

Hugo Fuks
Stephan Lukosch
Ana Carolina Salgado (Eds.)

LNCS 3706

Groupware: Design, Implementation, and Use

11th International Workshop, CRIWG 2005
Porto de Galinhas, Brazil, September 2005
Proceedings

 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

Hugo Fuks Stephan Lukosch
Ana Carolina Salgado (Eds.)

Groupware: Design, Implementation, and Use

11th International Workshop, CRIWG 2005
Porto de Galinhas, Brazil, September 25-29, 2005
Proceedings



Springer

Volume Editors

Hugo Fuks

Catholic University of Rio de Janeiro, Software Engineering Laboratory

R. Marquês de São Vicente, 225, 22453-900 Rio de Janeiro, Brazil

E-mail: hugo@inf.puc-rio.br

Stephan Lukosch

FernUniversität in Hagen, Computer Science Department

Universitätsstr. 1, 58084 Hagen, Germany

E-mail: stephan.lukosch@fernuni-hagen.de

Ana Carolina Salgado

Federal University of Pernambuco, Center for Informatics

Av. Prof. Luiz Freire S/N, Cidade Universitária, 50740-540 Recife-PE, Brazil

E-mail: acs@cin.ufpe.br

Library of Congress Control Number: 2005932312

CR Subject Classification (1998): H.5.2, H.5.3, H.5, K.3.1, K.4.3, C.2.4

ISSN 0302-9743

ISBN-10 3-540-29110-5 Springer Berlin Heidelberg New York

ISBN-13 978-3-540-29110-7 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

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper SPIN: 11560296 06/3142 5 4 3 2 1 0

Preface

This volume constitutes the proceedings of the 11th International Workshop on Groupware (CRIWG 2005). The conference was held in Porto de Galinhas (Recife), Brazil. The previous ten CRIWG workshops were organized in Lisbon, Portugal (1995), Puerto Varas, Chile (1996), El Escorial, Spain (1997), Buzios, Brazil (1998), Cancun, Mexico (1999), Madeira, Portugal (2000), Darmstadt, Germany (2001), La Serena, Chile (2002), Autrans, France (2003), and San Carlos, Costa Rica (2004). CRIWG workshops follow a simple recipe for success: good papers, a relatively small number of attendees, extensive time for lively and constructive discussions, and a high level of cooperation both within and between paper sessions. CRIWG 2005 continued this tradition.

This 11th CRIWG exemplified the continuing interest in the groupware research area. Groupware researchers from 16 different countries submitted a total of 67 papers. Each of the 67 papers was reviewed by at least three members of an internationally renowned Program Committee, using a double-blind reviewing process. Based on the reviewers' recommendations 29 papers were finally accepted: 16 long papers presenting mature work, and 13 short papers describing work in progress. The accepted papers were grouped into 8 themes that represent current areas of interest in groupware research: groupware development, collaborative applications, workflow management, knowledge management, computer-supported collaborative learning, group decision support systems, mobile collaborative work, and work modeling in CSCW. In addition, we were pleased to have Gerry Stahl from Drexel University in Philadelphia, USA, a renowned specialist in CSCL, as keynote speaker.

CRIWG 2005 would not have been possible without the work and support of a great number of people. First of all we thank all members of the Program Committee for their valuable reviews of the papers. We are grateful for the advice and support provided by the CRIWG Steering Committee. We extend a special acknowledgment to our sponsoring organizations: CIN/UFPE (Centro de Informática da Universidade Federal de Pernambuco), FACEPE (Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco), and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), all in Brazil.

Last, but certainly not least, we thank the attendees for their interest in CRIWG 2005, and hope they had an enriching experience at the conference.

September 2005

Hugo Fuks
Stephan Lukosch
Ana Carolina Salgado

Conference Organization

Program Committee Chairs

Hugo Fuks, Catholic University of Rio de Janeiro, Brazil
Stephan Lukosch, FernUniversität in Hagen, Germany

Program Committee

Pedro Antunes, Universidade de Lisboa, Portugal
Jaco Appelman, Delft University of Technology, The Netherlands
Nelson Baloian, Universidad de Chile, Chile
Marcos Borges, Federal University of Rio de Janeiro, Brazil
Patrick Brézillon, Université Paris 6, France
César A. Collazos, Systems Dept., Universidad del Cauca, Colombia
Bertrand David, Ecole Centrale de Lyon, France
Gert-Jan de Vreede, University of Nebraska at Omaha, USA
Dominique Decouchant, LSR-IMAG, Grenoble, France
Yannis Dimitriadis, University of Valladolid, Spain
Henrique João L. Domingos, Universidade Nova de Lisboa, Portugal
Thomas Erickson, IBM T.J. Watson Research Center, USA
Cléver Farias, Catholic University of Santos, Brazil
Jesus Favela, CICESE, Mexico
Christine Ferraris, Université de Savoie, France
Werner Geyer, IBM T.J. Watson Research, Cambridge, USA
Luis A. Guerrero, Universidad de Chile, Chile
Jörg M. Haake, FernUniversität in Hagen, Germany
Andreas Harrer, University of Duisburg-Essen, Germany
H. Ulrich Hoppe, University of Duisburg-Essen, Germany
Sten Ludvigsen, University of Oslo, Norway
Gloria Mark, University of California at Irvine, USA
Alberto L. Moran, Facultad de Ciencias — UABC, Mexico
Jose A. Pino, Universidad de Chile, Chile
Jean-Charles Pomerol, Université Pierre et Marie Curie, Paris, France
Nuno Preguiça, Universidade Nova de Lisboa, Portugal
Alberto Raposo, Catholic University of Rio de Janeiro, Brazil
Nicolas Roussel, Université Paris-Sud, France
Flavia Maria Santoro, UNIRIO, Brazil
Till Schümmer, FernUniversität in Hagen, Germany
Carla Simone, University of Milan, Italy
Robert Slagter, Telematica Instituut, The Netherlands

VIII Organization

José Valdeni de Lima, Universidade Federal do Rio Grande do Sul, Brazil
Aurora Vizcaíno Barceló, Universidad de Castilla-La Mancha, Spain
Jürgen Vogel, European Media Laboratory (EML) GmbH, Germany
Jacques Wainer, State University of Campinas, Brazil
Martin Wessner, Fraunhofer IPSI, Germany
Volker Wulf, Fraunhofer FIT, Germany

Doctoral Colloquium Chair

Gert-Jan de Vreede, University of Nebraska at Omaha, USA

Organization Committee Chair

Ana Carolina Salgado, Federal University of Pernambuco, Brazil

Organization Committee

Patricia Tedesco, Federal University of Pernambuco, Brazil
Carlos Ferraz, Federal University of Pernambuco, Brazil
Nelson Rosa, Federal University of Pernambuco, Brazil
Vaninha Vieira, Federal University of Pernambuco, Brazil

Sponsoring Institutions

Centro de Informática da Universidade Federal de Pernambuco, Brazil
Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco, Brazil
Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brazil

Table of Contents

Opening Keynote

Groups, Group Cognition and Groupware <i>Gerry Stahl</i>	1
---	---

Groupware Development

A Framework for Prototyping Collaborative Virtual Environments <i>Clinton Jeffery, Akshay Dabholkar, Kosta Tachtevrenidis, Yosep Kim</i>	17
---	----

Adaptive Distribution Support for Co-authored Documents on the Web <i>Sonia Mendoza, Dominique Decouchant, Alberto L. Morán, Ana María Martínez Enríquez, Jesus Favela</i>	33
---	----

Agilo: A Highly Flexible Groupware Framework <i>Axel Guicking, Peter Tandler, Paris Avgeriou</i>	49
---	----

Autonomous and Self-sufficient Groups: Ad Hoc Collaborative Environments <i>Joan Manuel Marquès, Leandro Navarro</i>	57
--	----

Empowering End-Users: A Pattern-Centered Groupware Development Process <i>Till Schümmer, Stephan Lukosch, Robert Slagter</i>	73
--	----

Integrating Synchronous and Asynchronous Interactions in Groupware Applications <i>Nuno Preguiça, J. Legatheaux Martins, Henrique Domingos, Sérgio Duarte</i>	89
---	----

Collaborative Applications

An Architectural Model for Component Groupware <i>Cléver R.G. de Farias, Carlos E. Gonçalves, Marta C. Rosatelli, Luís Ferreira Pires, Marten van Sinderen</i>	105
---	-----

An Architecture for Collaborative Geomodeling <i>Luciano P. Reis, Alberto B. Raposo, Jean-Claude Paul, Fabien Bosquet</i>	121
--	-----

Remote Control Point Motion Prediction in Internet-Based Real-Time Collaborative Graphics Editing Systems
Bo Jiang, Jiajun Bu, Chun Chen, Jianxu Yang 137

Synchronization Contexts as a Means to Support Collaborative Modeling
Niels Pinkwart 145

Tailoring Infrastructures: Supporting Cooperative Work with Configurable Email Filters
Volkmar Pipek, Markus Won, Roman Englert, Volker Wulf 153

Workflow Management

A Collaborative Framework for Unexpected Exception Handling
Hernâni Mourão, Pedro Antunes 168

A Workflow Mining Method Through Model Rewriting
Jacques Wainer, Kwanghoon Kim, Clarence A. Ellis 184

Design of an Object-Oriented Workflow Management System with Reusable and Fine-Grained Components
Gwan-Hwan Hwang, Yung-Chuan Lee, Sheng-Ho Chang 192

Modeling the Behavior of Dispatching Rules in Workflow Systems: A Statistical Approach
Gregório Baggio Tramontina, Jacques Wainer 208

Knowledge Management

Collective Knowledge Recall: Benefits and Drawbacks
Naiana Carminatti, Marcos R.S. Borges, José Orlando Gomes 216

Developing Shared Context Within Group Stories
Flávia Maria Santoro, Patrick Brézillon 232

Patterns of Collaboration and Non-collaboration Among Physicians
Claudia Barsotini, Jacques Wainer 248

Shared Knowledge: The Result of Negotiation in Non-hierarchical Environments
Oriel Herrera, David A. Fuller 255

Computer Supported Collaborative Learning

A Mediation Model for Large Group Collaborative Teaching <i>María Ester Lagos, Miguel Nussbaum, Francisca Capponi</i>	263
Analyzing the Organization of Collaborative Math Problem-Solving in Online Chats Using Statistics and Conversation Analysis <i>Alan Zemel, Fatos Xhafa, Gerry Stahl</i>	271
Collaboration for Learning Language Skills <i>Luis A. Guerrero, Milko Madariaga, Cesar Collazos, José A. Pino, Sergio Ochoa</i>	284

Group Decision Support Systems

Collaborative IS Decision-Making: Analyzing Decision Process Characteristics and Technology Support <i>Bjørn Erik Munkvold, Kristin Eim, Øyvind Husby</i>	292
Software Requirements Negotiation Using the Software Quality Function Deployment <i>João Ramires, Pedro Antunes, Ana Respício</i>	308
The Design and Field Evaluation of a Repeatable Collaborative Software Code Inspection Process <i>Pushpa G. Koneri, Gert-Jan de Vreede, Douglas L. Dean, Ann L. Fruhling, Peter Wolcott</i>	325

Mobile Collaborative Work

Handheld-Based Electronic Meeting Support <i>Gustavo Zurita, Nelson Baloian</i>	341
Sharing Information Resources in Mobile Ad-hoc Networks <i>Andrés Neyem, Sergio F. Ochoa, José A. Pino, Luis A. Guerrero</i>	351

Work Modeling in CSCW

Towards a Model of Cooperation <i>Adriana S. Vivacqua, Jean-Paul Barthès, Jano Moreira de Souza</i>	359
--	-----

Towards an Ontology for Context Representation in Groupware
Vaninha Vieira, Patrícia Tedesco, Ana Carolina Salgado 367

Author Index 377

Groups, Group Cognition and Groupware

Gerry Stahl

Drexel University, Philadelphia, USA

Gerry.Stahl@drexel.edu

<http://www.cis.drexel.edu/faculty/gerry>

Abstract. More than we realize it, knowledge is often constructed through interactions among people in small groups. The Internet, by allowing people to communicate globally in limitless combinations, has opened enormous opportunities for the creation of knowledge and understanding. A major barrier today is the poverty of adequate groupware. To design more powerful software that can facilitate the building of collaborative knowledge, we need to better understand the nature of group cognition—the processes whereby ideas are developed by small groups. We need to analyze interaction at both the individual and the group unit of analysis in order to understand the variety of processes that groupware should be supporting. This paper will look closely at an empirical example of knowledge being constructed by a small group and suggest implications for groupware design.

1 Individual Learning in Groups

Groupware is software that is specifically designed to support the work of groups.

