These measures are relevant to benchmark or study the dynamics of the process (e.g. seasonality).

Cost indicators have the objective of measuring the economics of the process. Actually, they measure the unit cost of input or output, the productivity of resources used by the process, and the usage of resources (i.e. the rate of used resource over available resources). The meaning of some measures changes dramatically depending on the stakeholder perspectives, as we show below in this paper.

Quality indicators have the objective of measuring the capacity of the process input or output of being consistent with the expected performance, and therefore include conformity measures, availability and customer satisfaction measures.

Finally, service indicators have the objective of measuring the performance against time in terms of response time, punctuality, perfect orders and flexibility. Most of these measures are clearly customer oriented.

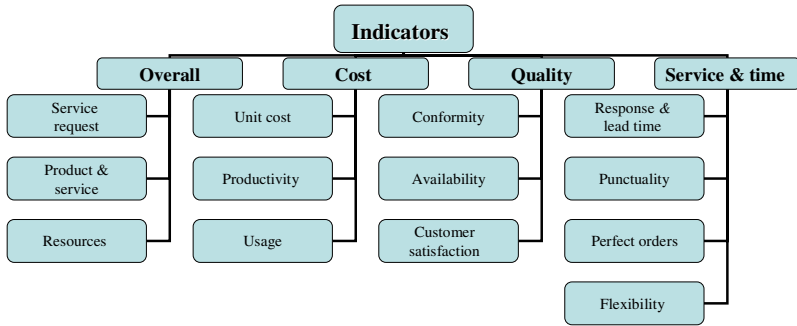


Fig. 1. A tree of process performance measures

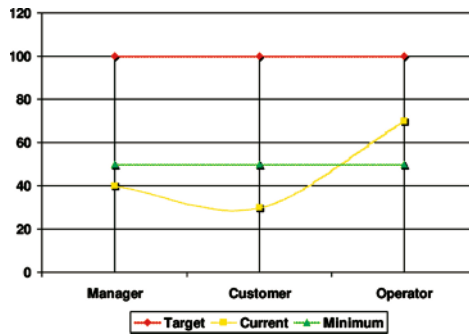


Fig. 2. A graphic overview of the stakeholder oriented performances

A process involves several stakeholders: the customer who receives the output, the manager who controls the process, the operator who works in the process. Each stakeholder views the process and would maximize the value from a different standpoint. The customer would minimize costs, maximize quality and squeeze times. The operator is motivated by a nice work environment, thus maximizing his own return from work. The manager would squeeze costs and maximize productivity and get the highest quality at the lowest cost.

Figure 2 summarizes the concept of the three perspectives. Each axis indicates a perspective and shows performance levels i.e. target, current, minimal. The target can be the performance expected by the stakeholder, a benchmark or an objective. The current value reflects the actual level in the existing process. The minimal value is the information on the minimal performance each stakeholder can allow and actually it is rather complicated to be calculated. The total percentage score within each perspective is given by the sum of the percentage of each individual performance measure.

To calculate this nice graphic, the analyst should map the general performance measures on the different stakeholders’ perspectives. So we have crossed the indicators with stakeholders, obtaining the tree shown in Figure 3. The resulting list helps the analyst to identify the performances of an individual process.

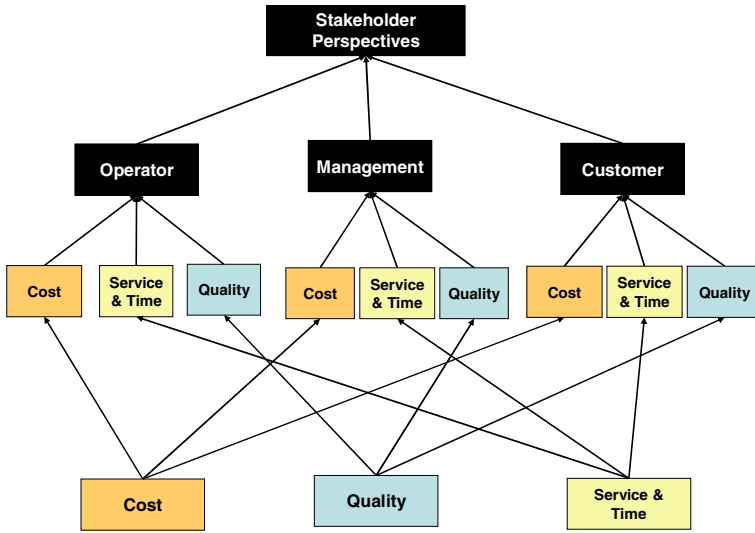


Fig. 3. Mapping generic indicators in stakeholder perspectives

3 Using the Performance Method

Some established process design methodologies, as of ARIS (19) consider different process views, that include activity flow, organization structure and information technology. In general, process innovation (5, 20,12) affects almost the whole range of organization variables and, in turn, the organizational setting affects performances (just recall the case of the assembly line against assembly islands). Therefore, we can assume the process performance is driven by some key organizational variables.

In the as-is analysis, the analyst considers the qualitative relation between the situation of organizational variables and the performance; in the to-be design, the analyst evaluates, by tests or simulation, if innovation can eventually give expected performances. If not, the analyst can come back to the design and modify it or, conversely, to re-analyze performances. We have defined a simple questionnaire, that helps the analyst to spot critical points (e.g. to identify no-value-added activities that affects efficiency and increase service times) (16).

The performance driven approach, supported by a multiple stakeholder perspective, is an iterative, almost heuristic, method. It requires an approach that differs from others used to implement Enterprise Systems, which are based on “best practices” or adapt a pre-designed normative process model to the individual process. This approach, largely used in ERP projects, gives very controversial results (14, 22).

The design of the actual process can also be oriented to the performance and to the stakeholder perspective. For example, UWA+ methodology (11) designs user experience and data models that incorporate the informational objectives of different stakeholder communities (e.g. the design of use cases is based on stakeholders goals, that in turn arise from stakeholders expected performance measures). Requirements stem from goals and are implemented in the use-cases.

Hence the performance analysis and the stakeholder view goes all the way down from the high level strategic analysis to the design of software.

4 Case Study

The performance model has been tested. First the overall modelling methodology has been tested in laboratory case studies (i.e. written case studies) that included an insurance company, the service process of a lift company, the equipment maintenance process of a transportation authority. Second, the models have been used on real-life cases, such a very large healthcare company and an e-government agency. In the last, only performance and information modelling have been used. Modelling has been proved easy to use and powerful. In the next section we discuss the case of e-government.

4.1 The Context

A Municipality manages the territory through a Planning Scheme, Technical Implementation Rules and Building Rules. A Government Decree (D.P.R. 380/01 “Testo Unico per l’Edilizia”) has introduced the One Stop Counter of Building Services (SUE - Sportello Unico per l’Edilizia). The Counter provides a complete front-end to whoever wants to build or modify a building. The service chain begins with an application to the Department of Planning and Community Development. The process ends with the permit and the payment of permit fees.

The case study considers a small Municipality in the South of Italy (16.000 citizens). The organization includes Departments and Divisions with specialized tasks. The process delivers a service assigned to the Department in charge of Public Works Services and Town Planning Services. In the latter Services 5 employees are employed, one Manager, and one technical expert as external contractor.

4.2 Goals and Indicators

Analysts have assessed the changes requested by new regulations that include the use of the Web for deploying services to citizens. From the management perspective, the introduction of Web has these goals: (a) To reduce the time of response from the Municipality, (b) Less errors in applications and related contentious and to provide citizens with a better interaction experience with Public Administration (e.g. reducing to zero the number of customers requesting information at the front end office), (c) keeping constant the number of clerks.

From the citizen/applicant (= customer) perspective, the goals are (a) applying and receiving the building permit and (b), in case of refusal, getting clarifications.

The quantification of the performances (including processes and web applications) are made by indicators. Indicators are from both interviews and experiences. We observe indicators of quality can be extended with specific indexes of Web systems usability, which depends on the way the service is delivered (front-end desk, Web, kiosks, etc.).

Table 1, 2 and 3 present stakeholder indicators. Table 1 shows the process owner’s ones, in which we include all the managers.

Table 1. Key Performance Indicators (KPI) of the Management Perspective

Indicator classes	Indicators	Measures
Process Costs	Unit cost	Headcount / applications (year)
	Time usage	Work time/ available time
Process Time and Service level	Process duration	Duration (days)
Process Quality	Incorrect documentation	Incorrect-missing applications / total applications
	Re-working rate	Applications reworked / complains
	Rate of special cases	Special cases/ applications

Table 2. Key Performance Indicators (KPI) of the Process Customer Perspective

Indicator classes	Indicators	Measures
Quality delivered to Customers	Complaints	# Complaints in a year
	Information on application status	Provided/ Not provided
	Contentious applications	Number of contentious applications
	Bureaucratic language simplification	Clearness in the presentation to a generic user
	Information availability	Time required to get updated about the application status
	Easiness of finding information	Qualitative scale
	Easiness of filling applications	Qualitative scale
	Easiness to find regulations corresponding to a case	Qualitative scale
Time and Service to Customers	Response Time	Time from submission to issue (days)
	Punctuality	Applications late/ Total applications
	Rate of worked requests	Worked requests / total requests
Cost of Customer	Customer Cost	Product cost / fee (Euros)
	Customer Time	(Time for information on application) + (time for following the application status) + (time for receiving the service)
	Information Access Cost	Time spent in asking for information about application and service (in days) Cost of information on service (Euro)
	Cost for the customer service use	Cost of use the service during the life cycle (Euro)

Table 2 shows the indicators of the customer. Please note that many indicators evaluate the performance of the customer independently from technology. As an example we show some indicators strictly linked to the Web application, and in particular to usability indicators, that we have inserted in the Quality delivered to Customer category.

Table 3. Key Performance Indicators (KPI)of the Process Operators Perspective

Indicator classes	Indicators	Measures
Quality delivered to Operators	Capability to prevent task errors	Task error rate
	Capability to prevent syntactic errors	Syntactic erro rate
	Flexibility	Capability to manage anomalies (not technical): present/ not present
	Integration Capability	Capability to integrate different information
Supports to Operators	Rate of process steps supported by computer systems	Number of process steps supported by computer systems/ Total number of steps
	Rate of process steps replaced by computer systems	Number of process steps replaced by computer systems/ Total number of steps
Costs and time to Operators	Number of elementary operations to complete the task	Number of elementary operations to complete the task
	Time for training on the procedure	Measured in hours
	Information access time	Seconds
	Time to accomplish a task	Measured in minutes
	Information sharing cost	Time for system data entry + time for system delivery output

Table 3 illustrates the indicators related to the Operator perspective. Even if included in the resources of the Building permit process, the Operator is a user of the IT systems. His or her major requirement is the capacity to have more efficient work and less prone to errors. As we can see the meaning of a same generic indicator (.e.g. cost) changes dramatically depending on the stakeholder, which could be the management (financial internal cost of executing the service chain), the operator (effort and time spent to accomplish a task) or the customer (cost of placing the service request, buying the service and using the service).

The case study on e-government shows how standard indicators can be customized on a specific case, and how powerful is the potential diagnosis the tool enables. In fact, by giving a value to the current and expected values of performance measures and simply summing up the resulting percentage, the analyst can get a radar diagram, and, in this specific case, would probably realize that the performances are below expectations from all the three perspectives. By eliciting goals from KPIs the analyst will design a system that reflects perspectives and expected performances.

5 Conclusion and Future Work

In this paper we have illustrated the concepts of a process modeling, based on the red thread of the stakeholders perspectives. We have discussed the key idea of customizing the performance according to stakeholders perspectives and the reason why a good design should balance the different interests of the stakeholders, i.e. management, operators and customers. Also we have suggested how the stakeholder perspectives can eventually be used all the way down, from a nearly strategic assessment of

the process to the design of the Web system. The case study has shown how modeling can be customized to evaluate an e-government service chain.

Our work is continuing. Our next objective is integrating process simulation. Actually, there are a variety of self-contained simulators that allow simulating performances in term of process duration and workload on process resources. The purpose of integration is to have a quick simulation of different performance alternative and process configuration. A second objective is to build a knowledge base on e-government and alike service processes wherefrom the analyst can directly pick process configurations and test them by simulation.

References

1. Bolstorff P., Rosenbaum R. Supply Chain Excellence: A Handbook for Dramatic Improvement Using the SCOR Model, Amacom (2003) ; Davenport T.H., Short J.E. "The new industrial engineering: Information Technology and Business Process Redesign", Sloan Management Review, vol. 31, n.1, 1990
2. Davenport T.H., Stoddard D.B., "Reengineering: business change of mythic proportions?", Harvard Business Review, March-April 1994
3. Davenport, T.H., "Putting the enterprise into the enterprise system," Harvard Business Review, vol. 76, no. 4, 1998, pp. 121-131
4. Gupta P., Six Sigma Business Scorecard, McGrawHill, New York , 2004
5. Hammer M., "Reengineering work: don't automate, obliterate", Harvard Business Review, July-August, 1990
6. Hammer M., "the Superefficient company" , Harvard Business Review, September/ October 2001, pp. 82-91
7. Kaplan R. S., Norton D.P., Balanced Scorecard, Harvard Business School Press, Boston, Ma, 1996
8. Kaplan R. S., Norton D.P., Strategy Maps: Converting Intangible Assets into Tangible Outcomes , Harvard Business School Press, Boston, Ma, 2003
9. Klein M., Herman G.A., Lee J., O'Donnel E., Malone T.H., "Inventing new business process using a process repository", in Malone T.H., Crowston K., Herman G.A. Organizing Business Knowledge - the MIT process handbook, The MIT press, Cambridge Ma., 2003
10. Longo A., Bochicchio M., Carducci M.,: "Reducing Normative And Informative Asymmetries In Fiscal Management For Local Administrations", in Digital Communities in a Networked Society: e-Commerce, e-Business and e-Government (ISBN 1-4020-7795-5), 2004
11. Longo A., Bochicchio M.,: UWA+: bridging Web systems design and business process modeling, HyperText 2004, Santa Cruz, August 2004
12. Markus M.L., Benjamin R.L., "The magic bullet theory of IT-enabled transformation", Sloan Management Review, vol. 38, n.2, 1994
13. Mabert Vincent A., Soni Ashok, Venkatraman M.A., "Enterprise Resource Planning: common myths versus evolving reality", Business Horizons, May-June, pp 69-75, 2001
14. Motwani J., Mircahandani D., Madan M., Gunasekaran A. "Successful implementation of ERP projects: evidence from two case studies", International Journal of Production Economics, 75, pp. 83-96, 2002

15. Motta G., “Il progetto dei sistemi informativi direzionali” (= Design of Management Information Systems) in Bracchi G., Motta G., Progetto di sistemi informativi (= Design of Information Systems) , Etas Libri, Milano 1993
16. Motta G., Barbieri T., Francalanci C. , “Business information and process modelling: an integrated modelling platform”, Dipartimento di Elettronica e Informazione, Politecnico di Milano, internal report 2004.27
17. Motta G, “Il metodo dei KPI” (= the KPI method) in: Bracchi G., Francalanci C., Sistemi Informativi e impresa digitale (= Information Systems in the Digital Enterprise); McGrawHill, Milano 2005
18. Rigby D.K., Anderson S.R, “CRM done right”, Harvard Business Review, vol 82., n11, november (2004).
19. Scheer A. W., ARIS - Business Process Modelling, Springer, 2000
20. Scott-Morton M (ed.) The Corporation of the 1990s: Information Technology and Organisational Transformation, Oxford University Press, New York, 1991
21. Seybold P., Marshak, R.T., The Customer Revolution (2000)
22. Mabert Vincent A., Soni Ashok, Venkatraman M.A. (2001), Enterprise Resource Planning: common myths versus evolving reality, *Business Horizons*, May-June, pp 69-75

Designing Business Process Variants – Using the BAT Framework as a Pragmatic Lens

Mikael Lind¹ and Göran Goldkuhl²

¹ University College of Borås, School of Business and Informatics, S-501 90 Borås, Sweden
Mikael.Lind@hb.se

² Linköping University, Department of Computer and Information Science,
S-581 83 Linköping, Sweden
ggo@ida.liu.se

Abstract. When designing business processes there is a need to identify and delimit different processes. There exist, however, unclear criteria for delimiting business processes. Business process analysts apply usually a sequential view of business processes. Besides process sequences, there is a need to acknowledge different process variants. Founded in a combined transformative and coordinative view on business processes, instruments for describing business process variants are put forward in this paper. Two instruments (matrices) are proposed. One business process division matrix, for distinguishing process variants in relation to each other. The other matrix, a business phase matrix, is to be used to reveal the content of each phase of the business interaction constituting each process variant.

1 Introduction

Many contemporary approaches for business and IT development emphasise a process oriented perspective of what is being done. A process oriented perspective on organisational work means that this work is divided into several “process components” in order to place the customer in focus. Several different methods for process modelling exist. It is important to have adequate method support when reconstructing current processes and redesigning new ones. The different methods are however based on different conceptual frameworks and thus different process notions.

Davenport [5] has identified a problem about process determination. According to Davenport [5] there can be many different ways of determining an organisation’s or several collaborating organisations’ business processes.

“Considerable controversy revolves around the number of processes appropriate to a given organization. The difficulty derives from the fact that processes are almost infinitely divisible; the activities involved in taking and fulfilling a customer order, for example, can be viewed as one process or hundreds. The ‘appropriate’ number of processes has been pegged out from two to more than one hundred” [5, pp. 27-28].

Unclear criteria for process division and delimitation can give rise to varying amount of processes when describing an organisation [5].

A process perspective puts the customer in focus. It is an impetus to analyse and design organisational work towards value for the customer. A process perspective usually also emphasizes a sequential view on organizational work. A process is considered to consist of sequences of activities. Input to the transformed through a series of stages (sub-processes or activities) to output aimed for customers [e.g. 15]. This sequential view on processes is illustrated in figure 1. A process is considered as a sequential workflow.

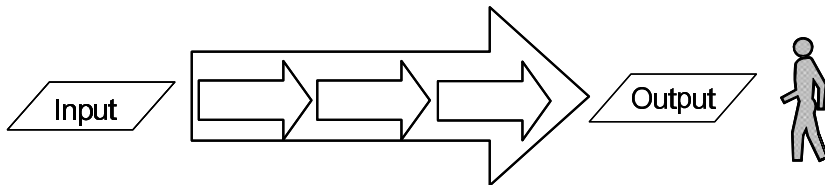


Fig. 1. A sequential view on business processes

Since the beginning of the nineties there has however been intensive debate of how to understand business processes. The main view on processes as a transformative workflow has been challenged. Keen & Knapp [16] have identified two main trends; either to comprehend business processes as transformation or to comprehend business processes as co-ordination [cf also. 13, 20]. As indicated above, the transformative dimension is about focusing the transformation of input (raw material) to output (the finished product to be utilised by the customer). The coordinative dimension is about focusing the creation, fulfilment and assessment of agreements between and within organisations. The latter dimension concerns patterns of interaction within and between organisations; how orders/assignments are given, accepted and forwarded. Communicative aspects are thus stressed in the coordinative perspective. Recent research has shown a potential in managing this task by including both transformative as well as coordinative dimensions of business processes [13, 20]. This means to view processes as both workflow and interaction.

It is also claimed that the sequential view does not give a just description of processes. A variant view has been put forward as a complement to a sequential view [cf. 20, 22]. This means that there will be alternative business processes in an organisation, i.e. there are different ways for performing business missions. An organisation usually performs different kinds of missions and this implies different types of business relationships between customer and supplier.

Unclear criteria for process division and process determination concern both criteria for distinguishing different types as well as different variants of business processes. In this paper we focus criteria for distinguishing process variants. For criteria concerning the distinction of different process types confer Lind [21].

In order to perform an adequate renewal and redesign of business processes it is important to distinguish different process variants. The purpose of this paper is to provide theoretical foundation as well as tools for determining such variants.

This paper is arranged as follows. In the next section theoretical foundations for understanding business interaction is presented. This section is followed by an empirical illustration in which process variants, i.e. several ways of performing business,

were identified. These variants will be conceptualised in a business phase matrix. Before concluding the paper some implications for business process design will be discussed.

2 The Different Phases of Business Interaction

Many organisations have alternatives for establishing and fulfilling agreements with customers. A focus, as emphasised in business process theories, on customer value implies an understanding of interaction patterns between supplier and customer. In this section we will therefore introduce some frameworks for understanding business interaction followed by choosing one of them as a basis for introducing an analytic tool to determine process variants. This tool will thus be founded in the supplier's alternative ways of interacting with its customers.

2.1 Different Frameworks for Understanding Business Interaction

Frameworks for business interaction have been proposed by a number of scholars; confer e.g. Ahlström [1] for an overview of some frameworks. A well-known reference model for electronic markets has been presented by Schmidt & Lindemann [30] and Lechner & Schmidt [19]. Within the language/action (L/A) tradition there are several business interaction frameworks, see for example Dietz [6], Goldkuhl & Lind [9, 10], Weigand & van den Heuvel [33], and Medina-Mora et al [27]; all building on the speech act insights from Searle [31]. These L/A approaches are important since they emphasize actions, communication and interactions in the relations between customer and supplier.

The L/A-tradition puts strong emphasis on analyzing patterns of inter-related business acts. There is pragmatic foundation with an emphasis on actions. Such thinking emanates from the Conversation-for-Action (CFA) schema [34]. It is also emphasized by all mentioned frameworks discussed above that business interaction can be divided into several phases; from offering and commitment to fulfillment and assessment. Goldkuhl & Lind [9, 10] has used this thinking of pattern as a foundation for their Business interAction & Transaction (BAT) framework covering business interaction patterns between two parties; the supplier and the customer.

It is common that a supplier have *different* kinds of actor relationships with its customers. Usually, the supplier offers, specifies, produces and delivers *different* kinds of products to its customers [20]. These differences give rise to process variants. Examples of different kinds of actor relationships are separate orders and frame contracting [9, 10]. Examples of different kinds of products are standardised vs. customised, transfer vs. letting out something, and moving vs. treating a client [12]. Dependent on the actor relationship and the product handled, the interaction between the business and the specific customer will vary. A unique combination of a certain kind of actor relationship and a certain kind of product determines a process variant. Each process variant includes and supports a particular interaction logic between the supplier and the customer.

To identify different logics of business interaction, i.e. different process variants, it is therefore necessary to use a business interaction theory that covers different

patterns of interaction dependent on the actor relationship. In order to arrive at a thorough understanding of such patterns it is necessary that a distinction is made between different types of business acts and how these are inter-related. A suitable framework for revealing the dimensions needed for determining process variants is the BAT-framework.

The BAT framework (sometimes an abbreviation for Business Action Theory) was originally presented by Goldkuhl [7]. It has later been refined several times; in Goldkuhl [8] and Goldkuhl & Lind [9, 10]. These revisions have been based on extensive empirical studies where the BAT framework has been applied in different settings [e.g. 2, 3, 11, 14, 17, 18, 22, 24, 25, 26]. The BAT framework has been compared with other frameworks; with Action Workflow of Medina-Mora et al [27] in Goldkuhl [7] and Verharen [32]; with DEMO of Dietz [6] in Reijswoud & Lind [29] and Verharen [32]; with ‘meta-patterns for electronic commerce’ of Weigand & van den Heuvel [33] in Lind & Goldkuhl [23]; with the MRM framework of Lechner & Schmidt [19] in Petersson & Lind [28].

2.2 The Business interAction and Transaction Framework

Different types of exchanges between business parties (customer and supplier) form the core of the BAT framework. First of all there is a distinction between interactions on a market level vs interactions on a dyadic level. On a market level a supplier interacts in relation to potential customers and vice versa. This interaction is called knowledge/contact search and exposure. On this market level the supplier directs its efforts towards potential customers, often many. When a contact is reached between a supplier and a customer this interaction may proceed to the dyadic interaction.

On the dyadic level there is a distinction made between frame contracting and business transactions. Frame contract means a contract concerning several subsequent business transactions that can be different sub deliveries. The frame contract level as well as the business transaction level consists of different type of exchanges between a particular supplier and a particular customer.

A frame contract is a long-term agreement. Such an agreement is established through exchange of proposals and commitments (figure 2). Exchange of proposals means negotiation between the two parties. Bids and counter-bids concerning particular products, prices and adjacent conditions are exchanged. Proposals may also include exchange of knowledge concerning other conditions relevant to the business interaction, i.e. different aspects of the parties’ capabilities. Exchange of commitments means the establishment of each party’s obligations within a frame contract. These obligations concern the expected future business actions of each party. The frame contract is an agreement that governs the subsequent recurrent business transactions in which the frame contract is gradually fulfilled. On the frame contracting level there is no exchange of value (products vs money). This occurs on the business transaction level. Experiences from the performance of business transactions may be a basis for an assessment of the frame contract and its fulfilments. Assessments can be made by each party and some of these can be exposed to the other party, i.e. assessments may be exchanged.

As shown in figure 2 there may be a recurrence of frame contracting over time. This means also a continual development of business relations. The frame contracting

will also be based on each party’s capabilities and through the process these capabilities will usually emerge. For example the negotiation can include development of new products. Based on the supplier’s general existing capability, new products might be specified which might lead to a development of the supplier’s capability. Such a new product might also influence the customer’s production process, which is part of the customer’s capability.

In figure 2 the exchanges of frame contracting are depicted. In this figure we also include an important part of the context – the relation to the business transaction level. Frame contracting cover exchanges of proposals, commitments, exchanges in the embedded business transactions, and assessments.

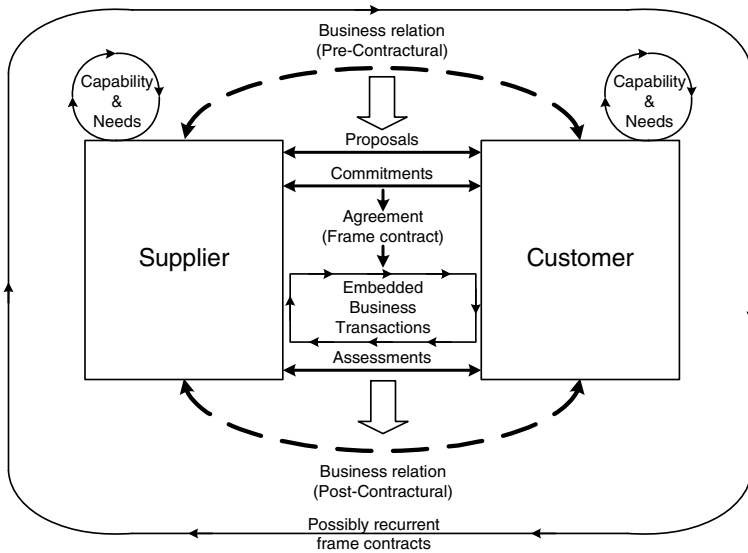


Fig. 2. The constituents of frame contracting (BAT frame contracting transaction model) [10]

Besides frame contracting the other level of business interaction on the dyadic level is the business transaction. Many times business transactions are instead governed by separate (single) transaction orders and no frame contracts exist. A business transaction comprises the establishment, fulfillment and assessment of a business agreement in order to satisfy one or several related product needs of the customer. This means exchanges of proposals, commitments, fulfillments and assessments (see figure 3).

The exchange of fulfilments means the exchange of value. It is only on this level that the exchange of value (goods and/or services in the exchange for money) occurs. If either part is not satisfied with the fulfilment, a reclaim might be directed to the other party, which occurs during the assessment phase. Of course, appreciative assessments may also be exchanged.

The business communication prior to the fulfilments will differ dependent on the contractual situation. In business transactions there can be either a frame contract

based sub-order or a separate (single) transaction order. If there exists a frame contract there will thus be a sub-order from the customer in accordance with this frame contract. In a frame contracting situation the need for exchanging proposals will decrease and be determined by the specifications in the frame contract. Many times parts of the proposal phase will be short-circuited when there exists a frame contract. One of the main intentions behind frame contracting is to decrease transaction costs through decreased interaction. Usually the contents and the transfer of the sub-order is standardised in ways to decrease transaction costs. Frame contracting is also used to reduce uncertainties and to ensure future procurement, production and sale.

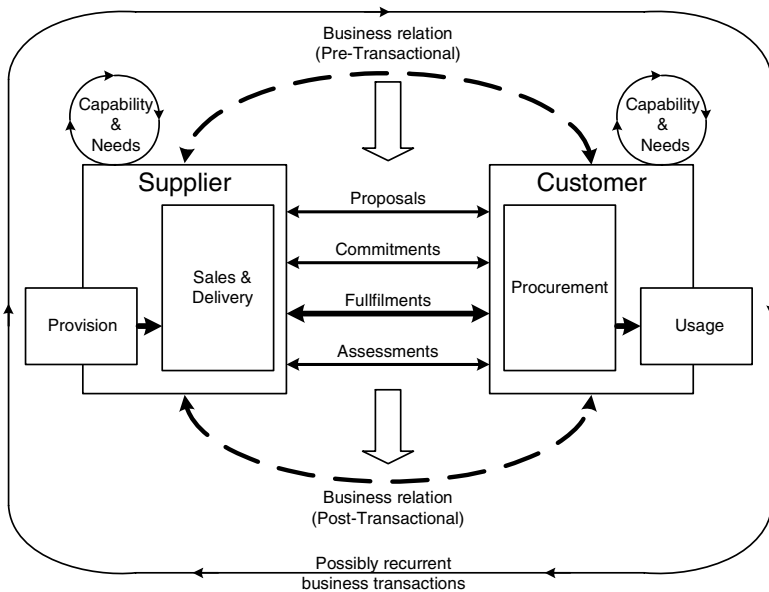


Fig. 3. The constituents of the business transaction (BAT business transaction model) [9]

If no frame contract exists we have the case with separate transaction order. In such a case the proposal and the commitment phases often need to be more elaborated compared to sub-orders within frame contracts. This is of course dependent on the character of the product and other important business circumstances. The strive for minimising transaction costs leads to standardisation of these types of transactions as well. This can be seen as one driving force for the development of e-commerce applications; transaction standardisation for transaction cost reduction.

The business transaction is dependent on the existing business relation between the business parties. Such relations can be deep if there exist prior interaction, and thin if no or little interaction has occurred. In the frame contract case, the frame contract functions as a regulator of the business relation between the two parties. The business relation does not only consist of these formal agreements, but also of the collected experiences of prior business interactions. The execution of the business transaction

will influence the business relation dependent on its performance. If there will be recurrent business transactions, the post-transactional relations will form the pre-transactional relations in the next business transaction. Such business transaction recurrence is dependent on the existence of a frame contract. The frame contract defines the occurrence of several recurrent business transactions. In the case of a separate transaction order, recurrence occurs when the two parties choose trade again.

The business interaction aims at improving both parties' capabilities in different respects. The customer wants to satisfy certain needs through the product usage. The purchased product will enable the customer to perform desired actions. The customer compensates the supplier for the delivery. This compensation will increase the financial capability of the supplier. The business interaction will however often have learning effects on both parties. Experiences from the execution of the business interaction may improve the capability for future business interactions. This capability improvement can apply both to this particular business dyad, but also to interaction with other business parties. Experiences from business interaction can be a basis for both continuous improvement on a daily basis and be incentives for more strategic developments.

Founded in the BAT framework the phases of interaction that can be distinguished are depicted in the table below (figure 4). These phases will form the basis for the model that will be used for distinguishing process variants (cf. section 3 & 4 below).

<i>Level</i>	<i>Type of exchange</i>
Market	Knowledge, contacts and business interests
Dyadic: Frame contracting	Proposals, Commitment, Assessments
Dyadic: Business transaction	Proposals, Commitment, Fulfillments, Assessments

Fig. 4. Different phases of interaction

3 Process Variants in Practice

How shall process variants be delineated in relation to each other and how shall they be described? We will address these issues by the use of a simple example. The example is based on an action-research oriented case study performed at a steel company, here named Steelco.

Steelco is a manufacturing company, which mainly transforms steel into pipes for hydraulic cylinders. Steelco has different ways of performing business, i.e. the company takes part in different business interactions. The strategy that Steelco enacts is to have a variety of interaction ways with their customers. One goal is to build long-term relationships with its customers.

In the case study several of Steelco's business processes were identified. These business processes coexisted and can therefore be addressed as process variants. The coexisting process variants were called:

- *Separate order – tailor made products (PV1)*, which consists of activities to produce and sell tailor-made products.
- *Separate order – standard products (PV2)*, which consists of activities that are performed when Steelco is selling standard products from the standard stock.

- *Frame contract – standard products (PV3)*, which consists of sales and production activities that are performed based on a customer prognosis of future orders covering several recurrent business transactions.
- *Separate order - traded products (PV4)*, which consists of activities that are performed when subcontractors of Steelco deliver products directly to Steelco’s customers. Steelco is not able to manufacture those products itself.

Figure 5 (a business process division matrix) shows the delimitation of the business processes in the case study. Four process variants; i.e. different principle ways for Steelco to perform its business, were identified. By using the two dimensions "Actor relationship" and "Product characteristics", it was possible to identify and classify the four process variants. This way of identifying and classifying business processes, by using these two dimensions, has been proven to be successful in different settings [c.f. 20, 21]. Based on extensive empirical evidence (summarised in [20]) these two dimensions has been found to be decisive for process variant determination. There are however other characteristics that determine the structure and functions of different business processes such as e.g. transformation logic, cash flow logic, the normative context and characteristics of the infrastructure [13].

Product char.	Tailor-made products	Standard products	Traded products
Actor relationship			
Separate order	PV1	PV2	PV4
Frame contract	---	PV3	---

Fig. 5. Business process division matrix: The different processes variants at Steelco

From the matrix (figure 5) it can be noted that there are two slots that do not have any content (frame contract - tailor-made products and frame contract – traded products). These empty slots gave rise to discussions concerning the potential of establishing new business opportunities. It turned out, however, that frame contracting on a basis of selling and delivering tailor-made products was unnatural. Steelco did however acknowledge the potential in adopting frame contracting for traded products. It was a question of using already established routines for frame contracting as well as the relationships established towards sub contractors. The business process division matrix can thus be seen as an instrument used for enhancing business development.

Each process variant can be characterised according to the different phases of business interaction identified in section 2.2. This characterisation is put forward in a business phase matrix (figure 6). Note that the process variant ‘separate order – traded products’ have not been included in the matrix for reasons of space.

From the business phase matrix it can be seen that different process variants are dependent on the actor relationship and the product characteristics. The content of the different phases might vary or be similar between the different process variants even if one of the dimensions is the same. Taking for example the two process variants of separate order handling, i.e. the same actor relationship, it can be noted that it is important for the tailor-made product process to establish design competence and in the case of standard product process there is a need for a standard stock.

Process variant Capability/Phase	Separate order – tailor-made products	Separate order – standard products	Frame contract – standard products
Action Capability	Flexible production equipment, design competence.	Own production of standardised products. Stock.	Own production of standardised products. Stock. Planning capability
Frame contracting			
Proposal	---	---	Proposal for long-term agreement
Commitment	---	---	Frame contract regulating both parties commitments
Assessment	---	---	Mutual assessment of the realisation of the frame contract
Business transaction			
Proposal	Products are designed based on customer needs. Prices are negotiated.	Offers of standard products. General and customer particular price lists as a basis for negotiation.	
Commitment	Customer order based on offer including product specification.	Customer order based on an offer or a price list	Sub order based on agreed frame contract
Fulfillment	Production based on order from the specific customer. No stock handling, only delivery.	Production for potential customers. Picking from stock and delivery is done based on the specific customer order	Production for potential customers. Picking from stock and delivery is done based on the specific sub order
Assessment	Potential claims are handled by Steelco.	Potential claims are handled by Steelco.	Potential claims are handled by Steelco.