Most software in the past, in contrast, has been designed to support the work of individuals. The most popular applications—such as word processors, Internet browsers and spreadsheets—are structured for use by one individual at a time. Software for communication among people—like an email program—assumes a model of communication as transmission of messages from one person to other individuals. Building on these examples, one could design groupware to support groups conceived of as sets of individuals. Such software would allow individuals to express their mental ideas, transmit these expressions to other people, receive expressions transmitted from other people and make sense of received messages as expressions of the ideas in the heads of the other people [as in 1]. Possibilities for improving these designs might be conceived in terms of “increasing the bandwidth” of the transmissions, possibly taking face-to-face communication as the “gold standard” of communication with a wide bandwidth of many channels (words, intonation, gaze, facial expression, gesture, body language).

Until recently, most research about groups has focused on the individual people in the group as the cognitive agents. For instance, research on cooperative learning in the 1970s [still in 2], assumed that knowledge resided in the individuals, and that group interaction was most useful as a way of transferring knowledge from one individual to another or as a way of motivating individuals to perform better. Educational research

on groups typically measured learning in terms of individual test outcomes and tried to study what is going on in the minds of the individuals through surveys, interviews and talk-aloud protocols. Similarly, research in social psychology about small groups conceptualized the groups as sets of rationally calculating individuals seeking to maximize their own advantages. This broad tradition looks to the individual as the unit of analysis, both to understand what takes place in group behavior and to measure quantitative learning or knowledge-building outcomes.

In the 1990s, the individualistic approach was thoroughly critiqued by theories of situated cognition [3], distributed cognition [4], socio-cultural activity theory [5] and ethnomethodology [6], building on the philosophies of phenomenology [7], mediation [8] and dialog [9]. These new approaches rejected the view that cognition or the construction of knowledge took place exclusively in the isolated minds of individuals, and showed how it emerged from concrete situations and interpersonal interactions. One consequence that could be drawn from this would be to analyze cognition at the small-group unit of analysis, as in many cases a product of social interaction within the context of culturally-defined rules or habits of behavior.

An alternative approach to designing groupware based on a group conception of cognition would provide functionality to support the working of a group as an organic whole, rather than just supporting the group members as individuals and treating the group as the sum of its parts. In the past, a number of researchers have tried to develop groupware that supports the functioning of the group itself, such as the formation of groups [10], intertwining of perspectives [11] and negotiation of group decisions [12; 13].

Here I would like to further develop the approach focused on the group that I presented in *Group Cognition* [14] and that is being investigated in the Virtual Math Teams (VMT) project at the Math Forum at Drexel University. In part I of the book, I present my own attempts to design software to support small-group interactions (building, of course, on previous work by others), and conclude that we need to better understand how groups work before we can effectively design groupware. In part II of the book, I then discuss how to analyze the methods that are used in groups to construct meaning and knowledge. Then I develop a concept of group cognition in part III to talk about what takes place at the group unit of analysis.

In this paper, I report on our preliminary analysis in VMT of a group of students working on a set of math problems in an online chat room. We are interested in seeing how they work together using a minimal system of computer support in order to see what forms of interaction might be supported by groupware with special functionality designed to increase the effectiveness of the collaboration.

In order to capture both the individual and the group contributions to discourse and to compare their results, we recently arranged an experiment with a combination of individual and group work. It consists of an individual phase where the knowledge of the individuals can be objectively assessed, followed by a group phase in which the references and proposals can be analyzed at both the individual and the group units of analysis. By seeing what the individuals knew before they participated in the group phase, it should be possible to see what the group interaction added.

In previous work at VMT, we have characterized two different general patterns of chat discourse: *expository narrative* and *exploratory inquiry* [15]. These are two common methods of conducting online discourse that embody different relationships

of the group to its individual members. We view online chat as a form of text-based interaction, where short texts respond to each other [16]. We analyze the chat discourse with a variation of conversation analysis—a scientific methodology based on ethnomethodological principles for analyzing everyday verbal conversation. In the VMT project, we have begun to adapt conversation analysis to chat by taking into account the consequences introduced by the textual medium, the math content, the physical separation and other differences from everyday conversation.

Expository narrative involves one person dominating the interchange by contributing more and longer texts [17]. Basically, the normal turn-taking procedures in which members take roughly equal and alternating turns is transformed in order to let one person narrate an extended story or explanation. For instance, if a student has already solved a math problem that the group is working on, that student might propose their solution or indicate that they have a solution and the others might request an explanation of the proposed solution. There would still be some forms of interaction, with members of an audience asking questions, encouraging continuation, indicating understanding, raising questions, etc. But in general, the proposer would be allowed to provide most of the discourse. In conversation, this kind of pattern is typical where one member narrates a story or talks in detail about some events or opinions [18]. Exposition in math has its own characteristics, such as providing mathematical warrants for claims, calculating values, addressing issues of formal logic, etc. But it follows a turn-taking profile similar to that of conversational narrative.

Exploratory inquiry has a different structure. Here, the group members work together to explore a topic. Their texts contribute from different perspectives to construct some insight, knowledge, position or solution that cannot be attributed to any one source but that emerges from the “inter-animation of perspectives” [9; 19]. Exploratory inquiries tend to take on the appearance of group cognition. They contrast with expository narratives in a way that is analogous to the broad distinction between *collaboration* and *cooperation* [20]. Collaboration involves a group of people working on something together, whereas cooperation involves people dividing the work up, each working by themselves on their own part and then joining their partial solutions together for the group solution. Expository narratives tend to take on the appearance of cooperation, where individuals contribute their own solutions and narrate an account of how they arrived at them. In a rough way, then, exploratory and expository forms of discourse seem to reflect group versus individual approaches to constructing shared knowledge.

I will now analyze our experiment involving a group of college students in an online chat discussing a series of math problems. I will try to tease apart the individual and the group contributions to meaning making, knowledge building and problem solving. We conducted the experiment using a set of well-defined math problems for which it is clear when an individual or a group arrives at the correct answer. We gave the individuals an opportunity to solve the problems on their own with pencil and paper. We then had them enter an online chat room and decide as a group on the correct answers. By collecting the individual papers and logging the chat, we obtained data about the individual and the group knowledge, which we can objectively evaluate and compare.

The students were given 11 problems on two sheets of paper with room to show their work and to give their answers. The problems were a variety of algebra and geometry problems, some stated as word problems. Most required some insight. They came from the Scholastic Aptitude Tests (SAT), which are taken by high school students in order to apply to colleges in the United States. They are primarily multiple choice questions with five possible answers, only one of which is correct.¹

For the individual phase of the experiment, the students had 15 minutes to complete the problems working silently with paper and pencil. Most students stopped work before the time was up. Their papers were collected and new sheets of paper with the same questions were distributed. The students were then instructed to work in randomly-assigned groups and solve the same problems online. They worked together in chat rooms for 39 minutes.

In this paper, I analyze the results of one group of five students who worked together in one chat room group. None of the students in this group did impressively well on the test as an individual. They each got 2 or 3 question right out of the 11 (see table 1) for a score of 18% or 27%.

Table 1. Problems answered correctly by individuals and the group

	1	2	3	4	5	6	7	8	9	10	11	Score
Hal		X	X					X				27%
Dan			X	X								18%
Cosi			X				X		X			27%
Mic					X		X					18%
Ben			X					X				18%
Group		X	X	X	X		X	X	X	X	X	82%

For the experiment's group phase, the students worked in a chat room using Blackboard's group chat facility without a shared whiteboard. The software is simple and familiar to the students. The students did not know each other and did not have any information about each other except for the login names. They had not worked together before and had not participated in a chat like this before. The result of the group work was that the group decided upon the correct answers to 9 of the 11 problems, for a group score of 82%. Thus, the group did considerably better than any of the individual students.

However, it seems that each of the correct group answers can be attributed to one of the students. Although each student got only 2 or 3 answers right, together at least one of them correctly answered questions 2, 3, 4, 5, 7, 8, 9. No one understood question 1, and the group did not get this answer either. Question 2 was correctly answered by Hal, who persuaded the group. Question 3 was correctly answered by everyone except Mic. Question 4 was correctly answered by Dan. Question 5 gave the group a lot of frustration because no one could figure it out (although Mic had gotten

¹ The 11 questions and the complete chat log are available at:

<http://www.cis.drexel.edu/faculty/gerry/publications/conferences/2005/criwg>.

The analysis in this paper is indebted to conversation analysis data sessions at the VMT project, led by Alan Zemel, and comments from Stephen Weimar and Martin Wessner.

it right on his paper); they eventually accepted the correct answer from someone outside the group. No one understood question 6, and the group got it wrong. They got question 7 right (following Cosi and Mic). Only Hal got question 8, but he persuaded the others. (Ben also got it on his paper, but did not participate in the group discussion.) Cosi got the answer to question 9. No one got questions 10 or 11, so the group had to work on these together. The discussion of question 10 was particularly interesting. As we will see, Cosi got the answer to question 10 and explained it to the others (although she had not gotten it on her paper). Hal got question 11 right and the others accepted it (although he had not gotten it on his paper).

So it appears as though the math problems were actually *solved by individuals*. The group responded to proposed answers. In instances where there were competing answers or other issues, the group required the proposer to give an account, defense or explanation. This resulted in an expository form of discourse where one member proposed an answer and explained why it was right. Although the group was not experienced in working together, they succeeded in selecting the best answers that their members could come up with. The result of the group cooperation was to achieve a sum of their individual results.

It is particularly interesting to observe how the group negotiated their group answers given proposals from various members. In some cases, everyone proposed the same answer and it was easy to establish a consensus. In certain other cases, only one person proposed an answer and the others simply went along with it. In more interesting cases, when someone proposed an answer that contradicted other people's opinions or was questionable for some other reason, the proposer was required to give an explanation, justification or accounting of their proposal. We do not have space here to analyze each of the negotiations: how they were begun, how people contributed, how the discussion was continued, how decisions were made and how the group decided to move on to a new problem. In particular, we cannot go into the integration of social chatter and math reasoning or fun making and decision making. Rather, we will take a look at the discussion of question 10, which was particularly interesting because no one had already solved this problem and because we can see the solution emerging in the discourse.

Question 10 is a difficult algebra word problem. It would take considerable effort and expertise to set up and solve equations for it. The group manages to finesse the complete algebraic solution and to identify the correct multiple-choice answer through some insightful reasoning. Question 10 is:

Three years ago, men made up two out of every three internet users in America. Today the ratio of male to female users is about 1 to 1. In that time the number of American females using the internet has grown by 30,000,000, while the number of males who use the internet has grown by 100%. By how much has the total internet-user population increased in America in the past three years?
(A) 50,000,000 (B) 60,000,000 (C) 80,000,000 (D) 100,000,000 (E) 200,000,000

The core discussion of this question takes place in the chat excerpts shown in Table 2.

Table 2. Excerpts from the chat discussion about problem 10

Line	Time	Name	Message	Interval
350	4:31:55	Mic	how do we do this..	
351	4:31:59	Mic	without knowing the total number	0:00:04
352	4:32:01	Mic	of internet users?	0:00:02
			
357	4:32:23	Dan	it all comes from the 30000000	
358	4:32:23	Mic	did u get something for 10?	0:00:00
359	4:32:26	Dan	we already know	0:00:03
360	4:32:44	Mic	30000000 is the number of increase in american females	0:00:18
361	4:33:00	Mic	and since the ratio of male to female	0:00:16
362	4:33:02	Mic	is 1 to 1	0:00:02
363	4:33:09	Mic	thats all i got to give. someone finish it	0:00:07
364	4:33:10	Mic	haha	0:00:01
365	4:33:18	Cosi	haha you jackass	0:00:08
366	4:33:20	Mic	haha	0:00:02
367	4:33:21	Dan	hahaha	0:00:01
368	4:33:26	Mic	u all thought i was gonna figure it out didnt	0:00:05
369	4:33:27	Mic	u	0:00:01
370	4:33:28	Mic	huh?	0:00:01
371	4:33:28	Hal	it would be 60,000,000	0:00:00
372	4:33:30	Mic	hal	0:00:02
373	4:33:31	Mic	its all u	0:00:01
374	4:33:33	Mic	see	0:00:02
375	4:33:34	Mic	i helped	0:00:01
376	4:33:54	Cosi	ok, so what's 11 – just guess on 10	0:00:20
			
386	4:34:45	Mic	lets get back to 5	
387	4:34:47	Cosi	i think it's more than 60,00000	0:00:02
388	4:34:57	Mic	way to complicate things	0:00:10
389	4:35:03	Cosi	haha sorry	0:00:06
390	4:35:05	Mic	life was good until you said that	0:00:02
391	4:35:07	Mic	:(0:00:02
392	4:35:18	Cosi	they cant get higher equally and even out to a 1 to 1 ratio	0:00:11
393	4:35:27	Cosi	oh, no wait, less than that	0:00:09
394	4:35:32	Cosi	50000000	0:00:05
395	4:35:34	Cosi	yeah, it's that	0:00:02
396	4:35:36	Cosi	im pretty sure	0:00:02
397	4:35:37	Mic	haha	0:00:01
398	4:35:38	Mic	how?	0:00:01

399	4:35:57	Cosi	because the women pop had to grow more than the men in order to even out	0:00:19
400	4:36:07	Cosi	so the men cant be equal (30)	0:00:10
401	4:36:11	Mic	oh wow...	0:00:04
402	4:36:16	Mic	i totally skipped the first sentencwe	0:00:05
403	4:36:16	Cosi	therefore, the 50,000,000 is the only workable answer	0:00:00
404	4:36:19	Dan	very smart	0:00:03
405	4:36:21	Cosi	Damn im good	0:00:02

We can see here that the group is meandering somewhat in trying to solve problem 10. Mic raises the question of how to solve it (lines 350-352). Dan suggests that the 30,000,000 figure is key, and Mic tries to build on this suggestion. But Mic ends his attempt with a laugh, clowning around that he was only pretending to figure out the problem. Hal proposes that the answer is 60,000,000 (line 371), but then Cosi complicates matters by questioning this answer (line 387).

Having rejected Hal's proposal, Cosi proceeds to solve the problem on her own. She reasons that the male and female population cannot grow by the same amount from uneven numbers to arrive at equal numbers (line 392). From this, she concludes that the answer is 50,000,000. She announces that she is "pretty sure" of this answer (line 396). At this point, it seems that Cosi has solved the problem on her own.

Mic responds to the statement that Cosi is only "pretty sure" and not positive by requesting an explanation of how Cosi arrived at her opinion that the answer is 50,000,000—and not the 60,000,000 that Hal proposed (line 398).

In the following lines (399, 400, 403), Cosi provides an account of her reasoning. If the females grew by 30,000,000 then the males must have grown by less than that. Therefore, the total growth must have been less than 60,000,000. The only answer listed that meets this condition is 50,000,000—so that must be the correct answer.

Cosi's extended turn providing an exposition of her thinking is interrupted only by Mic (lines 401, 402), who simultaneously affirms Cosi's approach, provides an excuse for not having solved the problem himself, and admits to not having read the problem carefully in the first place. In this way, Mic continues to move the group toward making good decisions about which proposed answers to accept while himself playing the fool. Dan speaks on behalf of the group (line 404), accepting Cosi's answer and proof by praising her as "very smart," to which she responds (line 405), "Damn, I'm good." In the subsequent discussion, both Hal and Mic agree with Cosi's solution. Cosi is anxious to move on to another problem and finally says (line 419), "ok great, im smart, lets move on."

From our analysis, we can see the advantages that have long been claimed by other researchers for collaborative learning [summarized in 21]. A number of students each contributed their best ideas. Some students knew some answers, some others, and together they arrived at a position where they effectively shared the whole set of best answers that any of them had to start with. In addition, the group work sustained their time-on-task beyond what any one student was willing to do, arriving at correct answers for the final two problems.

According to the foregoing analysis, the actual mathematical reasoning was done by individual minds. The group was able to take the results of these individual achievements and gather them together in a particularly effective way. In the end, all members of the group had the opportunity to know more correct answers than they could arrive at on their own. It may not be obvious that every student could then solve all the problems on their own, but there were a number of indications in the chat that students gained insights into aspects of the problem solving that we can assume would stay with them as individual learning outcomes.

In this experiment, we were able to see how the group took good advantage of the knowledge of its members, even though the group had not had any previous experience working together and had no external scaffolding from the teacher or the software in how to collaborate. As researchers, we know which students were able to solve which problems on their own and we could then observe how they interacted to solve the problems in the group context. Furthermore, we had a simple, objective measure of mathematical skill based on correct answers to standardized SAT problems. We observe that a group of students who individually scored 18-27% was able to score 87% when working together. Furthermore, this impressive result can be understood in terms of simply making good decisions about which proposals to listen to on each problem and then spending more engaged time-on-task on the two final problems.

2 Group Cognition in Online Math

In the previous section, the work of the student group was interpreted primarily at the individual unit of analysis. The problem solving was discussed as the accomplishment of individuals. The group decisions were discussed as a form of voting among people who largely made up their minds individually. In many cases, individuals did not hold strong opinions about the answers to the problems and therefore left the group decision up to other individuals—who might have a higher likelihood of knowing the correct answer—by remaining silent. However, it is possible to analyze the chat differently, taking the group as the unit of analysis.

The central point of the alternative approach is that the meaning constructed in a group discourse is often the result of the subtle ways in which utterances of different speakers or writers interact, rather than through a simple addition of ideas expressed or represented in the individual utterances.

Perhaps the greatest problem in understanding how groups work is to clarify the relation of individual to trans-individual contributions to the group meaning making. Clearly, individual group members may have ideas of their own that they introduce into the discourse. Their utterances may have to wait for the right moment in the conversational flow and they might have to design their contributions to fit into the discourse context in order to be accepted as useful proposals with a chance of being taken up, but they also may bring with them some premeditated meaning constructed by their proposer. Individuals also play a necessary role as the *interpreters* of the group meaning in an on-going way as they respond to the discourse [14, chapter 16]. On the other hand, the formative roles of adjacency pairs and other references among

utterances underline the importance of analyzing meaning making at the *group unit of analysis*, not just interpreting the utterances of individuals.

A more detailed analysis of the negotiations of the answers for questions 1 through 9 in the experiment shows that the group had methods for interacting that were quite effective in making good decisions. They had subtle ways of coalescing the individual group members into a collective that could work through the set of math problems, discover solutions and decide which solutions to adopt as the group's answers. This suggests that the problem solving methods used by the group of students is qualitatively different from the methods they use individually. Another way of putting it is that the group collaboration brings additional methods at the group unit of analysis that supplement the individual cognitive methods of problem solving. It may be important to distinguish these different classes of methods at the different levels of analysis, as well as to see subsequently how they work together.

In defining his concept of the *zone of proximal development*, Vygotsky strongly distinguished between what a student could accomplish individually and what that student could accomplish when working with others [8, p 86]: "It is the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers." Based on psychological experiments, Vygotsky argued that what children "could do only under guidance, in collaboration, and in groups at the age of three-to-five years they could do independently when they reached the age of five-to-seven years" (p. 87). In the chat, we have seen that older students can also achieve significantly more in collaborative groups than independently—and we have seen the methods of group interaction that one particular group adopted in one case study to accomplish this.

We can also revisit the solving of problem 10 as a group achievement. Of course, the sequence of recorded events—the lines in the chat log—are the same. But now we no longer attribute the source of the messages to the individuals as the "expression" of internal mental ideas that they have worked out in advance. Rather, we look for evidence in the details of the log of how messages are responses to each other.

Mic's opening question (lines 350-352) is based on the problem statement. The problem asks how much the population has increased. A straight-forward calculation of this increase might involve subtracting from the total number of Internet users now the corresponding figure for three years ago. But the two numbers needed for such a calculation are missing from the problem statement. The problem only gives indirect clues. The problem statement thereby calls for a less direct strategy. Mic's messages respond to this implicit requirement by making it explicit.

Dan responds to Mic's question by proposing an approach for coming up with a strategy. He says (lines 357 and 359), "It all comes from the 30,000,000 we already know." In other words, the strategic key is to start with the clue about the number of females having grown by 30,000,000.

(Note that to analyze the log we must disentangle line 358 from the middle of the two fragments of Dan's text and re-join Dan's text [22]. Mic's question (line 358) is posted at the same time as Dan's proposal, and as a consequence it is ignored and left as a failed proposal [14, chapter 21]).

Mic's next turn (lines 360-364) picks up on the 30,000,000 figure from Dan and tries to take it further by adding the fact that came before that figure in the problem

statement, namely that “Today the ratio of male to female users is about 1 to 1.” Mic puts this forward and asks for the group to continue to develop the strategy.

Mic’s contribution is not the expression of some rational problem solving that we might speculate took place in Mic’s mind. In fact, his contribution—if considered as an individual proposal with math content—only vaguely suggests a mathematical logic. It was primarily an interactive move to keep the group effort going. Following Dan’s posting to the chat, there was an unusually long pause of 18 seconds. In face-to-face conversation, a pause of a few seconds is embarrassingly long and exerts considerable pressure on the participants to make another contribution; in chat, 18 seconds can have a similar effect. So Mic repeats Dan’s reference to 30,000,000. Following another pause of 16 seconds, Mic adds the reference to the 1-to-1 ratio. He then explicitly calls on the other group members to join in. He admits that he cannot take it further himself, and he laughs.

Cosi, Dan and Mic have a good laugh at Mic’s expense, taking his contribution as a practical joke, as an attempt to look like he was making a significant mathematical contribution and then stopping short of delivering. This fills in an otherwise discouraging silence during which no one knows how to advance mathematically with the problem. The laughter lightens up the interaction, allowing people to throw ideas into the mix without worrying that they will necessarily be taken too seriously if they are only partial, or even wrong. After Mic’s jackass-like behavior, any other contribution would seem an improvement. In fact, Mic’s proposal and request are taken up.

Hal then proposes that the answer “would be 60,000,000” (line 371). This is a direct consequence of finishing Mic’s partial proposal. If there are 30,000,000 females (line 360) and the ratio of males to females is 1 to 1 (lines 361-362) and you want to know the total number (line 351), then the conclusion that “it would be 60,000,000” is at hand. Mic takes this to be the answer to problem 10 and tries to take partial credit for it by pointing out, “u see I helped” (lines 373-375).

At that point, Cosi suggests the group should go on to problem 11 and “just guess” on 10 (line 376). This declines to affirm Mic’s acceptance of 60,000,000 as the answer to question 10, but does so without raising this as a topic for further group discussion. Without making a decision about 10, the group goes on to all decide that the answer to problem 11 is C (lines 378-385, spanning just half a minute), as already stated by Hal in line 353.

Mic then summarizes the group’s status as: “So we got B for 10 and c for 11; lets get back to 5” (lines 384-386). At this point, Cosi objects to Mic’s continued assumption that Hal’s 60,000,000 is the answer to problem 10. Mic and Cosi joke about their disagreement. Again, the group’s light-hearted attitude avoids the potential of disagreements within the group becoming disruptive of the group functioning.

Cosi then formulates an argument (line 392) why the answer cannot be 60,000,000. The male and female populations cannot get higher equally (i.e., by 30,000,000 each) because they have to even out from unequal numbers according to the problem statement. After formulating this text, Cosi checks and then corrects her previous claim that “I think it’s more than 60,000,000” (line 387): “Oh, no wait, less than that: 50,000,000” (lines 393-394).

Cosi is somewhat hesitant about her revised claim. First she checks it and says, “Yeah, it’s that” (line 395), followed by the hedge, “Im pretty sure” (line 396). Mic

continues the laughter and then requests an account of how Cosi is pretty sure that the answer should be 50,000,000.

After a 19 second pause, Cosi takes the extended expository turn that Mic had offered her and the others had left open. She lays out a concise proof of her claim. Her argument concerns the increase in the number of females and the ratios of male to female users—the issues raised at the beginning of the group discussion by Dan and Mic. It is plausible that Cosi used the 19 second pause to reflect upon the solution that the group had come to and that her contributions had completed. Thus, her well-worked out retrospective account seems like the expression of her mental work in constructing the narrative explanation, although her earlier contributions to solving the math of the problem seemed more like spontaneous reactions to the flow of the group discourse.

A solution to problem 10 carried out from scratch using algebraic methods that translated the word problem into a set of equations to be solved for unknown values would have looked very different from Cosi's argument. Her contributions to the chat did not express an independent, individual approach to the problem. Rather, they were responses to preceding contributions. Cosi's texts performed checks on the previous texts and extended their arguments in directions opened up and called for by those previous contributions. Although Dan, Mic and Hal did not carry out the further steps that their own contributions required, they succeeded in starting a discourse that Cosi was able to repair and complete.

This analysis of the log excerpt gives a more group-centered view of the *collaborative solving* of the math problem by the group. Of course, at the level of individual postings, each contribution was that of an individual. But it is not necessary to see those contributions as expressions of prior private mental activities. Rather, they can be seen as responses to the previous texts, the context of the problem-solving task (e.g., the elements of the problem 10 text) and elicitations of contributions to come. These ties of the individual postings to the sequentially unfolding group discourse can be seen in the form of the postings themselves: single utterances do not stand on their own, but make *elliptical references* to previous mentionings, *indexical references* to matters in the physical and discourse situation and *projective references* to anticipated future responses or actions of other people [see 14, chapter 12]. The references weave a temporal fabric of discourse that defines the meaning of each text within its narrative context. Thus, the individual contributions are incorporated into a problem solving dialog at the group unit of analysis, which is where the meaning of the log is constructed.