Fig. 6. Business phase matrix

Further it can be noted that the frame contracting process includes activities for being more long-term oriented and thereby the business transactions covering sub-ordering processes become more efficient than the business transactions covering the separate order processes

It is also to be noted that the phases are inter-dependent for each process variant. This means that conditions for earlier phases must have been established for the success of the performance of the latter ones. The business phase matrix is therefore to be seen as a check list for covering all necessary aspects for realising one process variant. It is however as important to ensure that the process variants work together.

4 Implications for Business Process Design

Many times an organisation can be described by several process variants. In a business context this means that organisations have different ways of performing business. Process variants *co-exist* and they *co-utilise* limited resources. An organisation's process variants are thus superposed. This also means that the one and same sub process can be utilised in several process variants, but for different purposes. Such a

situation makes delivery promises complex since several product needs must be fulfilled by the same sub process.

During business process renewal and redesign it is therefore important to acknowledge the different co-existing process variants. This is important in order to manage the contextualisation, i.e. identifying the role that a certain sub process has in the work performed, of the parts of the process variants. The determination of such process variants should be made based on both the characteristics of the different products offered and supplied to customers and the actor relationship enhanced in the company. A certain process variant is delineated by actor relationship, i.e. frame contracting or separate order, and the product characteristics.

Founded in these two dimensions the content of the different phases of business interaction constituting a process variant can be determined. A business interaction is constituted by patterns of business acts covering a number of exchanges used for establishing, fulfilling and assessing agreements. During business process design it is essential to conceive organisational work based on these dimensions in order to arrive at a proper solution.

In this paper two instruments for identifying and designing process variants are proposed; the business process division matrix and the business phase matrix. By using existing or desired types of actor relationships and existing or desired types of products the business process division matrix should be used for delineating process variants. For each process variant the business phase matrix should be used for characterising the content of the phases constituting the process variants.

This way of working means that the point of departure is the business processes that directly involve the customer (cf. customer-facing processes according to Davenport [5]). Such point of departure stresses the question of “What actor relationships do we want to have and manage?” as well as “what products do we want to deliver and manage?”. Identification of other business process, such as e.g. internal business processes, should be founded in the identification of these process variants.

5 Conclusions

An unresolved task during business process design is criteria for determining one business process in relation to another. Two main dominators in the business process field have been identified; *business processes as transformation* and *business processes in sequence*. This was illustrated above in figure 1. In order to solve the task of finding criteria for process determination there has been a need to go beyond these two dominators. In this paper we have put forward the need of going *from* regarding business process sequences *to* regard process sequences *and* variants. We have also put forward the need of going *from* regarding business processes as transformation *to* regard business processes as transformation and coordination (interaction). Confer figure 7 and compare it with figure 1. Products are created based on input material (transformative dimension) and on orders (coordinative dimension).

As described in figure 7, there may exist several business process variants in an organisation. The notion of business process variants is to be seen as a complement to the dominating sequential view of business processes. This notion takes as its starting

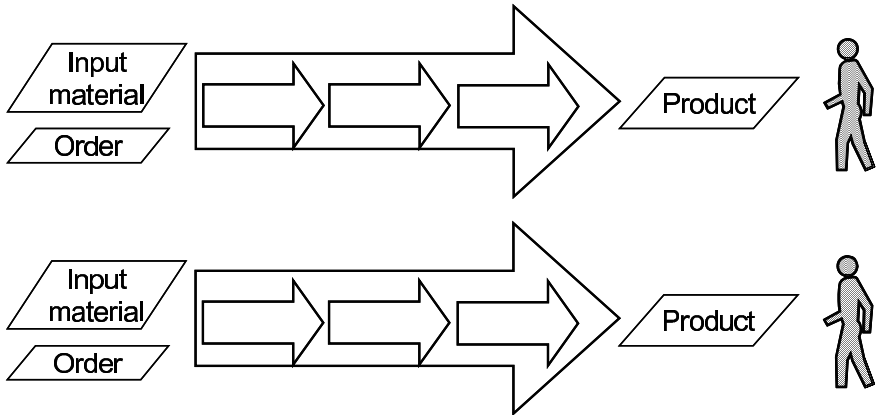


Fig. 7. A combined view on business processes (as transformation and coordination and as sequences and variants)

point that process variants co-exist in organisations and co-utilise the limited resources of organisations. Business process variants are determined by the two criteria product characteristics and actor relationship. This means that a coordinative dimension of business processes need to be taken into consideration as a complement to a transformative view. In this paper we have also put forward instruments to be used for renewal and redesign of such business processes. Instruments to be used during business process design are the business process division matrix and the business phase matrix. The business process division matrix put attention towards essential characteristics of business interaction. The business phase matrix put attention towards the content of the different phases of the business interaction patterns that constitute the process variants. These instruments are anchored in the BAT framework described earlier in this paper. The BAT model with its phases and roles is used as a lens for studying business processes. BAT is not just a theory describing business interaction. It is a practical theory [4] aimed for utilization in business process design. As such it may be helpful in identification, delineation and design of business process variants. BAT is a pragmatic lens in two respects; it emphasizes business *actions* and is intended to guide *practical* design.

References

1. Ahlström M. (2000) Offset management for large systems - a multibusiness marketing activity, Ph D Diss, Dep of Management and Economics, Linköping university
2. Axelsson K., Goldkuhl G., Melin U. (2000) Using Business Action Theory for dyadic Analysis, accepted to the 10th Nordic workshop on inter-organisational research, Trondheim
3. Axelsson K., Segerkvist P-A. (2001) Interaction between actors and information systems in web-based imaginary organisations – Experiences from two case studies, accepted to the 1st Nordic Workshop on Electronic Commerce, Halmstad University

4. Cronen V (2001) Practical theory, practical art, and the pragmatic-systemic account of inquiry, *Communication theory*, Vol 11, No 1
5. Davenport T.H. (1993): *Process Innovation – Reengineering Work through Information Technology*. Harvard Business School Press, Boston
6. Dietz J. L. G. (1999) Understanding and Modelling Business Processes with DEMO, Proc. 18th International Conference on Conceptual Modeling (ER'99), Paris
7. Goldkuhl G. (1996) Generic business frameworks and action modelling, In Proceedings of conference Communication modelling - Language/Action Perspective '96, Springer Verlag
8. Goldkuhl G. (1998) The six phases of business processes - business communication and the exchange of value, accepted to the 12th biennial ITS conference "Beyond convergence" (ITS '98), Stockholm
9. Goldkuhl G., Lind M. (2004a) Developing e-interactions – A framework for business capabilities and exchanges, Accepted to the 12th European Conference on Information Systems, June 14 – 16 2004, Turku, Finland
10. Goldkuhl G., Lind M. (2004b) The generics of business interaction - emphasizing dynamic features through the BAT model, in Aakhus M., Lind M. (Eds.) Proceedings of the 9th International Working Conference on the Language-Action Perspective on Communication Modelling, Rutgers University, The State University of New Jersey, New Brunswick, NJ, USA
11. Goldkuhl G., Melin U. (2001) Relationship Management vs Business Transactions: Business Interaction as Design of Business Interaction, accepted to the 10th International Annual IPSE Conference, Jönköping International Business School
12. Goldkuhl G., Röstlinger A. (2000) Beyond goods and services - an elaborate product classification on pragmatic grounds, in proc of Quality in Services (QUIS 7), Karlstad university
13. Goldkuhl G., Röstlinger A. (2003) The significance of workpractice diagnosis: Socio-pragmatic ontology and epistemology of change analysis, in Proc of the International workshop on Action in Language, Organisations and Information Systems (ALOIS-2003), Linköping University
14. Haraldson S., Lind M. (2005) Broken patterns, in Proceedings of the 10th Intl Conference on the Language Action Perspective, Kiruna
15. Harrington HJ (1991) *Business process improvement. The breakthrough strategy for total quality, productivity and competitiveness*, McGraw-Hill, New York
16. Keen P.G.W., Knapp E.M. (1996) *Every Manager's Guide to Business Processes – A Glossary of Key Terms & Concepts for Today's Business Leader*. Harvard Business School Press, Boston
17. Johansson B-M., Axelsson K. (2004) Communication media in distance selling. Business Interactions in a B2C Setting, in Proceedings of the 12th European Conference in Information Systems (ECIS), Turku
18. Johansson B-M., Axelsson K. (2005) Analysing Communication Media and Actions - Extending and Evaluating the Business Action Matrix, In Proceedings of the 13th European Conference on Information Systems, Regensburg
19. Lechner U., Schmidt B. F. (2000) Communities and Media – Towards a Reconstruction of Communities on Media, in Sprague E (Ed) *Hawaiian Intl Conf on System Sciences (HICSS'00)*, IEEE Press
20. Lind M. (2002) Dividing Businesses into Processes – Foundations for Modelling Essentials, In: Liu K., Clarke R. J., Andersen P. B., Stamper R. K. (Eds.) *Organizational Semiotics – Evolving a Science of Information Systems*, IFIP TC8/WG8.1, Kluwer Academic Publisher, pp. 211-230

21. Lind M. (2003): The Diversity of Work Practices - Challenging the Existing Notion of Business Process Types in Goldkuhl G., Lind M., Ågerfalk P. (2003, Eds.) Proceedings of Action in Language, Organisations and Information Systems (ALOIS), Linköping University, Linköping, Sweden, pp. 123-138
22. Lind M., Goldkuhl G. (1997) Reconstruction of different business processes - a theory and method driven analysis, In proc of the 2nd Intl Workshop on language/action perspective (LAP97), Eindhoven University of Technology
23. Lind M., Goldkuhl G. (2003) The constituents of business interaction - generic layered patterns, Data & Knowledge Engineering, Vol 47 (3), p 327-348
24. Lind M., Hjalmarsson A., Olausson J. (2003) Modelling interaction and co-ordination as business communication in a mail order setting, Proc of 8th Intl Working Conference on the Language Action Perspective (LAP2003), Tilburg
25. Melin U., Axelsson K. (2004) Emphasising Symmetry Issues in Business Interaction Analysis and IOS, in Proc of the Sixth International Conference on Electronic Commerce, ICEC'04, Delft University of Technology
26. Melin U., Goldkuhl G. (1999) Information Systems and Process Orientation - evaluation and change using Business Action Theory, in Wojtkowski W (eds, 1999) Systems Development Methods for Databases, Enterprise Modeling, and Workflow Management, Kluwer Academic/Plenum Publishers, New York
27. Medina-Mora R., Winograd T., Flores R., Flores F. (1992) The Action Workflow Approach to Workflow Management Technology, In: Turner J., Kraut R. (Eds.) Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW'92, ACM Press, New York
28. Petersson J., Lind M. (2005) Towards the concept of business action media: Frameworks for business interaction in an electronic market place setting, in Proceedings of the 3rd Intl Conf on Action in Language, Organisations and Information Systems (ALOIS), University of Limerick
29. Reijswoud VE van, Lind M. (1998) Comparing two business modelling approaches in the language action perspective, in Proc of Language Action Perspective (LAP'98), Stockholm
30. Schmidt B. F., Lindemann M.A. (1998) Elements of a Reference Model for Electronic Markets, in Sprague E. (Eds.) Proceedings of the 31st Hawaii Int. Conf. on System Science (HICSS'98), 193-201
31. Searle J. R. (1969) Speech Acts. An Essay in the Philosophy of Language, Cambridge University Press, London
32. Verharen E. (1997) A language-action perspective on the design of cooperative information agents, Ph D thesis, KUB, Tilburg
33. Weigand H., van den Heuvel W-J. (1998) Meta-patterns for Electronic Commerce Transactions based on FLBC, Proc. of 31st Annual Hawaii International Conference on System Sciences, pp. 261 – 270
34. Winograd T., Flores F. (1986) Understanding Computers and Cognition: A New Foundation for Design, Ablex, Norwood NJ

BPR Implementation: A Decision-Making Strategy

Selma Limam Mansar¹, Hajo A. Reijers², and Fouzia Ounnar³

¹Zayed University, Business Sciences College, P.O. Box 19282, Dubai, UAE
Selma.limammansar@zu.ac.ae

²Eindhoven University of Technology, Department of Technology Management,
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
h.a.reijers@tm.tue.nl

³Université Paul Cézanne - Laboratoire des Sciences de l'Information et des Systèmes -
UMR CNRS 6168 - Avenue Escadrille Normandie Niemen, 13397 Marseille cedex 20, France
fouzia.ounnar@lisis.org

Abstract. To support the efficient appraisal and selection of available best practices, this paper proposes a strategy for the implementation of Business Process Redesign (BPR). Its backbone is formed by the analytical hierarchy process (AHP) multi-criteria method and our earlier research on the popularity and impact of redesign best practices. Using (AHP) we derive a classification of most suitable best practices for the process being redesigned. Criteria such as the popularity, the impact, the goals and the risks of BPR implementation are taken into account. A case study of a municipality in the Netherlands is included. It discusses which best practices should be applied to redesign the invoicing process at the municipality.

1 Introduction

Few analytical tools exist to support the actual redesign of a business process. The aim of this work is to develop a tool that mimics the decision-making process practitioners apply to decide on which best practices to apply for redesigning a given process. The benefit of such a tool would be that it increases the efficiency of the redesign process itself and leads to a more systematic evaluation of best practices. To do so, the tool should (i) take into account the various factors that could lead to decide on one or the other best practice and (ii) provide a structured way to include these criteria and allow for an appreciation of their importance for the redesign. The presented work in this paper builds on our framework for BPR as introduced in [23] and validated in [14].

There have been a number of contributions in this field where mainly artificial intelligence algorithms have been used. Case-based reasoning and inference rules are examples of such results (see e.g. [17]). However, the majority of these contributions address only patterns of processes or specific processes for a given industrial or service sector.

Our project is to derive a decision-making process which output is a classification of most appropriate best practices for a specific process to be redesigned and where some redesign goals are addressed. The decision should take into account which redesign aspect is being investigated (i.e. to which component of our framework does

the best practice belong to), the known impact of the implementation of a best practice on a process, the established redesign goals for the process and the identified risks for the BPR implementation. The tool will not only provide the “best” best practice to apply, but also a ranking of all best practices for the process being redesigned.

In the sequel, section two will introduce the different aspects that should be taken into account when deciding which best practice should be implemented (i.e. the criteria). Section three introduces AHP as the multi-criteria decision-making method chosen for this study. Section four builds up the strategy for the implementation of BPR using AHP. Section five applies our findings to the case study of a Dutch municipality. Finally section six provides our conclusions and future work.

2 How to Decide Which Best Practice to Apply?

How do practitioners proceed? They certainly first observe the process and the organization. They identify with (top) managers the issues within the process and set up one or several goals for the redesign. They also consider implementation’s cost issues as this will impact the scope of the redesign. They then decide what to do.

In earlier work [14][23] we have established a framework for BPR implementation. The framework identifies seven components which need to be addressed during a redesign implementation: the customer, the information, the product, the operation view, the behaviour view, the organisation and the technology. The framework also provides some indications to the relative importance of the different components (cf. Table 2). Within this framework, practitioners have been using a number of best practices for BPR implementation. We gathered and classified these rules, identifying a list of top ten best practices in the field (They are classified in Table 3). The aim of this research was to initiate the development of a methodology for implementing BPR. So far the main results are:

- There are a number of **components** to address during a BPR implementation: The different components are not all equally vital for the redesign.
- There are a number of **popular best practices** that can be used for redesign *In general, and out of any context* (manufacturing/service, small/large organization, etc.
- We know the qualitative **impact** of each rule on four different dimensions: quality, time, cost and flexibility (positive/negative). **Figure 2** displays the example of the qualitative impact of the order assignment best practice.

Despite these results, there is still a long way to go before deciding which best practices to apply. For instance, the following information is still needed to decide which best practice to apply preferably:

- The BPR **goal**: Any redesign effort corresponds to a need for improvement on some specific areas. According to [12][13][16], goals usually fall in the following categories:
 - improve quality,
 - reduce costs,
 - reduce service time or production time,
 - improve productivity,

- increase revenue,
 - improve customer service,
 - use IT capabilities,
 - improve competitiveness.
- The **risk** factors identified for the BPR implementation: Before starting the redesign practitioners identify the factors that will challenge the redesign of the process. Some best practices might then become inappropriate because they would increase the identified risk. Research indicates for example that for BPR projects, top management commitment and managerial support are the most important factors.

Throughout the literature review, the following risk factors are often considered [1][6][12]:

- limited implementation time [9][11][21],
- poor information system architecture [7][9][11],
- limited funds [3][7],
- employee resistance [11],
- lack of managerial support and lack of top management commitment [2][8][11][21].

To summarize, deciding which best practice to apply is a complex process that involves looking at several criteria: the *component* the best practice belongs to, the best practice's *popularity*, its *impact* on a redesigned process, the initial redesign *goal* and the identified *risks*. In this research we investigate the usefulness of a multi-criteria decision making method for the choice of best practices and describe it hereafter.

3 A Multi-criteria Method Using AHP

The Analytical Hierarchy Process (AHP) is a method developed by Saaty [24] for complex multi-criteria problems for which quantitative and qualitative aspects must/could be taken into account. A survey on multi-criteria methods resulted in choosing AHP [19] as it is widely used to classify alternatives based on a range of criteria, ([20] have used it to select suppliers). It is a more descriptive and less normative analytical approach. The AHP process is performed in four phases:

(1) *Building a hierarchical process for the decision's problem:*

This method helps the decision-makers to structure the significant components of a problem in a hierarchical structure. This is based on the assumption that the identified entities can be grouped into disjoint sets. The elements in each group (also called level) of the hierarchy are assumed to be independent. The hierarchy of the decision-making process is defined by a quadruplet $\langle L1, L2, L3, L4 \rangle$ (cf. Figure 1) where:

- L1 = Global Objective;
- L2 = Criterion Level;
- L3 = Indicator Level;
- L4 = Alternative Level.

Then, the results are synthesized by decomposing complex decisions into a series of simple comparisons and arrangements (cf. Figure 1).

The following notations are used C_i = Criterion i ; I_{ik} = Indicator k of C_i ; A_j = Alternatives.

(2) *Pair wise comparison of each built hierarchical level's elements:*

AHP is based on determining a classification for the alternatives. The central 'ingredient' of the AHP method is comparisons. The pair wise comparison evaluates the relative importance of two elements for the decision maker. It contributes to the achievement of the adjacent higher level's objective. The AHP method scale of value is used [24]. It defines numerical values (1 to 9) corresponding to the importance of a factor against another factor. It is used for comparing qualitative data (Refer to Table 1).

(3) *Relative weight appraisal between the elements of each two adjacent levels which develops priorities for the alternatives.*

(4) *Relative weights aggregation of the different hierarchical levels to provide alternatives' classification of the decision:*

The pair-wise comparison being carried out, AHP calculates a vector of priority that classifies the alternatives in ascending or decreasing order. The classification by priority of the elements of the hierarchy level contributing to reaching an objective of the adjacent higher level is called 'relative weight' or 'order of priority'.

Table 1. Scale measurement for AHP [24]

Numerical Values	Definition
1	Equally important
3	Slightly more important
5	Strongly more important
7	Very strongly more important
9	Extremely more important
Reciprocals	Used to reflect dominance of the second alternative as compared with the first.

AHP has some disadvantages worth discussing here and mainly the possibility of Rank Reversal [4][5][10]. Even so, the method is still attractive for our purpose as the aim is not really to find *the* best practice but to get an idea about the most suitable ones for the redesign.

4 Building a Hierarchical Process

The first phase in applying the AHP method is to build a hierarchical process for BPR implementation (refer to phase (1), Section 3). The global objective is to find out the relevance of applying a best practice for a BPR implementation process (level 1 in Figure 1). We have identified in Section 2 a number of criteria that influence the choice of a best practice for the redesign and they are included in level 2 of Figure 1. For each of these criteria we also identify a number of indicators and will detail this in what follows (level 3 of Figure 1). In level 4 the different best practices (called alternatives in AHP) are assessed against the different indicators.

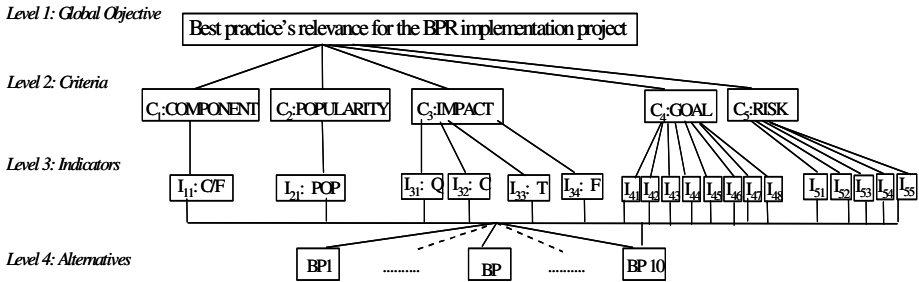


Fig. 1. Hierarchical classification for best practices evaluation

In what follows we list the criteria and their indicators:

1. Criterion C_1 : Component

For the component criteria we have identified one indicator only, I_{11} : Indicator C/F. It assesses a best practice by evaluating the importance in the framework of the component it belongs to. In [14] we have classified the components (refer to Table 2). We use the AHP scale to appreciate the importance of each component for the redesign. This value will be used for all best practices that belong to that component.

2. Criterion C_2 : Popularity

For the popularity criteria we have identified one indicator only, I_{21} = Indicator popularity. It assesses a best practice by evaluating its position in a top ten list of best practices [14]. The higher the position, the higher the ranking. For example, the order assignment best practice was tenth in our ranking. It is then allocated the value 1.

Note that despite the variability of the popularity criteria over time (today popular, a best practice might become unpopular in the future) we still believe it is relevant as the next criteria assesses the impact of the best practice thus dissociating how a best practice is currently “perceived” (popularity) and its real performance (impact).

Table 2. Assigning component Indicator’s values

Components’ classification	AHP-scale	Interpretation
1. Customer	9	Extremely important
2. Information	7	Very important
3. Product	7	Very important
4. Operation view	5	Important
5. Behaviour view	5	Important
6. Organisation	3	Slightly important
7. Technology	3	Slightly important

3. Criterion C_3 : Impact

To assess the qualitative impact of a best practice on a process we use the results of the Devil’s quadrangle [15]. A best practice may impact the quality (Q), the cost (C), the time (T) or the flexibility (F) of the process. They become indicators I_{31} , I_{32} , I_{33} and I_{34} , for the criteria Impact. We use the following evaluation ranking: -2, -1, 0, 1, 2 to summarize: a negative correlation between the BP and the impact it has on the

quality, cost, time or flexibility of the redesigned process (values -2, -1), a neutral influence (value 0) and a positive impact, i.e. the BP will improve the process’s quality, time, flexibility or cost (values 1, 2).

For example, based on its corresponding devil’s quadrangle (Refer to Figure 2) we can assign the values $I_{31} (Q) = 2$; $I_{32} (T) = 2$; $I_{33} (C) = 1$ and $I_{34} (F) = 1$ to the best practice Order assignment. This translates into: the order assignment best practice has a strong positive impact on the redesigned process’s quality and time and only a mild one on its cost and flexibility.

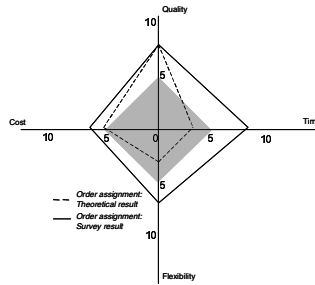


Fig. 2. Devil’s quadrangle for the Order assignment best practice

4. Criterion C_4 : Goal

This criterion assesses which goals are important for the BPR project under study. As explained in section 2 there are a number of potential goals for a BPR implementation which all become indicators for the criteria goal. They are I_{41} (Indicator Improve Quality), I_{42} (Indicator Reduce costs), I_{43} (Indicator Reduce service time or production time), I_{44} Indicator (Improve productivity), I_{45} (Indicator Increase revenue), I_{46} (Indicator Improve customer service), I_{47} (Indicator Use IT capabilities), I_{48} (Indicator Improve competitiveness). The AHP scale is used for this qualitative evaluation. For a studied BPR project we pair wise evaluate goals’ importance for that project, i.e. some goals are not relevant while others are the focus of the redesign.

5. Criterion C_5 : Risk

This criterion assesses for each identified potential risk, the extent to which applying a best practice will increase or not the identified risk for the implementation. We know that some risks are potentially more damageable than others for a BPR implementation. For example top managerial commitment is really important.

In section 2, we have identified the following risks which become indicators for the risk criterion: I_{51} (Indicator limited implementation time), I_{52} (Indicator poor information system architecture), I_{53} (Indicator limited funds, employee resistance), I_{54} (Indicator lack of managerial support) and I_{55} (Indicator lack of top management commitment). We use the following qualitative measurement scale -2, -1, 0, 1, 2 to summarize: a negative correlation between the BP and the risk (values -2, -1), a neutral influence (value 0) and a positive impact, i.e. the BP will help overcome that risk (values 1, 2).

A summary of the different criteria and their indicators is provided in Figure 1.

5 BPR Implementation Decision-Making: A Case Study

The process's organization we would like to redesign is a local municipality of 90,000 citizens in the Netherlands, in particular its Urban Management Service responsible for sanitation, parking facilities, green spaces, and city districts. This service employs over 300 civil servants. The process to be redesigned is the *Invoice processing* workflow. The municipality handled about 10,000 invoices in the years 2000 and 2001. In [14] we have described a simulation-based methodology to assess which best practices to implement in this case. It will be interesting to compare those results with the one that the decision-making algorithm will be providing. In the sequel we describe the different values we assign in relation to this case study and needed by the algorithm to deduce results. The values are chosen on the basis of our previous work [14] [23] and our own appreciation related to the addressed case study. Some values have been adjusted by the AHP algorithm to insure consistency amongst the values. A sensitivity analysis need further be performed to test the impact of values' changes on the overall result.

5.1 Level One

In this level we indicate what the global objective is. This depends on the project under study. Ten best practices have been selected for this case study (refer to Table 3). Our purpose is to find out, considering the case study's goals and risks, which best practice(s) should be recommended for implementation.

5.2 Level Two

In this level we assess the importance of criteria against each other (pair wise evaluation). This assessment is independent of the project under study and will be performed once and permanently entered in our AHP algorithm. It will be hidden from the users of the method. Table 1 is used as a scale of appreciation (1-9) as required by the AHP method. Results are summarized in Figure 3. In this figure, for example, Popularity is indicated as extremely more important than Component (value 9) and Impact is indicated as strongly more important than risk (1/5).

	<i>Component</i>	<i>Popularity</i>	<i>Impact</i>	<i>Goal</i>	<i>Risk</i>
<i>Component</i>	1				
<i>Popularity</i>	5	1			
<i>Impact</i>	7	1	1		
<i>Goal</i>	9	7	5	1	
<i>Risk</i>	5	1	1/5	1/9	1

Fig. 3. Pair wise comparison of the criteria

5.3 Level 3

In this level we assess for each criterion the importance of the indicators against each other (pair wise evaluation). This is not relevant for the component and popularity

criteria as they are described by one indicator only. Table 1 is used as a scale of appreciation (1-9) as required by the AHP method.

Impact Criteria

This assessment is independent of the project under study and will be performed once and permanently entered in our AHP algorithm. Results are summarized in Figure 4.

	$I_{31}(T)$	$I_{32}(Q)$	$I_{33}(F)$	$I_{34}(C)$
$I_{31}(T)$	1			
$I_{32}(Q)$	1/5	1		
$I_{33}(F)$	1/7	1/3	1	
$I_{34}(C)$	1	5	5	1

Fig. 4. Pair wise comparison of Impact’s indicators

Goal Criteria

This assessment depends on the project being studied and will have to be performed by practitioners if they use this method. For our case study, time and cost are the most popular redesign indicators for the criteria goal. Results are summarized in Figure 5.

	I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}
I_{41}	1							
I_{42}	7	1						
I_{43}	9	5	1					
I_{44}	1	1/7	1/9	1				
I_{45}	1	1/7	1/9	1	1			
I_{46}	1	1/7	1/9	1	1	1		
I_{47}	1	1/7	1/9	1	1	1	1	
I_{48}	1	1/7	1/9	1	1	1	1	1

Fig. 5. Pair wise comparison of Goal’s indicators

Risk Criteria

This assessment depends on the project being studied and will have to be performed by practitioners if they use this method. For our case study, Limited budget was the most threatening risk for the criteria Risk. Results are summarized in Figure 6.

	I_{51}	I_{52}	I_{53}	I_{54}	I_{55}
I_{51}	1				
I_{52}	1	1			
I_{53}	7	7	1		
I_{54}	1	1	1/7	1	
I_{55}	1	1	1/7	1	1

Fig. 6. Pair wise comparison of Risk’s indicators

5.4 Level 4

In this level we assess for each best practice the appropriate values for the indicators. The components values are derived from the classification in Table 2. The popularity values simply reflect the best practice position in the top ten table of best practices. The Impact values are derived from the devil’s quadrangles for each best practice. We have appreciated the risk’s indicators values. Finally, for the goal criterion’s indicators we have assigned a value of 1 on the AHP scale, indicating that we remain neutral in our judgment about this criterion as we want the decision-making algorithm to derive values for us. Table 3 indicates the values assigned to each best practice’s indicators.

Table 3. Indicator’s values for selected best practices

Best practice	Component C/F	Popularity popularity	Impact				Risk					Goal All indicators
			Q	T	F	C	A	B	C	D	E	
1. Task elimination	5	10	1	2	0	2	1	0	0	0	0	1
2. Task composition	5	9	1	2	1	1	0	0	0	0	0	1
3. Integral Technology	3	8	2	2	2	2	-2	-2	-1	0	0	1
4. Empower	3	7	1	2	2	1	-1	0	1	0	0	1
5. Order assignment	3	6	2	2	1	1	0	0	0	0	0	1
6. Resequencing	5	5	2	2	0	1	0	0	0	0	0	1
7. Specialist-generalist	3	4	2	2	2	2	-2	0	-1	0	0	1
8. Integration	9	3	2	2	1	1	-2	-2	0	0	0	1
9. Parallelism	5	2	0	2	0	1	-1	0	0	0	0	1
10. Numerical involvement	3	1	1	2	1	1	-1	0	1	0	0	1

5.5 Results

The application of the AHP algorithm delivered a ranking of best practices as shown in Table 4. It advises the implementation of the integration, task elimination and task composition/resequencing best practices as first choices. Apart from the integration

Table 4. Best practices classified by AHP for our case study

1. Integration
2. Task elimination
3. Task composition AND Resequencing
4. Parallelism
5. Order Assignment
6. Empower AND Numerical involvement
7. Specialist - Generalist

best practice, this is actually the set preferred in our earlier described case study using simulation [14]! The integration was originally identified as an applicable best practice but did not deliver good results on the simulation model. An explanation of the differences could be that the simulation did not take all the components of level 2 into account, such as e.g. the risk factor.

6 Conclusion

We see the presented parameterisation and application of the AHP-algorithm as a first step towards the development of decision-making strategy for BPR implementation. Based on earlier work of ourselves and others, we came up with substantiated comparisons and weights. Despite the well-known disadvantage of AHP in terms of effort required to rate criteria and options, the rating is finally restricted to level 1, goal and risk criteria in level 3 and only the risk values for each alternative as the other criteria are independent of the case under study. Our method was implemented in Java but is still in a prototype stage and needs improvement to be fully used in a business environment. Using a case study, we showed the feasibility and potential of the approach. Another case study is currently in progress. Furthermore, we will need to apply the method to all best practices (29) and not only to the top ten ones. A closer analysis of its outcomes should give us insight into the areas that could be improved. A major criterion to be included in our approach is *process characteristics*, identifying the set of properties of the current process that needs to be improved (see [18]).

References

- [1] Al-Mashari, M., Zairi, M.: BPR implementation process: an analysis of key success and failure factors. *Business Process Management Journal* (1999) Vol. 5, No. 1, 87-112.
- [2] Alter, A.: The corporate make-over, *CIO* (1990) Vol. 4, No. 3, 32-42.
- [3] Bashein, B., Markus, M., Riley, P.: Precondition for BPR success and how to prevent failures, *Information Systems Management* (1994) 7-13.
- [4] Belton, V., Gear, T.: On a Shortcoming of Saaty's Method of Analytic Hierarchies, *Omega*, (1983) Vol. 11, 228-230.
- [5] Bouyssou, D., Marchant Th., Perny P., Pirlot M., Tsouki`as A., Vincke Ph.: Evaluation and Decision models: a critical perspective", *Kluwer*, (2000).
- [6] Crowe, T.J., Fong, P.M., Bauman, T.A., Zayas-Castro, J.L.: Quantitative risk level estimation of business process reengineering efforts. *Business Process Management Journal*, (2002) Vol. 8, No. 5, 490-511.
- [7] Davenport, T.: *Process Innovation: Reengineering Work Through Information Technology*, (1993) Harvard Business School Press, Boston, MA.
- [8] Davenport, T., Short, J.: The new industrial engineering: information technology and business process redesign, *Sloan Management Review* (1990) Vol. 31, No. 4, 11-27.
- [9] Davidson, W.: Beyond re-engineering: the three phases of business transformation, *IBM Systems Journal* (1993) Vol. 32, No. 1, 65-79.
- [10] Dyer, J.S., Saaty, T.L., Harker, P.T., Vargas, L.G.: Remarks on the Analytic Hierarchy Process, *Management Science*, (1990) Vol. 36, No. 3, 249-275.
- [11] Grover, V., Jeong, S., Kettinger, W., Teng, J.: The implementation of business process reengineering, *Journal of Management Information Systems* (1995) Vol. 12, no. 1, 109-44.

- [12] Guimaraes, T. and Bond, W.: Empirically assessing the impact of BPR on manufacturing firms, *International Journal of Operations & Production Management* (1996) Vol. 16 No. 8, 5-28.
- [13] Hammer, M. and Champy, J.: *Reengineering the corporation: a manifesto for business revolution*. (1993) New York: Harper Business editions.
- [14] Limam Mansar, S., Reijers, H.A.: Best practices in business process redesign: validation of a redesign framework. *Computers in Industry* (2005), Vol. 56, No. 5, 457-471.
- [15] Limam Mansar, S., Reijers, H.A.: Best practices in business process redesign: Survey results amongst Dutch and UK consultants. *Proceedings of the 2004 Research Conference on Innovations in Information Technology*, Dubai 4-6 October (2004).
- [16] Maull, R.S., Tranfield, D.R. and Maull, W.: Factors characterising the maturity of BPR programmes, *International Journal of Operations and Production Management* (2003) Vol. 23 No. 6, 596-624.
- [17] Min, D.M., Kim, J.R., Kim, W.C., Min, D., Ku, S.: IBRC: Intelligent Bank Reengineering System. *Decision Support Systems* (1996), Vol. 18, No. 1, 97-105.
- [18] Nissen, M.E.: Redesigning reengineering through measurement-driven inference. *MIS Quarterly* (1988) Vol. 22, 509-534.
- [19] Ounnar, F.: *Prise en compte des aspects décision dans la modélisation par réseaux de Petri des systèmes flexibles de production*, PhD dissertation, National Polytechnique de Grenoble, (1999).
- [20] Ounnar, F., Pujo, P.: Supplier evaluation process within a self-organized logistical network. *International Journal of Logistics Management*, Accepted for publication, (2005).
- [21] Randall, A. (1993), *Business process redesign: how to do it*.
- [22] Reijers, H.A. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*. Springer-Verlag, Berlin, 2003.
- [23] Reijers, H.A., Limam Mansar, S.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega* (2005) Vol. 33, No. 4, August 2005, 283-306.
- [24] Saaty, T. *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

Applying Specialization to Petri Nets: Implications for Workflow Design

George M. Wyner¹ and Jintae Lee²

¹ Boston University School of Management, Boston, MA 02215, USA
gwyner@bu.edu

² Leeds School of Business, University of Colorado, Boulder, CO 80309, USA
Jintae.Lee@Colorado.Edu

Abstract. Inheritance has been suggested as a tool for managing changes in workflow systems. Van der Aalst and Basten [1] have identified four types of inheritance for workflows using a representation based on Petri nets. While they capture intuitions important for business process redesign, they suffer from their inability to state class-level constraints. This paper illustrates this limitation and proposes an extension that accommodates the class semantics and enables explicit representation of constraints on what variations in workflow are consistent with the original workflow. It also shows that the proposed approach subsumes the Van der Aalst and Basten's four inheritance types under a single framework and overcomes their limitation.