In weaving the discourse fabric, groups use different methods. We have discussed two methods of group discourse used in math problem solving in this chat: exploratory inquiry and expository narrative. In the excerpt concerning problem 10, we have seen that the group first explores a solution path by different students making small contributions that build on each other sequentially. When a candidate answer is reached that someone is “pretty sure” about, that person is asked to provide an extended account or proof of the answer. Thus, Cosi participates first in the joint exploratory inquiry and then provides an expository narrative. Both these methods are interactive discourse methods that involve responding to requests, structuring texts to be read by other group members and eliciting comments, questions and uptake.

Conversation analysts have identified *adjacency pairs* as a powerful way in which meaning is interactively constructed. An adjacency pair is a set of utterances by different people that forms a smallest meaningful unit [23]. For instance, a greeting or a question cannot meaningfully stand alone. You cannot meaningfully express a greeting or a question without someone else being there in the discourse to respond with a return greeting or an answer. The other speaker may ignore, decline or respond to your greeting or question, but your utterance cannot be a greeting or a question without it addressing itself to a potential respondent. The respondent may just be an imaginary dialog partner if you are carrying out the dialog in your mind [see 9]. Adjacency pairs are fundamental mechanisms of social interaction; even very young speakers and quite disabled speakers (e.g., advanced Alzheimer sufferers) often respond appropriately to greetings and questions. Adjacency pairs are important elements for weaving together contributions from different participants into a group discourse.

When I analyzed a different online chat of mathematics problem solving, I defined an adjacency pair that seemed to play a prominent role. I called it the *math proposal adjacency pair* [14, chapter 21]. In that chat, a math proposal adjacency pair consisted of a problem solving proposal by one person followed by a response. The proposal addressed the other students as a group and required one or more of them to respond to the proposal on behalf of the group. The proposal might be a tactical suggestion, like “I think we should start with the 30,000,000 figure.” Alternatively, it might be a next step in the mathematical solution, like “They can’t get higher equally and even out to a 1 to 1 ratio.” The response might simply be “k”—“okay, that’s interesting, what’s next?” The pattern was that progress in problem solving would not continue after someone made a proposal until the group responded to that proposal. If they responded affirmatively, a next step could then be proposed. If they responded with a question or an objection, then that response would have to be resolved before a next proposal could be put forward. It was important to the group that there be some kind of explicit uptake by the group to each proposal. A counter-example proved the rule. One participant made a failed proposal. This was an attempt to suggest a strategy involving proportions. But the proposer failed to formulate his contribution as an effective first part of a math proposal adjacency pair, and the rest of the group failed to take it up with the necessary second pair-part response.

In the chat we are analyzing now, the math proposal adjacency pairs have a somewhat different appearance. We can identify proposals in, for instance, lines 352, 357, 360, 362, 371, 387, 392 and 394. None of these is followed by a simple, explicit response, like “ok.” Rather, each is eventually followed by the next proposal that builds on the first, thereby implicitly affirming it. This is an interesting variation on the math proposal adjacency pair method of problem solving. It illustrates how different groups develop and follow different group methods of doing what they are doing, such as deciding upon answers to math problems.

If we combine the proposals from Mic, Dan, Hal and Cosi, they read like the cognitive process of an individual problem solver:

How can I figure out the increase in users without knowing the total number of internet users? It seems to all come from the 30,000,000 figure. 30,000,000 is the number of increase in American females. Since the ratio of male to female is 1 to 1, the total of male and

female combined would be 60,000,000. No, I think it must be more than 60,000,000 because the male and female user populations can't get higher at equal rates and still even out to a 1 to 1 ratio after starting uneven. No, I made a mistake, the total must be less than 60,000,000. It could be 50,000,000, which is the only multiple choice option less than 60,000,000.

Mathematical problem solving is a paradigm case of human cognition. It is common to say of someone who can solve math problems that he or she is smart. In fact, we see that taking place in line 404. Here, the group has solved the problem by constructing an argument much like what an individual might construct. So we can attribute group cognition or intelligence to the group [see 14, esp. chapter 19].

Unfortunately, the group of students in the chat log does not seem to attribute the problem solving intelligence to itself, but only to one of its members, Cosi. Because she takes the final step and arrives at the answer and because she provides the narrative account or proof, Dan says of her, "very smart" (line 404). Later (line 419), Cosi agrees, downgrading the self-praise by using it to close the discussion of problem 10 and of her role in solving it by proposing that the group move on to a remaining problem: "Ok great, im smart, lets move on." Casting Cosi as the smart one who solves problems leaves Mic cast as the jackass or class clown when in fact Mic is very skilled at facilitating the chat so that the whole group solves problems that neither Mic nor the others solved independently.

There is an ideology of individualism at work here that encourages both educational researchers and student participants to view problem solving as an accomplishment of individuals rather than groups. This has serious consequences for the design and adoption of groupware to support problem solving, as well as for research methodology and student learning. If groupware designers tried to support collaborative interactions, then they might design more than just generic communication platforms for the transmission of expressions of personal ideas. If researchers studying the use of groupware focused on processes of collaboration and the methods that groups used to solve problems—as opposed to treating exclusively individuals as cognitive agents—then research methods might focus more on conversation analysis [17], video analysis [24] and their application to discourse logs than on surveys and interviews of individual opinions. If students using groupware conceived of their work as interactively achieving a group solution, they might take more advantage of groupware collaboration features and might structure their textual contributions more explicitly as parts of an interwoven fabric of collaborative knowledge-building group discourse.

3 Groupware to Support Group Cognition

The first step in thinking about the design of groupware today is to understand the methods that groups use to accomplish problem solving, scientific inquiry, decision making, argumentation and the other tasks that they want to do. Generic communication platforms developed to meet the needs of corporations will continue to make new technologies available in response to market pressures. Within education, course management systems to support the administration of distance

education will proliferate under their own economic drives. But those developments are almost exclusively guided by a philosophy of individual cognition and the transfer of representations of mental contents.

The preceding analysis of a case study of group cognition suggests a variety of new design principles. Clearly, one or two case studies is not enough to inform a new approach to groupware design. This paper has only suggested the kind of analysis that is needed to investigate and characterize the methods that groups of students might use to do their work collaboratively. Different age groups, tasks, cultures and environments will introduce considerable variety in how groups constitute themselves, define their work, socialize, problem solve, persuade, guide, decide, conclude, etc. Nevertheless, a number of principles can already be suggested. It is important to start thinking about groupware design because ideas for innovative functionality and prototypes of new components will have to be tried out with online groups and the resultant logs analyzed. One cannot know how new technologies will lead to new member methods without such investigation.

Here are some very preliminary suggestions for groupware design principles:

Persistency and Visibility. Make the group work visible and persistent so that everyone in the group can easily see what has been accomplished by all members. Ideally, important contributions should stand out so that people do not have to search for them, but are made aware of them with little or no effort. This is a non-trivial requirement, since the work of a group quickly becomes too extensive for everyone to read and keep track of. The software must somehow help with this.

Deictic Referencing. As discussed above, the references from one message to another or to objects in the problem context are essential to the meaning making. Software could make these references visible under certain conditions. Patterns of references among proposals, adjacency pairs and responses between different group members could also be displayed in order to give participants indicators about how their group interaction is going.

Virtual Workspaces. Ideally, the groupware would encourage noticing, recognizing and reflecting on related contributions. There should certainly be group workspaces for different kinds of work to be done together, creating shared artifacts. For instance, there could be group workspaces for taking notes and annotating them, for jointly navigating the Internet, for constructing shared drawings, for building formal arguments together, for collecting annotated bibliographies and other lists or collections. Issues of turn-taking, ownership and control become important here.

Shared and Personal Places. It may be useful to distinguish and sometimes to separate individual and group work [13]. However, it may be important to make even the individual work visible to everyone. Group accomplishments build on the individual contributions. Even contributions that the proposer does not consider significant may, as we have seen above, provide a key to progress of the group. In addition, group members often want to know what people are doing when they are not active in the group. Content should move fluidly from place to place.

Computational Support. Of course, a major advantage of having groupware systems running on computers is that they can provide computational support to the work of

their users. They can filter or tailor different views or computational perspectives [14, chapter 6] of materials in the chat or workspaces, as well as providing search, browsing and annotating facilities. They can play various moderator roles.

Access to Tools and Resources. Another advantage of the networked computer infrastructure is that groupware can provide structured access to information, tools and other resources available on the Internet, for instance in relevant digital libraries and software repositories.

Opening New Worlds and (Sub-)Communities. Finally, Internet connectivity allows for groups and their members to participate in larger online communities and to interact with other groups—either similar or complementary. Groupware could facilitate the building of open-ended networks of individual, group and community connections, or the definition of new sub-communities.

Allowing Natural Language Subtleties. While computer support brings many potential advantages, it also brings the danger of destroying the extreme flexibility and adaptability of the natural language used in conversation and group interactions. Groupware designs should be careful not to impose rigid ontologies and sets of allowable speech acts for the sake of enabling automated analyses. It should permit the use of overloaded, multiple functioning, subtle linguistic expression that is not reified, stereotyped, coded or packaged, but that opens space for interpretation, engagement, creativity, problem solving. As we saw in the chat, even a simple laugh can perform multiple complex roles simultaneously. Chat is a vibrant form of human interaction in which people exercise their creativity to invent linguistic novelties such as abbreviations, contractions, emoticons and new ways of interacting textually. Groupware should support this, not cramp it.

References

- [1] Shannon, C. & Weaver, W. (1949) *The Mathematical Theory of Communication*, University of Illinois Press, Chicago, IL.
- [2] Johnson, D. W. & Johnson, R. T. (1989) *Cooperation and Competition: Theory and Research*, Interaction Book Company, Edina, MN.
- [3] Suchman, L. (1987) *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge University Press, Cambridge, UK.
- [4] Hutchins, E. (1996) *Cognition in the Wild*, MIT Press, Cambridge, MA.
- [5] Engeström, Y. (1999) Activity theory and individual and social transformation. In Y. Engeström, R. Miettinen, & R.-L. Punamäki (Eds.), *Perspectives on Activity Theory*, Cambridge University Press, Cambridge, UK, pp. 19-38.
- [6] Garfinkel, H. (1967) *Studies in Ethnomethodology*, Prentice-Hall, Englewood Cliffs, NJ.
- [7] Heidegger, M. (1927/1996) *Being and Time: A Translation of Sein und Zeit*, (J. Stambaugh, Trans.), SUNY Press, Albany, NY.
- [8] Vygotsky, L. (1930/1978) *Mind in Society*, Harvard University Press, Cambridge, MA.
- [9] Bakhtin, M. (1986) *Speech Genres and Other Late Essays*, (V. McGee, Trans.), University of Texas Press, Austin, TX.
- [10] Wessner, M. & Pfister, H.-R. (2001) Group formation in computer-supported collaborative learning, In: *Proceedings of ACM SIGGROUP Conference on Supporting Group Work (Group 2001)*, Boulder, CO, pp. 24-31.

- [11] Stahl, G. & Herrmann, T. (1999) Intertwining perspectives and negotiation, In: Proceedings of *International Conference on Supporting Group Work (Group '99)*, Phoenix, AZ, pp. 316-324. Available at: <http://www.cis.drexel.edu/faculty/gerry/cscl/papers/ch07.pdf>.
- [12] Vogel, D., Nunamaker, J., Applegate, L., & Konsynski, B. (1987) Group decision support systems: Determinants of success, In: Proceedings of *Decision Support Systems (DSS '87)*, pp. 118-128.
- [13] Stahl, G. (2002) Groupware goes to school. In J. H. J. Pino (Ed.) *Groupware: Design, Implementation and Use: Proceedings of the 8th International Workshop on Groupware (CRIWG '02), Volume LNCS 2440*, Springer, La Serena, Chile, pp. 7-24. Available at: <http://www.cis.drexel.edu/faculty/gerry/cscl/papers/ch11.pdf>.
- [14] Stahl, G. (2006) *Group Cognition: Computer Support for Building Collaborative Knowledge*, MIT Press, Cambridge, MA. Available at: <http://www.cis.drexel.edu/faculty/gerry/mit/>.
- [15] Zemel, A., Xhafa, F., & Stahl, G. (2005) Analyzing the organization of collaborative math problem-solving in online chats using statistics and conversation analysis, In: Proceedings of *CRIWG International Workshop on Groupware*, Racife, Brazil.
- [16] Garcia, A. & Jacobs, J. B. (1999) The Eyes of the Beholder: Understanding the Turn-Taking System in Quasi-Synchronous Computer-Mediated Communication, *Research on Language and Social Interaction*, 34 (4), pp. 337-367.
- [17] Sacks, H. (1992) *Lectures on Conversation*, Blackwell, Oxford, UK.
- [18] Bruner, J. (1990) *Acts of Meaning*, Harvard University Press, Cambridge, MA.
- [19] Wegerif, R. (2005) A dialogical understanding of the relationship between CSCL and teaching thinking skills, In: Proceedings of *International Conference of Computer Support for Collaborative Learning (CSCL 2005)*, Taipei, Taiwan.
- [20] Dillenbourg, P. (1999) What do you mean by "collaborative learning"? In P. Dillenbourg (Ed.) *Collaborative Learning: Cognitive and Computational Approaches*, Pergamon, Elsevier Science, Amsterdam, NL, pp. 1-16.
- [21] Strijbos, J.-W., Kirschner, P., & Martens, R. (Eds.) (2004) *What We Know about CSCL ... and Implementing it in Higher Education*, Kluwer Academic Publishers, Dordrecht, Netherlands.
- [22] Cakir, M., Xhafa, F., Zhou, N., & Stahl, G. (2005) Thread-based analysis of patterns of collaborative interaction in chat, In: Proceedings of *international conference on AI in Education (AI-Ed 2005)*, Amsterdam, Netherlands.
- [23] Duranti, A. (1998) *Linguistic Anthropology*, Cambridge University Press, Cambridge, UK.
- [24] Koschmann, T., Stahl, G., & Zemel, A. (2005) The Video Analyst's Manifesto (or The Implications of Garfinkel's Policies for the Development of a Program of Video Analytic Research within the Learning Sciences). In R. Goldman, R. Pea, B. Barron, & S. Derry (Eds.), *Video Research in the Learning Sciences*. Available at: <http://www.cis.drexel.edu/faculty/gerry/publications/journals/manifesto.pdf>.

A Framework for Prototyping Collaborative Virtual Environments

Clinton Jeffery, Akshay Dabholkar, Kosta Tachtevrenidis, and Yosep Kim

Department of Computer Science, New Mexico State
{jeffery, adabholk, ktachtev, ykim}@cs.nmsu.edu

Abstract. *Unicron* is a platform for rapidly developing virtual environments that combine two popular forms of collaboration: a 3D collaborative virtual environment fostering meetings, appointments, whiteboard sessions and lectures, along with a 2D development environment including collaborative software design, text editing, and debugging tools. This paper presents novel aspects of the *Unicron* design and implementation.

Keywords: Collaborative virtual environments.

1 Introduction

Collaborative 3D virtual environments (CVEs) are highly successful in the area of entertainment, in games and social environments such as Everquest and There.com. This success suggests enormous potential in many other domains, especially those involving geographically-dispersed participants, such as large-scale software development efforts, or distance education.

In order to apply CVE's successfully to these other domains, several major obstacles must be overcome. This paper addresses two of those obstacles: the high cost of developing new CVE's which constitutes a barrier to entry, and the importance of bringing existing 2D collaborative tools along to get the "real" domain-specific work done. We have constructed a CVE platform called *Unicron* in order to explore the tools and methods needed to solve these problems.

2 Motivation

Unicron was initiated for the purpose of computer science distance education. Existing distance education tools such as WebCT lack domain-specific support, such as the need for an instructor to look at an individual student's debugging session and explain how to read what the student sees on their screen. Collaborative editors abound, but tools for collaborating over compiler diagnostics or debugger sessions are not common.

We turned to collaborative virtual environments because they have proven highly effective at bonding people together and providing effective substitutes for in-person social necessities such as chatting, making appointments, and holding

meetings. Many users of CVEs work productively every day with people they have never met face to face. The high cost of developing a CVE was seemingly prohibitive, so we developed a research project aimed at reducing this cost. Unicorn's feature set was adapted several times during the construction of our first "low cost" CVE by surprising actual "high cost" tasks encountered during development.

The initial goal for Unicorn was to produce a CVE that brings a Computer Science department experience to remote locations, including as many aspects as possible of the freshman-through-junior year experience. The virtual community is modelled after the appearance and actual dynamic content of the 1st floor of Science Hall at NMSU. Remote students gain access to the department's tools and expertise (such as CS software tools, seminars and meetings, lab assistants).

Access to CS computers and software tools can already be accomplished by existing 2D tools, and seminars and meetings can be held using teleconferencing rooms, so a 3D virtual community is in principle unnecessary, but in practice we feel it offers several advantages. Others' experience with Viras [15] supports the idea that a 3D space provides better social awareness than 2D places of comparable size and complexity. In addition, we believe 3D virtual spaces borrow successfully from users' expertise at remembering and navigating 3D real spaces to provide superior ease of learning and use.

Compared with Viras' use of an abstract and customizable archipelago, it seems unimaginative to base a 3D space on a real academic department. While the Unicorn framework can be used to build abstract spaces, using a real space also has advantages. Using the Science Hall model, transfer students already know the physical environment by the time they arrive at NMSU. Also, faculty and students who become familiar with the real Science Hall, such as prospective students from around the state who visit for a five week summer session, will already know their way around the 3D model when they see it. Together, this helps students and faculty correlate distance education experiences with on-campus CS education experiences, and simplifies their orientation into the NMSU CS department.

A virtual community both complements and contrasts with traditional video teleconferencing, and with web-based tools such as WebCT. While it is impossible to set up cameras from every angle in every room, it is very possible to supplement the traditional distance-learning cameras and digital capture devices in key locations with a 3D model of the department. Special-purpose Internet-based collaborative versions of CS laboratory software, such as text editors and compilers, are part of this virtual CS community. Integration of existing distance-education technology into this 3D virtual community is also important.

To the user, Unicorn looks and feels like an interactive 3D videogame application on their PC, within which they can move around, examine signs and notices, go to class or to an instructor's office hours, and go to a virtual lab to work on assignments. Within the virtual lab, the view often shifts to a higher-resolution 2D view of the tools, but the collaborative nature of the environment (chatting, asking the TA or instructor questions, etc.) remains consistent. Text

and voice chat modes can be set to local proximity (seen heard automatically by those nearby in the 3D environment) or to select individuals by name (closer to a phone or private chat style of communication).

Perhaps the “killer application” targeted by the virtual environment is to see who is available to provide (remote) technical assistance, and then obtain that assistance (say, with debugging a program) in a 2D view of code. The convenience factor of being able to perform that task in real time despite the students or instructor being far away or simply away from the office is tremendous. This replaces long sequences of e-mails in which student and instructor struggle to get to the bottom of the student’s problem which the student may not understand well enough to describe precisely, or which the student may not have the writing skill to describe in less than a long and painful message.

3 Design

Unicron consists of several components that form the middle layer in a rapid prototyping framework for collaborative virtual environments:

- in the bottom layer, a very high level language was augmented with ultra-simple 3D, network, and audio API’s.
- the Unicron class library provides infrastructure for the networked 3D environment, e.g. the behavior of doors, whiteboards, and avatars; the Unicron server enables user interaction and shares state between clients; Unicron also provides simple “builder” tools to generate a virtual environment from inputs such as 2D floor plan data and extract textures from digital camera photos.
- the application layer for any particular CVE generated using Unicron consists of: a 3D model produced semiautomatically using Unicron builder tools; a set of domain collaboration tools; and a set of user accounts, created on the Unicron server.

Figure 1 shows several aspects of the Unicron client-server architecture.

Unicron was developed as a prototyping framework rather than a particular CVE because the same properties that are needed in order to easily modify and experiment with the integration of the CVE and the collaborative 2D tools also make it relatively easy to construct new CVE’s. Objects in the virtual environment are built using a simple 3D API in a very high-level general purpose applications prototyping language. A very high-level language has been employed by other systems for rapid prototyping virtual reality environments, such as Alice [14]. Our emphasis on collaboration is different from that effort’s emphasis on graphics and animation.

3.1 Language Layer

Some of the research effort in developing Unicron was conducted in terms extending a very high level language to support collaborative virtual environments.

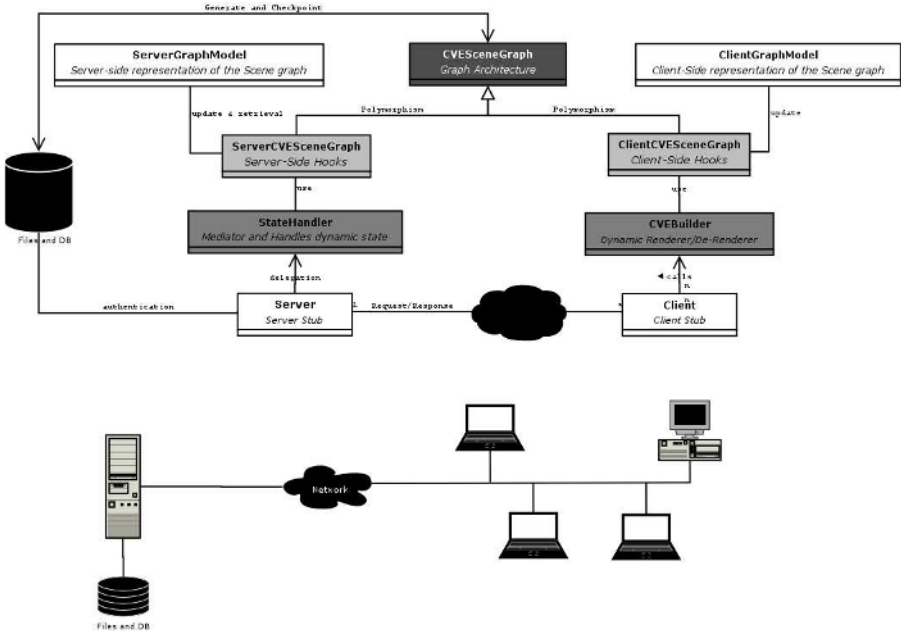


Fig. 1. Unicorn CVE Architecture

Simple high level interfaces to graphics, networking, and audio were added or extended to the Unicon programming language in order to build Unicorn [10]. Unicon is described at <http://unicon.org>.

Language extension is selected instead of writing libraries or modules when a feature is general, or when it has the need or the potential to interact with other features in the language virtual machine or runtime system. Once a given hardware capability is sufficiently ubiquitous, adding control structures and built-in syntax to access it is not just a notational convenience, but an enabling technology.

3.2 Class Library Layer

The Unicorn class library primarily serves to model virtual environment functionality, independent of its views and controls. The research contribution here is not to invent new paradigms, but to explore the simplest implementation techniques that provide sufficient performance on current hardware. This design bias, combined with the very high level language used, make it easy to add new features and conduct experiments. This is appropriate since CVE technology is still in its infancy and only gradually moving towards adolescence: many novel domain features need to be explored in the next decade.

The virtual environment model is a graph data structure, in which “room” objects are nodes, and doors and openings are edges with interesting properties.

It includes relatively static data that is replicated to all clients at install- or load-time and usually not modified during the middle of a user session, as well as dynamic data that can be modified by users. The model is controlled and maintained on a master server, which sends each client that small subset of the dynamic data that is needed for the user's current projection. The server sends dynamic information only to those users' who are in proximity or otherwise need be aware of that information. Where possible, multiple dynamic state changes are bundled together into a single network packet. These approaches reduce the server's network load. As with most CVE's the server remains the main limit to scalability despite these techniques.

We developed our own portable multiplatform representation of the model, after trying many state of the art technologies that we expected to use, such as SQL, XML, VRML, and X3D. We were surprised not to be using VRML or X3D, but they proved much too low-level and cumbersome for our purpose, due to their generality. Avoiding SQL or simpler database technologies such as Access or GDBM allows our server to be brought up trivially in new environments. This aids in testing, makes creating new Unicron-based worlds trivial, and simplifies tasks such as migrating server state to a new architecture, as we found when moving from a 32- to 64-bit platform. The static and dynamic state is all represented using text files in subdirectories in the file system. So far we have not observed a performance problem due to this choice, although one does have to be sensible about how often the server saves frequent dynamic state changes such as avatar moves.

Examples of static and dynamic model data are presented below; first a Room object, then a Door out of that room, then the dynamic state for that door. These files are simple enough to edit or debug easily by hand, and easily generated by a level editor. The coordinate system is a simple cartesian system with units of 1 meter with the origin at the northwest corner of our building.

```
# static properties of a room
Room {
name SH 167
x 29.2
y 0
z 0.2
w 6
h 3.05
l 3.7
floor Rect { texture floor2.gif }
obstacles [
  Box { # window sill
    Rect {coords [29.2,0,.22, 29.2,1,.22, 35.2,1,.22, 35.2,0,.22]}
    Rect {coords [29.2,1,.22, 29.2,1,.2, 35.2,1,.2, 35.2,1,.22]}
  }
]
decorations [
  Rect { # window
```



```

    texture wall2.gif
    coords [29.2,1,.22, 29.2,3.2,.22, 35.2,3.2,.22, 35.2,1,.22]
  }
  Rect { # whiteboard
    texture whiteboard.gif
    coords [29.3,1,2.5, 29.3,2.5,2.5, 29.3,2.5,.4, 29.3,1,.4]
  }
]
}

# static properties of a door
Door {
x 33
y 0
z 3.9
height 2.3
plane 3
rooms [SH 167, cooridoor 167]
}

# dynamic state of a door
link {
name link1
openness 1.0
delta 0
direction 1
}

```

The intent of this human readable format was to be concise and human-maintainable as part of the larger goal of reducing the effort to develop CVE's. In practice graphical tools are usually used to generate models, but human-readable output is still useful. The entire Science Hall first floor static model data is around 30KB, making it feasible to send out new models in-session or at program startup as part of the patcher.