1 Introduction

In light of the increasing need to adapt to ever-changing requirements, the workflow community has identified as a priority the ability to redesign existing workflow definitions while preserving a set of constraints. In particular, Van der Aalst and Basten [1] have proposed four types of inheritance rules for workflows using a variant of Petri net called Workflow Process Definition (WPD). However, an analysis of the semantics of specialization reveals that a WPD, or more generally a Petri net, may precisely define the behavior of the current workflow but does not adequately define the features that must be preserved when it is specialized. We describe an approach that overcomes this limitation in the later sections, but first elaborate on the problem that motivated this research in the rest of this section.

Consider the workflow process definition for reviewing a loan given in Fig. 1. We have simplified this process considerably as it is intended for illustration only:

Suppose that we need to enhance or revise this workflow to accommodate a new technology (e.g. eCommerce) or a new regulation (e.g. Sarbanes-Oxley). It would then be nice to have a method that can tell us whether the revised workflow still reinforces the constraints that needs to be preserved (e.g. that the cash cannot be dispersed without the loan approval). Consider, for example,

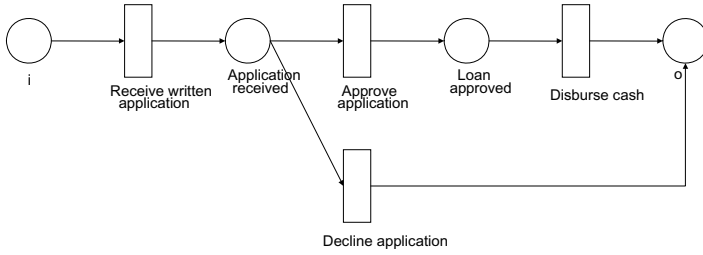


Fig. 1. Workflow process definition for *reviewing a loan*

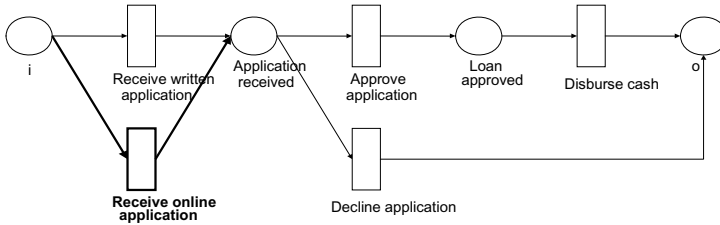


Fig. 2. Specialization consistent with original workflow

two possible extensions of the workflow in Fig. 1: one that adds a possible on-line application step (Fig. 2) and one that adds an "expedite application" step (Fig. 3). We would like then a method that can tell us that the workflow in Figure 2 is acceptable but the one in Fig. 3 is not because the latter does not preserve the constraint that the loan must be approved before dispersed. As will be discussed in Section 2, Van der Aalst and Basten [1] (henceforth referred to as VdAB) have proposed a solution to this problem by defining workflow specialization rules. Under these rules, any specialization of a workflow would then be substitutable for the original workflow. In particular, of the four specialization types defined in [1], Workflow 2 (Fig. 2) would be a specialization of Workflow 1 (Fig. 1) under the *protocol inheritance rule*; thus the former would be substitutable for the latter. However, a problem with this solution is that Workflow 3 (Fig. 3) also qualifies as a specialization under the same rule even though it skips the loan approval process. On the other hand, the *projection inheritance rule* disqualifies Workflow 3 as a non-specialization but then also disqualifies Workflow 2 as well. In fact, as will be shown in Section 3, none of the four specialization rules proposed by VdAB can define Workflow 2 as a valid specialization while disqualifying Workflow 3 as invalid.

The rest of the paper presents an extension of Petri net that enables class-level representation of constraints to be preserved and a notion of specialization that builds on this representation to overcome the above limitation of the VdAB approach. This notion of specialization is also consistent with the the subsumption-based definition of specialization that is more traditional in object-oriented framework. To do so, we first describe the formal apparatus of the VdAB

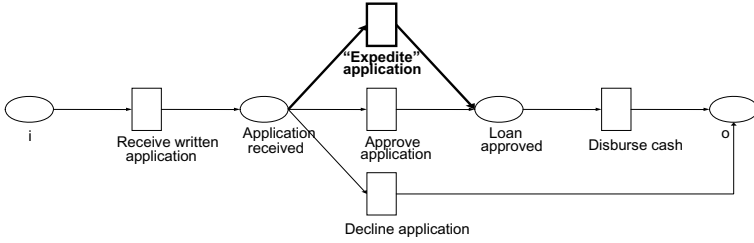


Fig. 3. Specialization inconsistent with original workflow

approach (Section 2) and its limitations (Section 3). We then describe our approach that extends the VdAB work (Section 4) and discuss how it overcomes its limitation (Section 5). We conclude the paper with a brief discussion of the related work (Section 6) and a summary of the contribution (Section 7).

2 The Van der Aalst & Basten Approach to Petri Net Specialization

Van der Aalst & Basten [1] (henceforth referred to as VdAB) define four kinds of inheritance for workflow process definitions (WPDs).¹ A WPD is a sound WF-net. A WF-net is an L-labeled P/T-net $N = (P, T, F, \ell)$. P here refers to the places in the net. T refers to the transitions. $F \subseteq (P \times T \cup T \times P)$ is a relation which specifies directed arcs between transitions and places. $\ell : T \rightarrow L$ assigns labels (from L , a set of labels). A WF-net has an initial input place i (corresponding to the initial creation of a new case to be handled by the workflow), an output place o (corresponding to the completion of a case), and is "strongly connected."

A sound WF-net satisfies the following requirements: when a case is completed (token in o), there are no other tokens, any reachable marking can lead to completion, and all transitions can be reached from the starting state. We will make use of the following additional results and notations introduced by VdAB:

The marking of a WPD is represented in the form $[p^j q^k \dots]$, where p, q (and additional terms) correspond to places in P and the exponents correspond to the number of tokens in each place. In particular, $[i]$ corresponds to a net with a single token in the input place, and similarly $[o]$ corresponds to a net with a single token in the output place.

The preset of a node $e \in P \cup T$ is denoted by $\bullet e = \{g \in P \cup T \mid gFe\}$. Similarly, the postset of e is denoted by $e\bullet = \{g \in P \cup T \mid eFg\}$.

Given a net N with marking s , one or more transitions t may be enabled so that one of the enabled transitions can fire. To be enabled a transition t must have a token in each place in its preset (i.e. $\bullet t \leq s$). We then say $(N, s) [t]$.

¹ Throughout this paper we adopt the terminology and definitions used in [1]. In the interests of brevity we include here only the notation and terminology essential to our arguments. Please see [1] for a full formal treatment of workflow process definitions.

An enabled transition may fire in which case the marking of the net becomes $s - \bullet t + t \bullet$. A firing sequence σ is a sequence of transitions $\sigma : [0, 1, \dots, n - 1] \rightarrow T$ such that $\sigma(0)$ is enabled at the start and each transition enables the one which follows. If such a sequence is enabled by marking s , we say $(N, s) [\sigma]$.

While we will make some use of these formalisms, it is impractical to give here a detailed exposition of the inheritance relationships themselves so we must content ourselves with a very informal exposition. This informal approach will be sufficient for the proof sketches in section 5 below:

VdAB start with the premise that a workflow specialization should add additional capabilities (i.e. new transitions and possibly places) while preserving all the capabilities of the original workflow. This leads to two distinct kinds of specialization. Let $w_1 = (P_1, T_1, F_1, \ell_1)$ and $w_2 = (P_2, T_2, F_2, \ell_2)$ be two WPDs. Then:

Protocol inheritance is satisfied if there is a set of transitions in T_1 which, if removed from w_2 , yield a net which is behaviorally equivalent to w_1 . In this case we say that $w_2 \leq_{pt} w_1$. The intuition is that w_2 can simulate w_1 by blocking these “extra” transitions.

Projection inheritance is satisfied if there is a set of transitions in T_1 which, if ignored (i.e. removed from the execution trace), yield a net which is behaviorally equivalent to w_1 . In this case we say that $w_2 \leq_{pj} w_1$. The intuition is that w_2 can simulate w_1 by hiding these “extra” transitions.

Protocol/projection inheritance is satisfied if $(w_2 \leq_{pt} w_1) \wedge (w_2 \leq_{pj} w_1)$ in which case we say that $w_2 \leq_{pp} w_1$.

Life-cycle inheritance is satisfied if we can identify one set of transitions to be blocked and another to be hidden in w_2 such that the resulting net is behaviorally equivalent to w_1 , in which case we say that $w_2 \leq_{lc} w_1$.

3 Analysis of Van der Aalst & Basten Approach

We draw on our previous work on process specialization to analyze the VdAB proposal. Our approach [2,3] is based on the view that specialization is a subsumption relationship. We formalize this approach as follows:

Consider some language or graphical representation intended to describe features of our world. For the moment we don't concern ourselves with the internal structure of this representation (i.e. its syntax). We will just consider the set \mathcal{W} of well formed expressions in this language.

Let D be a set of objects which together constitute a domain of interest and M be a map from \mathcal{W} to $\mathcal{P}(D)$ (the powerset of D). Then given M , each w in \mathcal{W} can be said to represent a class whose extension is $M(w)$.

We can then define the *specialization* relation \prec on \mathcal{W} :

$$w_1 \prec w_2 \leftrightarrow M(w_1) \subseteq M(w_2) \tag{1}$$

This definition formalizes the subsumption approach to specialization: w_1 is a specialization of w_2 if and only if all instances of w_1 are also instances of w_2 . In other words: the extension of w_1 is a subset of the extension of w_2 .

It follows that to define specialization for some representation \mathcal{W} we need to identify the target domain D and also the mapping M . In other words, we need to be able to say what counts as an instance for any description w in \mathcal{W} . Note also that specialization is a property of the classes represented by \mathcal{W} and is not a property of instances in D since it's a relationship among subsets of D . Finally, note that for a given \mathcal{W} and D there are many possible mappings M , each of which yields a different specialization relationship on \mathcal{W} .

We will refer to $\langle \mathcal{W}, D, M \rangle$ as defining an *extension semantics* on \mathcal{W} . Any extension semantics E defines the associated specialization relationship \prec_E . We will omit the subscript when the context is clear. Conversely, given some proposed specializing transformation, we require that there be a consistent extension semantics. We now apply this approach to the VdAB results on workflow inheritance.

3.1 Specialization of Workflow Process Definitions

Our discussion of specialization above implies that any notion of specialization for WPDs will require us to identify an extension semantics. We adopt the approach that each instance of a Petri net is itself a Petri net, typically an exact copy. This approach actually fits well with how Petri nets are used in workflow systems, as VdAB observe [1, pp. 157-8]. For example, in a loan approval workflow, when each new loan arrives, a new copy of the loan approval WPD would be instantiated and initialized. At any time there may be multiple such instances, all having the same set of places and transitions but each corresponding to a different loan application likely in a different state with a different execution history. We will thus adopt the approach that D is itself \mathcal{W} (the set of all WPDs) and M is a map from \mathcal{W} to $\mathcal{P}(\mathcal{W})$.

We observe that a WPD should at least include exact copies of itself as instances, that is $w \in M(w)$. It turns out, however, that if w has specializations (other than itself), then those specializations will also correspond to instances in $M(w)$.

Proposition 1. *Membership in $M(w)$ should include the set of all WPDs which are specializations of w .*

Proof. $w_1 \prec w \rightarrow M(w_1) \subseteq M(w)$. Since, as noted above, $M(w)$ should at least include w itself, it follows that $w_1 \in M(w_1)$ and hence $w_1 \in M(w)$. \square

Based on Proposition 1 we will define $M(w)$ to be exactly the set of all specializations of w . Since VdAB have given formal relationships corresponding to each of the four types of inheritance, we can precisely state which WPDs are specializations of any workflow process definition. Thus we can immediately define M for each of these types of specialization:

Definition 1. $M_{LC}(w) = \{v | v \leq_{LC} w\}$, and similarly for the other three definitions.

3.2 Limitations of WPD as a Process Class Description

Once we adopt the notion that a workflow can be specialized, it follows that the original WPD above represents not only a particular workflow, but also a class of possible workflows. The question of which workflows are to be included in that class is a question about the intention of the WPD designer. The designer needs to choose one of the four possible VdAB specializations (and extension semantics) in order to specify which workflows ought to be considered as specializations (i.e. as consistent with the requirements embodied in the original design).

Thus we should consider which choice of extension semantics is consistent with the designer's intentions. Table below indicates which specializations are permitted under each of the four approaches to specialization proposed by VdAB. As indicated in the table, there is no VdAB specialization which permits the desired specialization while excluding the other.

One might argue that this is not a problem. That one can simply evaluate the two specializations permitted under (say) protocol specialization and determine which is acceptable. This might not be practical for a complex process with many such processes and furthermore it seems evident that none of the alternatives above really capture the implicit constraints intended by the designer.

To overcome this problem, we need a fundamental change in our approach to specialization, a new way to define WPD classes so that these important properties can be represented and preserved. In section 4 we propose such an approach.

Table 1. Specialization of the Loan Review Process

Inheritance Type	Apply online?	Skip approval?	Comments
Protocol (blocking)	Allowed	Allowed	New trans. can be blocked.
Projection (hiding)	Forbidden	Forbidden	New trans. lead to observable difference in behavior.
Protocol & Projection	Forbidden	Forbidden	Specializations do not satisfy protocol inheritance.
Life cycle	Allowed	Allowed	

4 Proposed Extension to Van der Aalst & Basten

We will extend the notation for WPDs to allow a finer-grained specification of class membership. This additional notation will have no bearing on the execution of a given workflow process definition but instead restricts the membership of its extension and hence the ways in which the WPD can be specialized. We will refer to such an extended WPDs as a *Workflow Process Class Definition* (WPCD).

The intuitive interpretation of our extension is that when you need to change the existing workflow in order to handle some additional requirements you need to know what you can't do if you want to be consistent with the intention of the existing workflow. That is, our approach provides support for handling exceptions to the existing workflow.

4.1 Defining WPCD Syntax

Definition 2. We define a WPCD as the tuple $(P, T, F, \ell, E_L, P_B, F_L)$, where:

(P, T, F, ℓ)	is a WPD (i.e. a sound WF - net)
$E_L \subseteq (P \cup T)$	is a set of "frozen" nodes
$P_B \subseteq P - E_L$	is a set of "externally frozen" places
$F_L \subseteq F$	is a set of "frozen" arcs

The behavior of WPCD as a workflow is entirely described by (P, T, F, ℓ) . The additional elements of the definition describe which variations are permitted, and hence which specializations will be permitted. We present below an informal interpretation of each of the new components.

E_L is a set of places and transitions which are "frozen," meaning that the preset and postset of these nodes cannot be changed in a specialization. For example, a specialization could ordinarily add a new transition (e.g. under protocol inheritance) but such a transition cannot be connected to a frozen place since that would involve a modification to its preset or postset. We represent a frozen node by adding two small boxes to the node's representation (circle or rectangle).

P_B is a set of places which are not frozen (transitions can be added) but which can only add "internal transitions." That is, any transition or sub-net which is connected to $p \in P_B$ does not connect to any other node in $P \cup T$. These externally frozen places are represented by adding a small triangle to the circle representing that place.

F_L is a set of arcs which cannot be refined. That is, a frozen arc cannot be modified by adding intervening places or transitions.

These new notations are shown in Figure 4 below.

Having defined this new kind of net, we now give it an appropriate extension semantics. Specifically we define the function M_{CD} for this net.

4.2 Defining Extension Semantics

The extension of a WPCD w_0 is defined to include every WPD w which satisfies the following four conditions:

1. *Life-cycle inheritance.* w is a valid instance of life-cycle inheritance for w_0 .
2. *Frozen elements.* The presets and postsets of all transitions and places marked as frozen are identical in w and w_0 .
3. *Externally frozen places.* Transitions added in w to the preset or postset of an externally frozen place do not connect to other places or transitions found in w_0 .

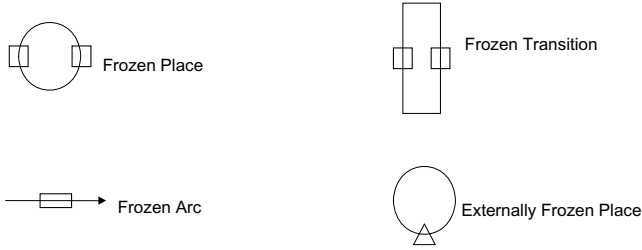


Fig. 4. WPCD Notation

4. *Frozen arcs.* Any arc marked as frozen in w_0 must also appear in w . That is, such arcs cannot be modified by adding intervening places or transitions.

5 Analysis of Extension

It turns out that the WPCD notation is sufficiently rich to capture the restrictions a designer may wish to impose on membership in the class associated with a WPD and hence with what counts as a specialization. In particular we will show the following:

1. For the *Loan Review* example (Figure 1 above), we will show that these semantics suffice to permit the specialization in Figure 2 while excluding the specialization in Figure 3.
2. We will show that all four kinds of VdAB specialization can be handled as special cases of this semantics.

5.1 The Loan Approval Example Revisited

Figure 5 below shows how the Loan review process can be represented as a WPCD. The only change from the original (Figure 1) is that the place *Loan approved* has been frozen. This means that the only way that a token can arrive in *Loan approved* is via the existing transition *Approve application*. Since *Disburse cash* has *Loan approved* in its preset, this means that *Disburse cash* cannot fire unless *Approve application* has fired previously.

5.2 Representing VdAB Inheritance with WPCDs

Each type of inheritance identified by VdAB can be captured by constraining the use of various elements of the WPCD notation. In what follows we will limit ourselves to describing the form of WPCD required for each inheritance type along with an informal argument that each form will result in the desired form of inheritance. Protocol and projection inheritance are illustrated using the review loan workflow.

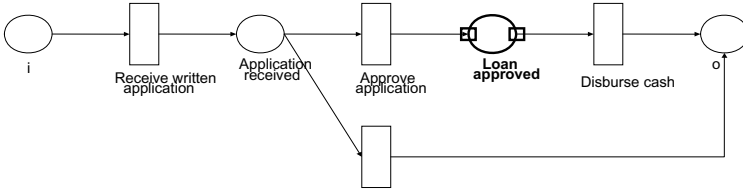


Fig. 5. Representation of the Loan review process using WPCD

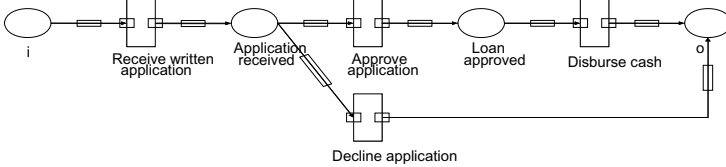


Fig. 6. Protocol Inheritance

We can specify *life-cycle inheritance* by refraining from using any of the additional notations introduced in WPCD. Under those circumstances the semantics for $M()$ reduces to condition 1 (see the definition in section 4.2 above) which is equivalent to \leq_{lc} (life-cycle inheritance).

We can specify *protocol inheritance* by requiring that all transitions and arcs be frozen. Since arcs are frozen, new transitions can only connect to the existing net through its places. If we block these additional transitions then the only changes in the state of the net must result from the firing of the original transitions. Since we have frozen these transitions, their presets and postsets will be unchanged and thus their firing rules will be unchanged. As a result the behavior of any specialization obtained under this WPCD will be equivalent to the original net when new transitions are blocked.

We can specify *projection inheritance* by requiring that all places be externally frozen. This allows specializations corresponding to three kinds of changes: (1) A subnet can be connected to a single place. (2) A subnet can be connected to one or more transitions. (3) A subnet can be inserted into an arc (between an existing place and transition). These changes correspond to the three "transformation

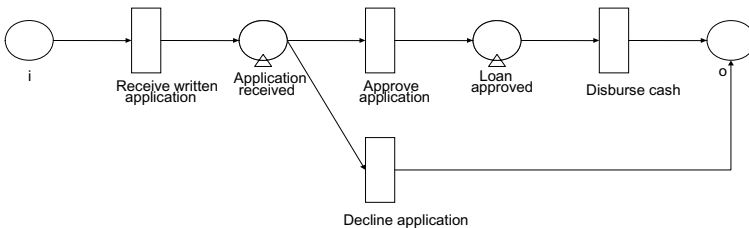


Fig. 7. Projection Inheritance

rules” PPS, PJS, and PJ3S which VdAB have shown to result in projection inheritance.

By following rules for *both* Protocol and Projection inheritance as described above, any specializations of the WPCD will satisfy both kinds of inheritance and thus will satisfy *Protocol/projection* inheritance.

6 Related Work

This study builds on previous work of ours. In [4], we propose the use of process specialization as a way of categorizing and analyzing processes for process design/redesign. In [3] we point out the importance of formalizing the notion of process specialization and a need for its own set of semantics that is different, though compatible, with that for object specializations. We apply this approach to state diagrams and later to dataflow diagrams [2].

In addition to Van der Aalst and Basten’s work on workflow inheritance [1] our work is closely related to Nierstraz’s work on active object subtyping [5], and Schrefl & Stumptner’s work on Object Life Cycle specialization [6]. The relation between our approach and the first two are discussed in [3]. Schrefl and Stemptner define “object life cycle” as “an overall description of how instances evolve over time.” After pointing out the need for criteria for object life cycle specialization, they present them in the context of Object Behavior Diagram [7]. This work is similar to ours in many respects. However, it differs from our study in that it treats a process as part of an object (i.e. its behavior) and focuses on behavioral consistency, whereas in our approach a process is treated as a class with its own extensions.

7 Conclusions

This paper builds on a growing body of research on workflow inheritance, which can be a powerful tool for managing changes in workflow system design. As demonstrated in the object-oriented modeling and programming area, inheritance enables us to introduce changes while preserving a given set of constraints. In order to do so, we need to formalize the notion of inheritance among workflow definitions and identify the rules governing it.

The recent work of Van der Aalst and Basten [1] has provided us with a solid framework within which to tackle this task. In particular, they have given us the behavioral definition of inheritance and four types of inheritance rules.

This paper builds on this work by proposing a diagrammatic extension of the Petri net that enables us to use it as a class representation without affecting its behavioral characteristics. With this extended formalism, we showed the comprehensiveness of our approach by identifying a set of syntactic rules which can be adapted to each of Van der Aalst and Basten’s four inheritance rules. We showed the potential usefulness of our approach by applying it to a simple loan processing example.

In addition to the need for a complete formal analysis of our proposed extensions, our current results have a number of limitations and unresolved issues that set the stage for future work:

1. Are the notations we propose for the Workflow Process Class Definition complete or at least rich enough to express the sorts of process requirements we would want to capture in practice? One issue in particular is whether it should be possible to freeze the preset or postset of a node individually (our current approach freezes them both). For example, in the loan review process WPCD shown in Figure 5 we stipulate that loans can only be approved by firing the transition *Approve application*, but we also disallow alternatives to *Disburse cash* as the next step in the process. If we could freeze just the preset to *Loan approved*, then we would still prevent cash from being disbursed without loan approval but would allow for alternatives to immediate disbursement of cash. Beyond this specific issue, it would be interesting to identify other types of constraints than those discussed so far (for example, specifying a disjunction of transitions such that one of them must be fired) and then examine how our approach might be used to represent and preserve such constraints through specialization.
2. Do the WPCD notations constitute a minimal set or is there a smaller set of notations which is as expressive?
3. The current analysis is based on trace equivalence rather than the more discriminating (and widely preferred) notion of bisimulation [8].
4. We do not fully consider the role that decomposition ought to play in workflow specialization. Our previous work suggests that decomposition is often an important aspect of the specialization of process models [2,3].

We plan to address these issues in future work as well as to explore how the approach can be applied to actual workflow redesign situations.

Acknowledgments

The first author would like to thank the Systems Research Center and the Dean's Office at Boston University's School of Management for their support of this research. Both authors wish to thank the BPD 2005 workshop participants for their insightful comments which have contributed to this paper.

References

1. van der Aalst, W.M.P., Basten, T.: Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science* **270** (2002) 125–203
2. Lee, J., Wyner, G.M.: Defining specialization for data flow diagrams. *Information Systems* **28** (2003) 651–671
3. Wyner, G.M., Lee, J.: Process specialization: Defining specialization for state diagrams. *Computational and Mathematical Organization Theory* **8** (2002) 133–155

4. Malone, T.W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Quimby, J., Osborn, C.S., Bernstein, A., Herman, G., Klein, M., O'Donnell, E.: Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science* **45** (1999) 425–443
5. Nierstrasz, O.: Regular types for active objects. In Paepcke, A., ed.: *Conference on Object-Oriented Programming: Systems, Languages, and Applications*, Washington, D.C., Association for Computing Machinery (1993) 1–15
6. Schrefl, M., Stumptner, M.: Behavior-consistent specialization of object life cycles. *ACM Transactions on Software Engineering and Methodology* **11** (2002) 92–148
7. Kappel, G., Schrefl, M.: Object/behavior diagrams. In: *Proceedings of the 7th International Conference on Data Engineering (ICDE'91)*, Kobe, Japan, IEEE Computer Society Press (1991)
8. Van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. *Journal of the ACM* **43** (1996) 555–600

“Intelligent” Tools for Workflow Process Redesign: A Research Agenda

Mariska Netjes, Irene Vanderfeesten, and Hajo A. Reijers

Technische Universiteit Eindhoven, Department of Technology Management,
PO Box 513, NL-5600 MB Eindhoven, The Netherlands
m.netjes@tm.tue.nl, i.t.p.vanderfeesten@tm.tue.nl,
h.a.reijers@tm.tue.nl

Abstract. Although much attention is being paid to business processes during the past decades, the design of business processes and particularly workflow processes is still more art than science. In this workshop paper, we present our view on modeling methods for workflow processes and introduce our research aiming for the development of an “intelligent” software tool for workflow process redesign. This tool uses two approaches to redesign workflows: an evolutionary approach, focussing on local updates to a given process, and a revolutionary approach, starting with a clean-sheet of paper.

1 Introduction

Business Process Redesign (BPR) is a popular methodology for companies to boost the performance of their operations. In essence, it combines a radical restructuring of a business process with a wide-scale application of information technology [8]. A common practice to apply BPR is that management consultants encourage specialists, employers and managers within the setting of a workshop to think of alternatives to the existing business process or to think of completely new processes. The role of the external consultants is to manage the workshop and to stimulate people to abandon the traditional beliefs they may have about the process in question, e.g. using creativity techniques. A well-chosen delegation of internal specialists and managers should ensure that all expertise is available that is required to make a process design.

Popular as this approach may be, it is questionable whether it will lead to the best possible redesign. We identify the following problems with this approach:

- It is *subjective*: The identification of problem areas is strongly influenced by the composition of the workshop group and their individual expertise. Furthermore, it is difficult to assess for the facilitator to identify the opportunities of change.
- It encourages *high-level designs*: An abstraction from complex procedures and coordination mechanisms is made to more easily reach consensus.
- It is often *not reproducible*: Even if there is a rationale for design decisions, it is often difficult at a later stage to understand why a specific solution was favored by the workshop group.

For more background and a further discussion of these issues, see [20].

The concrete problem this research project addresses is the lack of sound scientific foundations for the way that business processes are redesigned in practice. The research project aims at an extension of the capabilities of an existing, widely used process modeling tool in industry with “intelligent” capabilities to suggest favorable alternatives to an existing process. We refer to “intelligent” because the tool should be able to produce good alternatives itself. Together with the initial process model, this approach also requires an explicit specification of the performance targets that are aimed for, e.g. a process with the lowest average operational cost or a process with the lowest average lead time. The proposed capabilities of the “intelligent” tool are twofold:

- On the one hand, the tool can suggest *evolutionary*, local updates to an existing workflow design. These updates only gradually improve its performance. In this evolutionary approach the existing process is taken as starting point and is gradually refined or improved. Improvements are suggested on the basis of ‘best practices’ that have accumulated in the literature on BPR over the last decade (see [22]).
- On the other hand, there is the possibility to generate a *revolutionary* new alternative of an existing workflow. A clean-sheet of paper is taken to design the complete process from scratch. This approach provides possibilities to “re-do” the whole process and make radical changes in the process model, on the basis of an analysis of the essential information processing underlying the process (see [21]).

The type of processes that will be redesigned with the “intelligent” tool are workflow processes [20,2]. Workflows are commonly found within administrative settings such as banks and insurance companies and typical examples are mortgage requests or damage claims. In this paper we use the evaluation of mortgage requests as a simple, but clear example to illustrate the two approaches.

This workshop paper is organized as follows. In the next section, an overview is given of relevant literature to clarify the background of this research. Further, the evolutionary (Section 3) and revolutionary approach (Section 4) are discussed in more detail. The paper ends with a conclusion.

2 Literature

In their seminal work [8], Hammer and Champy identified IT as a key enabler for redesigning business processes. This new role of IT “represents a fundamental departure from conventional wisdom on the way an enterprise or business process is viewed, examined, and organized” [9] and triggered many articles on the role of IT in the context of BPR. However, as pointed out in [7], most of the studies deal with conceptual frameworks and strategies - not with modeling and analysis of business processes with the objective of improving the performance of reengineering efforts.

The use of IT to actually support a redesign effort can take on various forms and a variety of tools is available in the market place. Kettinger, Teng and Guha compiled a list of 102 different tools to support redesign projects [10]. Building on this study, Al-Mashari, Irani and Zairi classified BPR-related tools and techniques in 11 major groups [3]. These groups cover activities such as project management, process modeling, problem diagnosis, business planning, and process prototyping. Gunasekaran and Kobu reviewed the literature from 1993-2000 and came to the following classification of modeling tools and techniques for BPR: (i) conceptual models, (ii) simulation models, (iii) object-oriented models, (iv) IDEF models, (v) network models, and (vi) knowledge-based models [7]. More recently, Attaran linked the various available tools to three different phases in a BPR program: before a process is designed, while the process is being designed, and after the design is complete [4]. In these phases, a tool can respectively act as a facilitator (e.g. as an inspirator for a new strategic vision), as an enabler (e.g. for mapping the process, gathering performance data, and simulation), and as an implementor (e.g. for project planning and evaluation). Using the classifications of these authors, our interest in this paper lies with the role of IT as an *enabler* of process design, more in particular “to help identify alternative business processes” [4], in the form of *knowledge-based models* that “facilitate the process of reengineering by minimizing the complexity of the modeling and analysis of BPR” [7].

There are various papers addressing the state of the art on BPR tools and techniques (e.g. [10,3]). Despite their vast supply, Nissen states that typically “such tools fail to support the deep reengineering knowledge and specialized expertise required for effective redesign” [16]. Similarly, Bernstein, Klein and Malone observe that “today’s business process design tools provide little or no support for generating innovative business process ideas” [5].

The most important reasons to work towards the development of an “intelligent” tool for process redesign is that new design alternatives can be developed easier [7], more cost-effectively [15], quicker [13,16] and more systematically [13,19,24]. At this point in time few tools qualify on these requirements. The ProcessWise methodology, although promoted as “advanced” and supported by an integrated tool, does not offer any guidance for the design itself [6]. Case based reasoning (CBR) systems are presented in [11,14]. They enable an efficient search and retrieval of earlier redesign solutions that hopefully fit the aims of a new BPR effort. However, it is the human designer who must still weigh their applicability and perform the adaptations to the current situation. Another drawback is that the cases are typically restricted to a certain business domain, e.g. banking.

More promising seems the approach on the basis of the MIT Process Handbook as presented in [13]. The process recombinator tool is implementing this approach [5]. Through the notions of (i) process specialization and (ii) coordination mechanisms, new designs can be systematically generated on the basis of an identified list of core activities. It is the end user who then can select the most satisfactory process. In contrast to the earlier CBR approaches, the existing

design knowledge extends over multiple business domains and the end user is supported in a meaningful way to generate alternatives.

In research that is associated to the MIT Process Handbook, a quite different yet promising approach is presented in [12,19]. It attempts to capture the *grammar* underlying a business process. Just like natural language consists of words and rules to combine these words, business processes are seen as consisting of activities that can be combined using rewrite rules. A clear advantage of this approach would be that different process variants can be systematically generated and explored. However, it seems difficult to identify the rules and constraints that apply for certain categories of processes and to represent these in a grammatical framework.

The last approach particularly worth mentioning here is the KOPeR tool described in [15,16]. The idea is that a limited set of process measures (e.g. process length, process handoffs, etc.) can be used to identify process pathologies in a given process (e.g. a problematic process structure, fragmented process flows, etc.). These process pathologies can then be matched to redesign transformations known to effectively deal with these pathologies. Although the tool does not generate new designs itself, e.g. by visualizing the effect of a suggested transformation on an existing design, experiments suggest that the tool “performs redesign activities at an overall level of effectiveness exceeding that of the reengineering novice” [17].

In summary, although the existence and importance is acknowledged of tools to support redesigners with the technical task of generating new process designs, few tools exist that can match this task. Elements common to the few existing “intelligent” redesign tools are (i) the use of earlier redesign cases, (ii) the exploration of a deeper process structure, and (iii) the application of general (i.e. non-business specific) transformation rules/best practices. In particular, the application of the latter two ingredients seems to lead to the most effective results.

3 Evolutionary Approach

This section addresses the evolutionary approach to workflow process design. The evolutionary approach is based on the application of general best practices or heuristic rules. The best practices support practitioners in developing a workflow design by making evolutionary, local updates to an existing design.

3.1 Introduction

Best practices are the basis for the evolutionary approach. A best practice has essentially the following parts: some kind of construction or pattern that can be distinguished in the existing workflow, an alternative to be incorporated for the redesign and a context-sensitive justification for this alternative. An extensive literature survey has taken place to collect all best practices for evolutionary process improvement of workflows (see [22]). Reijers and Limam Mansar [22] have identified 29 best practices. For each best practice the authors present a qualitative description, its potential effects and possible drawbacks. Next to this,

they present a business process redesign framework. Both the description of the best practices and the classification of the best practices based on the presented framework are starting points for the development of the “intelligent” tool. In the next sections we will give an example of the use of a best practice and present our research outline and issues.