3.3 Application Layer

The Unicorn client, called NSH, renders 3D environment sessions in one of many tabbed buffers, which can also include collaborative editing, compilation, execution, debugging, and UML design sessions. The 3D CVE sessions support lecture, lab, and office hour academic functions, serving largely to coordinate more detailed collaborative activities using the other tools. A subgraph of the whole client projection of the CVE is rendered, based on a simple proximity heuristic; as a user moves into a new room, new nodes in the graph become visible, other nodes are deemed invisible and derendered, and the server changes the set of dynamic data for which the client will be informed.

The initial proximity heuristic (the null heuristic) rendered all static data and all available dynamic data at each step. This worked surprisingly well on

reasonable OpenGL implementations, but did not scale well on the typical Windows clients we expect to find in users' homes. A second naive heuristic rendered all nodes within a distance k in the Room graph, unless a closed door prevented visibility. This works sufficiently, but can be improved with very little effort by varying k as the graph is traversed, based on the direction the user is facing and the edge properties being traversed. For example, people cannot see around corners, although they may hear around them. The rendering traversals can be precomputed during initialization when the static data is loaded. Other high performance algorithms such as traversals of binary space partition trees are not implemented in Unicron yet because graphics fidelity has been a lower priority than networking and collaboration aspects of the CVE framework thusfar.

The graphics fidelity of the Unicron virtual environment is on the low side. High graphical fidelity requires an order of magnitude more effort in terms of texture art as well as programming, reducing our ability to add experimental domain-specific features, without changing functionality much from our distance education perspective. This tradeoff is acceptable for Unicron because the CVE serves largely for the coordination of collaborative 2D tool sessions, and because it needs to run on stock PC's without special hardware, rather than a high-fidelity virtual reality system with head mounted display and data gloves or other VR peripherals.

Unicron's network protocol and architecture is simple, but features several hybrid aspects. Unicron's architecture decouples simulation from rendering, as is the case for Alice [14]. The simulation is performed on a shared server cluster which also handles voice transmission and recording processes.

Client messages are classified as vital or transient and sent by TCP or UDP accordingly. Private communications (both chat and audio) are sent peer-to-peer if possible, but a central server is used for forwarding between peers that cannot



Fig. 2. Avatars

communicate directly due to firewalls, as well as maintaining shared state of many forms. Secondary servers similar to the booster boxes advocated by IBM Zurich are deployed at sites where substantial bandwidth sharing is enabled by their use [2]. In our case this includes the campuses of our educational partners, a consortium of two-year colleges.

3.4 Avatars

Unicon's avatars are graphically simple, but customizable communication tools (Figure 2). Avatars require about as many graphical primitives as would a LEGO(tm) figure, and feature the ability to point, an identifying label, and a visual indication when audio or text chatting is performed.

Creating an avatar is a simple exercise. Using the GUI, users can customize their clothing using colors or textures, and provide a GIF or JPG image to present their face in an ordinary rectangle or texturemapped within an egg-shaped head, which is more recognizable from the side or rear.

4 Implementation

Unicon is written in the Unicon programming language [11], a descendant of Icon [7]. Unicon features a set of 3D graphics facilities based on OpenGL that are profoundly simpler and easier to learn and use than OpenGL or Java3D [12].

In addition to those features already described, Unicon's first incarnation as a CS education tool features:

- Audio stream access between selected locations, allowing hands-free verbal communication.
- Dynamic classroom projector and electronic whiteboard output are integrated into the 3D environment
- Passive capture and recording (via microphones and high-resolution digital cameras) of classroom lectures. The audio and conventional black/white-board content are captured and integrated into the 3D environment in near real-time.
- A collaborative software development environment, developed to include special purpose, multi-user shared-view versions of compilers (for C, C++, and Java), programmer's editors, debuggers, and software design (UML) diagramming tools.

4.1 An Immersive 3D Graphical Environment

Unicon's graphical environment is cartoon-like (similar to popular games), rather than aiming at being photorealistic as for many CVEs such as *Le Deuxieme Monde*. We believe there will be cultural side-benefits compared with traditional distance learning: for example, students will not have to engage in eye contact, and will have less fear of embarrassment while asking questions. Figure 3 shows example scenes students might see in this environment.

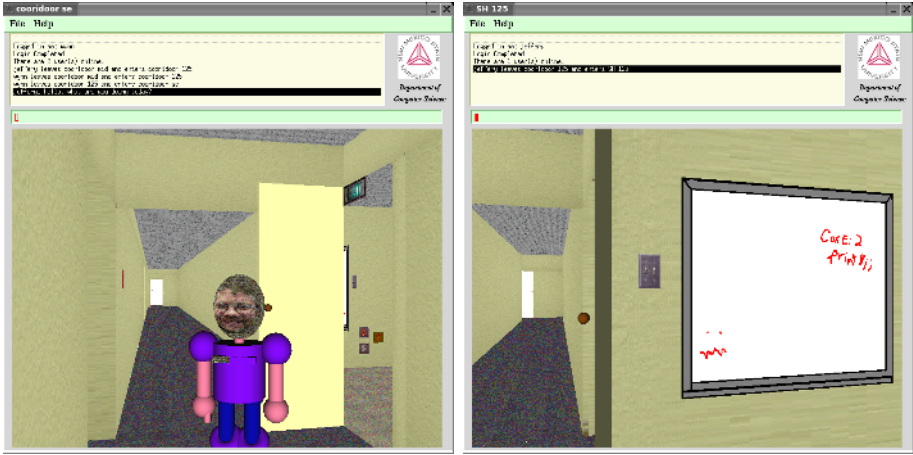


Fig. 3. Virtual Academia with Integrated Whiteboards, Chat and Voice

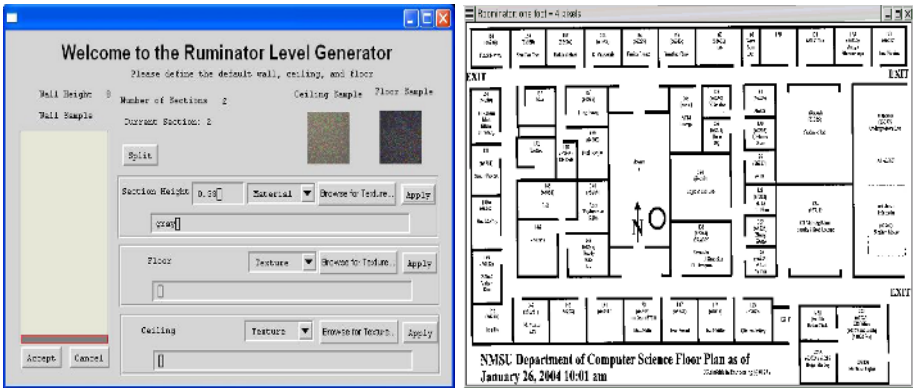


Fig. 4. Layout Editing using Ordinary Floor Plans

The virtual whiteboard is a fairly standard collaborative tool; NMSU's electronic classroom features custom local software that feeds the whiteboard content up in Adobe SVG (an XML) format via HTTP, where it is available to the CVE along with other SVG plugin-equipped browsers.

Besides whiteboards and their pens which naturally relate to interactions with 2D drawings, other interactive objects within this environment include avatars (other users), books (on-line documentation), and virtual workstations. Virtual workstations serve to integrate and interconnect with 2D collaborative applications; for example, to join someone's 2D collaborative editor session, walk over to the workstation their avatar is at, and click on their machine. Visible indications of which machines are occupied, and which printers have long wait-queues are other example uses of virtual objects, enabling remote users to select a computer on which to remote login or select a printer to receive a print job.

While open-source image manipulation and 3D tools for artists are abundant, free tools dedicated to simply and rapidly generating 3D virtual buildings from floor plans are scarce. After exploring several alternatives including VRML- and X3D-capable 3D tools, we failed to find what we needed and developed a crude “level wizard” that generates our environment directly from a dialog specifying the default ceiling, walls, and floors, followed by a semiautomatic room layout extractor. The layout extractor allows the developer to specify the rooms in the model directly from an ordinary floor plan image file, which may have been scanned-in, drawn by hand, or captured from another tool as a screen shot.

4.2 Collaborative Software Development Environment

Computer Science faculty working with students at a distance need more than a bulletin board or chat facility, they need to see the contents of the software development tools that are running on the student’s screen, to advise the student about compiler errors and runtime errors they may be experiencing, and show them how to perform problem solving tasks such as debugging. Collaboration software such as Centra and NetMeeting allow shared views of documents, but not within the context of software development. The remote-control genre of commercial software (such as PC Anywhere or RealVNC) provides shared views of applications, but at a considerable cost in bandwidth. For Computer Science instruction, a cross-platform solution (for Linux, UNIX, and Windows) is often

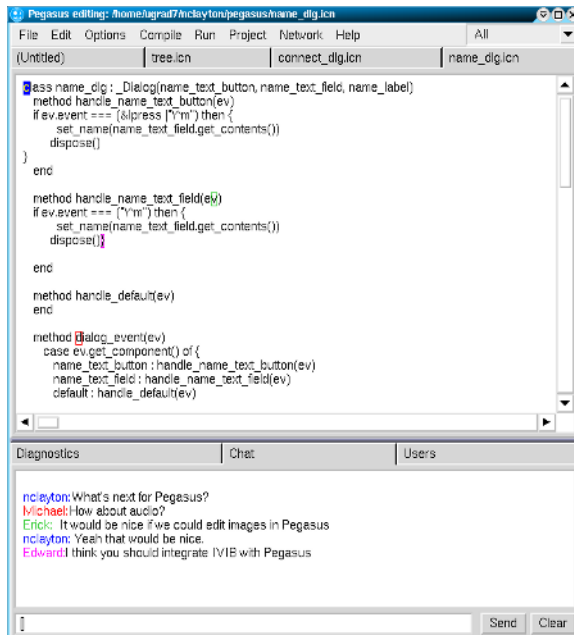


Fig. 5. A collaborative development environment

needed; specifically, one that is open source, allowing it to be integrated into the virtual community environment; RealVNC might be a good candidate. In New Mexico most of the population is rural and many students still use dialup, where a minimum-bandwidth solution is needed.

Unicon's collaborative development environment prototype called Pegasus is shown in Figure 5. The prototype allows users to edit text, watch each other, and chat. We are in the process of adding hands-free audio communication, and collaborative views of other areas of software development such as UML design diagrams, compilation, execution, and debugging output. Pegasus will also be integrated into the 3D environment in the preceding section. Instructors will be able to walk around a virtual lab, talk with students, and look at their virtual screens, zooming in to the high resolution collaborative environment view when questions require on-screen details.

5 Innovations

The primary innovations thusfar in Unicon are not in the user-visible application experience, nor in the CVE architecture, but in the runtime system support, class library and tools that reduce the programming effort necessary to build a CVE. This support includes both graphics, networking, audio and the integration of these subsystems.

CVE's have a classic problem for event-driven systems, namely, that they have multiple event streams. In Unicon, this consists of a TCP chat connection, a UDP connection for most 3D environment updates, and the window system input handling. Performance requirements dictate that these connections be checked continuously; but polling is too expensive. On UNIX/X11 the `select()` is supposed to be good for this, but on MS Windows `select()` takes only network connections, and on both platforms we found it necessary to write our own runtime system support to avoid polling.

6 Related Work

Unicon is related to both 3D collaborative virtual environments and 2D collaborative applications. Both Unicon's 3D environment and 2D collaborative tools owe much to fundamental internet communication technologies such as chat, mailing lists, and especially MUDs and MOOs featuring different rooms for different topics or tasks [20]. Sony's EverQuest (www.everquest.com) popularized multi-user virtual environments that add an immersive first-person 3D graphical world onto a collaboration tool that still largely consists of MUD-like text chat. EverQuest handles thousands of users and works adequately over 28.8K modems. Digital Space Traveler, available from www.digitalspace.com, demonstrates the potential for voice and 3D sound in virtual environments. Adobe Atmosphere, Apple QuickTime VR, There.com and Meet3D are other commercial efforts. These packages and game engines are attractive, but without source

code it is impossible to customize them to integrate CS programming tools such as compilers and editors.

There are many 3D research CVE's, such as MASSIVE (-1/-2/-3)/HIVE [17][6], and DIVE [4]. These systems have often emphasized the avatars at the expense of the work people are trying to collaborate on. Instead of taking a VR-centric view where everything is done in the VR, or the opposite view of VR as just a weak collaboration tool whose main purpose is to augment reality by providing awareness of other users [18], Unicron takes the position of *augmented virtuality*: the VR models real-world places and activities, and 2D tools are integrated into the VR where a person would be turning to a computer or other device (such as a whiteboard) to do some work in the non-VR version of things. This allows for relatively seamless transitions. As much as possible, "integration" of 2D tools means bundling their functionality directly into the CVE, rather than switching to a separate window. For example, the collaborative views of text are provided by tabs in the CVE Window, to minimize the user's cognitive context switching costs, and provide continuity in the overall environment. Text and voice chat controls are uninterrupted by switching between 3D view and code view. See Figure 6. Voice chat buttons are upper left (disable, room-mode, and phone-mode). Text chat in upper right provides context as tabs shift between 3D and 2D main area views. Use of the navigation bar in lower left varies with the current mode.

Some of CVE's and other VR systems are especially interesting because they are open source or otherwise publically available, such as University of Manchester's Maverik [8] (aig.cs.man.ac.uk/maverik/), Planeshift (www.planeshift.it), VR Juggler (www.vrjuggler.org), or Alice [14]. One related CS education project that is using Alice is Saint Joseph University's JABRWOC project, which uses virtual reality as an instructional domain, as opposed to Unicron's goal of creating a virtual community for software development collaborations such as CS distance education. Another related education project is Viras [15], a CVE for education built using Active Worlds (www.activeworlds.com). While it is unclear that Active Worlds is customizable to the extent needed for domain-specific education CVE tasks, it is a tremendous resource for generic CVE construction.

The term "collaborative programming environment" can refer generically to any tool that assists programmers to work in teams, such as the (asynchronous) document revision features found in Microsoft Word, or the set of software tools provided by Source Forge (www.sourceforge.net). The collaborative software design and programming environment we envision for the virtual community is more closely related to fully interactive systems from the field of computer supported cooperative work (CSCW). Similar systems from that domain include Microsoft's NetMeeting, NetEdit [21], RECIPE [19], and others. Among 2D collaborative tools, CROCODILE is especially related, as it provides a virtual academic enterprise rendered as hypertext 2D graphs [13]. The CSCL community is also actively exploring uses of 3D environments as educational tools [9]. We are not aware of any systems that integrate a collaborative programming environment into an immersive 3D virtual environment as is the case for Unicron.

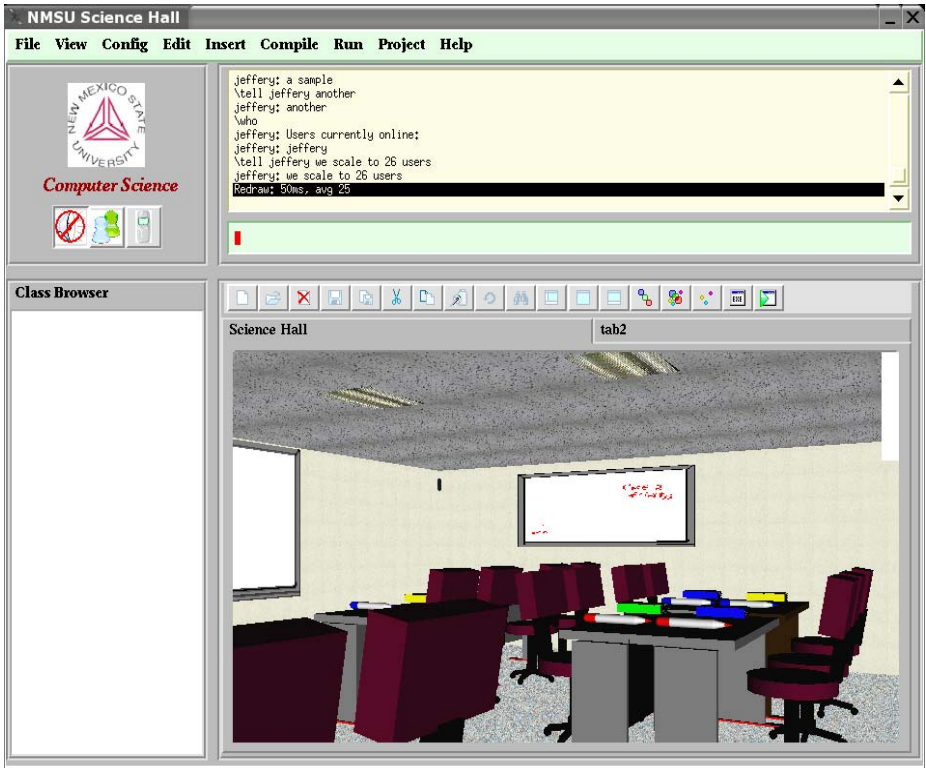


Fig. 6. Integrated view of the IDE and CVE

7 Experience Gained

Quantitative evaluation of Unicorn is ongoing and will be the subject of another paper. Anecdotal experiences learned fall into categories, including (a) experiences building the NSH CVE, (b) discoveries about hardware and software limitations of current mainstream platforms that pertain to CVE applications, and (c) experiences in the integration of domain-specific instructional tools.

The claimed ease of use of the programming language Unicorn used to build the Unicorn framework has been validated by developer experiences, with caveats. A group of average Master's students were able to learn Unicorn and then build, with direction from faculty, both the CVE framework/tools and the working CVE described in this paper. However, the high-level language does not substitute or solve issues of code quality or software engineering, all it does is pose fewer lines of code for instructors or peers to have to code review and debug. Naive student code that works for a few users is different from expert code that runs well for many users.

Many experiences running the CVE have proven useful and occasionally surprising. As expected, performance has been an issue, especially in scaling to

larger numbers of users. However, the relatively slow speed of the very high-level language has not been the issue it was expected to be, because CVEs are heavily I/O bound. Because of this, both the client and the server had to be rewritten the same way: in both cases a simple event-driven I/O multiplexor had to be changed to a batch/step processor that minimized the number of interactions with the operating system by processing all input prior to sending all output for each step. This change alone scaled our system from handling 4 users to handling 26 users on midrange Pentium 4 hardware.

We expected the server written in Unicon to be the bottleneck, but found that the server written in a very high level language has not been a problem; server load has seldom exceeded 10% and is generally far less. The CVE built using this framework remains I/O bound, but the network is not the problem any more: current scalability is limited by clients' graphics rendering code, dominated by VM runtime system OpenGL code written in C. Scaling to hundreds of simultaneous users in the same virtual classroom will involve reducing the graphics rendering cost they impose on each other, for example by abstracting groups of far-away users, and disabling their ability to "distract" each other with individual movements. We found the hard way that wireless internet technologies perform substantially worse than wired technologies for such collaborative tools, corroborating reports from gamers. This is unfortunate since NMSU's electronic classroom is equipped with wall-to-wall wireless laptop machines.

Our experiences integrating domain-specific tools are modest so far. Direct integration of the text editor widget and supporting network code for the collaborative IDE was fairly straightforward. Integrating command-line and textual tools is abstracted by means of pseudo-terminals running inside editor widgets. However, many of the tools we want to integrate involve graphical user interfaces and require either reimplementing as a collaborative tool or else they must be open enough to be extended enough to talk to our framework using a mechanism such as [16] [5].

8 Conclusions and Future Work

On the positive side, in rapid prototyping a collaborative virtual environment with Unicon, we were amazed at the ease with which the smoothly animated first-person view of the environment was developed, initially in 300 lines or so. The multi-user interaction capabilities came together similarly rapidly and simply; our first n-user chat client and server were less than 200 lines of code.

Forseeably but inconveniently, to go from a floor plan and digital camera images to a working prototype virtual environment of an entire building or more requires more sophisticated tools for texture management automation. Texture management (reducing the size of textures, making them tile nicely, etc.) is a seemingly straightforward process that should be automatable by many methods, but most tutorials on the subject advocate texture manipulation by hand using a tool such as the Gimp, or PhotoShop [1]. The Gimp is great, but building a virtual world requires a lot of textures. We implemented some simple automated

tools based on techniques described by Bourke [3] and are working on further refinements.

The Unicron platform is highly suitable for quickly prototyping CVEs that work well for a modest number of simultaneous users, such as the class sizes of 5-30 that we anticipate using it for. We are gradually improving it to scale to larger numbers of users, but we have not applied the kind of hardware or software engineering resources that go into commercial games that handle hundreds or thousands of users.

Our future efforts include: splitting master server responsibilities across multiple machines to improve scalability; refinement and packaging of open source world-building tools; implementing well-known higher performance graphics algorithms to improve rendering; collaboration support for additional CVE domains and other academic disciplines; and performance tuning and fault resistance, especially to network fluctuations.

Acknowledgments

Joe Pfeiffer developed our electronic classroom's whiteboard application, with a Linux device driver written by Mike Wilder. Numerous students assisted with early Unicron development; Wynn Winkler and Nolan Clayton developed noteworthy prototypes. This work was supported in part by the New Mexico Alliance for Minority Participation, and by NSF grants EIA-0220590, EIA-9810732, and DUE-0402572.

References

1. anonymous. *Making Titable & Seamless Textures*. HighPoly3D.com, 2005.
2. D. Bauer, S. Rooney, and P. Scotton. Network infrastructure for massively distributed games. In *Proceedings of the 1st Workshop on Network and System Support for Games, NetGames 2002*, pages 36–43. ACM, April 2002.
3. P. Bourke. *Tiling Textures on the Plane*. Swinburne Centre for Astrophysics and Supercomputing, Australia, 1992.
4. E. Frecon. Dive: Communication architecture and programming model. *IEEE Communications Magazine*, 42(4):Xpp, April 2004.
5. D. Garlan and E. Ilias. Low-cost, adaptable tool integration policies for integrated environments. In *Proceedings of the fourth ACM SIGSOFT symposium on Software development environments*, pages 1–10. ACM, 1990.
6. C. Greenhalgh and D. Snowdon. Hive distribution api. *Technical Report*, 0:5pp, November 1997.
7. R. E. Griswold and M. T. Griswold. *The Icon Programming Language*. Peer to Peer Communications, San Jose CA, 1997.
8. R. J. Hubbold, X. Dongbo, and S. Gibson. Maverik — the manchester virtual environment interface kernel. In *Proceedings of the 3rd Eurographics Workshop on Virtual Environments*, pages x–x+y, February 1996.
9. R. Hmlinen, P. Hkkinen, S. Jrvel, and T. Manninen. Computer-supported collaboration in a scripted 3-d game environment. In *Proceedings of CSCL 2005*, Taipei, Taiwan, 2005.

10. C. Jeffery, A. Dabholkar, K. Tachtvrenidis, and Y. Kim. Programming language support for collaborative virtual environments. In *Proceedings of 18th International Conference on Computer Animation and Social Agents*. CGS, 2005.
11. C. Jeffery, S. Mohamed, R. Pereda, and R. Parlett. *Programming with Unicon*. Unicon Project, unicon.sf.net, 2004.
12. N. Martinez and C. L. Jeffery. Unicon 3D Graphics User's Guide and Reference Manual. *Unicon Technical Report*, 9(a):28pp, July 2003.
13. Y. Miao. *Design and Implementation of a Collaborative Virtual Problem-Based Learning Environment*. M.S. Thesis, Technical University of Darmstadt, 2000.
14. R. Pausch and colleagues. Alice: A rapid prototyping system for 3d graphics. *IEEE Computer Graphics and Applications*, 15(3):8–11, May 1995.
15. E. Prasolova-Frland and M. Divitini. Collaborative virtual environments for supporting learning communities: an experience of use. In *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, pages 58–67. ACM, 2003.
16. S. P. Reiss. Connecting tools using message passing in the field environment. *IEEE Software*, 7(4):57–66, July 1990.
17. D. Roberts and P. Sharkey. Maximising concurrency and scalability in a consistent, causal, distributed virtual reality system, whilst minimising the effect of network delays. In *Proceedings of the IEEE WETICE'97*, pages x–x+y. IEEE, June 1997.
18. M. Robinson, S. Pekkola, J. Korhonen, S. Hujala, T. Toivonen, and M.-J. O. Saari-nen. Extending the limits of collaborative virtual environments. *Collaborative Virtual Environments: Digital Places and Spaces for Interaction*, page XX pp, 2001.
19. H. Shen and C. Sun. Recipe: a prototype for internet-based real-time collaborative programming. In *Proceedings of the Second International Workshop on Collaborative Editing Systems*, 2000.
20. J. Smith. *Basic Information about MUDs and MUDding (MUD FAQ)*. Oklahoma State University Dept. of Math, Oklahoma, OK, 1999.
21. A. Zafer. *NetEdit: A Collaborative Editor*. M.S. Thesis, Virginia Polytechnic Institute, 2001.