3.2 Example of Best Practice

The evaluation of mortgage requests described in [1] will be used to illustrate the development of a design alternative. In the example only one best practice will be used. This best practice is based on the knock-out principle, used to decide whether a case should be accepted or rejected. We have chosen the knock-out best practice, because this best practice is already formalized and quantitatively supported by Van der Aalst [1]. The example is a simplified version of the example used by Van der Aalst [1] and describes the process of granting a mortgage for buying a house. The process consists of five checking tasks currently placed in the sequence: (A) ‘check salary of mortgagee’, (B) ‘check current debts’, (C) ‘check mortgage history’, (D) ‘check collateral’ and (E) ‘check insurance’. Each check has two possible outcomes: OK or NOK (i.e., not OK). If for a specific request the outcome of a task is NOK, the request is rejected immediately. The request is only accepted and the mortgage granted if all checks are positive. The process variables are stated in Table 1. We assume there are enough resources

Table 1. Process variables for the knock-out problem

Task	Reject probability	Processing time (min.)	Ratio
A	0.10	35	0.003
B	0.15	30	0.005
C	0.20	20	0.01
D	0.15	15	0.01
E	0.20	20	0.01

present in the process to avoid bottlenecks. The knock-out best practice will be used to find the ordering of the five tasks of the mortgage process with the lowest average lead time. The knock-out best practice states “order knock-outs in an increasing order of effort and in decreasing order of termination probability” [20]. Tasks should be ordered in descending order using the ratio ‘reject probability / processing time’ (see Table 1 for the ratios). Tasks with the same ratio should be ordered using the reject probability (in descending order) to obtain the process with the lowest average lead time. Using the knock-out best practice shows that two optimal sequential process designs exist: process E.C.D.B.A. and process C.E.D.B.A.

This example illustrates clearly the suitability of the knock-out best practice to improve a workflow design with checking tasks. It is easy to apply a best practice on a situation suited for it, but this does not mean that application of the best practices on real workflows will be easy. Identified issues will be addressed in the next section.

3.3 Research Outline and Issues

The best practices and the use in an “intelligent” tool are the main focus of the research on the evolutionary approach. The first step will be the selection of best practices that seem fruitful to be developed further for inclusion in the tool. A main selection criterion could be the level of performance improvement that is pursued with such a best practice. We will consider the performance dimensions costs, time, quality and flexibility and provide the user with the possibility to enter his performance goals for the redesign. Simulation seems to be a suitable performance analysis technique for investigation of the changes in performance when applying the various best practices to workflow processes. Issues related to the selection are the large number of best practices to be taken into consideration, the level of required information and obtaining this information (e.g. from the end users).

Some of the best practices are already formalized and quantitatively supported, but most of them are more qualitative by nature. The logical next step would be the formalization and quantification of the selected best practices. Formalized transformation rules are necessary for an automatic recognition of sub-optimal patterns within a workflow design and the replacement of such a pattern. The justification of a replacement is done with the quantitative support for the use of the best practice. Finally, an implementation of the formalized best practices in the “intelligent” tool should take place.

An important issue during the research will be combining the best practices in the tool. First of all, it might be difficult to incorporate different classes of best practices in the same tool, because each class requires different types of information. Secondly, not only insight in the effects of an individual best practice is necessary, but also on the interaction effects when applied together with other best practices.

4 Revolutionary Approach

This section focuses on a revolutionary approach to workflow process redesign, i.e. the process model is designed from scratch. This approach can also be classified as an approach that is based on the exploration of a deeper structure of the workflow product, as will be shown in the next sections.

4.1 Introduction

At the basis of the revolutionary approach are the principles of Product Based Workflow Design (PBWD) [20,21,23]. PBWD focuses on the workflow product that is produced during the workflow process, instead of on the process of production itself. As explained in the introduction, workflow processes are business processes in an administrative setting, in which a lot of information is processed. The structure of information processing can be captured in a tree structure, called the product data model (PDM). The PDM can be compared to the concept of a Bill of Material (BoM) in manufacturing [18]. Like a BoM for instance

describes the physical assembly of a car (i.e. a car is made out of an engine and a subassembly, which consists of four wheels and a chassis), the PDM describes the administrative “assembly” of for example a mortgage.

Until now PBWD has not been widely applied in practice because there is a lack of methods and tools to use this theory to redesign workflow processes. This research will focus on how the underlying structure of the process can be used in the redesign of the workflow process. The next section will amplify on the product data model for the mortgage process which we used earlier as an example. Next, we will shortly introduce our plans for future research in this area.

4.2 Example of Product Data Model

As a simple example we will elaborate on a product data model for the mortgage process that is described in [1]. Before the mortgage is approved five checks have to be executed. Of course all the checks, i.e. decisions, are based on information. The information in turn can be divided into several pieces of information that are built on other pieces of information or are provided by the applicant. In a PDM the pieces of information are called information elements and the different information elements are related to each other through operations.

We will now consider (a part of) the product data model for the mortgage process. The model is depicted in Figure 1. We have only elaborated the tree structure for the salary check in the mortgage process for reasons of clarity. The tree structures for the other checks are comparable to the one for the salary check.

We will explain the PDM from Figure 1 in a bottom-up way. The salary check starts with information on the income and expenses of the mortgage applicant and finally ends with the decision whether the salary is sufficient to buy the desired property. For instance the gross salary per month (information element number 19) and the percentage of holiday payment (20) together determine the gross salary per year (17). In a similar way the other information elements are determined and finally the check on salary (2) is performed. When the outcome for the check on salary is negative, the mortgage can be rejected immediately (indicated by the single link between “check salary” and “mortgage OK?”). When the check on salary is positive, four other positive checks are needed before the mortgage is approved (depicted by the connected links between the five checks and the mortgage decision).

4.3 Research Outline and Issues

The research on the revolutionary approach proposed by this position paper focuses on deriving workflow process models based on the product data model. The first step in this research project is to study the theory of PBWD more carefully in order to identify the most pressing issues. At this point in time some of the issues are already revealed.

In the first place there is a need for practical cases to apply the PBWD view. Until now, few investigations have been done to the retrieval of the product data model in a practical situation. Issues that arise here are the level of detail to

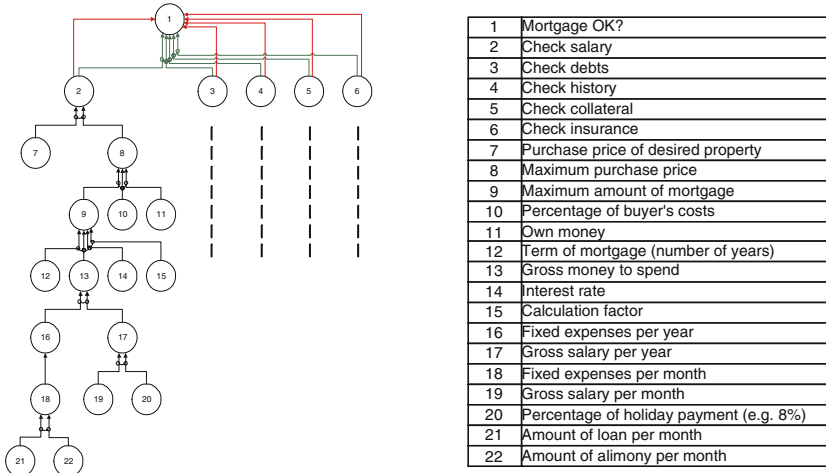


Fig. 1. Product data model for mortgage process

which the product data model has to be described, the amount of information available and needed, and the way in which data can be gathered (from systems, working instructions, laws, regulations, etc.) to retrieve the product data model. Another issue is the way in which a process model (consisting of tasks) can be derived from the product data model. At this point in time cohesion and coupling measures exist that more or less define the quality of a workflow design based on the manually clustering of information elements and operations [23]. The aim of this research is eventually to find an algorithm that (semi)automatically groups the information elements and operations of a product data model in activities. In this way we want to develop an objective and prescriptive method for redesigning workflow processes that can suggest favorable alternatives to a workflow process based on the product data model.

Moreover, at the moment the method only focuses on structural properties. Duration, costs and other performance indicators are not yet considered in the derivation of the best alternative process model. Next, the cohesion and coupling measures and the developed method should be validated by means of practical examples and/or experts. Finally, the developed prescriptive method has to be incorporated in a workflow redesign tool to support workflow designers. We also aim for the development of a prototype with this functionality.

5 Conclusion

In this paper we proposed two approaches to develop “intelligent” tools for workflow process redesign. The first approach focuses on local improvements of an existing workflow process through identified best practices. The second approach takes a clean-sheet of paper to design the workflow process focusing on a deeper process structure of the information that is processed in the workflow process.

By using these two approaches we think the most pressing difficulties that we identified in popular practice in workflow process redesign can be overcome. The approaches we aim for are objective, they aim for a more detailed view and they are reproducible.

References

1. W.M.P. van der Aalst. Re-engineering knock-out processes. *Decision Support Systems*, 30, pp. 451-468, 2001.
2. W.M.P. van der Aalst, K.M. van Hee. *Workflow management: models, methods and systems*. MIT Press, 2002.
3. M. Al-Mashari, Z. Irani, M. Zairi. Business Process Reengineering: A Survey of International Experience. *Business Process Management Journal*, 7(5), pp. 437-455, 2001.
4. M. Attaran. Exploring the Relationship between Information Technology and Business Process Reengineering. *Information & Management*, 41(5), pp. 585-596, 2004.
5. A. Bernstein, M. Klein, T.W. Malone. The Process Recombinator: A Tool for Generating New Business Process Ideas. In: T.W. Malone, K. Crowston, G.A. Herman, editors, "Organizing Business Knowledge: The MIT Process Handbook", MIT Press, Massachusetts, pp. 403-422, 2003.
6. P. Calvert. An Advanced BPR Methodology with Integrated, Computer-based Tools. In: B.C. Glasson, I.T. Hawryskiewicz, B.A. Underwood, R.A. Weber, editors, "Business Process Re-engineering: Information Systems Opportunities and Challenges", Elsevier Science, Amsterdam, pp. 161-170, 1994.
7. A. Gunasekaran, B. Kobu. Modelling and Analysis of Business Process Reengineering. *International Journal of Production Research*, 40(11), pp. 2521-2546, 2002.
8. M. Hammer, J. Champy. *Reengineering the Corporation*. London, Nicholas Brealy, 1993.
9. Z. Irani, V. Hlupic, G. Giaglis. Business Process Reengineering: A Design Perspective. *The International Journal of Flexible Manufacturing*, 12(4), pp. 247-252, 2000.
10. W.J. Kettinger, J.T.C. Teng, S. Guha. Business Process Change: A Study of Methodologies, Techniques, and Tools. *MIS Quarterly*, 21(2), pp. 55-80, 1997.
11. S. Ku and Y.H. Suh. An Investigation of the K-tree Search Algorithm for Efficient Case Representation and Retrieval. *Expert Systems with Applications*, 11(4), pp. 571-581, 1996.
12. J. Lee, B.T. Pentland. Grammatical Approach to Organizational Design. MIT Center for Coordination Science Working Paper 215, 4144, pp. 1-35, December 2000.
13. T.W. Malone, K. Crowston, J. Lee, B. Pentland. Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. *Management Science*, 45(3), pp. 425-443, 1999.
14. D.M. Min, J.R. Kim, W.C. Kim, D. Min, S. Ku. IBRC: Intelligent Bank Reengineering System. *Decision Support Systems*, 18(1), pp. 97-105, 1996.
15. M.E. Nissen. Redesigning Reengineering Through Measure-Driven Inference. *MIS Quarterly*, 22(4), pp. 509-534, 1998.
16. M.E. Nissen. An Intelligent Tool for Process Redesign: Manufacturing Supply-Chain Applications. *The International Journal of Flexible Manufacturing Systems*, 12(4), pp. 321-339, 2000.

17. M.E. Nissen. An Experiment to Assess the Performance of a Redesign Knowledge System. *Journal of Management Information Systems*, 17(3), pp. 25-43, 2001.
18. A. Orlicky. Structuring the Bill of Materials for MRP. *Production and Inventory Management*, december, 1972, pp. 19-42.
19. B.T. Pentland. Grammatical Models of Organizational Processes. In: T.W. Malone, K. Crowston, G.A. Herman, editors, “Organizing Business Knowledge: The MIT Process Handbook”, MIT Press, Massachusetts, pp. 191-214, 2003.
20. H.A. Reijers. Design and Control of Workflow Processes: Business Process Management for the Service Industry. *Lecture Notes in Computer Science* 2617. Springer-Verlag, Berlin, 2003.
21. H.A. Reijers, S. Limam Mansar, W.M.P. van der Aalst. Product-Based Workflow Design. *Journal of Management Information Systems*, 20(1), pp. 229-262, 2003.
22. H.A. Reijers and S. Limam Mansar. Best Practices in Business Process Redesign: An Overview and Qualitative Evaluation of Successful Redesign Heuristics. *Omega: The International Journal of Management Science*, 33(4), pp. 283-306, 2005.
23. H.A. Reijers, I.T.P. Vanderfeesten. Cohesion and Coupling Metrics for Workflow Process Design. *Proceedings of the 2nd International Conference on Business Process Management*, pp. 290-305, Potsdam, 2004.
24. E.S. Yu, J. Mylopoulos. From E-R to “A-R” - Modelling Strategic Actor Relationships for Business Process Reengineering. In: “Proceedings of the 13th International Conference on the Entity-Relationship Approach”. *Lecture Notes in Computer Science* 889. Springer-Verlag, Berlin, pp. 548-565, 1994.

Risk Management in the BPM Lifecycle

Michael zur Muehlen and Danny Ting-Yi Ho

Howe School of Technology Management,
Stevens Institute of Technology,
Castle Point on the Hudson, Hoboken, NJ 07030
{mzurmuehlen, tho2}@stevens.edu

Abstract. Business Process Management is considered an essential strategy to create and maintain competitive advantage by streamlining and monitoring corporate processes. While the identification of critical success factors for the management of business process related projects has been addressed by some research projects, the risks associated with these projects have received considerably less attention. This is a concern: Although BPM projects contain phases that relate to traditional software development and deployment projects, the application of risk mitigation strategies found in software engineering ignores the subsequent process management phases that follow upon the implementation and automation of processes. This paper provides an overview of risks associated with BPM projects along the phases of the BPM lifecycle. After a classification of the risks identified with individual lifecycle phases and transitions we discuss four strategies to deal with these risks: avoidance, mitigation, transfer, and acceptance. The outlook of this paper discusses how assessment frameworks such as CobIT and COSO can be applied to risk management in the context of BPM.

1 Motivation

The specification and improvement of corporate processes is a measure that helps functional organizations improve the hand-off points of work items between departments. After the majority of the 1990s reengineering projects put business processes at the center of reorganization strategies, process management has more recently been realigned with continuous improvement efforts that go back to the Total Quality Management initiatives of the 1980s, and continuous improvement efforts that have their roots in W. Edwards Deming's work in the 1950s.

Business Process Management covers the lifecycle of process discovery, specification, implementation, execution, monitoring and controlling. While corporate reorganization often focuses on the makeup of structural entities such as departments and divisions, the core processes enacted to deliver products are services tend to remain a core binding element for organizations. Consequently, structuring organizations around business processes is a popular topic both in management and the technical literature. A study conducted by Grover indicates that, even with enormous time and investment devoted, 7 out of 10 surveyed business process

projects failed [1]. Such a high failure rate implies that in addition to understanding what should be done in process reengineering projects, the avoidance of things that should *not* be done deserves equal attention. We are particularly interested in describing the risks that endanger the success of business process projects, such as those listed in [2, 3].

In this paper, we describe specific risks that BPM projects are exposed to along the BPM lifecycle. Based on four risk management strategies we discuss the options that a BPM project manager has in dealing with these risks. Finally, we outline the role of existing frameworks such as COSO and CobIT in identifying existing risks and planning for their mitigation.

2 Business Process Management

The general notion of the term business process is widely understood, but there exist almost as many definitions of the term as there are authors writing about the topic. In general, processes transform input into output along a path of activities, which may invoke or consume resources such as people or materials. Depending on the position of the process within the corporate supply chain, core and support processes can be distinguished. Core processes (sometimes called identity processes) are the main value-creating pipelines of an organization; they are triggered by interaction with external parties such as suppliers or customers, and their output is directed at consumers outside the organization. Support processes are mainly internal to an organization, and enable the execution of core processes. They do not produce results that are of direct value to customers or suppliers. To formalize these notions we propose the following definition of a process: A process is a sequence of activities that is necessary to manipulate an object of economic interest to the organization, and that achieves a specific goal. The components of a process are both the structure of the process (i.e. the control flow among activities, data flow dependencies, and business rules that cover constraints in the execution of the process), its goals, as well as ancillary elements such as resources, input and output.

Management in general is a cross-sectional function that controls the use of resources and choreographs the operational activities of the enterprise. Management functions follow a lifecycle of planning, organizing, staffing, directing and controlling, and budgeting. Business Process Management is the application of this management cycle to an organization's business processes. While Business Process Management has gained interest in industry over the last few years (compare e.g. [4]), its roots are not new. For instance, Levin proposed the ideas of automatic process control in physical processes to office work in 1956 [24].

Zairi and Sinclair see BPM as "a structured approach to analyze and continually improve fundamental activities such as manufacturing, marketing, communications and other major elements of a company's operations" [5]. Elzinga et al. emphasize that no matter how continuous improvement is performed, it must be based on the quality of products and services that will be evaluated by the customers. Consequently, they define BPM as "a systematic, structured approach to analyze, improve, control, and manage processes with the aim of improving the quality of

products and services” [6]. Harmon echoes this idea: “BPM refers to aligning processes with the organization's strategic goals, designing and implementing process architectures, establishing process measurement systems that align with organizational goals, and educating and organizing managers so that they will manage processes effectively” [7].

The above definitions all point to the core task of Business Process Management: To create alignment among the individual process components input, output, resources, process structure, and process goals. If such alignment is achieved, the overall process performance of the organization should increase both in terms of process quality (e.g. less waste, idle time, rework) and quantity (e.g. shorter cycle times, faster adjustment to environmental changes). Alignment is seldom achieved through a one-time process. Instead, an iterative approach in form of a continuous process management lifecycle helps organizations achieve, maintain, and improve the quality of their processes. This lifecycle is shown in figure 1.

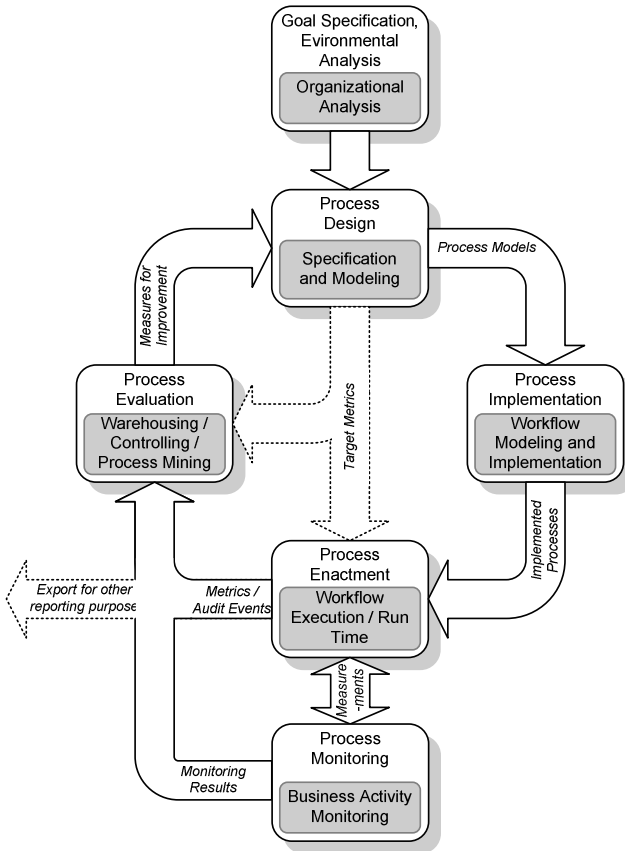


Fig. 1. Business Process Management Lifecycle

The process lifecycle starts with a definition of organizational and process goals, and an assessment of environmental factors and constraints that have an effect on the business processes of an organization. The purpose of the following process design phase is the identification of those processes an organization wishes to analyze, redesign, and/or automate. The details of these processes are specified and mapped using (semi-)formal modeling methods. Before processes are designed or redesigned, it is necessary to identify and clarify variables that will influence the process design. Internally, these variables include the purpose and deliverables of the process, known limitations of the process and the affected organization. External variables reflect the influence of outside parties such as suppliers, customers, competitors and governmental agencies. The completeness of the goal specification and the organizational analysis defines the parameters and thus the constraints for the desired process design.

During the process implementation phase the specified process models are transferred into the operational environments which can either be manual (e.g. via procedure handbooks) or automated (e.g. via BPM or workflow software). Finally, individual process instances are derived from the process specification and executed; their performance is monitored in real time. For the purpose of process control and improvement, audit trails produced during the process enactment and monitoring stages can be used in the evaluation stage. During this stage data from multiple process instances is aggregated to discover temporal trends and design flaws. Feedbacks and contingency plans for process improvement can be formulated based on the results of process measurement and evaluation.

3 Risks and Risk Management

In classical decision making theory, risk is conceived as “reflecting variation in the distribution of possible outcomes, their likelihoods, and their subjective values” [8]. By this definition, risk can be expressed mathematically as “the probability of occurrence of loss/gain multiplied by its respective magnitude.” [20] The Project Management Institute defines risk as “an uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective” [23]. Since risks are commonly associated with negative outcomes [8], the distinction between risks and problems often remains unclear. Charette claims that a risk is not a problem, but at most a “potential problem” that may result from making a particular decision. To some extent “risk is the probability of unwanted consequences of an event and decision” [10].

3.1 Risk Management

The purpose of risk management is to “reduce or neutralize potential [risks], and simultaneously to offer opportunities for positive improvement in performance.” [22] A general risk management framework is composed of 3 main action phases: identification, analysis, and control [3]. In practice, the risk identification phase is typically conducted by an expert group through brainstorming or techniques such as

fault tree or event tree analysis, cause consequence analysis, or failure mode and effect analysis.

Risks are caused by uncertainties [12], thus it is often difficult to frame risks in a precise fashion. One way to do so is to characterize risks using properties such as impact, probability, time frame, and coupling with other risks [12]. Four risk-handling strategies are suggested in the literature: mitigation [13], avoidance, transfer, and acceptance /assumption [14], table 1 summarizes these strategies in detail.

Table 1. Risk Management Strategies

RISK MGMT. STRATEGY	DEFINITION	EXAMPLES
Mitigation	To reduce the probability of a risk and/or the impact that an occurrence of the risk may bear. Risk limitation aims at the implementation of controls that dampen the effects of risk occurrences, while not completely alleviating them.	<ul style="list-style-type: none"> • Standardized process routing • Formalized exception handling • Complete kit processing • Collaboration, checks & balances
Avoidance	To eliminate the probability of a specific risk before its occurrence. This strategy is normally realized by trading the risk for other risks that are less threatening or easier to deal with.	<ul style="list-style-type: none"> • Process redesign
Transfer	To shift risk or the consequences caused by the risk from one party to another. Also called “risk sharing”. Risk transfer may involve the purchase of an insurance policy, or the outsourcing of risky project parts.	<ul style="list-style-type: none"> • Process Outsourcing • Insurance Policies
Acceptance/ Assumption	To adapt to the risk when it becomes a problem. The enactment of a risk contingency plan is required in this strategy.	<ul style="list-style-type: none"> • Adaptation to regulatory requirements

3.2 Common Taxonomies of Risk in Enterprise Projects

The notion of risk in enterprise projects has been dealt with extensively in the academic literature. The most popular taxonomy of risks in enterprises looks at the risk context. Typically, a business entity is always threatened by natural risks, human risks, and environmental risks [14]. Similarly, in the field of business process management projects, risks can be categorized into three groups: people risks, management risks, and technical risk [3]. Nevertheless, Davenport points to organizational/human resources and information technologies as two major enablers of process innovation [15]. This implies that the enablers of process innovation can produce negative impacts on businesses if they are not managed properly.

In their model of risk factors in Enterprise Systems implementations, Scott and Vessey add external business context to the risk factors identified above [16]. They suggest that risks can produce a positive impact on businesses if they are well

managed within the organization and if the organization is able to react to outside changes. In Sumner’s research, the general risk context is broken down into smaller groups: skill mix, management structure and strategy, software system design, user involvement and training, technology planning, project management, and social commitment [17]. Figure 2 provides a taxonomy of risk that was derived from the above sources and a review of risk management literature as provided in [25].

Property	Value			
Cause	External		Internal	
Likelihood	Certain	Highly Probable	Moderately Probable	Improbable
Severity	Loss of asset, capability, process	major delay of process, loss of data	Minor process disruption	Delayed detection of misconduct
Affected Area	Financial	Technical	Functional	Organizational
Cause of Error	Skill-based	Knowledge-based	Rule-based	
Detectability	Prior to effect	At time of effect	After the effect	

Fig. 2. Taxonomy of risk properties

Risks may originate either from within the organization, or they may be caused by external factors. In the case of BPM projects, examples for these cases are a lack of BPM capabilities among the members of the project team, or the choice of an external supplier that is unable to deliver the required technology. The likelihood of a risk occurrence can range from certain risks, in which case error handling procedures should be in place, to improbable risks. The effects of risk can vary in severity. Some risks may jeopardize the entire BPM effort (e.g. loss of executive support) while others are just a delayed recognition of minor risks, such as documentation or governance issues. The area affected by risk ranges from financial aspects, technical capabilities, functional capabilities, to organizational issues. Risks do not materialize by themselves, but they reflect the outcome of some (intentional or unintentional) mistake. Such mistakes can be caused because the necessary skills are absent, i.e. the project staff is lacking training in the tools and methods applied. They may happen because the knowledge to manage a new context is absent, i.e. if the project staff does not recognize the context of a problem to find an appropriate solution. And finally, the project management rules may force participants to work in a particular fashion that is not suited to mitigate a given problem, in other words, policies and procedures did not allow for proper risk mitigation.

4 Risks Specific to BPM Projects

While the lifecycle shown in figure 1 is the depiction of an ideal continuous process management strategy, its execution is subject to numerous risks that need to be managed. Some of these risks occur within the phases of the lifecycle, while others are specific to the transition between two phases.

The following table lists common risks encountered in and between these phases. The majority of the risks identified lie in a) a mismatch of methods employed in the different phases of the process lifecycle, b) a lack of clarity who is responsible for the individual phases or their results, and c) a mismatch of process design, automation, and evaluation objectives (i.e. goal mismatch). Managers of BPM projects should pay particular attention to these areas.

The lifecycle-based classification of risks in BPM projects is useful from a managerial perspective, as it allows BPM project managers to address specific risks that relate to the current phase of the BPM project, there is some overlap among risks that occur across different lifecycle phases. In order to identify these risks, a more functional classification of BPM project risks is needed. These functional categories cluster risks that have common antecedents, e.g. a lack of training, general project management skills, or technology choices. By managing the common root causes of these risks, a BPM project manager may be able to control more effectively risks that affect several lifecycle phases.

Table 2. Lifecycle-specific Risks in BPM Projects

Lifecycle Phase	BPM-specific Risk
Analysis	<ul style="list-style-type: none"> • Conduct analysis without a view on enterprise/process/task strategy • Failure to define process goals/values in a language understandable for process stakeholders • Overemphasis of technical variables • Failure to relate systematic/organizational risks to the analysis • Analysis language is not capable to represent observed process semantics
Analysis → Design	<ul style="list-style-type: none"> • Failure to properly map analysis outcomes to design models • Loss of information during the mapping processes
Design	<ul style="list-style-type: none"> • Implementation modeling languages are not capable to represent desired process semantics • Design using incompatible modeling technologies • Lack of communication between process designers and process stakeholders • Designers ignore the organizational perspective of process design • Risk handling mechanisms are missing in the design • Modelers use different levels of abstraction
Design → Implementation	<ul style="list-style-type: none"> • Wrong translation from process models to implementation plans • Mismatch of design method and implementation method/perspective

Implementation	<ul style="list-style-type: none"> • Lack of a high level implementation view (for executives) • Lack of process management knowledge at the management level • Overemphasis on technical issues • Resulting models do not fit the current infrastructure • Resulting models do not fit the current organizational structure • Failure to relocate resources (plans for transformation) • Failure to rearrange/reassign roles and responsibilities to process stakeholders (instantiate process management) • Process stakeholders assume they know the new processes and their roles without review of the redesign
Execution	<ul style="list-style-type: none"> • Lack of communication and a common language among stakeholders • Resistance from stakeholders to perform process-oriented activities • Stakeholders take too long to adapt to process-oriented work style • Stakeholders are unable to collaborate across organizational boundaries • Stakeholders feel uncomfortable under process-oriented leadership • The composition of stakeholders changes during the runtime • System is unstable in the runtime environment • Service vendors merge or go out of business • New regulatory requirements make current process practices illegal
Monitoring	<ul style="list-style-type: none"> • Lack of monitoring strategies, plans, objectives, and methods • Stakeholders/Laws prohibit process transparency (monitoring) • Flawed monitoring information produced by stakeholders • Absence of a precise information filtering policies • Monitoring without a qualitative perspective (i.e. numerical focus) • Monitored objectives differ from original design objectives
Monitoring & Execution → Controlling	<ul style="list-style-type: none"> • Information overload of monitoring recipients • Failure to translate raw audit data into useful information • Lack of management in merging multiple information channels • Unrecorded human interference in the process • Failure to report critical issues to allow timely response
Controlling	<ul style="list-style-type: none"> • Missing standards for evaluation policies/methods • Controlling objectives are different from process design objectives • Misinterpretation of audit data • Missing link from audit data to business data • Failure to relate the evaluation to strategic and external variables
Controlling → Design	<ul style="list-style-type: none"> • Lack of well-defined feedback mechanisms • Inability to recognize problems from process evaluation • Failure to derive contingency plans form the evaluation • Controlling and process improvement conducted by different stakeholders

The following table contains a classification of risk categories that we subsequently apply to the risks listed in table 2. The risk categories described in this table are based on a review of the related literature discussed earlier.

Table 3. Risk Classification

Risk Factor	Definition
Method	Lack of understanding or misuse of methods in the planning, design, implementation, enactment, evaluation phase.
Communication	Lack of communication among BPM stakeholders and participants. This Includes conversations, meeting, training, reporting, and communication in all other forms [3, 17, 18]
Information	Absence of information efficiency, effectiveness, security, flexibility for both transfers between lifecycle phases and process monitoring and controlling efforts. [17, 18]
Change Management	Inability to manage/perform changes [1, 3, 17, 18]
System / Technology	Failure of system/technology implementation due to the system/technology’s nature or through improper human interference [1, 17, 18]
Leadership / Management	Failure to display strong leadership and/or proper project management [1, 3, 17]
Resource / Skill	Lack of desired resource/skill sets or the misuse of resources/skills [1, 3, 17, 18]
Strategy	Failure to define vision, goals, functions of all BPM stakeholders, participants, and components involved [1, 3, 17, 18]

Table 4. Mapping of BPM Risks to Risk Taxonomy

Risk Factor	Life-Cycle Risks
Method	<ul style="list-style-type: none"> • Invalid process analysis/design methods [1], [2] • Invalid mapping methods (problem to solution, solution to implementation) [1, 2], [2, 3] • Invalid process modeling methods [2, 3] • Invalid process implementation methods [3] • Invalid evaluation methods [5] • Inconsistency of evaluation/measurement methods [5], [6] • Invalid feedback mechanism [5, 2]
Communication	<ul style="list-style-type: none"> • Miscommunication of goals [1,2] • Lack of communication among stakeholders [ALL] • Hidden assumptions in design and implementation [1,2,3]
Information	<ul style="list-style-type: none"> • Misusage of information [1,2], [4,6], [5] • Inadequate information [ALL] • Invalid information [1, 2], [2, 3], [5, 2] • Invalid information conversion [6, 5]
Change Management	<ul style="list-style-type: none"> • Failure to redesign jobs/functions [1, 2] • Failure to perform necessary changes [2] • Inability to recognize problems [5, 2] • Inability to react to designated changes [ALL]
System / Technology	<ul style="list-style-type: none"> • Lacking technology acceptance [ALL] • Misusage of technology [ALL] • Lack of technology flexibility [ALL] • Lack of technology compatibility [ALL] • Lack of technology scalability [ALL]

Leadership / Management	<ul style="list-style-type: none"> • Lack of leadership/management [ALL] • Inconsistency of leadership/management [ALL] • Absence of leadership/management [ALL]
Resource / Skill	<ul style="list-style-type: none"> • Absence of resource/skill [ALL] • Misusage of resource/skill [ALL] • Inability to use resource/skill [ALL]
Strategy	<ul style="list-style-type: none"> • Inaccurate strategic definition [ALL] • Unclear strategic definition [ALL] • Absence of strategic definition [ALL]

Now that we have established a classification for different types of risk, we can map the BPM-specific risks from the previous section to these categories. The numbers behind the lifecycle specific risk example denote the lifecycle phase in which the risk was identified [1=organizational analysis, 2=design, 3=implementation, 4=execution, 5=monitoring, 6=controlling].

A look at table 4 shows that while some of the categorized risks apply to specific lifecycle phases and transitions, all of the risks associated with the categories system/technology, leadership/management, resource/skill, and strategy affect all phases of the BPM lifecycle, i.e. they are orthogonal to the progress any BPM project makes through the lifecycle. Consequently, a BPM project manager should address these orthogonal risks prior to the start of the project, while risks in the other categories are specific to individual lifecycle phases, and may lend themselves to a deferred mitigation approach.

5 Other Approaches to Risk Management: ERM and COBIT

Kliem claims that risk management should consist of three actions: risk identification, risk analysis, and risk control [3]. By the same token, Peltier suggests a complete risk management lifecycle that should include the following key concepts: analysis, design, construction, test, and maintenance [14]. In either case, there is consensus that a lifecycle concept is essential and fundamental to risk management.

ERM is a framework designed by the Committee of Sponsoring Organizations of the Treadway Commission (COSO) that helps businesses assess and enhance their internal control systems. The term “internal control system” includes all policies and procedures that an organization adopts to achieve management’s objective of ensuring the orderly and efficient conduct of business [11]. COSO defines ERM as “... a process, effected by an entity’s board of directors, management and other personnel, applied in strategy setting and across the enterprise, designed to identify potential events that may affect the entity, and manage risk to be within its risk appetite, to provide reasonable assurance regard in the achievement of entity objectives” [11]. COSO claims that in order to minimize the impact of risks risk management must address four major areas: strategy, operations, reporting, and compliance. In addition to these areas, eight individual risk components have to be reviewed. These are the internal environment, objective setting, event identification, risk assessment, risk response, control activities, information and communication, and monitoring.

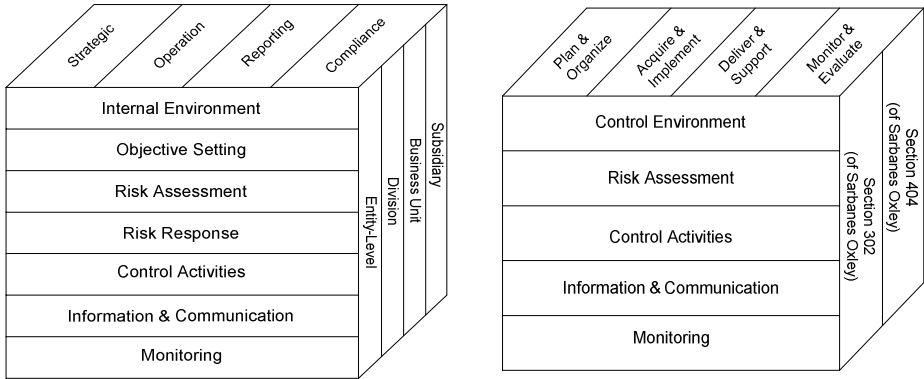


Fig. 3. COSO Enterprise Risk Management Framework (left) and CobIT (right)

COSO’s ERM has been broadly adopted by businesses since the Sarbanes-Oxley Act was ratified 2002. The act requests businesses to assure the quality of financial reports as well as the existence and adherence to internal control policies.

The Control Objectives for Information and related Technologies (CobIT) created by the IT Governance Institute (ITGI) is a set of audit-oriented guidelines that helps businesses improve their IT governance [19]. ITGI believes that effective management of information and related technology infrastructure will improve business performance. In addition to the effective and efficient delivery of information, IT governance is charged with realizing risk management by improving information security, accountability, and integrity. The CobIT Framework consists of four high level control objectives: Planning and Organization, Acquisition and Implementation, Delivery and Support, and Monitoring. With regard to the individual components within these objectives, ITGI has adopted the COSO ERM framework. However, only five of the original eight components are applied by ITGI: Control environment, risk assessment, control activities, information and communication, and monitoring. The COSO framework is based on the assumption that during the implementation of CobiT control objectives, business entities will be able to identify risks that endanger the usage of information throughout the organization and further mitigate these threats before they result in financial or organizational damages.

Both COSO ERM and CobIT are useful frameworks that can guide the actions of a BPM project manager to structure risk recognition and mitigation activities. While the ERM framework is more comprehensive in the sense that it lists individual risk areas in more detail, the CobIT framework is more closely aligned with the lifecycle concept that guides most BPM efforts.

6 Summary and Outlook

In this paper we have discussed risks that are part of the Business Process Management Lifecycle. Based on an analysis of risk taxonomies in the academic literature we have classified risk factors that can be associated with the individual phases of the BPM lifecycle. The majority of risk factors identified relate to the

composition of a BPM project: The selection of stakeholders, mismatches between design and implementation methods, and the mismatch of organizational, process, implementation, and evaluation goals and metrics. By mapping the lifecycle risks to a functional framework we have shown that some risks are specific to individual lifecycle phases, while system, leadership, resource, and strategy-related risks affect the BPM lifecycle in its entirety. Our study shows that BPM projects are faced with risks both within individual lifecycle phases, as well as with risks during the transition between lifecycle phases. While we did not elaborate on practical risk mitigation strategies for BPM projects in this paper, our future work focuses on the mapping of these risks to activities within the COSO and CobIT frameworks. This continued effort will hopefully lead to practical risk mitigation strategies for BPM projects.

References

- [1] Grover, V.: From Business Reengineering to Business Process Change Management: A Longitudinal Study of Trends and Practices. *IEEE Transactions on Engineering Management* 46 (1999) 36-46
- [2] Clemons, E. K., Thatcher, M. E., Row, M. C.: Identifying Sources of Reengineering Failures: A Study of the Behavioral Factors Contributing to Reengineering Risks. *Journal of Management Information Systems* 12 (1995) 9 - 36
- [3] Kliem, R. L.: Risk Management for Business Process Reengineering Projects. *Information Systems Management* 17 (2000) 71-73
- [4] Smith, H., Fingar, P.: *Business Process Management - The Third Wave*. Meghan Kiffer Press, Tampa, FL (2003)
- [5] Zairi, M., Sinclair, D.: Business Process re-engineering and process management. *Business Process Re-engineering & Management Journal* 1 (1995) 8 - 30
- [6] Elzinga, D. J., Horak, T., Lee, C.-Y., Bruner, C.: Business Process Management: Survey and Methodology. *IEEE Transactions on Engineering Management* 42 (1995) 119 - 128
- [7] Harmon, P.: Evaluating an Organization's Business Process Maturity. *Business Process Trends* 2 (2004)
- [8] March, J. G., Shapira, Z.: Managerial Perspectives on Risk and Risk Taking. *Management Science* 33 (1987) 1404-1418
- [9] Wieggers, K.: Knowing your enemy: software risk management. *Software Development* 6 (1998)
- [10] Charette, R.: *Applications Strategies for Risk Management*. McGraw-Hill, New York (1990)
- [11] COSO: *Enterprise Risk Management - Integrated Framework*. Executive Summary. Committee of Sponsoring Organizations of the Threadway Commission, (2004)
- [12] Gemmer, A.: Risk management: moving beyond process. *Computer* 30 (1997) 33 - 43
- [13] Adler, T. R., Leonard, J. G., Nordgren, R. K.: Improving Risk Management: Moving from Risk elimination to Risk Avoidance. *Information and Software Technology* 41 (1999) 29-34
- [14] Peltier, T. R.: Risk Analysis and Risk Management. *The EDP Audit, Control, and Security Newsletter* 32 (2004)
- [15] Davenport, T. H.: *Process Innovation*. Harvard Business School Press, Boston, Massachusetts (1993)

- [16] Scott, J. E., Vessey, I.: Managing Risks in Enterprise Systems Implementations. *Communications of the ACM* 45 (2000) 74-81
- [17] Sumner, M.: Risk Factors in Enterprise-wide/ERP projects. *Journal of Information Technology* 15 (2000) 317-327
- [18] Somers, T. M., Nelson, K. G.: A Taxonomy of Players and Activities across the ERP Project Life Cycle. *Information and Management* 41 (2002) 257 - 278
- [19] IT Governance Institute (ITGI): IT Control objectives for Sarbanes-Oxley. http://www.itgi.org/template_ITGI.cfm?template=/ContentManagement/ContentDisplay.cfm&ContentID=14133
- [20] Jaafari, A.: Management of Risks, Uncertainties and Opportunities on Projects: Time for a Fundamental Shift. *International Journal of Project Management* 19-2 (February 2001) 89-101.
- [21] Miller, R., and Lessard, D.: Understanding and Managing Risks in Large Engineering Projects. *International Journal of Project Management* 19 (2001) 437-443
- [22] Ward, S., and Chapman, C.: Transforming Project Risk Management into Project Uncertainty Management. *International Journal of Project Management* 21-2 (1994) 97-105
- [23] Project Management Institute: A Guide to the Project Management Body of Knowledge (PMBOK Guide), 2000 edition. Project Management Institute
- [24] Levin, H.S.: *Office Work and Automation*. John Wiley & Sons, New York 1956.
- [25] zur Muehlen, M.; Rosemann, M.: Integrating Risks in Business Process Models. In: *Proceedings of the 2005 Australasian Conference on Information Systems (ACIS 2005)*, Manly, Sydney, Australia, November 30-December 2, 2005.

Preface

(BPRM 2005)

Reference models for business processes have been a successful means for designing, redesigning, tailoring, and implementing business processes. Still, there is no common understanding of reference models for business processes:

- What is a reference model?
- What makes them different from a business process model?
- What should be covered by a reference model?
- What is their purpose and how should they be used?
- How should they be designed and presented?

The Workshop on Business Process Models (BPRM 2005), which was held as a satellite event of the Third International Conference on Business Process Management, Nancy, France, September 5, 2005, aimed at discussing these questions and coming to a common understanding of the terms and the interesting research issues in this field. Up to now, business process reference modelling has been mainly a German issue. One of the goals of the workshop was to discuss the issue internationally and to create an international ‘Reference Modelling Community.’

The workshop brought together people from different application areas, using different notations and formalisms, in order to present and discuss their point of view. There were two contributions discussing the term and the goals of reference modelling. And there were two contributions that dealt with formalisms for defining and customizing configurable process models. Since these papers were directly focused on business process reference models, they have been selected for publication in these post-proceedings (see Table of Contents). In addition, there were two papers that took a look at reference models from a broader perspective; due to their different focus, they have not been included in this volume.

We are happy that the workshop on Business Process Reference Models was accepted as a satellite event of BPM 2005, and we would like to thank the invited speaker Jörg Becker for his overview on business reference models and on research directions and open issues. Moreover, we would like to thank the Program Committee and the reviewers for selecting the papers, and we appreciate the support of the Local Chairs, Claude Godart and Olivier Perrin, in organizing this event. Finally, we would like to thank the Publication Chair, Armin Haller, for compiling these post-proceedings.

Organization

Workshop Organization Committee

E. Kindler, Paderborn, Germany
M. Nüttgens, Hamburg, Germany

Program Committee

W. v. d. Aalst, Eindhoven, The Netherlands
J. Becker, Münster, Germany
J. v. Brocke, Münster, Germany
C. Bussler, Galway, Ireland
P. Dadam, Ulm, Germany
S. Dustdar, Vienna, Austria
E. Kindler (Co-chair), Paderborn, Germany
P. Loos, Mainz, Germany
M. Nüttgens (Co-chair), Hamburg, Germany
A. Oberweis, Karlsruhe, Germany
F. Rump, Emden, Germany
O. Thomas, Saarbrücken, Germany

Sub-reviewers

O. Adam
P. Fettke
B. Kaffai
C. Seel, Dipl.-Kfm.
C. Seel, MScIS
J. Zwicker

Business Process Reference Models: Survey and Classification

Peter Fettke¹, Peter Loos², and Jörg Zwicker²

¹ Johannes Gutenberg University Mainz,
Information Systems and Management, 55099 Mainz, Germany
fettke@isym.bwl.uni-mainz.de
<http://isym.bwl.uni-mainz.de>

² Institute for Information Systems (IWi) at the German
Research Center for Artificial Intelligence (DFKI), 66123 Saarbrücken, Germany
{loos, zwicker}@iwi.uni-sb.de
<http://iwi.uni-sb.de/>

Abstract. Within the Information Systems field, reference models are well-known for many years. The aim of this paper is to survey and to describe reference models for business processes. Our analysis of 30 process reference models is based on a framework consisting of criteria such as application domain, used process modeling languages, model's size, known evaluations and applications of process reference models. Furthermore, we identify model domains, which have been dealt with, describe similarities and differences between the available process reference models, and point to open research questions.

1 Introduction

Information modeling is a core vehicle to analyze, design, implement, and deploy information systems [1]. However, the modeling process is often resource consuming and faulty. As a way to cope with these failures and to improve the development of enterprise-specific models, the idea of reference modeling was born [2-4].

While an application model represents a particular enterprise system, a conceptual model represents a class of similar enterprise systems. It is a conceptual framework that can be used as a blueprint for information system construction [5]. To use a particular reference model, it must be adapted to the requirements of a particular enterprise. Reference models are also called universal models, generic models, or model patterns. The term reference model for business processes refers to a specific type of reference model. A process reference model represents dynamic aspects of an enterprise, e.g. activity sequences, organizational activities required to satisfy customer needs, control-flows between activities, particular dependency constraints [6].

In publicly available sources, numerous more or less elaborated process reference models are proposed. The main objective of this paper is to identify, to survey and to describe the well-known process reference models. Compared to existing reference model surveys [5, 7], this study is more comprehensive and focuses on reference models for business processes.

Our study is of both practical and theoretical importance. From a practical viewpoint, the selection of an appropriate process reference model is complicated. One

presumption of reusing a reference model is to know its availability, its application domain, its potentials and limitations etc. A model survey can offer such information. Thus, this instrument fosters a rational and systematic model selection process.

Beside the practical relevance, surveys of reference models are of importance for the theory of enterprise modeling in general and for the theory of reference modeling in particular. Surveys of reference models can show varieties, gaps and areas of improvements. The results of a survey represent a meaningful basis for new and advanced reference models. Even if such an investigation does not take place in conjunction with the development of a new reference model, at least the scope of already developed reference models should be made clear by such a survey afterwards. Therefore, a survey of process reference model stimulates the scientific progress of reference modeling.

The paper unfolds as follows: Section 1 motivates this piece of research. We introduce the theoretical background of this study in Section 2. In Section 3, we use the proposed framework to describe 30 well-known process reference models. The obtained results are discussed in Section 4. The paper ends with some concluding remarks.

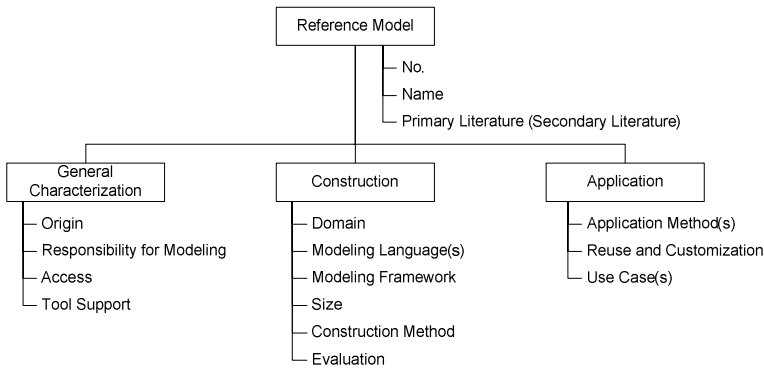


Fig. 1. Criteria for describing process reference models

2 Theoretical Background

Framework. Existing reference models can be structured regarding numerous points of view. Figure 1 illustrates and structures the here considered criteria for characterizing process reference models. Beside universal characteristics, suitable for the complete spread of reference models, the description and classification of process reference models requires particular consideration of process-related criteria. The universal and process-specific criteria of the framework are described in the following:

- *Identification:* The identification of reference models is made by running numbers and reference model names. References, wherein the reference models are described, are also specified (primary literature). This information is completed with additional references (secondary literature) wherein certain reference model

properties are explained. The specification of secondary literature particularly supports providing information about limited accessible reference models.

- *General Characterization*: The following four criteria generally characterize a reference model.
 - *Origin*: The origin informs about the classification of the person(s) who have developed the reference model. In this regard, both science and practice can be distinguished.
 - *Responsibility for Modeling*: This criterion describes the persons or organizations that developed the reference model.
 - *Access*: The access specifies the accessibility to the reference model by third parties. If the reference model is completely obtainable over usual ways of librarianship the access is classified as “open”. The access is “closed”, if the responsible person(s) or institution provides no possibility for using and recognizing the reference model by third parties. If the access is neither open nor closed the access is classified as “limited”. This is the case, e.g., if the reference model can be purchased as standalone product. If the access to the reference model is closed the information of all aforementioned and following criteria is based on statements from the specified primary and secondary literature.
 - *Tool Support*: This criterion describes whether the reference model can be automatically used by a software tool or whether the reference model is only available in paper or digital copy.
- *Construction*: The following six criteria address the construction of process reference models:
 - *Domain*: The domain describes the field of application from perspective of the person(s) or institution responsible for developing the reference model. The criterion is distinguished into domain differentiation and domain description. Specifying the domain differentiation serves to distinguish varying principles of domain classification. So far, several differentiation approaches have been proposed. Using [8] in this framework, a widely elaborated approach is considered. With this, different principles of differentiation can be identified: Institutional differentiation is based on institutional characteristics of the intended business system (e.g. “Industrial Enterprise” or “Bank”); functional differentiation is realized through business functions as differentiation characteristic (e.g. classical business functions: “Distribution Logistic”, “Production Planning and Control”; newer functions: “Facility Management”, “Knowledge Management”); object-driven differentiation where business objects serves as differentiation characteristic (e.g. “Life Insurance” or “Branch Business”); enterprise type-driven differentiation is based on special enterprise characteristics (e.g. a book publisher can be considered as a special type of a publisher). Also universal reference models exist which cannot be classified based on the aforementioned principles. Beside the domain differentiation, the domain description specifies the intended field of the reference model’s application using short descriptions.
 - *Modeling Language(s)*: The language criterion states the modeling language(s) used to represent the reference model. To address the particular consideration and description of process reference models, modeling languages or diagram

types used to represent process models of the reference model are particularly specified. Further modeling languages are additionally described.

- *Modeling Framework*: This criterion describes whether a modeling framework is part of the reference model. A framework can structure relevant elements esp. diagrams of a reference model and their relationships at a higher level of abstraction. This reduces complexity and provides an overview of elements and relationships within the reference model.
 - *Size*: So far, appropriate size metrics for models of different modeling languages do not exist [7]. To give a vague impression about the size of the described reference models, several metrics can be used. The number of represented diagrams and views pose as general attributes. As a process-related metric, the number of process steps within represented process diagrams is stated. The mentioned sizes of smaller models (<30) shall be counted, the sizes of bigger models can be estimated and rounded off to full decade. If the access to the model is closed the information is based upon statements of given references.
 - *Construction Method*: This criterion states the modeling concept used by the responsible person(s) or institution for developing the reference model.
 - *Evaluation*: This criterion describes the used methods for evaluating the reference model by the person(s) or institution responsible for developing the reference model or by third parties. Evaluation methods are only considered, if they are explicitly intended for model evaluation by the evaluator. Besides the method, it is stated whether the result of performed evaluation is inter-subjective verifiable.
- *Application*: The following three criteria address the application of process reference models:
- *Application Method(s)*: This criterion describes the known method resp. concept for applying the reference model.
 - *Reuse and Customization*: This criterion lists concepts for reusing and customizing of model elements in the scope of the model's application.
 - *Use Case(s)*: The use case(s) describes how often the reference model was applied to construct an application model. As with the evaluation method, this criterion is completed by the information whether the number and extent of use cases are inter-subjective verifiable.

Method. The method of the performed survey does not possess of a rigorously defined procedure. The identification of the models was realized in a more explorative way. We identified and describe 30 business process reference models which are well-known from our view. The 30 models were selected out of the plenty of well-known reference models using the modeling language representing the reference models. If one of the reference models is represented in one or more established process modeling languages it is recorded as process reference model. The necessity of a more strictly methodical procedure is not essentially given in the scope of this paper. Rather we want to propose a framework to describe business process reference models, demonstrate those applicability and analyze process reference models based on an appropriate plenty of explorative surveyed models.

3 Results

3.1 General Characterization

The identified reference models are depicted in the table within the appendix of this paper. In the following, the models are characterized regarding the criteria origin, responsibility for modeling, access and tool support:

- *Origin*: Practitioners developed eight reference models. The design of the other 22 models was partly or completely done by scientists.
- *Responsibility for Modeling*: Fifteen reference models were developed by one person, the author of the primary reference, and eight models by several persons. Furthermore, enterprises are responsible for the development of four models, two models were developed by associations and one model was created by an independent office of the British government.
- *Access*: The access is open to 16 reference models of the survey, closed to eight models and limited to six models.
- *Tool Support*: Fifteen reference models are only published as paper copy, and eight models can directly be handled in a tool. Finally, no statement is given by the responsible person(s) to the remaining seven models.

3.2 Construction

Domain. Due to missing standards, reference models cannot be described based on a consistent domain framework. A classifying description of the results and an associated quantitative analysis is only possible through the subjective-driven specification of the domain differentiation. So, 11 reference models are provided for institutional context. Further 12 models are classified into the functional context. Finally, seven models cannot be exactly classified based on the differentiation principle as described in chapter 2.

Using the domain description from the responsible designer(s), reference models for the information systems' development in industrial enterprises exist (e.g. "Aachener PPS"-model or SCOR). Further subjects of reference modeling are financial service providers (e.g. insurer or banks), book publishers or special business functions like knowledge management, logistic and environmental data management.

Modeling Language(s). To represent reference models, several modeling languages are used. Widely-accepted modeling languages, such as the Entity-relationship Model (ERM) and the Unified Modeling Language (UML), are applied. Furthermore, modeling languages like the Semantic Object Model (SOM), function trees or special object-oriented languages, are used. Some designers use languages which are exclusively developed to construct the correspondent reference model.

To model the business process view of the reference models, Event-driven Process Chains (EPC) are used in many cases. Also parts of further languages and language frameworks are utilized. For example, the activity and use case diagrams as parts of the UML are particularly often used. SOM and the Multi-Perspective Enterprise Modeling (MEMO) similarly possess views for business process modeling. Further

languages and diagram types are e.g. the process chain diagram (VKD), value chain diagram, task chain diagram or proprietary languages.

Modeling Framework. In 18 cases, the designer(s) of the models and/or the authors of the literature references make statements about a framework to structure the elements and relationships of the reference model. In 10 of the 18 cases, the author explicitly negates the existence of a framework. In the remaining eight cases, the existence is stated or the framework is represented and described.

Size. With the determination of the model's size, values have only been acquired, if the process reference models are represented within the given references or the author(s) specifies the number. Where the metrics could be determined: The number of used diagrams ranges from one up to estimated 450. The number of views ranges from one to four. Finally, the number of process steps as process-related size ranges from estimated 50 to 300 and in one case to 1500 steps.

Construction Method. Statements on the development process are identified for 14 of the analyzed reference models. Four of these cases explicitly refer to a used procedure model, build up on such a model or introduce an own model. The designers of the remaining 10 reference models shortly describe the applied procedure without comprehensively explicating the chosen procedure. For example, the designers describe their reference models as “deductively derived” or “constructed on case examples”.

Evaluation. Methods for evaluating the quality are only determined for 15 reference models. These cases can be distinguished into two groups:

1. *Evaluation approach:* In two cases, information on how an evaluation of the reference models can be done is stated without documenting concrete results. The proposals cover a comparison of a reference model with an enterprise or the annotation of a necessary empirical evaluation.
2. *Results:* In regard to 13 reference models, results of evaluations are described by authors of according literature references. The results base on different procedures:
 - In three cases, the reference model was evaluated through the prototypical implementation within a software product.
 - In three cases, the reference model was used for case studies: The spread ranges from simple, fictitious conditions to reality-similar utilizations.
 - In one case, a questioning of model users was organized to determine the possibilities of utilizations.
 - In two cases, an ad hoc evaluation was carried out compiling arguments to show preferences and limitations of the reference model from the evaluator's view.
 - In two cases, a thought experiment was performed by the author. This is a way to evaluate the reference model through demonstrating exemplary application within a hypothetical context.
 - In further two cases, a prototypical or exemplary application of the reference model within a fictitious context is described without a real application.

3.3 Application

Application Method. Possible potentials of process reference modeling only unfold with applying the reference models. Thirteen of the identified process reference

models cover proposals including configurational options for the model's application process. Most of them (twelve authors) develop a model-based procedure model for specific application purposes. Typical examples are reference model-based procedure model: for knowledge management [9], for developing information and communication architecture [10]. For the SAP R/3 reference model, a model-based procedure model is not proposed. Instead, contributions exist wherein the application of the model is exemplary described [11-13].

Reuse and Customization. Statements on concepts for reusing and customizing of elements within the reference models are only provided for nine of the entire 30 models. Similar to the modeling language, one reference model can comprise more than one concept. The specialization of the developed models and the usage of build-time operators are often proposed concepts. Beside others, a particular case is the usage of model variants for different application contexts in one model.

Use Case(s). Use cases are also a way of evaluating similar to case studies. Though, the real application projects are not construed as ex ante evaluating studies rather than the project results are used as ex post evaluation. In nine of the entire 30 cases, the reference models were used within real projects. In the remaining 21 cases, statements on real applications are not available, although, in one case, the author explicitly states that no real application has taken place.

4 Discussion

4.1 Identified Reference Models

Within this survey, 30 process reference models are described and classified. The quantification does not raise the claim of a comprehensive survey. In fact, because of the lack of space, this paper shall document 30 well-know process reference models. From a practical point of view, the determination whether a reference model is a process reference model, is difficult and bases on subjective distortions:

- It is complicated to answer whether a reference model is a collection of many individual models or an overall model. For example, it can be argued that the SKO-reference model consists of two reference models, the SKO-reference *data* model and the SKO-reference *process* model. Moreover, it is unclear how to describe model variants; either as part of a comprehensive reference model or as several individual reference models.
- Further difficulties arise with presenting one reference model in several publications. A decision is necessary whether these models are similar representations of different reference models or different representations resp. versions of the same model. For example, this interpretational problem arise with the “Handels-H”-model which is presented in two book editions, one from 1996 and the other from 2004. Moreover, interpretational problems arise with incompletely published models.
- Finally, the decision whether a “reference model” is a reference model in the here implied intuitive conception, is often complicated.

4.2 General Characterization

In publicly available sources, numerous more or less elaborated process reference models are proposed. Most of them were developed within science. In spite of this, it can be suspected that process reference models can be found in the reality of enterprise modeling. Nevertheless, the survey and classification illustrates a lack of implementation of the analyzed reference models in real environments. This can depend on several reasons:

- Many reference models are partly not accessible or only limited accessible. This fact is plausible in regard to reference models developed within practice, but the limitation of reference models from science is inappropriate. Moreover, 20 of the analyzed reference models with origin in science do not possess tool support. From the view of reference modeling objectives, the propagation and application of the models is prohibited by the lack of access and missing tool support.
- From a practical point of view, the selection of an appropriate process reference model is difficult and complicated. One presumption of reusing a reference model is to know its availability and application-relevant information. It can be suspected that many available reference models, in particular the models developed in science, are not publicly known. Application methods resp. procedure models to apply reference modeling in practical context do not regard the multiplicity of existing models; at least we do not know such methods resp. procedures. Moreover, the field of reference modeling is not comprehensively established within practical enterprise modeling.

Further reasons for the low application of the reference models are associated with characteristics like used representation language, possibility of inter-subjective evaluation, existence of appropriate application method etc. Primary gaps are identified within the correspondent, following sections.

4.3 Construction

Domain. The differentiation of application domains is done in several ways. Using the principles of differentiation introduced in chapter 2, the analyzed reference models can only be classified as institutional and functional domains. Furthermore, some reference models cannot be exactly classified. These two issues do not show a lack of reference models for certain domains. In fact, it points out difficulties regarding the differentiation. So, the introduced differentiation criteria are not exclusive but partly overlap. For example, a book publisher can be both a special enterprise type and an institution. Also enterprises which perform production planning and control functions are regularly classified as industry. The difficulty cannot be deepened any further. Instead, it shall be stressed that the differentiation of a reference model is not trivial and it has to be done with utmost diligence: Finally, the differentiation of the domain determines the intended field of application and as consequence the reference model's potential.

Modeling Language(s). Reasons like more or less objective properties, personal preferences, available tools etc. limit the development of standardized modeling languages.

Nevertheless, it is desirable to explicate the special requirements to a modeling language before using to design a reference model. Only this guarantees inter-subjective proving and makes the language a subject of criticism. It should also be pointed out which constructs a language has to provide for serving as efficient reference modeling language. Furthermore, the question exists whether configuration mechanism for modeling languages can be usefully constituted in reference modeling. Currently, only few authors evaluate languages before designing a reference model.

Construction Method. So far, only few authors explicit the procedure of their model's construction. Two types of procedure models can be generally distinguished:

- Empiric-oriented design methods develop reference models based on a class of real enterprises.
- Deductive-oriented design methods derive a reference model from formal-logical and mathematical inferences.

A rating of both procedures is ambivalent: Empiric-oriented methods neglect possible, but, up to now, unrealized design concepts of business systems. On the other hand, deductive methods suggest a compelling nature which is not present within the reality of model design. Although, this procedure does not perform invulnerable inferences, it based on simple plausibility deliberations.

Advantages and disadvantages of both procedures shall not be discussed in more detail. Though, we do not know any work investigating this problem from an empirical point of view. An intensive analysis of effects of several design methods for reference models is desirable.

Evaluation. The evaluation of reference models is of high importance and an extraordinary challenge. Both acceptable evaluation criteria and methods are not established [14]. On the one hand, the scientific perspective demands precise, consistent and complete reference models. On the other hand, from perspective of application, simplicity and understandability are of relevance. Hence, conflicts of objectives can arise.

Although, several results of reference model evaluation exist, considerable more need for research is noticed. Existing evaluation results cannot often be evaluated by third parties. For example, some evaluators only argue that the reference model stands the practical application. Furthermore, standardized methods and criteria for evaluation do not exist.

Meanwhile upcoming contributions with reference model evaluation by third parties and by the author are a favorable development (e.g. [15]). In some cases, these contributions can be critically assessed from a methodical point of view (e.g. low validity), but we recommend such evaluations: These contributions are essential to evaluate the potentials of reference models. Only evaluations by third parties ensure the reference models' independency and usability.

Evaluation by third parties has often failed because of practical limitations: As already mentioned and criticized in section 4.2, the reference models are partly not accessible or only limited accessible. For the evaluation of the reference models by third parties, it is necessary to fully publish the models.

4.4 Application

As several procedures are known within the field of reference model design, no standardized application methods have been presented. Although, it is obvious that only one design method can be used for the development of one reference model, it is possible that several application methods can be used to apply a reference model. Nevertheless, the realization of similar task at the reference models' application can be assumed:

- *Selection of one reference model*: Identified application methods abstract from this question and only take one given reference models into account. Although, Schwegmann originally proposes to decide according to instinct whether a new reference model shall be developed or an existing reference model shall be selected and reused [16].
- *Configuration and adaptation of one reference model*: For this purpose, approaches like a configurational reference modeling are described [17], although they are not widely implemented by existing reference models.

5 Conclusion and Further Research

This paper analyzes the body of process reference models available in public sources. The main contribution of our work is three-folded: First, we propose a new framework to describe business process reference models. In this study, we use this framework to survey well-known business process reference models. However, the proposed framework can be used to guide the developing process of *new* models, too. Second, we demonstrate the applicability of the framework by describing 30 business process reference models. This survey fosters the model selection process during application model development. Third, our analysis of the obtained results points to open research questions. For instance, the development of language constructs for reusing and customizing of model elements in the scope of the model's application or the evaluation of languages before designing a reference model.

Our work has some limitations: First, we do not introduce the term reference model formally. Instead, our analysis is based on a rather intuitive conception, which may lead to misunderstandings. However, we believe it is not easy to give an acceptable explication of the term reference model, because, e.g., the term is both used as a one- and two-place predicate [18]. Second, our survey is mainly based on a literature review. We suspect that business process reference models can be found in the reality of enterprise modeling, too. However, we only survey models that are already described in literature. So, our study is just based on secondary information. Third, the framework used to describe reference models is limited. May be, it will be necessary to extend the framework and to define the used criteria in a more rigorously way. Also the justification of the used criteria has to be more stringent in future.

In the future, we try to overcome the mentioned limitations. Our long-term research objective is to develop the conceptual foundations for reference model catalogs. Reference model catalogs are inspired by construction catalogs used in engineering disciplines and provide systematic and comprehensive information about

all known reference models. To achieve this objective, we will develop a formalized notion of the term reference model first. Second, we are preparing empirical studies to describe and to explain reference modeling processes found in reality. Third, we will use ontology technology to capture our framework and to describe the known body of reference models.

Acknowledgement. This paper presents results from the research project “Reference modeling with reference model catalogs” funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation).

References

1. Wand, Y., Weber, R.: Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda. *Information Systems Research* 13 (2002) 4, 363-377
2. Mertins, K., Bernus, P.: Reference Models. In: Schmidt, G. (ed.) *Handbook on Architectures of Information Systems*. Springer, Berlin et al. (1998) 615-617
3. Mišić, V. B., Zhao, J. L.: Evaluating the Quality of Reference Models. In: Storey, V. C. (ed.) *Conceptual Modeling - ER 2000 - 19th International Conference on Conceptual Modeling*, Salt Lake City, Utah, USA, October 9-12, 2000 Proceedings. Springer, Berlin et al. (2000) 484-498
4. Scheer, A.-W., Nüttgens, M.: ARIS Architecture and Reference Models for Business Process Management. In: Oberweis, A. (ed.) *Business Process Management - Models, Techniques, and Empirical Studies*. Springer, Berlin et al. (2000) 376-389
5. Fettke, P., Loos, P.: Classification of reference models - a methodology and its application. *Information Systems and e-Business Management* 1 (2003) 1, 35-53
6. Becker, J., Kugeler, M., Rosemann, M.: *Process Management*. Springer (2003)
7. Van Belle, J.-P. W. G. D.: *A Framework for the Analysis and Evaluation of Enterprise Models*. University of Cape Town (2003)
8. Mertens, P., Lohmann, M.: Branche oder Betriebstyp als Klassifikationskriterien für die Standardsoftware der Zukunft? Erste Überlegungen, wie künftig betriebswirtschaftliche Standardsoftware entstehen könnte. In: *Verbundtagung Wirtschaftsinformatik 2000*, 110-135
9. Warnecke, G., Gissler, A., Stammwitz, G.: Referenzmodell Wissensmanagement - Ein Ansatz zur modellbasierten Gestaltung wissensorientierter Prozesse. *IM Information Management & Consulting* 13 (1998) 1, 24-29
10. Rohloff, M.: Das Prozessrahmenwerk der Siemens AG: Ein Referenzmodell für betriebliche Geschäftsprozesse als Grundlage einer systematischen Bebauung der IuK-Landschaft. In: Becker, J., Knackstedt, R. (eds.): *Wissensmanagement mit Referenzmodellen: Konzepte für die Anwendungssystem- und Organisationsgestaltung*. Physica-Verlag, Heidelberg (2002) 227-235
11. Keller, G., Lietschulte, A., Curran, T. A.: Business Engineering mit den R/3-Referenzmodellen. In: Nüttgens, M. (ed.) *Electronic Business Engineering - 4. Internationale Tagung Wirtschaftsinformatik 1999*. Physica-Verlag, Heidelberg (1999) 397-423
12. Lietschulte, A., Keller, G.: Modellgestützte R/3 Einführung. In: Mertens, P. (ed.) *Referenzmodellierung '98: Anwendungsfelder in Theorie und Praxis*, 14. Juli 1998. Forschungsinstitut für Rationalisierung an der RWTH Aachen, Aachen (1998) 5-1 - 5-8

13. Curran, T. A., Keller, G.: SAP R/3 Business Blueprint - Business Engineering mit den R/3-Referenzprozessen. Addison-Wesley, Bonn et al. (1999)
14. Fettke, P., Loos, P.: Multiperspective Evaluation of Reference Models - Towards a Framework. In: Jeusfeld, M. A., Pastor, Ó. (eds.): Conceptual Modeling for Novel Application Domains - ER 2003 Workshops ECOMO, IWCMQ, AOIS, and XSDM, Chicago, IL, USA, October 13, 2003. Springer, Berlin et al. (2003) 80-91
15. Taylor, C., Probst, C.: Business Process Reference Model Languages: Experiences from BPI Projects. In: Proc. INFORMATIK 2003 - Innovative Informatikanwendungen, Band 1, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 29. September - 2. Oktober 2003 in Frankfurt am Main (2003) 259-263
16. Schwegmann, A.: Objektorientierte Referenzmodellierung - Theoretische Grundlagen und praktische Anwendung. DUV, Wiesbaden (1999)
17. Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuroпка, D.: Configurative Process Modeling - Outlining an Approach to Increased Business Process Model Usability. In: Proc. Information Resources Management Association Conference (2003) 615-619
18. Fettke, P., Loos, P.: Referenzmodellierungsforschung. Wirtschaftsinformatik 46 (2004) 5, 331-340
19. Luczak, H., Kees, A.: Das Aachener PPS-Modell. In: Proc. Referenzmodellierung '98: Anwendungsfelder in Theorie und Praxis, 14. Juli 1998, RWTH Aachen (1998) 2-1 - 2-9
20. ten Voorde, H.: Dynamic Enterprise Modeling. In: Proc. Referenzmodellierung '98 - Anwendungsfelder in Theorie und Praxis, 14. Juli 1998, RWTH Aachen (1998) 7-1 - 7-22
21. Kremer, H., Dold, G., Fischer, H., Strobel, M., Seifert, E. K.: Informationssysteme für das Umweltmanagement - Das Referenzmodell ECO-Integral. Oldenbourg (2000)
22. Frank, U.: Modeling Products for Versatile E-commerce Platforms - Essential Requirements and Generic Design Alternatives. In: Arisawa, H., Kambayashi, Y., Kumar, V., Mayr, H. C., Hunt, I. (eds.): ER 2001 Workshops, HUMACS, DASWIS, ECOMO, and DAMA. Springer, Berlin et al. (2002) 444-456
23. Becker, J., Schütte, R.: Handelsinformationssysteme. Landsberg/Lech (1996)
24. Hochstein, A., Hunziker, A.: Serviceorientierte Referenzmodelle des IT-Managements. HMD - Praxis der Wirtschaftsinformatik (2003) 232, 45-56
25. Kaiser, T. M., Bach, V., Vogler, P., Österle, H.: Eine Methode für die Konzeption von Intranets. HMD - Praxis der Wirtschaftsinformatik (1999) 209, 94-104
26. Buchwalter, J.: Elektronische Ausschreibungen in der Beschaffung - Referenzprozeßmodell und prototypische Realisierung. Eul, Lohmar, Köln (2002)
27. Gerber, S., Mai, A.: Ein Referenzmodell für das Filialgeschäft von Banken als betriebliche Wissensplattform. In: Becker, J., Knackstedt, R. (eds.): Wissensmanagement mit Referenzmodellen: Konzepte für die Anwendungssystem- und Organisationsgestaltung. Physica-Verlag, Heidelberg (2002) 195-206
28. Haas, C., Ahlemann, F., Hoppe, U.: Organisationale Integration von E-Learning in Unternehmen - ein Referenz-Informationsmodell. In: Schoop, E. (ed.) Wirtschaftsinformatik 2003/Band I - Medien - Märkte - Mobilität. Physica, Heidelberg (2003) 707-726
29. Herrmann, G.: Verlässlichkeit von Geschäftsprozessen - Konzeptionelle Modellbildung und Realisierungsrahmen. Logos, Berlin (2002)
30. Kluger, M. A.: Beitrag zur effizienten Anwendung der dynamischen Unternehmensmodellierung. Verlag Praxiswissen, Dortmund (1999)
31. Krömker, M.: Werkzeug zur durchgängigen Systemunterstützung der Angebotserstellung in der Unikat- und Kleinserienfertigung. Verlag Mainz, Aachen (2000)
32. Kruse, C.: Referenzmodellgestütztes Geschäftsprozessmanagement: ein Ansatz zur prozessorientierten Gestaltung vertriebslogistischer Systeme. Gabler, Wiesbaden (1996)
33. Mertens, P.: Integrierte Informationsverarbeitung 1 - Administrations- und Dispositionssysteme in der Industrie. 12 edn. Gabler, Wiesbaden (2000)

34. Mertens, P., Griese, J.: Integrierte Informationsverarbeitung 2 - Planungs- und Kontrollsysteme in der Industrie. 9 edn. Gabler, Wiesbaden (2002)
35. Neumann, S.: Workflow-Anwendungen in technischen Dienstleistungen - Eine Referenz-Architektur für die Koordination von Prozessen im Gebäude- und Anlagenmanagement. Logos, Berlin (2003)
36. Pumpe, D.: Ein Referenzmodell zur Planung und Steuerung der Abläufe in Seehafen-Containerterminals. Mensch-und-Buch, Berlin (2000)
37. Remme, M.: Konstruktion von Geschäftsprozessen: ein modellgestützter Ansatz durch Montage generischer Prozesspartikel. Gabler, Wiesbaden (1997)
38. Rüffer, T.: Referenzgeschäftsprozeßmodellierung eines Lebensversicherungsunternehmens. In: Proc. Modellierung betrieblicher Informationssysteme - Proceedings der MobiS-Fachtagung 1999, 14. und 15. Oktober 1999, Universität Bamberg (1999) 86-107
39. Schaich, C.: Informationsmodell zur fachübergreifenden Beschreibung intelligenter Produktionsmaschinen. Utz, München (2000)
40. Schlagheck, B.: Objektorientierte Referenzmodelle für das Prozess- und Projektcontrolling - Grundlagen - Konstruktion - Anwendungsmöglichkeiten. DUV, Wiesbaden (2000)
41. Tzouvaras, A.: Referenzmodellierung für Buchverlage: Prozess- und Klassenmodelle für den Leistungsprozess. Cuvillier, Göttingen (2003)
42. Keller, G., Teufel, T.: SAP R/3 Process Oriented Implementation - Iterative Process Prototyping. Addison-Wesley, Harlow et al. (1998)
43. Gerber, S., Hiestermann, A., Kittlaus, H.-B.: Management von Prozeßmodellen dezentraler BPR-Projekte mit Hilfe eines zentralen Referenzprozeßmodells. In: Nüttgens, M. (ed.) Electronic Business Engineering - 4. Internationale Tagung Wirtschaftsinformatik 1999. Physica, Heidelberg (1999) 375-395
44. Eisenreich, A.: Das SKO-Datenmodell - ein Referenzmodell für die Sparkassenorganisation. In: Becker, J., Knackstedt, R. (eds.): Referenzmodellierung 2002: Methoden - Modelle - Erfahrungen. Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster, Münster (2002) 121-132
45. Gerber, S., Müller-Luschnat, G.: Sind Referenzprozeßmodelle in der betrieblichen Praxis sinnvoll? - Ein Beispiel aus der Dienstleistungsbranche. In: Schürr, A. (ed.) Modellierung '99 - Workshop der Gesellschaft für Informatik e. V. (GI), März 1999 in Karlsruhe. Teubner, Stuttgart, Leipzig (1999) 27-42
46. Supply-Chain Council Inc. SCOR Overview. Overview of the SCOR Model v3.0.[Online]. Available: www.supply-chain.org
47. Stephens, S.: The Supply Chain Council and the Supply Chain Operations Reference Model. Supply Chain Management 1 (2001) 1, 9-13
48. Holten, R., Melchert, F.: Das Supply Chain Operations Reference (SCOR)-Modell. In: Becker, J., Knackstedt, R. (eds.): Wissensmanagement mit Referenzmodellen: Konzepte für die Anwendungssystem- und Organisationsgestaltung. Physica-Verlag, Heidelberg (2002) 207-226
49. Huan, S. H., Sheoran, S. K., Wang, G.: A review and analysis of supply chain operations reference (SCOR) model. Supply Chain Management - An International Journal 9 (2004) 1, 23-29
50. GDV (Ed.). Anwendungsarchitektur der deutschen Versicherungswirtschaft. Gesamtverband der deutschen Versicherungswirtschaft e. V. [Online]. Available: <http://www.gdv-online.de/vaa/>
51. Scheer, A.-W.: Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse. 7 edn. Springer, Berlin et al. (1997)
52. Scheer, A.-W.: 20 Jahre Gestaltung industrieller Geschäftsprozesse. Industrie Management 20 (2004) 1, 11-18

Appendix

Identification			General Characterization				Construction		
No.	Name	Primary Literature (Secondary Literature)	Origin	Responsibility for Modeling	Access	Tool Support	Domain Differentiation	Domain Description	Modeling Language(s)
									Process Modeling Language(s)
1	*Aachener PPS*-Model	[19]	Science	Authors	Closed	Yes	Function	Production, Planning and Control Systems	Proprietary Process Model
2	Baan Reference Model	[20]	Practice	Baan	Closed	Yes	Others	n.S.	Proprietary Process Model
3	ECO-Integral	[21]	Science	Authors	Open	No	Function	Operational Environmental Protection	EPC
4	Enterprise Modeling for E-Commerce (ECOMOD) Reference Model	[22]	Science	Authors	Limited	n.S.	Others	Internet Platform for Commerce	MEMO-OrqML
5	*Handels-H*-Model	[23]	Science	Authors	Open	No	Institution	Enterprises doing Commercial Functions	EPC
6	Information Technology Infrastructure Library (ITIL) PROMET I-NET Reference Model	[(15, 24)]	Practice	Office of Government Commerce	Limited	No	Function	IT-Management	Verbal
7		[25]	Science	Authors	Closed	No	Others	Intranet Conception	Proprietary Process Model
8	Process Framework of Siemens AG	[10]	Practice	Siemens AG	Closed	n.S.	Others	Development of Information and Communication Landscape	Graphical and Verbal
9	Buchwalter's Reference Model	[26]	Science	Author	Open	No	Function	Electronical ITB-Systems in Procurement	Value Chain Diagram, Task Chain Diagram
10	Reference Model of Gerber/Mai	[27]	Practice	Authors	Closed	Yes	Institution	Branch Business of Banks	Process Hierarchy-Diagrams
11	Reference Model of Haas et al.	[28]	Science	Authors	Closed	n.S.	Function	E-Learning Processes in Enterprises	EPC
12	Herrmann's Reference Model	[29]	Science	Author	Open	n.S.	Others	Reliability Requirements for Business Processes	UML Activity Diagram
13	Kluger's Reference Model	[30]	Science	Author	Limited	Yes	Function	Vehicle-based Transport System	Proprietary Process Model
14	Krömker's Reference Model	[31]	Science	Author	Open	n.S.	Institution	Creation of Offers for Unicums and Small-sized Series	IDEF0 with Process Character
15	Kruse's Reference Model	[32]	Science	Author	Open	No	Function	Distribution Logistic	EPC
16	Reference Model of Mertens/Griese	[33, 34]	Science	Author	Open	No	Institution	Industrial Enterprise	EPC
17	Neumann's Reference Model	[35]	Science	Author	Open	No	Function	Technical Facility Management	EPC
18	Pumpe's Reference Model	[36]	Science	Author	Open	No	Institution	Seaport Container Terminal	EPC
19	Remme's Reference Model	[37]	Science	Author	Open	No	Others	Management Organization	EPC
20	Rüffer's Reference Model	[38]	Science	Author	Open	No	Institution	Primary Insurer at the Example of Life Insurance Domain	Semantic Object Model (SOM) using Interaction-Schema (IAS) for Structure and Transaction-Event-Schema (VES) for Dynamic
21	Schaich's Reference Model	[39]	Science	Author	Open	n.S.	Institution	Production Machinery	UML Use Case Diagram
22	Schlagheck's Reference Model	[40]	Science	Author	Open	No	Function	Controlling	UML Activity Diagram
23	Schwegmann's Reference Model	[16]	Science	Aithor	Open	No	Function	Warehouse Management	EPC
24	Tzouvaras's Reference Model	[41]	Science	Author	Open	No	Institution	Service Processes at Book Publishers	UML Activity Diagram
25	Reference Model of Warnecke et al.	[9]	Science	Authors	Closed	n.S.	Function	Knowledge Management	Proprietary Process Model
26	SAP R/3 Reference Model	ARIS for R/3 of IDS Scheer AG [42] [(11-13)]	Practice	SAP AG	Limited	Yes	Others	n.S.	EPC
27	*Sparkassenorganisation (SKO)*-Reference Model Supply Chain Operations Reference Model (SCOR-Model)	[43-45]	Practice	Information Center of *Sparkassen-organisation GmbH*	Closed	Yes	Institution	German *Sparkassen*	EPC
28		[46]([47-49])	Practice	Supply Chain Council Inc.	Limited	Yes	Function	Supply Chain Management	Graphical and Verbal
29	Insurance Architecture (VAA)	[50]	Practice	Gesamtverband der deutschen Versicherungswirtschaft e. V. (GDV, German Insurance Association)	Limited	Yes	Institution	Insurer	Verbal Description, UML Use Case Diagram
30	Y-CIM Model	[51] ([52])	Science	Author	Open	No	Institution	Industrial Enterprise	EPC, Process Chain Diagram (VKD)

Legend:
 * - Number is estimated
 n.S. - no statement

No.	Modeling Lang.		Construction Size				Construction Method	Evaluation / Inter-subjective Verifiable	Application		
	Further Language(s)	Modeling Framework	Number of Diagrams	Number of Views	Process-related Size	Application Method(s)			Reuse and Customization	Use Case(s) / Inter-subjective Verifiable	
1	Task Model, Function Model, Data Model, Object Model	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	Multiple / No	
2	Funktion Model, Organizational Model	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	
3	Function Tree, ERM	Yes	100*	3	230*	Case studies	Case Studies / No	Procedure Model	n.S.	3 / No	
4	MEMO	n.S.	n.S.	n.S.	n.S.	n.S.	Prototype, Critical Argumentation / Partly	n.S.	n.S.	n.S.	
5	ERM, Function Tree	Yes	100*	3	1500*	n.S.	n.S.	Procedure Model for Development of an Information Strategy	Variants	n.S.	
6	Verbal	Yes	n.S.	n.S.	n.S.	n.S.	Questioning (15) / Yes	n.S.	n.S.	According to [15] Multiple Applied / No	
7		n.S.	n.S.	n.S.	n.S.	n.S.	Case Studies / Partly	PROMET-related Procedure Model	n.S.	n.S.	
8		Yes	n.S.	n.S.	n.S.	n.S.	n.S.	Procedure Model for Developing of Information and Communication Architecture	n.S.	Real Application / No	
9		No	16	2	130	Analysis of Existing Systems	Prototype / Partly	n.S.	n.S.	n.S.	
10	Class Diagrams	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	
11	ERM	n.S.	n.S.	n.S.	n.S.	Case Examples	Empirical Verification is proposed	n.S.	n.S.	n.S.	
12	UML	No	n.S.	n.S.	n.S.	Schütte's Procedure Model	n.S.	n.S.	n.S.	0 / No	
13	Data Model	n.S.	n.S.	n.S.	n.S.	Following the Construction Method for Technical Products (VDI 2222)	Prototypical Application	Procedure Model	n.S.	1 / Partly	
14		No	16	1	-	Actual Survey and Weak-point Analysis	n.S.	Procedure Model for Introducing	n.S.	3 / Yes	
15	Function Tree, ERM, Organigram	Yes	12	4	70	n.S.	n.S.	Procedure Model	Composition of Reference Modules, Customization of Model Contents	n.S.	
16	Function Tree, ERM	No	1	1	-	n.S.	n.S.	n.S.	n.S.	n.S.	
17	Value Chain, ERM	No	50	3	210*	Analysis of Existing Reference Models	Thought Experiment / Yes	n.S.	Process Extensions	n.S.	
18	Class Diagrams	No	19	2	50*	Empirical	Ad Hoc Evaluation / Partly	n.S.	n.S.	n.S.	
19		No	9	1	50*	Analysis of Design Decisions	Thought Experiment / Partly	Procedure Model	Placeholder and Specialization	n.S.	
20		No	8	3	-	Deductive	Model Comparison in Practice (Proposal)	n.S.	n.S.	n.S.	
21	UML	n.S.	n.S.	n.S.	n.S.	Balzer's Object-oriented Analysis (OOA)	Exemplary Application	n.S.	n.S.	n.S.	
22	UML Class Diagram	Yes	20	2	-	Procedure Model	Prototype / Partly	Procedure Model	Model Specialization, Build-time Operators	n.S.	
23	UML Class Diagram	No	16	2	80*	Procedure Model	Ad Hoc Evaluation / Partly	Procedure Model	Model Specialization, Build-time Operators	n.S.	
24	UML, Value Chain Diagram	Yes	32	2	-	Procedure Model	Two Case Studies / Partly	n.S.	Build-time Operators	n.S.	
25	Object Model	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	Procedure Model	n.S.	n.S.	
26	ERM, Function Tree	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	[11-13]	n.S.	n.S.	
27	Function Tree, ERM	Yes	n.S.	n.S.	n.S.	n.S.	n.S.	Procedure Model	Modeling Level, Specialization	According to [44] 30 / No	
28		n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	Specialization	Multiple	
29	ERM, Function Tree, UML Class Diagram	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	n.S.	
30	ERM, Function Tree	Yes	450*	4	300*	n.S.	n.S.	n.S.	n.S.	According to [52] Multiple / No	

Legend:
 * - Number is estimated
 n.S. - no statement

Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation

Oliver Thomas

Institute for Information Systems (IW_i)
at the German Research Center for Artificial Intelligence (DFKI),
Saarland University, Stuhlsatzenhausweg 3, Building D 3.2,
D-66123 Saarbruecken (Germany)
thomas@iwi.uni-sb.de
<http://www.iwi.uni-sb.de>

Abstract. The heart of every scientific discipline is its own unique, uniform and acknowledged terminology. As an application-oriented mediator between business administration and computer science, information systems research in particular is in need of a theoretical foundation and an instrument capable of translating basic theoretical knowledge into practical applications. Its dependency on and proximity to actual practice, as well as the rapid development of information technology often get in the way of the sound, systematic and consistent formation of concepts. Reference modeling is especially in need of a theoretical foundation. Due to the strong influence of implementation-oriented thought within this field, a gap has resulted between research and practice which has often led to undesirable developments. The high expectations organization and application system developers have on the reutilization of reference models are often disappointed. Apparently, the recommendations made by reference model developers often do not meet the expectations of potential model-users. One reason for this is the non-uniform grasp of the term reference model. This article attempts to counteract this deficiency by way of a detailed analysis of the way the term reference model is used and understood.

1 Initial Situation and Problem

Information systems are mediators between business frameworks and information technology and can be characterized using in-depth system-theoretical attributes. For example, the complexity of information systems can be seen as a significant system-theoretical attribute. Put simply, this complexity can be attributed to the fact that information systems work on a business, as well as on a technical level. By constructing models, the attempt is made to create manageable artifacts with which the complexity of information systems becomes controllable. The information models created thereby have a tradition of more than thirty years [2; 10; 16]. From today's perspective, these models have established themselves in information systems research as a vital medium for describing operational information systems [18; 20; 23; 28; 34; 40; 41].

The application possibilities inherent in information models range from software design and the implementation and configuration of standard software to business process reengineering.

Due to the possibility of their reutilization, in many cases the construction of information models is connected to the demand to abstract from enterprise-specific characteristics. Therefore, one differentiates between enterprise-specific information models and reference models. The term “enterprise-specific” characterizes only the individual character of the corresponding information model; there is no restriction to legally independent companies connected with it. For reasons of linguistic clarity it is therefore better to speak of specific information models in order to allow for the fact that the specificity of the models does not result exclusively from the enterprise-context but rather, for example, also from a project-context. To emphasize this context one can also speak, for example, of project-specific models.

In contrast to this, a reference model – in the sense of an initial conceptual approach – is a point of reference for the development of specific models because it represents a category of applications [5, p.90; 35, p.66, pp.69–74; 39, pp.31–38]. Prominent examples of this in the scientific field are the reference model for industrial enterprises (Y-CIM-Model) from SCHEER [32], as well as the SAP R/3-reference model [11] resulting from commercial practice. On the one hand, the possibility of orienting oneself on the technical content of such reference models promises the model-users savings in time and costs, while on the other the quality of the model to be constructed can be increased by the use of a reference model.

Despite these benefits often attributed to reference models in literature, no uniform grasp of the term “reference model” exists. In research and practice different types of models are referred to as reference models. For HARS for example, the term reference model “belongs to a class of terms used often but rarely defined clearly” [17, p.12]. Even a decade after this assertion the situation has barely changed. Although the term reference model was defined more precisely at the end of the 1990ies during the conference *Reference Modeling* – a summary of the conference series is available under the URL <http://www.wi.uni-muenster.de/is/Tagung/> – and the dissertation from SCHÜTTE [35] – that is at least in German-speaking regions – the tendency in literature towards generally declaring information models as reference models still exists. In this respect, the assertion from LEHNER that “in a sense every model can be understood as a reference model” [22, p.126] is not surprising. The question as to why recommended models “warrant” the attribute “reference” in literature often goes unanswered. Here, we have singled out one of the many current unfounded reference model declarations from the German-language information systems community: Using a reference model AHLEMANN, HAAS, HOPPE [3] show how e-learning can be systematically integrated in the further education of companies by establishing it in the operational planning system. Although they explain their grasp of the term reference model according to SCHÜTTE [35, p.69], why they refer to their model as a reference model and not an information model is not explained.

The following analysis on the way the term reference model is understood in information systems research takes this situation into account. It is structured in the following manner: Section 2 first lists “early” considerations to the term reference model from a historic perspective, as well as giving an etymological analysis of the term. Following this, in Section 3, the current attribute-based characterizations of the term

reference model in the literature of today will be discussed critically. The insights resulting from this will flow into a set-theoretic illustration, as well as an explanation of the way the term is understood in Section 4. A critical discussion of the findings in Section 5 shows the consequences resulting from the definition presented here for the use of reference models. The article ends with a conclusion in Section 6.

2 Etymology and History of the Term Reference Model

From an etymological view, the term “reference” has a double significance. In addition to its meaning as a recommendation, the word “reference” is also used in the sense of bearing a relation to something, quoting something or alluding to something. The term “reference” was initially used in the business language of the 19th century to denote a person or company able to give information concerning the trustworthiness of a business partner. The definition of a person or place to whom or where one could appeal for his or her (social) recommendation came later [1, p.464].

In linguistics “reference” also refers to the relationship between linguistic symbols and their contributor in the extra-linguistic reality. In economics, “reference” is used to describe a state which can not be achieved in reality or a state of affairs of exemplary nature. Thus for example, the model of perfect competition, discarded to a large extent due to its restrictive assumptions, is accepted as a reference. In information modeling, one also speaks of a model being consulted as an ideal type of reference object or as a recommendation for the development of other models.

The historic roots of the term reference model in information systems research can only be traced with difficulty. Nevertheless, early clues to the basic idea of reference modeling can be found in the literature which today essentially consists in the systematic structuring and reutilizing of operational tasks for their data processing support.

The significance of graphic models valid for a class of applications was discussed early on in business administration literature. Already 1931 NORDSIECK characterized in *Grundprobleme und Grundprinzipien der Organisation des Betriebsaufbaus* so-called *Aufgabengliederungspläne*¹ as follows: “Usually, a task structuring plan already has a relatively universal character because it is created according to logical principles, i.e. it is not only valid for the company being studied but rather – with a few changes – for companies with similar aims and the same branch of trade” [24, p.160].

Also, the *ideal models* described by KOSIOL in an analysis of the relationships between business administration and operations research come close to today’s term reference model. He explains: “So-called real models which try to represent objects of empirical reality are opposed to ideal models which exhibit no reference to reality or leave this open” [21, p.755]. He adds, that “ideal models are the constructs of operations research which represent a larger area of possible real-life situations and serve as prefabricated solutions or standard recipes for certain categories of decision problems in coping with practical problems” [21, p.758].

¹ Today the term “Aufgabengliederungspläne” which may be translated as “task structuring plan” has gone out of use. The technical terms “function hierarchy diagram” resp. “function tree” have won recognition as terms in the meaning of the corresponding modeling language.

Another early paraphrase for the fundamental idea of reference modeling can be found in the environment of the *System Dynamics* approach going back to FORRESTER. This is a concept founded on the systems theory for the model-based description and simulation of dynamic systems. In 1968 FORRESTER wrote retrospectively: “A person applying the industrial dynamics approach to actual corporate problems seems to do so by drawing heavily on his mental library of the systems which he has previously studied. If others are to be able to do the same, such libraries of examples must be put in orderly written form. Such a series of structures would identify those relationships which are found repeatedly in industry. [...] Such a treatment of systems should concentrate on the minimum structure necessary to create a particular mode of behavior.” [13] FORRESTER thus characterizes an attribute of reference models which attempt to abstract from individual characteristics in order to make themselves reusable.

The question however still exists, as to which origins the term reference model can be traced back to. There is a consensus in literature on the fact that the terminological foundation for “reference model” – in terms of a reference information model – was laid with the *Köln Integration Model (KIM)* [15; 16]. However, neither of these publications speaks of a “reference model”. Instead they speak of the development of a “universal model for an integrated data processing system” [16, p. VII], a “basic model” [16, p. X] or a “model template” [15, p. 44]. These terms characterize models “that are generalized in a way, that they are not specific to an individual company, but rather characteristic for all resp. the lion’s share of companies from a certain group or branch of trade” [15, p. 43]. These models should serve in helping companies to create their own individual information system [16, p. X].

Despite these early references to the significance of universal models and their usefulness as templates for the derivation of enterprise-specific models, the technical term “reference model” first established itself in literature towards the end of the 1980ies [14; 26; 29; 43]. This chronological correlation can be supported by looking at different editions of the book *Business Process Engineering* from SCHEER [32]. In the first edition, the data model developed therein is referred to as an integrated database schema resp. an enterprise-wide data model [29]. Then, in the preface of the second edition, SCHEER states that the consideration of the company data model was complemented by practical experience gained in the between-time using the model as a basis for enterprise-specific data models [32, p. VIII]. At another point in the same edition he makes this statement more precise by remarking, that the model had already been used several times as a reference model in setting up enterprise-wide data models [32, pp. 542ff.].² The acceptance of the model as a reference model in practice even prompted SCHEER to give the book a different subtitle in the second edition *Reference Models for Industrial Enterprises* [32]. This publication was material to the coinage of the term “reference model” – that is, in the realm of German-speaking information systems research.

² Compare this assessment to the statement from ÖSTERLE, BRENNER, HILBERS, who believe that the data models from [29] are primarily to be consulted as reference models for the development of enterprise-specific models [25, p. 71].

3 Characterization of the Term Reference Model Based on Attributes

The proposed reference model terms in information systems literature are generally based upon attributes which characterize these reference models, in particular, the attributes “universality” and “recommendation character” [39, pp. 31 ff.].

3.1 The Attribute Universality

The demand for universality as a constituent attribute of the term reference model can be found in many works [5, p. 90; 17, p. 15; 19, p. 12; 35, p. 69; 42, p. 127]. For purposes of simplification these publications also talk of the universality of reference information models. HARS for example, sees the universality of a reference model as a prerequisite for it serving as a source for the creation of a specific model [17, p. 15]. JOST explicitly emphasizes that the character of universality is the most significant attribute of a reference model [19, p. 12]. The fact however, that the universality of a reference model can not be understood in the sense of the model’s claim to absoluteness, i.e. a claim to universal validity, often goes unrecognized. A reference model can only be (universally) valid with regard to a certain category of applications, for example a category of enterprises or a category of projects. Already in 1980 BRETZKE differentiated in his analysis *Der Problembezug von Entscheidungsmodellen* between two types of models, concrete and common decision models [8, pp. 10 ff.]. If one transfers his remarks to enterprise-specific models and reference models, then a reference model “is characterized by the fact that it applies to a certain category of situations. It is not universal because it is always valid, but rather because it is always valid under certain circumstances (contained within itself)” [8, p. 11]. To speak of the universality of a reference model is therefore seen as being inexpedient in this article. Thus, the allowance for a corresponding constituent attribute for the term reference model was not considered here.

3.2 The Attribute Recommendation Character

In addition to universality, it is possible in other works to find the demand for a recommendation character as a constituent attribute for the term reference model [4, pp. 25 f.; 5, pp. 86, 90; 7, p. 428; 27, pp. 16 f.; 35, p. 69]. Authors connect such a recommendation with the fact that reference models have a standard character for a certain class of applications. They serve as a default solution, from which enterprise-specific concretizations can be derived (economically). Similar to the argumentation in the previous section, the demand for a recommendation character for reference models also proves to be critical. For example, it is unclear how the quality of a recommendation for a reference model can be verified – in this regard VOM BROCKE [39, p. 32] also speaks of the lack of assessability for the content of a recommendation: Which model can be granted or even denied recommendation character subject to which attributes? Which demands can be made on the recommendation or those making the recommendation? These questions make it obvious that this is a question of a non-operational aspect. The user cannot decide upon the recommendation

character of a model objectively, but rather only subjectively within the scope of its application. Therefore, this attribute must also be seen as non-constituent for the term reference model in this article.

4 Implications for the Term Reference Model

4.1 Set-Theoretic Illustration of the Way the Term Reference Model Is Understood

Since both attributes “universality” and “recommendation character” have been excluded as constituent attributes for the term reference model, the question remains as to how a model becomes a “reference”. To answer this question we must first look at model-theoretic principles [36] in which a developer and a user perspective on models are taken into consideration. Using these perspectives one can discern whether a model is *declared to be* a reference model (developer’s perspective) or whether it is *accepted* as a reference model (user perspective). “Or” is not used here in its colloquial sense, but rather should be understood in a Boolean sense as an adjunction (non-excluding “or”), so that the case of the developer-sided declaration *and* the user-sided acceptance is also taken into consideration. Elementary set-theoretic considerations were consulted in order to illustrate possible situations. These are illustrated in Fig. 1 and will be explained in the following.

The basic set seen in Fig. 1 is the set of all information models IM . As subsets of this set, the set of the information models declared to be reference models by the developers of the models $RM_{\text{Declaration}}$, as well as the set of the information models used by model-users for the construction of specific models $RM_{\text{Acceptance}}$ are plotted. For the characterization of reference models three situations are conceivable:

1. $RM_{\text{Declaration}} \cap \overline{RM_{\text{Acceptance}}}$: The elements of this set are declared as reference models without being accepted by a user. In this case, the property of being a reference model is based upon the assertion of the developers.
2. $\overline{RM_{\text{Declaration}}} \cap RM_{\text{Acceptance}}$: It is conceivable that users consult a model for the construction of specific models, although the model’s developers did not initially intend this. Corresponding information models are characterized by this set.
3. $RM_{\text{Declaration}} \cap RM_{\text{Acceptance}}$: The intersection of both sets takes the information models declared to be reference models by the developers, as well as those accepted by the users as such into account. A consensus between developer and user exists in regard to the characterization of the elements of this set as reference models.³

³ This point of view precludes the case that a model seen neither from the developer’s side nor from the user-side as a reference can be declared a reference model [39, p.32, fn.139]. Moreover, it remains unclear in this case whose task it is to make the declaration resp. the model as a reference model.

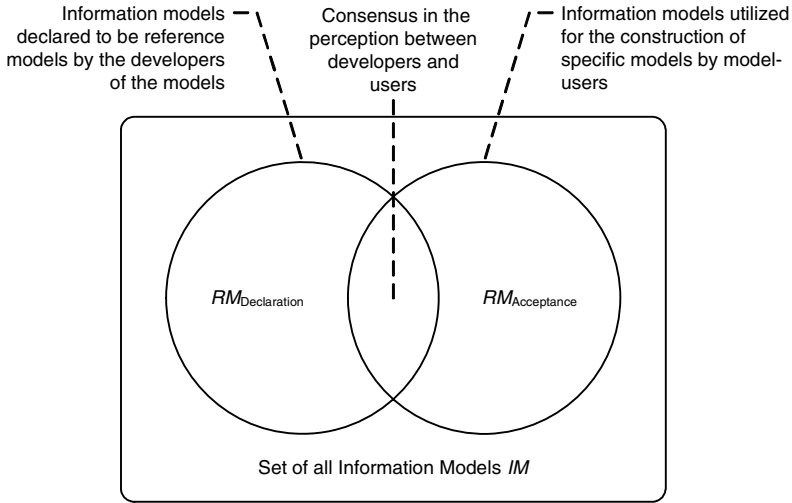


Fig. 1. Set-theoretic illustration of the term reference model

In this article, the developer-sided declaration as reference model (cp. set $RM_{Declaration}$ in Fig. 1) is seen neither as a necessary, nor as a sufficient criterion for the characterization of a reference model. A developer’s assertion that he has constructed a universally valid and recommendable model remains meaningless for the time being. In this context, VOM BROCKE also speaks of “reference character at plan level” [39, p.33, fn. 140]. This attribute can ultimately be proved only by way of the model being applied at least once. SCHEER argues similarly. He concretizes the demands on a reference model from a user’s point of view to the effect that at least one application must be conceivable for the use of the model, unchanged, as a specific model [33, p.4]. In an economic sense, a reference model that goes unused undoubtedly falls short of its basic intention. The use of an information model by a model-user for the derivation of specific models, i.e. its acceptance as a reference (cp. set $RM_{Acceptance}$ in Fig. 1) can thus be seen as a necessary criterion for the characterization of a model as a reference.

To clarify whether it can also be acknowledged as a sufficient criterion, two cases can be distinguished from a set-theoretic point of view. Either the model was also recommended by the developer as a reference, i.e. it is contained in the set $RM_{Declaration} \cap RM_{Acceptance}$ or the developer did not intend this, i.e. it is contained in the set $\overline{RM_{Declaration}} \cap RM_{Acceptance}$ (cp. Fig. 1). The first case can be seen as ideal and thus as uncritical due to the consensus between the developer and user. This paper however, also recognizes the second case as constituent for the term reference model. As a result, only the user can make the decision as to whether a model can be recognized as a reference. User-sided acceptance can be seen as a sufficient criterion. Consequently, it is possible for a model to become a reference model at its initial application; if need be without the knowledge of its developer [39, p. 34].

It would be ideal for the constructor of a model to declare his model a reference model only when its application is known to him in at least one case. This grasp of the term is justified by the example from literature already discussed in Section 2. SCHEER also initially developed a data model which he then recommended for the derivation of enterprise-specific models [29]. It has however, turned out that in practice the model's recommendation character was accepted. Thus for example, a field report from BÜRLI et al. [9] was published on the derivation of a specific model based upon the information model from SCHEER for the field of production planning and control. This then prompted SCHEER to declare the model to be a reference model [32].

4.2 Explanation of the Term Reference Model

The term reference model can be explained as a concretion of the term "information model" on the basis of the constituent attribute of user-sided acceptance: A *Reference model* – specifically: reference information model – is an information model used for supporting the construction of other models.

This definition stands in the tradition of early definitions and emphasizes the benefits of reference models "as a fundamental starting point for the development of new information systems" [30, p.94]. HARS also emphasizes the user-sided acceptance by stating that "every reference model is a model which can be consulted for the development of other models" [17, p.15]. SCHEER later abstracts from information models and sees a reference model "as a model which can serve as the starting point for the development of solutions based on concrete problems" [33, p.3]. A corresponding tendency in emphasizing the use of reference models can also be observed in the more recent literature of reference modeling. Thus BECKER, KNACKSTEDT refer to information models used as initial solutions for the development of project-specific models [6, p.415], as reference models.

Consequently, the author pleads for a use-oriented reference model term. Every model resp. partial model which can be used in supporting the construction of another model can be seen in this sense as a reference model. The reutilization of reference models connected with this can be seen as a fundamental idea resulting from the paperless, tool-supported data-processing consulting at the beginning of the 1990ies [31] and must be emphasized as a fundamental characteristic of reference models.

5 Discussion on the Term Reference Model as Defined Here

5.1 Consequences for Reference Modeling Research

Studies in the field of reference modeling must often deal with the fundamental problem of finding and locating reference models. Because reference models are understood as special information models, the search can initially be limited to information models. When an information model is found one must then decide whether it is a case of a reference model or not. In making this decision, the person searching for the model is confronted with a problem in two respects. Firstly, one can only subjectively decide whether a model is a reference model. However, even if one person accepts a reference model as such, this does not mean that the next person will also do so. And

secondly, identifying criteria such as universality or recommendation character must be dismissed as constituent characteristics of a reference model. This examination follows the use-oriented reference model term from Section 4, which is directed at the model's use. The models declared exclusively as being reference models are not accepted as such.

Were one to use this "restrictive" understanding of the term in reference modeling research the number of actual reference models would be small, because by close interpretation the existence of at least one application – moreover: its documentation – would be essential. The focus of contextual studies in reference modeling should therefore be extended to the models only declared as being reference models. This corresponds in two ways with the pragmatic orientation expounded upon at the beginning of this article. On the one hand, the topic of this article is not to judge whether models declared to be reference models in literature should actually be accepted as such. And on the other, the deduction of future research guidelines can only be possible by way of analyzing prevailing perceptions.

5.2 Consequences for the Management of Reference Models

It is irrelevant for a model's user – in the sense of the term reference model used in this paper – whether a model, whose content he wishes to reuse, has been recommended for use by the model's developer i. e. was declared to be a reference model or not. He orients his decision on the use of a reference model only on whether he can recognize a potential benefit from the model. In order to make this decision the reference model must be made available to the user. An important prerequisite for the structuring of this availability is the systematic management of the current stock of reference models [37; 38].

Despite the variety of existing reference models there are very few studies in literature with the verification and documentation of actual reference models as their subject. Based on this, there is also a lack of studies regarding the question of which reference models should be used in which situations. A very small number of approaches deal with the systematization of reference models, whereby it is in fact the tabulation of reference models that is meant here and not so much the survey-like textual description of the actual stock of reference models found in literature. The most comprehensive results were delivered by the analyses from FETTKE and LOOS [12] on the catalog-based reutilization of reference models in which the authors transfer the concept of a construction catalog used in engineering to reference modeling. These so-called *reference model catalogs* represent without a doubt, a meaningful tool for the systematic management of reference models.

However, it must be pointed out with respect to the cataloging only of reference information models, that those involved in the development and administration of a reference model catalog also have the problem mentioned in Section 5.1: They must decide which information model can be accepted as a reference model and thus be cataloged. Taking into consideration the exact interpretation of the term reference model which is the basis of this article, only models for which at least one application exists could then be cataloged. This implicates that the users of such a catalog, limited only to reference models, would be principally refused access to enterprise-specific models. This circumstance contradicts the pragmatic focus of the use-oriented refer-

ence model term in this study, because it remains unconsidered that information models generally – even when the developer has declared it a reference model or it has already been used – are used to support the construction of other models. This results, for the design of a reference model catalog, in the need for an expansion in the direction of a systematic organization of information models, independent of their contextual individuality.

5.3 Consequences for the Creation of Reference Modeling Languages

The use-oriented reference model term underlined in this publication emphasizes the use of a reference model for the construction of enterprise-specific models. The user's task during construction, which can be supported by IT tools, consists in the adaptation of the reference model. In a figurative sense, the derivation of specific models from a reference model characterized by this term is equivalent to the creation of different variants of the reference model [35, pp.207–209]. Thus, for example, the enterprise-specific models *information model product-oriented Manufacturing Enterprise E₁* or *information model process-oriented Manufacturing Enterprise E₂* could be derived as variants of the reference model *Manufacturing*.

The management of variants derived from reference models is especially interesting in two respects. Firstly, the storage of the variants in connection with the adaptation-premises also administrated can speed up the future development of enterprise-specific models for comparable applications. Secondly, this also allows for a similarity analysis of the variants whose results can then be used for the development of new reference models.

Reference modeling languages must therefore be created so that they support model-variant management. However, contradictory opinions exist in literature as to which construction technique should be used for reference model-variant management. While for example, SCHÜTTE ties variant management *ex ante* to the construction technique of the configuration [35, pp.207ff.] and also refers to a variant as a configured output in his terminology [35, p.207, fn.91], VOM BROCKE [39, p.101] argues against the coupling of variant management with individual construction techniques and proposes further construction techniques with aggregation, specialization, instantiation, and analogy construction [39, pp.235ff.]. These construction techniques for the adaptation of models must be embedded in modeling languages. The effort needed for the expansion of these languages is however, so high that it can, by all means, more than make up for the benefits which can be achieved by adapting reference models within the framework of modeling projects. Reference modeling research must therefore dedicate itself more heavily to the question of profitability in the application of reference models in the future.

6 Final Conclusion

The topic of this article was the detailed analysis of the understanding of reference models in the information systems discipline. The author did not intend to create a comprehensive and universally valid definition of the term. In fact, the author's aim was to examine the term "reference model" from different perspectives and on the

basis of this, create an understanding which he hopes will prove to be useful in the context of information systems research. The author hopes to have contributed a valuable share in answering the question “What is a reference model?” put in the call of the *Workshop on Business Process Reference Models (BPRM 2005)*.

The need clearly remains for more fundamental research in order to understand the effects connected to the creation and use of reference models in research and practice. However reference modeling research approaches this topic in the future, the compilation of improved knowledge on the application systems and organizations remains a central topic in this field of research. A terminological foundation for the management of this knowledge represented by reference models has been made available by the insights gained within the scope of this article.

Acknowledgements. This paper presents results from the research project “Reference Model-Based Customizing with Vague Data”, abbreviated “Fuzzy-Customizing”, funded by the Deutsche Forschungsgemeinschaft (German Research Foundation) as part of the initiative BRID². The author would like to thank the anonymous reviewers for their comments which helped to improve an earlier version of this paper.

References

1. Drosdowski, G. (ed.): *Duden Etymologie : Herkunftswörterbuch der deutschen Sprache*. 2nd ed. Mannheim : Dudenverl., 1989 (in German)
2. Abrial, J.-R.: Data Semantics. In: Klimbie, J. W.; Koffeman, K. L. (eds.): *Data Base Management : Proceeding of the IFIP Working Conference Data Base Management, Cargèse, Corsica, France, 1–5 April, 1974*. Amsterdam : North-Holland Pub. Co, 1974, pp. 1–60
3. Ahlemann, F.; Haas, C.; Hoppe, U.: Organisationale Integration von E-Learning in Unternehmen – ein Referenz-Informationsmodell. In: Uhr, W.; Esswein, W.; Schoop, E. (eds.): *Wirtschaftsinformatik 2003 : Medien – Märkte – Mobilität ; Band 1*. Heidelberg : Physica, 2003, pp. 707–726 (in German)
4. Becker, J.; Delfmann, P.; Knackstedt, R.; Kuropka, D.: Konfigurative Referenzmodellierung. In: Becker, J.; Knackstedt, R. (eds.): *Wissensmanagement mit Referenzmodellen : Konzepte für die Anwendungssystem- und Organisationsgestaltung*. Heidelberg : Physica, 2002, pp. 25–144 (in German)
5. Becker, J.; Holten, R.; Knackstedt, R.; Schütte, R.: Referenz-Informationsmodellierung. In: Bodendorf, F.; Grauer, M. (eds.): *Verbundtagung Wirtschaftsinformatik 2000*. Aachen : Shaker, 2000, pp. 86–109 (in German)
6. Becker, J.; Knackstedt, R.: Konstruktion und Anwendung fachkonzeptioneller Referenzmodelle im Data Warehousing. In: Uhr, W.; Esswein, W.; Schoop, E. (eds.): *Wirtschaftsinformatik 2003 : Medien – Märkte – Mobilität ; Band 2*. Heidelberg : Physica, 2003, pp. 415–434 (in German)
7. Becker, J.; Schütte, R.: Referenz-Informationsmodelle für den Handel: Begriff, Nutzen und Empfehlungen für die Gestaltung und unternehmensspezifische Adaption von Referenzmodellen. In: Krallmann, H. (ed.): *Wirtschaftsinformatik '97 : Internationale Geschäftstätigkeit auf der Basis flexibler Organisationsstrukturen und leistungsfähiger Informationssysteme*. Heidelberg : Physica, 1997, pp. 427–448 (in German)
8. Bretzke, W.-R.: *Der Problembezug von Entscheidungsmodellen*. Tübingen : Mohr, 1980 (in German)

9. Bürli, A.; Jaccottet, B.; Knolmayer, G. F.; Myrach, T.; Küng, P.: Vorgehen beim Aufbau von CIM-Datenmodellen. In: *io Management Zeitschrift* 61 (1992), no. 12, pp.82–86 (in German)
10. Chen, P. P.-S.: The entity-relationship model – toward a unified view of data. In: *ACM Transactions on Database Systems* 1 (1976), no. 1, pp.9–36
11. Curran, T. A.; Keller, G.; Ladd, A.: *SAP R/3 business blueprint : Understanding the business process reference model*. Upper Saddle River, NJ : Prentice Hall PTR, 1998
12. Fettke, P.; Loos, P.: Classification of Reference Models – A Methodology and its Application. In: *Information Systems and e-Business Management* 1 (2003), no. 1, pp.35–53
13. Forrester, J. W.: Industrial Dynamics – After the First Decade. In: *Management Science* 14 (1968), no. 7, pp.398–415
14. Gersting, J.; Kinsley, K.; McDonald, N.; North, J.; Sastry, M.; Stull, E.: Reference model for DBMS user facility. In: *ACM SIGMOD Record* 17 (1988), no. 2, pp.23–52
15. Grochla, E.: Das Konzept des Kölner Integrationsmodells. In: Grochla, E. (ed.): *Integrierte Gesamtmodelle der Datenverarbeitung : Entwicklung und Anwendung des Kölner Integrationsmodells (KIM)*. München : Hanser, 1974, pp.35–46 (in German)
16. Grochla, E.; Garbe, H.; Gillner, R.; Poths, W.: Grundmodell zur Gestaltung eines integrierten Datenverarbeitungssystems : Kölner Integrationsmodell (KIM). In: Grochla, E.; Szyperski, N. (eds.): *Arbeitsberichte des Betriebswirtschaftlichen Instituts für Organisation und Automation (BIFOA) an der Universität zu Köln*, no. 71/6, Köln : WISON Verl., 1971 (in German)
17. Hars, A.: *Referenzdatenmodelle : Grundlagen effizienter Datenmodellierung*. Wiesbaden : Gabler, 1994 (in German)
18. Hay, D. C.: *Requirements analysis : From business views to architecture*. Upper Saddle River, NJ : Prentice Hall PTR, 2003
19. Jost, W.: *EDV-gestützte CIM-Rahmenplanung*. Wiesbaden : Gabler, 1993 (in German)
20. Kilov, H.: *Business models : A guide for business and IT*. Upper Saddle River : Prentice Hall, 2002
21. Kosiol, E.: Betriebswirtschaftslehre und Unternehmensforschung : Eine Untersuchung ihrer Standorte und Beziehungen auf wissenschaftstheoretischer Grundlage. In: *Zeitschrift für Betriebswirtschaft* 34 (1964), no. 12, pp.743–762 (in German)
22. Lehner, F.: Modelle und Modellierung. In: Lehner, F.; Hildebrand, K.; Maier, R. (eds.): *Wirtschaftsinformatik : Theoretische Grundlagen*. München : Hanser, 1995, pp.73–164 (in German)
23. Mylopoulos, J.: Information Modeling in the Time of the Revolution. In: *Information Systems* 23 (1998), no. 3/4, pp.127–155
24. Nordsieck, F.: Grundprobleme und Grundprinzipien der Organisation des Betriebsaufbaus. In: *Die Betriebswirtschaft* 24 (1931), no. 6, pp.158–162 (in German)
25. Österle, H.; Brenner, W.; Hilbers, K.: *Unternehmensführung und Informationssystem : Der Ansatz des St. Galler Informationssystem-Managements*. 2nd ed. Stuttgart : Teubner, 1992 (in German)
26. Peckham, J.; Maryanski, F.: Semantic data models. In: *ACM Computing Surveys* 20 (1988), no. 3, pp.153–189
27. Rosemann, M.; Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung. In: Becker, J.; Rosemann, M.; Schütte, R. (eds.): *Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung : Proceedings zur Veranstaltung vom 10. März 1997*. Münster : Institut für Wirtschaftsinformatik, Westfälische Wilhelms-Universität, 1997, pp.16–33 (in German)

28. Rossi, M.; Siau, K.: *Information Modeling in the new Millennium*. Hershey : Idea Group Publishing, 2001
29. Scheer, A.-W.: *Enterprise-wide data modelling : Information systems in industry*. 1st ed. Berlin : Springer, 1989
30. Scheer, A.-W.: Unternehmensdatenmodell. In: *Information Management 5* (1990), no. 1, pp.92–94 (in German)
31. Scheer, A.-W.: Papierlose Beratung – Werkzeugunterstützung bei der DV-Beratung. In: Scheer, A.-W. (ed.): *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, no. 81, Saarbrücken : Universität des Saarlandes, 1991 (in German)
32. Scheer, A.-W.: *Business Process Engineering : Reference Models for Industrial Enterprises*. 2nd ed. Berlin : Springer, 1994
33. Scheer, A.-W.: ARIS – House of Business Engineering: Konzept zur Beschreibung und Ausführung von Referenzmodellen. In: Becker, J.; Rosemann, M.; Schütte, R. (eds.): *Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung : Proceedings zur Veranstaltung vom 10. März 1997*. Münster : Institut für Wirtschaftsinformatik, Westfälische Wilhelms-Universität, 1997, pp.3–15 (in German)
34. Scheer, A.-W.: *ARIS – business process modeling*. 2nd ed. Berlin : Springer, 1999
35. Schütte, R.: *Grundsätze ordnungsmäßiger Referenzmodellierung : Konstruktion konfigurations- und anpassungsorientierter Modelle*. Wiesbaden : Gabler, 1998 (in German)
36. Stachowiak, H.: *Allgemeine Modelltheorie*. Wien : Springer, 1973 (in German)
37. Thomas, O.: Reference Model Management. In: *6th International Conference The Modern Information Technology in the Innovation Processes of the Industrial Enterprises MITIP 2004 : September 9–10, 2004 Prague, Czech Republic ; Proceedings*. Prague : Czech Technical University, 2004, pp.33–36
38. Thomas, O.; Adam, O.; Seel, C.: SAP Business Process Model Management. In: *Business Process Innovation : Proceedings ; SAP Innovation Congress Americas '04, February 28-March 1, 2004*. Orlando : SAP AG, 2004
39. vom Brocke, J.: *Referenzmodellierung : Gestaltung und Verteilung von Konstruktionsprozessen*. Berlin : Logos, 2003 (in German)
40. Wand, Y.; Weber, R.: Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda. In: *Information Systems Research 13* (2002), no. 4, pp.363–376
41. Weber, R.: *Ontological foundations of information systems*. Melbourne : Coopers & Lybrand and the Accounting Association of Australia and New Zealand, 1997
42. Wolf, S.: *Wissenschaftstheoretische und fachmethodische Grundlagen der Konstruktion von generischen Referenzmodellen betrieblicher Systeme*. Aachen : Shaker, 2001 (in German)
43. Wollnik, M.: Ein Referenzmodell des Informations-Managements. In: *IM Information Management 3* (1988), no. 3, pp.34–43 (in German)

On the Syntax of Reference Model Configuration – Transforming the C-EPC into Lawful EPC Models

Jan Recker¹, Michael Rosemann¹,
Wil van der Aalst^{1,2}, and Jan Mendling³

¹ Queensland University of Technology,
126 Margaret Street, Brisbane QLD 4000, Australia
{j.recker, m.rosemann, w.vanderaalst}@qut.edu.au

² Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

³ Vienna University of Economics and Business Administration,
Augasse 2-6, 1090 Vienna, Austria
jan.mendling@wu-wien.ac.at

Abstract. Enterprise Systems need to be *configured* to fit organizational requirements and to provide support for their business operations. Reference models aim at supporting this task but fail in providing adequate conceptual support due to missing configurability of the models themselves. Our research extends the work on *a configurable reference modeling approach*. In previous research we developed a conceptual notation for configurable reference models. This paper considers a syntactic perspective of reference model configuration. We discuss the lawful environments of configurable nodes and report about syntactic implications of model configuration in these environments. We then apply these findings in the design of an *interchange format for configurable reference models* and discuss its applicability for the XML-based design of tool support, which ultimately will facilitate the automatic verification and transformation of reference process models to executable workflow specifications.

1 Reference Models and Enterprise Systems

Many organizations suffer problems from poorly implemented Enterprise Systems (ES) [1]. Both academia and industry state that these problems result from a misalignment gap between business and IT, which, once closed, would lead to significantly improved business performance [2]. The notion of (mis-)alignment primarily embraces the process dimension, i.e. the alignment of IT functionality to the actual business processes of an organization. In many cases, it is observed that the system hampers the normal way of handling processes instead of supporting it. This is even more surprising given the fact that business

process orientation as a concept has been a major topic in both academia and practice at least since the 1990's [3, 4]. Alongside this trend, the IS community has experienced the proliferation of an enormous number of process modeling methods, including the Event-Driven Process Chains (EPC) [5], which itself is used within the Enterprise System SAP.

The term Enterprise Systems represents integrated information systems that aim at holistically supporting the operational processes of organizations. Though ES packages are distributed as Commercial Off-The-Shelf (COTS) software, their implementation often results in tremendous configuration efforts. Given the fact that the alignment of “generic” ES solutions to “specific” organizational needs denotes a highly complex task, it was found that a model-driven solution would provide a more intuitive approach towards configuring, adapting and customizing ES software to customer demands. Such a model-driven approach naturally would take on existing *reference models*, which have already been developed by ES vendors in order to improve the understandability of their systems. In the context of Enterprise Systems, such *application reference models* that describe structure and functionality of software solutions on different levels of conceptual abstraction are of particular interest. Due to their prescriptive nature, i.e. application reference models usually depict the complete functionality of the system [6], they are however only of limited use to the ES configuration process, mainly due to a lack of conceptual support in the form of a configurable modeling language underlying the reference models.

Addressing this issue, we have developed a new reference modeling approach which considers the configurable nature of an Enterprise System. The representation language of this approach is called *Configurable EPCs* (C-EPCs). While previous research efforts have focused on the meta model and the notation of C-EPCs [7], this paper discusses syntactical problems of C-EPCs in the process of reference model configuration. The *scope of our paper* is the translation of (configured) C-EPCs to lawful (regular) EPCs. We will show that the application of C-EPC in the process of ES reference model configuration leads to syntactic problems and we will outline an approach how to handle these problems when translating C-EPC models into lawful process models. More specifically, the *aim of our paper* is to outline a XML schema-based approach using the EPC Markup Language (EPML) [8] for the task of syntactical validation of reference process model configuration.

The remainder of our paper is structured as follows: Section 2 presents issues and shortcomings of the EPC notation in light of reference model configuration and introduces the notion of a configurable reference process modeling technique. Also, it briefly reports on related work in the field of configurable reference modeling. Section 3 discusses problems that occur when configuring reference process models. We present a XML-based specification of C-EPCs on which the design of tool support for syntax validation and automatic model translation will be based. We briefly summarize our work in Section 4 and propose some conclusions.

2 A Configurable Reference Modeling Language

2.1 On the Syntax and Semantics of EPCs

In order to gain an understanding for the C-EPC notation and to raise awareness of problems we encounter during reference process model configuration, we briefly outline the notion of classical EPC models and discuss some issues related to the informal semantics and syntax of EPC.

The EPC language was developed at the University of Saarland, Germany, in collaboration with SAP AG (see [5]). A simple EPC consists of events as passive states, functions as active transformations, and logical connectors that connect events and states through control flow. EPCs have - amongst others - been used for the design of the reference process models in SAP [6]. As discussed quite intensively in academia, see e.g. [9,10], the definition of EPC in [5], on which we based our research on the C-EPC language, leads to syntactic and semantic problems. The syntax of EPCs as deployed in our research context can be found in [7]. However, this definition does not cover behavioral aspects of EPCs and thus may contain semantic ambiguities. For instance, the informal semantics of an OR-join causes confusion as a joining OR-connector may or may not synchronize incoming process flows [10]. While these problems have been addressed in academic contributions, see e.g. [9,11,12], and while there exist approaches to provide semantics to EPCs, see e.g. [13], there is not yet a generally accepted solution to the issue of EPC semantics.

Considering such problems before the background of ES configuration, the informal semantics of EPC lead to severe issues: EPC models, which depict those process scenarios that are deemed relevant to a particular organization, need to be translated into executable process specifications, which an Enterprise System can execute at run-time. Or, consider a workflow management system that defines, executes, manages and controls business processes based on these models. In whatever case, it is of paramount importance to have syntactically correct, i.e. *lawful* EPC process models as an outcome of the configuration process.

Yet, we did not want to further complicate the semantics of EPCs by introducing new semantic elements to the language specification but instead decided to express the semantics of Configurable EPCs in terms of traditional EPCs. Hence, we seek to validate the behavior of configurable processes through their *translation* to regular EPCs. Then, any of the formalization approaches mentioned in [9,11,12,13] can be used as a semantic foundation, and we may stop the discussion of semantics here. However, we later need to some semantic implications when translating Configurable EPCs into lawful process models.

2.2 On Configurable Reference Process Models: The C-EPC Notation

Current reference modeling languages lack configuration support. As an example, the SAP reference model [6], which is depicted in the EPC notation, covers in the version 4.6 more than 1,000 business processes and inter-organizational business scenarios. As the main objective of reference models is to streamline the design

of particular models, they are coined by the “Design by Reuse” paradigm. To increase their applicability, such models typically not include merely one proposed alternative for conducting business in a certain domain but a range of often mutually exclusive alternatives. Hence it denotes an ‘upperbound’ of process models that may possibly be implemented in a particular enterprise. As an organization might merely favor one of the depicted alternatives, they potentially only refer to a subset of ES functionality to be implemented and accordingly only to a subset of the reference model. Until today, however, these types of decision cannot be reflected within the ‘upperbound’ reference model due to lacking configuration support of the underlying reference modeling language. Existing reference modeling techniques neither support the *highlighting* nor *selection* of (process) configuration alternatives. This lack of expressiveness obviously denotes a major issue for reference model users.

Addressing these issues, this section introduces *Configurable EPCs* (C-EPCs) as an extension to the popular EPC modeling technique [5]. Focus was spent to the active parts of process models, i.e. functionality (functions, tasks, transitions, and the like) and control flow. We have not examined the configurability of events (or states) as more passive parts of processes since they cannot actively be influenced by an organization. It is the reaction to events that can be influenced and this reaction is covered in C-EPCs. The notion of a Configurable EPC has been introduced and formalized in [7], therefore we only discuss the basic notation here. Fig. 1 shows an example of a C-EPC model, with the left part showing the configuration alternatives, the middle part showing one selected alternative after configuration, and the right part showing a possible lawful EPC model resulting from the configuration.

In a C-EPC functions and connectors can be configured. Notation-wise, these configurable nodes are highlighted by bold lines. *Configurable functions* may be included (*ON*), excluded (*OFF*), or conditionally skipped (*OPT*). To be more specific, for configurable functions, a decision has to be made whether to perform this function in every process instance at run-time, whether to exclude this function permanently, i.e. it will not be executed in any process instance, or whether to defer this decision to run time, i.e. for each process instance it has to be decided whether or not to execute the function.

Configurable connectors subsume possible build-time connector types that are less or equally expressive. Hence, a configurable connector can only be configured to a connector type that restricts its behavior. A configurable OR-connector may be mapped to a regular OR-, XOR-, or AND-connector. Or, the configurable OR-connector may be mapped to a single sequence of events and functions (indicated by SEQ_n for some process path starting with node n). A configurable AND-connector may only be mapped to a regular AND-connector. A configurable XOR-connector may be mapped to a regular XOR-connector or to a single sequence SEQ_n .

In order to depict inter-dependencies between configurable EPC nodes, the concept of *configuration requirements* has been introduced. Inter-related configuration nodes may be constrained by such requirements. Consider the example

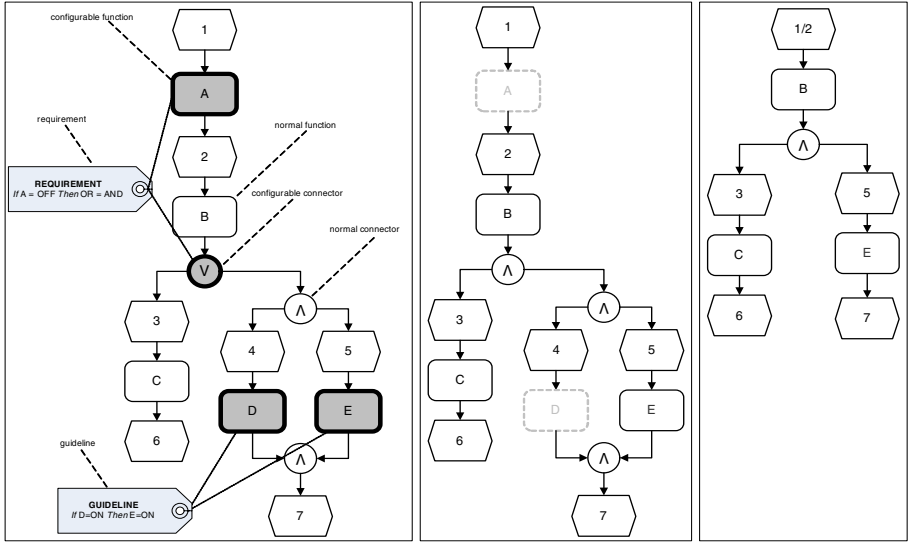


Fig. 1. A simple C-EPC (before configuration, after configuration, and resulting EPC)

given in the left part of Figure 1. If the configurable function *A* is excluded, the inter-related configurable OR-connector must be mapped to a regular AND-connector. Such configuration requirements are best defined via logical expressions in the form of *If-Then*-statements.

Additionally, *configuration guidelines* provide input in terms of recommendations and proposed best practices (also in the form of logical *If-Then*-expressions) in order to support the configuration process semantically. Consider again the example given in the left part of Figure 1. A recommendation could be that if function *D* is included, then so should be function *E* (but not necessarily vice versa). Summarizing, requirements and guidelines represent hard (*must*) respectively soft (*should*) constraints.

Concluding, we introduced a configurable reference modeling notation which potentially facilitates a model-driven selection and modification of process flows and process activities.

2.3 Related Work

Related work on configurable reference modeling includes the perspectives-based configurative reference process modeling approach by Becker et al. [14]. This approach focuses on adaptation mechanisms and proposes several mechanisms for automatically transforming a reference model into an individual model. While the work of Becker et al. focuses on generic adaptation mechanisms, this research pursues a reference model-driven approach towards ES configuration.

Soffer et al.’s suggestions on ERP modeling [15] can also be regarded as close to our proposed ideas. Following the concept of scenario-based requirements

engineering, they evaluate the Object-Process Modeling Methodology in order to determine a most appropriate ERP system representation language. The so-called argumentation facet, related to the ability of a modeling language to express optionality-related information, is just one of many of their criteria. Their work does not comprehensively analyze requirements related to modeling ERP configurability and focuses on technique evaluation rather than on the development of a more appropriate technique.

Gulla and Brasethvik [16] introduce three process modeling tiers to manage the complexity of process modeling in comprehensive ERP Systems projects. Their functional tier dimension deals with the functionality of the Enterprise System. However, they do not study how reference models fit into in this tier.

Based on this brief review we find that the notion of a C-EPC is the only dedicated Configurable Modeling approach that supports systems configuration aspects on a conceptual level. This paper extends our preceding work on the notation and formalization of C-EPCs [7] and on the process of Enterprise Systems configuration using C-EPCs [17] in the way that it considers more technical aspects of model configuration and translation.

3 On the Syntax of Reference Model Configuration

3.1 Configuration Using the C-EPC Modeling Language

The task of configuring reference models that have been deemed configurable by highlighting variation points in the model embraces both a semantic and a syntactic dimension. While the former is concerned with making business configuration decisions in order to match organizational strategy and requirements, the latter is concerned with maintaining syntactical correctness within the configured models to ensure a lawful translation to executable workflow specifications at run-time. We will show, that these dimensions are inter-related during configuration as syntactic considerations of implementing the models have semantic, i.e. business consequences and must hence be considered during configuration and translation.

We have described the semantic dimension of configuration in [17]. Basically, through the use of the C-EPC notation, process scenarios and process alternatives that are deemed desirable for a particular organization are selected. This is done by switching configurable nodes within a C-EPC model to a desired setting. Configuration requirements and configuration guidelines restrict respectively aid this task. The outcome of this phase is a C-EPC model where all configurable nodes have been switched to a certain setting. What, however, hasn't been ensured yet, is that these configured C-EPC models apply to the formal syntax of regular EPC. As an example, the middle part of Fig. 1 shows a configured version of the C-EPC model shown in the left part, where the configurable OR-connector has been switched to a regular AND-connector and where function *A* and *D* have been excluded (shaded grey). As can be seen, the resulting process model would be syntactically inconsistent: Consider function *A*: Assuming the control flow is reconnected where the excluded function is missing, two events

would follow each other. This is syntactically incorrect. Or, consider function D : Its exclusion leads to an “empty” branch. As this branch is subsequent to an AND-connector, it has to be removed because it does not make sense to do “something” (i.e. executing function E) while at the same time to do “nothing” (i.e. propagating a process folder without any transformation from event 4 to event 7).

Inadvertently, the step beyond semantic configuration of C-EPC models from a business perspective is the task of re-establishing syntactical correctness and consistency, i.e. the translation of configurable process model into lawful regular process specifications (as an example refer to the right part of Fig. 1).

3.2 Translating C-EPCs to EPCs: Syntactical and Semantic Problems

Now, in order to approach the syntactic and inherent semantic problems that arise due to the configuration of C-EPCs, we need to develop a translation approach that maps a configured C-EPC to a lawful regular EPC. As discussed above, this is a delicate task due to the semantic problems of EPCs themselves. There are in principle several options to approach this task:

- Refine the EPC specification to arrive at rigorously and unambiguously defined semantics for EPCs and thus, for C-EPCs.
- Ignore the semantics of EPCs and merely focus on specifying an unambiguous translation of C-EPCs to EPCs, which themselves may then be further discussed.

Here, we opted for the latter alternative: We wanted to *extend* the work on reference modeling techniques rather than developing new ones. Due to its popularity for the design of reference models and referring to the extensive academic work on its formalization and definition we deemed it better to take EPCs as both starting and ending point for our design of configurable process models instead of proposing yet another semantic and syntactic definition of EPCs.

Looking at the configuration of reference process models, this task can be divided into *global* and *local* decisions, with the former being based on the general model context and which can be made without studying the individual process model. Local decisions on the other hand require an explicit study of the relevant (parts of) process models. Our forthcoming discussion is focusing on the *local* aspects of configuration. We do not deem it necessary to explicitly address global decisions for the following reasons:

1. EPCs and thus C-EPCs can be hierarchically structured by decomposing single EPCs into more detailed sub-models. Analogously, each (C-) EPC may be generalized to a simpler model on a coarser level of detail. Hence, all contexts of configurable nodes may eventually be drilled up to the smallest possible local environment, as will be discussed below.
2. The notion of C-EPCs provides explicit representations for the depiction of inter-dependencies between configurable nodes. Hence, global dependencies

between processes depicted in separate process models may be expressed, thereby not needing an explicit addressing of a global process context.

3. As current practice shows (consider e.g. the configuration of the SAP system), the process of reference model configuration starts at a very coarse level of detail with industry sector-spanning process models (in the SAP context: collaborative business scenarios). At this stage, configuration refers to deleting dispensable processes from high-level process models. It can be seen as more of a scoping exercise in a pre-implementation stage. Hence, global configuration decisions merely are decisions as to the inclusion or exclusion of processes, the former of which then need to be locally configured.

Concluding, we argue that configured C-EPC models can be transferred into lawful EPC models in accordance to laws based on the *local syntactic environment* of configurable nodes. We must, for the purpose of this paper, limit some of the discussions to examples. A complete discussion of all local environments for configurable nodes and the entire resulting process model variants would require more space and is furthermore deemed unnecessary for making our argument.

Configurable Functions. Firstly, we investigate the local environments of configurable functions. As an EPC consists of events (E), functions (F), and splitting (S) respectively joining (J) connectors, there are nine different local environments for a configurable function A (see Fig. 2).

Studying the local environments of configurable functions reveals that, once a configurable function A has been switched to a desirable setting, the syntactical clean-up of the process model is not a purely technical decision. Due to missing formal semantics of the EPC notation - e.g. the EPC modeling language does not explicitly differ between triggering and resulting events that pre-/succeed a function - removals or inclusions of process model elements may have semantic and thus, business-related consequences. Bearing that in mind, syntactic validation may lead to various syntactically lawful yet semantically different process models.

Consider the following example. Referring to the local environment ‘Event-Function-Event, EFE’ - the configurable function A is embedded in the context of a preceding event E_P and a succeeding event E_S - configuration and syntactic validation may lead to the process model variants shown in Fig. 3. Now, as can be seen in Fig. 3, the syntactic handling of switching configurable functions ON or OFF are simple, according to the definitions in [7]. Functions mapped to OPT , however, are trickier.

Consider the configuration decision of switching the function A to OPT . The resulting process model must cater for a run time decision to either bypass the function or execute it. Due to the informal EPC semantics, it is not necessarily obvious whether the succeeding event E_S denotes a triggering state for a subsequent business function or a resulting state for A . In the former case, the bypass does not need to include E_S (variant 1). In the latter case, E_P needs not to be bypassed (variant 2). Maybe both states surrounding A may be bypassed, thereby passing a new state $E_{P/S}$ (variant 3). Another syntactically valid solution is to introduce a ‘dummy’ function $skipA$ which just propagates a process folder from

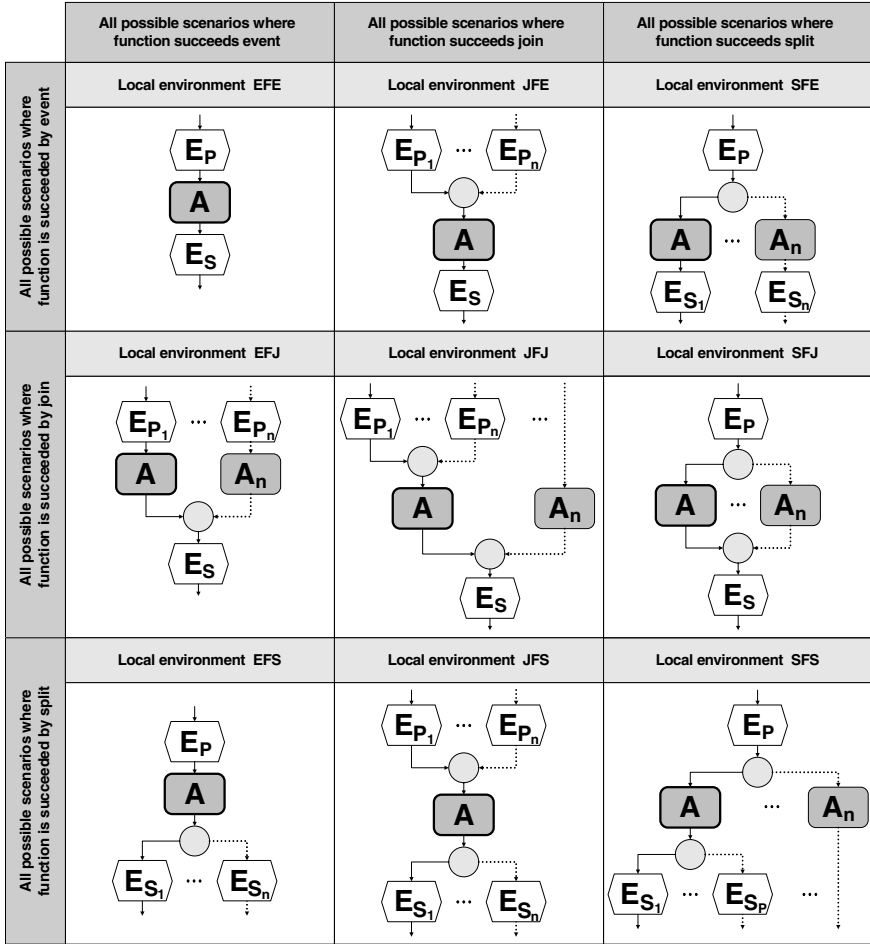


Fig. 2. Local environments for configurable functions

E_P to E_S without any transformation (variant 4). Or, a new decision function Z and an additional event E_x are introduced to augment the configuration decision of switching A to OPT (variant 5). This case, obviously, requires the inclusion of knowledge external to the model in order to specify the decision function Z .

Configurable Connectors. Considering configurable connectors and referring back to the configuration constraints described in Section 2.2, these nodes may appear in any of the local environments shown in Fig. 4.

According to the syntax rules of lawful EPCs, some local environments are restricted to the AND connector, since both OR- and XOR-connectors need to be linked to a preceding function that allows for the decision which process path to take. With respect to syntactically lawful process variants for these local environments, configurable connectors are relatively easy to handle, as shown in

figuration in whatever local environment there can only exist one corresponding syntactically lawful process variant.

Synopsis. The syntactic alternatives for all other local environments of configurable nodes, as depicted in Fig. 2 and Fig. 4 are constructed in a similar way. We examined the lawful environments of all configurable nodes and constructed syntactic alternatives for all combinations of predecessors and successors. As already mentioned, we cannot discuss them in detail here.

Yet, as can be shown through our examples, the syntactic clean-up of configured reference process models bears some semantic decisions in itself. The syntactical validation of C-EPC models may lead to several syntactically lawful yet semantically different EPC model variants. Since we decided not to modify the EPCs but instead base our work on the (arguably ambiguous) traditional EPC definition, it is sufficient to design adequate tool support that facilitates and aids the translation process from C-EPCs to EPCs. We will thus, in the next section, address this translation task by presenting a XML-based schema specification of C-EPCs that will be used to aid the syntax validation and translation of C-EPCs to regular lawful process models.

3.3 Towards Tool Support for Reference Model Configuration

Research towards tool support for C-EPCs based on an interchange format was motivated by two facts:

- The configuration of a C-EPC should correspond to a concrete EPC [7]. However, as we discussed in this paper, it is not possible to automate such mapping, hence adequate tool support is needed to facilitate and aid this task.
- EPCs and C-EPCs are not executable and thus cannot serve as direct specifications for process or workflow execution engines - which would, however, be desirable especially in the context of Enterprise Systems. In order to facilitate the interchange of configured reference process models to other process specifications, a standardized interchange format for “cutting-edge” process execution languages is needed.

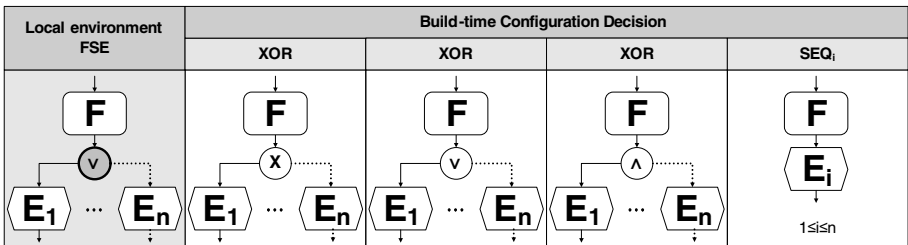


Fig. 5. Lawful alternatives for configuring an OR-connector in the FSE environment

Contemplating available options, we deemed a design specification based on a XML schema to be the best alternative. In particular, we opted for the *EPC Markup Language* (EPML) [8]. This selection was made for the following reasons:

1. EPML is able to perform syntax validations of EPCs [18].
2. EPML leverages the interchange of EPCs to other process modeling and execution languages [19].
3. EPML can be generated from the ARIS Markup Language (AML) [20] and is also supported by open source modeling solutions such as EPC Tools. Hence, tool platforms are available for implementing reference model configuration tool support based on C-EPCs.

Due to space limitations we cannot describe the EPML definition in any detail, which can be found in [8]. Instead, we merely introduce the main extensions to EPML to cater for the C-EPC specification (see Table 1).







As can be seen from Table 1, for each configurable node we introduce an EPML representation element. A configurable function is defined as an extension to a regular EPC **function** in EPML, merely annotating a new attribute element **configuration**, which is optional and may take a value of *on*, *off*, or *opt*. Configurable connectors are likewise specified as extensions to regular **connectors**, with the option of setting the attribute element configuration to a concrete value - in accordance to the definitions outlined in Section 2.2. Specifically, if for a configurable connector the value *seq* is selected, an attribute *goto* specifies the ID of the starting EPC node of the process path selected. Configuration requirements and guidelines, respectively, are defined as logical expressions involving a number of configurable nodes. In EPML they are thus defined as part of the root **epc** element, with a list containing the IDs of involved elements (*idRefs*). The logical expressions themselves can be modeled via XPath expressions, for instance

```
<configurationRequirement idRefs="2 4">
  <if xpath="function[@id='2']//configuration[@value='off']">
    <then xpath="function[@id='4'] / /configuration[@value='on']">
</configurationRequirement>
```

Note that this specification allows for a representation of C-EPCs both *before* configuration (such as the one depicted in the left part of Fig. 1), and *after* configuration (such as the one depicted in the middle part of Fig. 1). Also, as our definitions are mere extensions to the traditional EPC specification in EPML such that one the one hand traditional EPC models represented in EPML can still be validated against the extended EPML schema, and on the other hand EPML tools that are not aware of configuration aspects are still able to process C-EPCs as traditional EPCs by simply ignoring the additional configuration element information.

Now, based on these EPML specifications, reference model configuration tool support may be designed that facilitates the model-driven configuration and translation of C-EPCs. In particular, the EPML specifications will be used to:

Table 1. EPML representations for the C-EPC notation

C-EPC representation	EPML representation	C-EPC representation	EPML representation
	<pre><xs:element name="configurableFunction"> <xs:complexType> <xs:choice minOccurs="0"> <xs:element name="configuration"> <xs:complexType> <xs:attribute name="value" use="optional"/> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="on"/> <xs:enumeration value="off"/> <xs:enumeration value="opt"/> </xs:restriction> </xs:simpleType> </xs:attribute> </xs:complexType> </xs:choice> </xs:complexType> </xs:element></pre>		<pre><xs:element name="configurableConnector"> <xs:complexType> <xs:choice minOccurs="0"> <xs:element name="configuration"> <xs:complexType> <xs:attribute name="value" use="optional"/> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="or"/> <xs:enumeration value="and"/> <xs:enumeration value="xor"/> <xs:enumeration value="seq"/> </xs:restriction> </xs:simpleType> </xs:attribute> <xs:attribute name="goto" type="xs:integer"/> </xs:complexType> </xs:element> </xs:complexType> </xs:element></pre>
	<pre><xs:element name="configurableConnector"> <xs:complexType> <xs:choice minOccurs="0"> <xs:element name="configuration"> <xs:complexType> <xs:attribute name="value" use="optional"/> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="xor"/> <xs:enumeration value="seq"/> </xs:restriction> </xs:simpleType> </xs:attribute> <xs:attribute name="goto" type="xs:integer"/> </xs:complexType> </xs:element> </xs:choice> </xs:complexType> </xs:element></pre>		<pre><xs:element name="configurableConnector"> <xs:complexType> <xs:choice minOccurs="0"> <xs:element name="configuration"> <xs:complexType> <xs:attribute name="value" default="and" use="optional"/> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="and"/> </xs:restriction> </xs:simpleType> </xs:attribute> </xs:complexType> </xs:choice> </xs:complexType> </xs:element></pre>
	<pre><xs:element name="configurationRequirement"> <xs:complexType> <xs:sequence> <xs:element name="if"> <xs:complexType> <xs:attribute name="xpath" type="xs:string"/> </xs:complexType> </xs:element> <xs:element name="then" maxOccurs="unbounded"> <xs:complexType> <xs:attribute name="xpath" type="xs:string"/> </xs:complexType> </xs:element> </xs:sequence> <xs:attribute name="idRefs"> <xs:simpleType> <xs:list itemType="xs:integer"/> </xs:simpleType> </xs:attribute> </xs:complexType> </xs:element></pre>		<pre><xs:element name="configurationGuideline"> <xs:complexType> </xs:sequence> <xs:element name="if"> <xs:complexType> <xs:attribute name="xpath" type="xs:string"/> </xs:complexType> </xs:element> <xs:element name="then" maxOccurs="unbounded"> <xs:complexType> <xs:attribute name="xpath" type="xs:string"/> </xs:complexType> </xs:element> </xs:sequence> <xs:attribute name="idRefs"> <xs:simpleType> <xs:list itemType="xs:integer"/> </xs:simpleType> </xs:attribute> </xs:complexType> </xs:element></pre>

- leverage the modeling of C-EPCs via existing modeling tools, such as ARIS or the open source platform EPC Tools,
- design a XML schema-based tool for checking the validity of configurations,
- implement an EPML-based algorithm for translating C-EPCs to EPCs, and
- facilitate the interchange of C-EPCs to other process specifications.

4 Summary and Conclusions

This paper reported on syntactical and semantic challenges of reference model configuration, using the example of translating C-EPC models to lawful regular EPCs. We showed that both syntactical and semantic perspectives must be considered when mapping configurable nodes to desired regular EPC nodes. Resulting from these elaborations, we presented initial conceptual work towards adequate tool support for the configuration of process models. Based on our research, such tool support can be designed that embeds our recommendations and

thereby guides users when configuring Enterprise Systems based on configurable reference process models.

Our research has a few limitations. First, our conceptual approach needs to be empirically validated to prove its feasibility and applicability. However, we are currently undertaking this task and already conducted a laboratory experiment with postgraduate IT students on the perceived usefulness and perceived ease of use of C-EPCs in comparison to EPCs. Initial results show that C-EPCs are in fact perceived as more useful and easier to use for the task of reference model configuration [21]. Second, we focused the EPC notation and neglected the question of its executability. However, we selected the EPML interchange format as a basis for tool support for good reason, as it may facilitate the translations from (C-) EPCs to other executable process specifications.

Acknowledgements. The authors would like to express their gratitude towards the continuous fruitful contributions of Alexander Dreiling and Wasim Sadiq to the C-EPC research project and Markus Nüttgens to the EPML initiative. The research on the C-EPC has been funded by SAP Research and Queensland University of Technology with the Strategic Link with Industry project “Modelling Configurable Business Processes”. SAP is a trademark of SAP AG, Germany.

References

1. Scott, J.E., Vessey, I.: Managing Risks in Enterprise Systems Implementations. *Communications of the ACM* 45 (2002) 74–81
2. Sabherwal, R., Chan, Y.E.: Alignment Between Business and IS Strategies: A Study of Prospectors, Analyzers, and Defenders. *Information Systems Research* 12 (2001) 11–33
3. Davenport, T.H., Short, J.E.: The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review* 31 (1990) 11–27
4. Hammer, M., Champy, J.: *Reengineering the Corporation: A Manifesto for Business Revolution*. Harpercollins, New York (1993)
5. Keller, G., Nüttgens, M., Scheer, A.-W.: Semantische Prozessmodellierung auf der Grundlage ”Ereignisgesteuerter Prozessketten (EPK)”. Working Papers of the Institute of Information Systems, No. 89. University of Saarbrücken, Saarbrücken (1992)
6. Curran, T., Keller, G., Ladd, A.: *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Prentice Hall PTR, Upper Saddle River (1997)
7. Rosemann, M., van der Aalst, W.: A Configurable Reference Modelling Language. *Information Systems* (2005) In Press
8. Mendling, J., Nüttgens, M.: EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). *Information Systems and e-Business Management* (2005) In Press
9. van der Aalst, W.: Formalization and Verification of Event-driven Process Chains. *Information and Software Technology* 41 (1999) 639–650
10. van der Aalst, W., Desel, J., Kindler, E.: On the semantics of EPCs: A vicious circle. In: Nüttgens, M., Rump, F.J. (eds.): *Proceedings of the GI-Workshop EPK 2002. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, Trier (2002) 71–79

11. Langner, P., Schneider, C., Wehler, J.: Petri Net Based Certification of Event-Driven Proc-ess Chains. In: Desel, J., Silva, M. (eds.): Proceedings of the 19th International Conference on Application and Theory of Petri Nets. Lecture Notes in Computer Science, Vol. 1420. Springer, Lisbon (1998) 286–305
12. Dehnert, J., Rittgen, P.: Relaxed Soundness of Business Processes. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.): Proceedings of the 13th International Conference on Advanced Information Systems Engineering. Lecture Notes In Computer Science, Vol. 2068. Springer, Interlaken (2001) 151–170
13. Kindler, E.: On the semantics of EPCs: Resolving the vicious circle. *Data & Knowledge Engineering* (2005) In Press.
14. Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuropka, D.: Configurative Process Modeling - Outlining an Approach to increased Business Process Model Usability. In: Khosrow-Pour, M. (ed.): Proceedings of the 14th Information Resources Management Association International Conference. IRM Press, New Orleans (2004) 615–619
15. Soffer, P., Golany, B., Dori, D.: ERP modeling: a Comprehensive Approach. *Information Systems* 28 (2003) 673–690
16. Gulla, J.A., Brasethvik, T.: On the Challenges of Business Modeling in Large-Scale Reengineering Projects. In: Chen, P.P., Embley, D.W., Kouloumdjian, J., Liddle, S.W., Rod-dick, J.F. (eds.): Proceedings of the 4th International Conference on Requirements Engineering. IEEE, Schaumburg (2000) 17–26
17. Dreiling, A., Rosemann, M., van der Aalst, W., Sadiq, W., Khan, S.: Model-Driven Process Configuration of Enterprise Systems. In: Sinz, E.J., Ferstl, O.K. (eds.): Proceedings of the 7th International Tagung Wirtschaftsinformatik. Gesellschaft für Informatik, Bamberg (2005) 691–710
18. Mendling, J., Nüttgens, M.: EPC Syntax Validation with XML Schema Languages. In: Nüttgens, M., Rump, F.J. (eds.): Proceedings of the 2nd GI Workshop on Event-Driven Process Chains. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Bamberg (2003) 19–30
19. Mendling, J., Nüttgens, M.: Exchanging EPC Business Process Models with EPML. In: Nüttgens, M., Mendling, J. (eds.): Proceedings of the 1st GI Workshop XML4BPM. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Marburg (2004) 61–79
20. Mendling, J., Nüttgens, M.: Transformation of ARIS Markup Language to EPML. In: Nüttgens, M., Rump, F.J. (eds.): Proceedings of the 3rd GI Workshop on Event-Driven Process Chains. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Luxembourg (2004) 27–38
21. Recker, J., Rosemann, M., van der Aalst, W.: On the User Perception of Configurable Reference Process Models Initial Insights. In: Proceedings of the 16th Australasian Conference on Information Systems. Sydney (2005) forthcoming.

Configurable Process Models as a Basis for Reference Modeling

W.M.P. van der Aalst^{1,3}, A. Dreiling², F. Gottschalk¹,
M. Rosemann³, and M.H. Jansen-Vullers¹

¹ Department of Technology Management, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands

w.m.p.v.d.aalst@tm.tue.nl

² European Research Center for Information Systems, University of Münster,
Leonardo-Campus 3, 48149 Münster, Germany

³ Queensland University of Technology, 126 Margaret St,
Brisbane, QLD 4000, Australia

Abstract. Off-the-shelf packages such as SAP need to be configured to suit the requirements of an organization. Reference models support the configuration of these systems. Existing reference models use rather traditional languages. For example, the SAP reference model uses Event-driven Process Chains (EPCs). Unfortunately, traditional languages like EPCs do not capture the configuration-aspects well. Consider for example the concept of “choice” in the control-flow perspective. Although any process modeling language, including EPCs, offers a choice construct (e.g., the XOR connector in EPCs), a single construct will not be able to capture the time dimension, scope, and impact of a decision. Some decisions are taken at run-time for a single case while other decisions are taken at build-time impacting a whole organization and all current and future cases. This position paper discusses the need for *configurable process models* as a basic building block for reference modeling. The focus is on the control-flow perspective.

1 Introduction

The main objective of reference models is to streamline the design of particular models by providing a generic solution [19]. The application of reference models is motivated by the “Design by Reuse” paradigm. Reference models accelerate the modeling and configuration process by providing a repository of potentially relevant models. These models are ideally “plug and play” but often require some customization/configuration to be adjusted to individual requirements [7]. A configurable process model provides rules defining how a reference model can be adapted. Such a generating adaptation must be distinguished from non-generating adaptations as, e.g., aggregation, specialization or instantiation [5]. Unfortunately, the languages used for reference modeling [4,8,18] provide little or no support for configuration. The goal of this position paper is to discuss the need for *configurable process models*.

One of the most comprehensive models is the SAP reference model [8]. Its data model includes more than 4000 entity types and the reference process models cover more than 1000 business processes and inter-organizational business scenarios [19]. Most of the other dominant ERP vendors have similar or alternative approaches towards reference models. Foundational conceptual work for the SAP reference model has been conducted by SAP AG and the IDS Scheer AG in a collaborative research project in the years 1990-1992 [13]. The outcome of this project was the process modeling language Event-Driven Process Chains (EPCs) [13,14], which has been used for the design of the reference process models in SAP. EPCs also became the core modeling language in the Architecture of Integrated Information Systems (ARIS) [21,22]. It is now one of the most popular reference modeling languages and has also been used for the design of many SAP-independent reference models (e.g., the ARIS-based reference model for Siebel CRM or industry models for banking, retail, insurance, telecommunication, etc.). *Despite its success, the basic EPC model offers little support for process configuration.* It contains (X)OR connectors but it is unclear whether the corresponding decisions need to be taken at run-time (e.g., based on the stock-level), at build-time (e.g., based on the size of the organization using SAP), or somewhere in-between (e.g., based on the period of the year or resource availability). Therefore, we developed the so-called *Configurable EPCs* (C-EPCs) [19,9], a generic-monolithic approach for constructing re-usable models [10]. Indeed C-EPCs are extending the configuration opportunities of build-time operators [23,20,17]. However, they only provide a partial solution as they are only a representation variation, based on a specific language (EPCs), allowing the user to select or hide elements [5,6]. In this position paper we would like to trigger a discussion on requirements for configurable process models in a broader perspective.

The remainder of the paper is organized as follows. First, we elaborate on the concept of “choice” which is essential for configurable process models. Second, we approach the problem from a more theoretical viewpoint, i.e., we depict what the essence of configuration is. Finally, we briefly discuss Configurable EPCs as a first step towards such configurable models.

2 Configuration: It Is All About Making Choices

This paper focuses on configurable process models, i.e., we restrict ourselves to the control-flow perspective [12]. There are many languages to model processes ranging from formal (e.g., Petri nets and process algebras such as Pi calculus) to informal (flow charts, activity diagrams, EPCs, etc.). Each of these languages provides some *notion of choice* (e.g., two transitions sharing a single input place in a Petri net or an (X)OR-split connector in an EPC). Typically, it is not possible to describe the nature of such a choice. At best one can either specify a Boolean condition based on some data element (data-based decision) or one can specify events that have to occur for triggering paths (event-based decision) [16]. The usual interpretation is that a choice is made at run-time, based on such

a Boolean condition or based on occurring events. *In the context of reference models, this interpretation is too narrow.*

The *scope* of a decision can vary. For example, if a hospital uses a rule like “If a patient has high blood pressure a day before the planned operation, the operation will be canceled”, then the scope of each choice (operate or not) is limited to a single patient. There may also be choices which affect more cases, e.g., consider the rule “If there is a major disaster in the region, all planned operations will be canceled.” or also an entire process, e.g., “The admittance process requires patients to pre-register.”. There may even be choices that affect all processes in some organizations. The classical process modeling languages, e.g., the languages used in workflow management systems [2,12], allow only for one level of choices. Reference models have to allow for a broader spectrum of choices. Such choices are called configuration choices and are made at build-time. Configuration choices also affect choices at run-time. For example, at build-time one can choose not to use specific functionality offered by the system. Then no choice needs to be made at run-time anymore. But it may also be possible to use the functionality conditionally (e.g., depending on the workload). In this case the choice must be made at run-time. One can view configuration as *limiting choices by making choices*. Seen from this viewpoint, process modeling languages need to distinguish between run-time choices and configuration choices (i.e., at build-time). Note that the borderline between run-time choices and configuration choices may be a bit fuzzy as the following examples show.

- Based on the volume of the order, the goods are shipped by truck or mail.
- On Saturday, goods are shipped by truck.
- If stock is below 100 items, only preferred customers are serviced.
- The Dutch branches require a deposit, while this is not needed for branches in other countries.
- The organization chooses not to allow for pre-shipments.

Each of these choices is at another level. However, the processes in e.g. the SAP reference model show only one type of choice: the (X)OR-split connector. This triggered us to develop the so-called C-EPCs.

3 Configuration: A Theoretical Perspective

As described above a reference model provides a generic solution that needs to be configured for a specific situation. A generic-monolithic approach for model reuse should guide the user to a solution fitting to the individual requirements [10]. Therefore the reference model must be able to provide a complete, integrated set of all possible process configurations. This means the reference model is the least common multiple of all process variations, which leads to *inheritance of dynamic behavior* [1,3]. A reference model can be seen as a subclass of all concrete models. A concrete model itself is a superclass of the reference model. This may create confusion as the term “super” is intuitively connected to the bigger and at first existing reference model (e.g., in [24] traditional inheritance

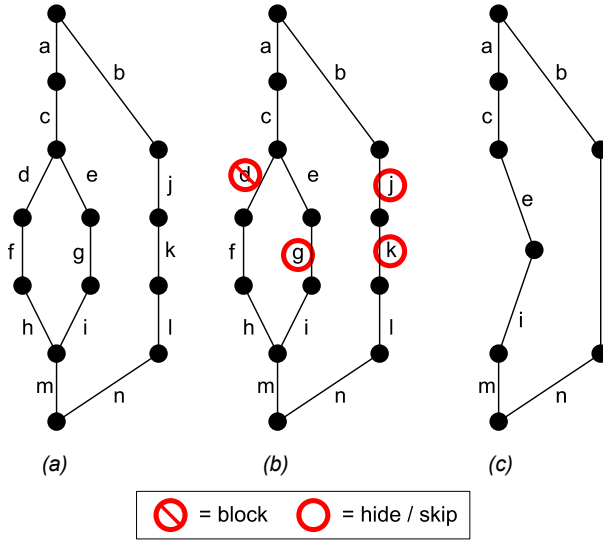


Fig. 1. Three labeled transition systems: (a) the initial model (e.g., the reference model), (b) a particular configuration hiding and blocking specific edges/labels, and (c) the resulting model

was altered to depict the reference model as superclass). However, it corresponds to the traditional notion of inheritance in which the subclass adds things to the superclass (e.g., additional methods or attributes). So configuration can be described as the reverse of inheritance. This allows us to use some of the ideas described in [1,3], in particular we use the idea of *hiding* and *blocking*.

Any process model having formal semantics can be mapped onto a labeled transition system. The nodes in a labeled transition system represent states, the directed edges represent transitions, and each transition has a label denoting some event, action or activity. Traditional choices in the process model, correspond to nodes in the labeled transition system with multiple output arcs. Consider Figure 1(a) showing a labeled transition system. In the initial state (the top node, edges go from top to bottom) there is a choice between *a* and *b*. If *a* is selected, the next step is *c* and then there is a choice between *d* and *e*, etc. If we consider Figure 1(a) to be a reference model, a configuration of this model should select the desired parts. This can be done by blocking and hiding edges or labels. In Figure 1(b) one edge is blocked and three edges are hidden. Hiding and blocking should be interpreted as in [1,3], i.e., hiding corresponds to *abstraction* and blocking corresponds to *encapsulation*. If an edge is blocked, it cannot be taken anymore. By hiding an edge the path is still possible but the associated label is no longer relevant, i.e., it is renamed to a silent step τ . One can think of the latter as simply skipping the edge. Figure 1(c) shows the resulting model after blocking and hiding the edges indicated in Figure 1(b).

A configurable process model should allow for the specification of which edges/labels can be blocked and hidden/skipped. An interesting question is

whether it should be possible to defer this decision to run-time. In the latter case, there would be two more options: *optional blocking* and *optional hiding* (to be decided at run-time).

4 Configuration: An Example of a Language

To conclude this position paper we introduce *Configurable EPCs* (C-EPCs) as an example for a configurable process modeling language. C-EPCs are an extension of the classical EPCs [13]. A classical EPC consists of functions (i.e., the activities), events and connectors. Functions follow events and events follow functions. Moreover, to model splits and joins in a process connectors may be used. There are three types of connectors: AND, OR and XOR. AND-splits and AND-joins may be used to model parallel routing. XOR-splits and XOR-joins may be used to model the selection of specific routes (e.g., an “if then else” construct). OR-splits and OR-joins may be used to model a mixture of conditional and parallel routing. (However, the semantics of the OR-join is still debated [14].)

In a C-EPC *both functions and connectors may be configurable*. Configurable functions may be included (ON), skipped (OFF) or conditionally skipped (OPT). Configurable connectors may be restricted at build-time, e.g., a configurable connector of type OR may be mapped onto an AND connector. Local configuration

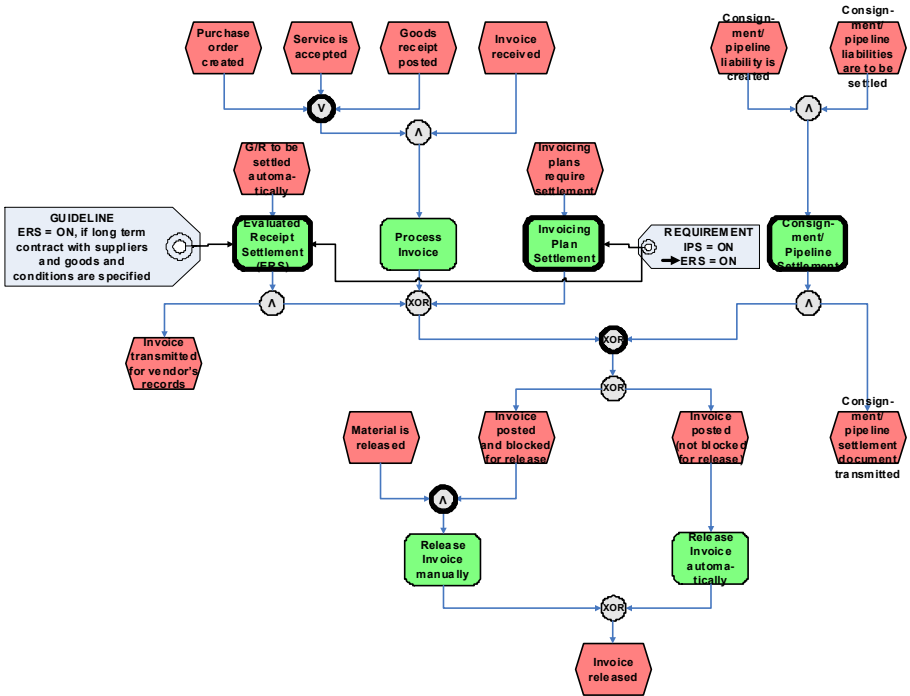


Fig. 2. A Configurable EPC

choices like skipping a function may be limited by configuration requirements. For example, if one configurable connector c of type OR is mapped onto an XOR connector, then another configurable function f needs to be included. This configuration requirement may be denoted by the logical expression; $c = OR \Rightarrow f = ON$. In addition to these requirements it is possible to add guidelines, supporting the configuration process.

Figure 2 shows a C-EPC describing an invoice verification process. The classical EPC is extended with configurable functions and connectors (indicated using thick lines). For example function *Invoicing Plan Settlement* is configurable, i.e., it may be included (ON), skipped (OFF) or conditionally skipped (OPT). The diagram shows also some configurable connectors. In this position paper we do not further elaborate on C-EPCs. For more information, we refer to [19,9]. The important thing to note is that it is possible to extend a language like EPCs with configurable elements. Moreover, there are two types of choices: (1) configuration choices made at build-time and (2) “normal” choices made at run-time.

C-EPCs can be seen as a rather naive, but very intuitive, configuration language that allows (optionally) blocking and hiding of edges/labels at build-time for specifying the configuration of the model. Using the theory developed in [1,3] and basic notions such as simulation, bisimulation, and branching bisimulation [11,15] on the one hand and practical experiences using C-EPCs on the other hand, we hope to develop more mature configuration languages.

The aim of this position paper is to trigger a discussion on configurable process models. To do this we argued that configuration is strongly related to the timing and scope of choices. We also showed an example of a language (C-EPCs). However, to allow for a more language-independent discussion we also tried to capture the essence of configuration in terms of (optional) hiding and blocking of edges or labels.

References

1. W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.
2. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
3. T. Basten and W.M.P. van der Aalst. Inheritance of Behavior. *Journal of Logic and Algebraic Programming*, 47(2):47–145, 2001.
4. J. Becker, M. Kugeler, and M. Rosemann, editors. *Process Management: A Guide for the Design of Business Processes*. Springer-Verlag, Berlin, 2003.
5. J. Becker, P. Delfmann, R. Knackstedt. Konstruktion von Referenzmodellierungssprachen: Ein Ordnungsrahmen zur Spezifikation von Adaptionsmechanismen für Informationsmodelle. In *WIRTSCHAFTSINFORMATIK*, 46(2004)4, pages 251–264.
6. J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, D. Kuropka. Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In *Proceedings of the 15th Information Resources Management Association International Conference*. New Orleans, 2004.

7. P. Bernus. *Generalised Enterprise Reference Architecture and Methodology, Version 1.6.3*. IFIP/IFAC Task Force on Architectures for Enterprise Integration, 1999.
8. T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
9. A. Dreiling, M. Rosemann, W.M.P. van der Aalst, W. Sadiq, and S. Khan. *Model-driven process configuration of enterprise systems*. In O.K. Ferstl, E.J. Sinz, S. Eckert, and T. Isselhorst, editors, *Wirtschaftsinformatik 2005. eEconomy, eGovernment, eSociety*, pages 687–706, Physica-Verlag, Heidelberg, 2005.
10. P. Fettke and P. Loos. *Methoden zur Wiederverwendung von Referenzmodellen – Übersicht und Taxonomie*. In J. Becker, R. Knackstedt, editors, *Referenzmodellierung 2002: Methoden – Modelle – Erfahrungen*, Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 90 (in German), pages 9–33. University of Münster, Münster, 2002.
11. R.J. van Glabbeek and W.P. Weijland. *Branching Time and Abstraction in Bisimulation Semantics*. In *Journal of the ACM*, 43(3):555–600, 1996.
12. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
13. G. Keller, M. Nüttgens, and A.W. Scheer. *Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK)*. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.
14. E. Kindler. *On the Semantics of EPCs: A Framework for Resolving the Vicious Circle*. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97. Springer-Verlag, Berlin, 2004.
15. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1980.
16. M. Owen and J. Raj. *BPMN and Business Process Management – Introduction to the New Business Process Modeling Standard*, Popkin Software, 2003.
17. M. Rosemann. *Komplexitätsmanagement in Prozessmodellen: methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung (in German)*. Gabler, Wiesbaden, 1996.
18. M. Rosemann. *Application Reference Models and Building Blocks for Management and Control (ERP Systems)*. In P. Bernus, L. Nemes, and G. Schmidt, editors, *Handbook on Enterprise Architecture*, pages 596–616. Springer-Verlag, Berlin, 2003.
19. M. Rosemann and W.M.P. van der Aalst. *A Configurable Reference Modelling Language*. In *Information Systems* (to appear, also available from BPMCenter.org), 2005.
20. M. Rosemann and R. Schütte. *Grundsätze ordnungsmäßiger Referenzmodellierung*. In J. Becker, M. Rosemann, R. Schütte, editors, *Entwicklungsstand und Perspektiven der Referenzmodellierung*, Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 52 (in German), pages 16–33. University of Münster, Münster, 1997.
21. A.W. Scheer. *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer-Verlag, Berlin, 1994.
22. A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.
23. R. Schütte. *Grundsätze ordnungsmäßiger Referenzmodellierung – Konstruktion konfigurations- und anpassungsorientierter Modelle (in German)*. Gabler, Wiesbaden, 1998.
24. A. Schwegmann. *Objektorientierte Referenzmodellierung: theoretische Grundlagen und praktische Anwendung (in German)*. Gabler, Wiesbaden, 1999.

Author Index

- Aberer, Karl 18
Albani, Antonia 305
Albert, Patrick 102
Altenhofen, Michael 116
Alves de Medeiros, A.K. 203
Andersson, Birger 257
Ardagna, Danilo 32
- Barros, Alistair 61
Basanta, Pablo 329
Berio, Giuseppe 245
Börger, Egon 116
Bosin, Andrea 281
Boufaïda, Mahmoud 269
Boukhelfa, Kamel 269
Braun, Peter 89
Bruno, Giorgio 75
- Campos, Cristina 353
Chabrol, Michelle 375
Chalmeta, Ricardo 233, 353
Chauvet, Julie 375
Chen, Pin 293
Cimpian, Emilia 130
Coltell, Óscar 353
- Dadam, Peter 177
Delgado-Kloos, Carlos 329
Dessi, Nicoletta 281
Dreiling, A. 512
Dubois, Eric 342
Dumas, Marlon 61
Dustdar, Schahram 216
- Estevez-Ayres, Iria 329
- Farahbod, Roozbeh 144
Féniès, Pierre 375
Fettke, Peter 469
Franczyk, Bogdan 89
Fugini, Maria Grazia 281
- Garcia-Valls, Marisol 329
Glässer, Uwe 144
Godart, Claude 363
- Goldkuhl, Göran 408
Gombotz, Robert 216
Gottschalk, F. 512
Gourgand, Michel 375
Grangel, Reyes 233, 353
- Hauswirth, Manfred 18
Henocque, Laurent 102
Ho, Danny Ting-Yi 454
- Incoul, Christophe 342
Jansen-Vullers, M.H. 512
- Kleiner, Mathias 102
Kloeckner, Sebastian 305
Komoda, Norihisa 385
Kowalczyk, Ryszard 89
Kutvonen, Lea 317
- La Rosa, Marcello 75
Lee, Jintae 432
Lemcke, Jens 116
Liberati, Diego 281
Limam Mansar, Selma 421
Lind, Mikael 408
Longo, Antonella 399
Loos, Peter 469
Ludwig, André 89
Ly, Linh Thao 177
- Marjanovic, Zoran 390
Matsuyama, Norikazu 385
Mendling, Jan 497
Mocan, Adrian 130
Motta, Gianmario 399
- Nakayama, Kojiro 385
Netjes, Mariska 444
- Oaks, Phillipa 3, 61
Oba, Michiko 385
Ounnar, Fouzia 421
- Panetto, Hervé 257
Peña, Iñaki 233

- Pernici, Barbara 32
Perrin, Olivier 363
Pes, Barbara 281
- Recker, Jan 497
Reichert, Manfred 177, 191
Reijers, Hajo A. 421, 444
Rinderle, Stefanie 177, 191
Rosemann, Michael 497, 512
Rozinat, A. 163
Ruokolainen, Toni 317
- Schmitt, Michael 342
Schuster, Stefan 233
Selk, Bernhard 305
- ter Hofstede, Arthur H.M. 3
Thomas, Oliver 484
- Vajihollahi, Mona 144
van der Aalst, W.M.P. 163, 203, 497, 512
Vanderfeesten, Irene 444
Vu, Le-Hung 18
Vujasinovic, Marko 390
- Weber, Barbara 191
Weijters, A.J.M.M. 203
Wild, Werner 191
Wohed, Petia 257
Wyner, George M. 432
- Yildiz, Ustun 363
Yu, JianJun 47
- Zhou, Gang 47
zur Muehlen, Michael 454
Zwicker, Jörg 469