Adaptive Distribution Support for Co-authored Documents on the Web

Sonia Mendoza¹, Dominique Decouchant¹, Alberto L. Morán²,
Ana María Martínez Enríquez³, and Jesus Favela⁴

¹ Laboratoire “Logiciels Systèmes, Réseaux”, Grenoble, France
{Sonia.Mendoza, Dominique.Decouchant}@imag.fr

² Facultad de Ciencias, UABC, Ensenada, B.C., Mexico
almys@uabc.mx

³ Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, D.F., Mexico
ammartin@cinvestav.mx

⁴ Departamento de Ciencias de la Computación, CICESE, Ensenada, B.C., Mexico
favela@cicese.mx

Abstract. In order to facilitate and improve collaboration among co-authors, working in the Web environment, documents must be made seamlessly available to them. Web documents may contain multimedia resources, whose management raises important issues due to the constraints and limits imposed by Web technology. This paper proposes an adaptive support for distributing shared Web documents and multimedia resources across authoring group sites. Our goal is to provide an efficient use of costly Web resources. Distribution is based on the current arrangement of the participating sites, the roles granted to the co-authors and the site capabilities. We formalize key concepts to ensure that system's properties are fulfilled under the specified conditions and to characterize distribution at a given moment. The proposed support has been integrated into the PIÑAS platform, which allows an authoring group to collaboratively and consistently produce shared Web documents.

1 Introduction

Distribution concerns the way in which the shared data objects of a groupware application are distributed across the cooperating sites. Previous efforts (e.g. [4]) identified flexible distribution support as an important issue in groupware design and development. However, most platforms only support one distribution strategy that is applied to all data objects of a single application. Some platforms (e.g. [9] [11]) take flexibility into account for their distribution support, but they are mainly focalized on supporting synchronous work. Consequently, they do not address other important issues (e.g. persistency) that arise from the need to support disconnected or nomadic work.

In spite of the progress in groupware design and development, only few applications are globally used. Being the most used application of the Internet, the Web offers a potential as a viable technology for groupware development, deployment and evaluation [1]. Current research efforts on the Web (e.g. [6]) focus on providing asyn-

chronous support for document co-authoring. However, these solutions follow a hybrid distribution strategy (e.g. centralized data objects and replicated programs) which makes them difficult to ensure document availability within an unreliable environment. Moreover, the Web technology also imposes constraints raising important issues of distributed document management (e.g. client site and user identification is not yet integrated into the access mechanism).

In order to provide an efficient use of costly Web resources, we propose an adaptive support for distributing them across co-author sites. Our support takes into account the current arrangement of the involved sites, the roles granted to the co-authors and the site storage capabilities. We have integrated this dedicated management service into the PIÑAS platform [2], which is suitable for supporting collaborative authoring on the Web. Such autonomous support allows the application developer to implement different distribution strategies, which are not only applied to a collection of applications, but also to individual data objects of a single application.

This paper is organized as follows. After discussing related work in Section 2, we describe our approach for distributing shared Web entities in Section 3. Particularly, we present the data model of the Web document entity and its components. Then, we introduce the principles to arrange co-author's sites. Later, we describe how Web documents and their resources are shared among co-authors and how they are distributed across their sites. Finally, Section 4 presents conclusions and future work.

2 Related Work

Most groupware platforms support one distribution strategy, which applies to the whole application. For instance, Rendezvous [7] provides a centralized strategy, while GroupKit [13] is based on a replicated one. Suite [3] offers a hybrid strategy, which centralizes the shared data objects and replicates the application programs. Only few platforms, such as GEN [11] and DreamObjects [9], provide flexible support for distributing shared data objects, which allows the application developer to implement different distribution strategies.

By default, GEN offers implementations for centralized and replicated strategies. A shared data object can be represented as a proxy or replica in order to respectively implement a centralized or replicated data object. By means of the open implementation technique [8], this platform allows the developer to specify how a shared data object is distributed and maintained consistent.

DreamObjects provides replicated, asymmetric and adaptive strategies. In contrast to the centralized strategy, the asymmetric strategy does not require a well-known server. The adaptive strategy dynamically changes the distribution characteristics of shared data according to the user working style. By applying the substitution principle, this platform allows the developer to specify the distribution and consistency strategies of a shared data object at runtime. Although GEN and DreamObjects target on providing several distribution strategies, they are mainly concerned with synchronous work support. Consequently, they do not address other important issues (e.g. persistency and access control) that arise from the need to support disconnected work.

Additionally, document managers like Lotus Notes [14], Bayou [5] and DOORS [12] support asynchronous work. However, even when these systems take persistency support into account, they are constrained to follow a replicated strategy. Except for Lotus Notes, none of the previous platforms is well adapted for the Web.

Current research efforts, such as WebDAV [6] and BSCW [1], focus on providing functions for accessing and publishing Web documents on remote servers. Users are able to lock documents, to get and modify them, and finally to send them back to the server. However, these functions are only designed for a hybrid strategy, which makes them difficult to guarantee document availability within an unreliable environment (e.g. Internet).

We aim at going a step further by proving an efficient use of shared multimedia resources. Our platform offers an adaptive distribution support which takes into account important issues, such as storage capabilities, authoring roles and current arrangement of the cooperating sites. These issues arise from the need to support collaborative work in disconnected and nomadic mode on the Web environment. Nevertheless, they have not been previously addressed by other platforms in a comprehensive way.

3 Distribution of Shared Web Entities

In this section, we formally define the underlying concepts and relations of our approach for distributing shared Web entities¹. First, we present the data model of the Web document entity and its components. Then, we introduce the underlying principles to arrange co-author's sites and characterize the distribution of a document across co-author sites. Afterwards, we present the phases of the distribution protocol. Finally, we describe how Web documents and their resources are shared among co-authors and how they are distributed across their sites.

3.1 Data Model of Shared Web Entities

A Web document entity consists of one or more fragment entities. In turn, a fragment entity can refer to multimedia resource entities. A document has to hold at least one fragment, while a fragment may not include any resources. A fragment can belong to a document only once, and similarly a resource can be a member of a fragment only once. The *Document-Fragment* and *Fragment-Resource* relations are refined as follows. Let D denote the set of Web documents and H represent the set of fragments. Allow F to designate an injection from D to $(\mathcal{P}(H) \setminus \{\emptyset\})$ such that, for every document $d \in D$, the $F(d)$ set contains the component fragments. On the other hand, let I be the set of multimedia resources. The E function, such that $E \in H \rightarrow \mathcal{P}(I)$, maps a fragment to a set of resources. Therefore, for every fragment $f \in H$, the $E(f)$ set holds the resources referred by the fragment. The F and E mappings are defined using different types of binary relations as two different documents can not maintain the same total sequence of fragments. However, two distinct fragments may contain common resources.

¹ Refer to appendix A for the formalization notation.

An entity consists of a set of properties and a body (see Fig. 1). In turn, a body holds a sequence of hypertext links to other entities. A document body maintains a HTML, XHTML or XML description about a sequence of fragment references. Similarly, a fragment body holds a description about a sequence of resource references. At creation time, a document just consists of one fragment, whose body can be initialized with a possible empty description. A resource body can contain MIME types, e.g. a "GIF/JPEG" image, a "basic/tone" audio or a "MPEG/pointer" video, etc.

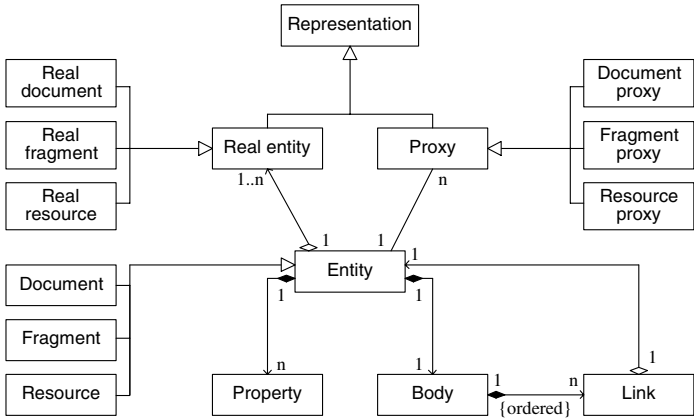


Fig. 1. Representations of the document, fragment and resource entities

The data model supports two types of representations at the local site, proxy and real entity, which permit the application developer to respectively implement centralized or replicated data objects. Representing an entity as proxy allows to remotely refer to it. Conversely, creating a real entity allows it to be locally referenced. The main difference between these representations resides on the entity body: a real entity actually holds a body, while a proxy just comprises a reference to the body.

3.2 Arrangement of Co-author Sites

We distinguish a working site from a storage site to carry out distribution of co-authored documents [10]. A working site is an end-user computer, running PIÑAS-based applications and possibly standard Web applications, where an author consults and modifies documents. A storage site is a Web server host, running PIÑAS-based services, where documents are stored. The *Author-Storage Site* and *Author-Working Site* relations are defined as follows. Let A be the set of authors, SS represent the set of storage sites hosting PIÑAS services and WS denote the set of working sites running PIÑAS applications. The S function, such that $S \in A \rightarrow (\mathcal{P}(SS) \setminus \{\emptyset\})$, maps an author to a non-empty set of storage sites. Similarly, the W function, such that $W \in A \rightarrow (\mathcal{P}(WS) \setminus \{\emptyset\})$, maps an author to a non-empty set of working sites. Hence, for every

author $a \in A$, the $S(a)$ and $W(a)$ sets respectively contain his available storage and working sites. According to the S and W functions, several authors can share a working and/or storage site. Moreover, a single site can act both as a storage and a working site in order to allow collaborative autonomous work in a degraded mode.

The multi-site author principle stated above allows authors to work in nomadic mode. An author can establish alternative connections to/from different pre-specified sites as he/she moves. Thus, he/she is able to transparently work on his/her documents from different sites, and dynamically retrieve his/her working environment at any moment. Based on this principle, we specify a first level of nomadic work features by relating a working site to a set of storage sites. Formally, the T_a function, such that $T_a \in W(a) \rightarrow (\mathcal{P}(S(a)) \setminus \{\emptyset\})$, maps a working site to a non-empty set of storage sites. Therefore, for every working site $w \in W(a)$, the $T_a(w)$ set holds the storage sites that can handle requests from author a working at the w site.

The following scenario illustrates these concepts (see Fig. 2): authors, *Ray*, *Beth*, *Kurt* and *Saul*, dispersed around the world, have several working and storage sites at their disposal. *Ray* can produce/consult documents on the *progreso.mx*, *koln.de* and *versailles.fr* working sites and download/upload them from the *cancun.mx*, *bavaria.de* and *louvre.fr* storage sites. *Ray*'s sites are located in different countries, i.e. *Mexico*, *Germany* and *France*, to allow him to work in nomadic mode. Based on his current physical location, e.g. *progreso.mx*, he can access documents from a close site, e.g. *cancun.mx*, using a more reliable environment.

According to the W , S and T_a functions, *Ray*'s working and storage sites and the relations between these sites are defined as follows:

- $W(\text{Ray}) = \{\textit{progreso.mx}, \textit{koln.de}, \textit{versailles.fr}\}$
- $S(\text{Ray}) = \{\textit{cancun.mx}, \textit{bavaria.de}, \textit{louvre.fr}\}$
- $T_{\text{Ray}} = \{(\textit{progreso.mx}, \textit{cancun.mx}), (\textit{koln.de}, \textit{bavaria.de}), (\textit{versailles.fr}, \textit{louvre.fr})\}$

Such an arrangement provides *Ray* with high availability of his documents by distributing them across several sites. As depicted by the gray shadow, he can rely on an infrastructure that automatically distributes documents to make them available during a working session and to persistently store them. In addition to these features, the underlying infrastructure ensures that *Ray*'s documents will be updated and kept consistent, even in an unreliable environment (e.g. Internet). Moreover, different authors' working sites, e.g. *Ray*'s site, *versailles.fr*, and *Beth*'s site, *orsay.fr*, can download/upload documents from a common storage site, e.g. *louvre.fr*. Thus, several authors located in relatively close places (e.g. in the same LAN) can share site capabilities (e.g. storage). Finally, the *tulum.mx* site can simultaneously act as working and storage site to allow *Saul* to work in an autonomous way.

An author can have several Web documents, which are organized into a documentation base. We formally define the *Author-Document* relation as follows. The B function, such that $B \in A \rightarrow \mathcal{P}(D)$, maps an author to a set of documents. Hence, for every author $a \in A$, the $B(a)$ set contains his documents. According to the B function, a documentation base may be empty at a given moment (e.g. at author's registration time). Moreover, different author's bases can contain common documents in order to allow document sharing. Private documents are stored in the owner author's sites, while shared documents also are saved in those of his colleagues.

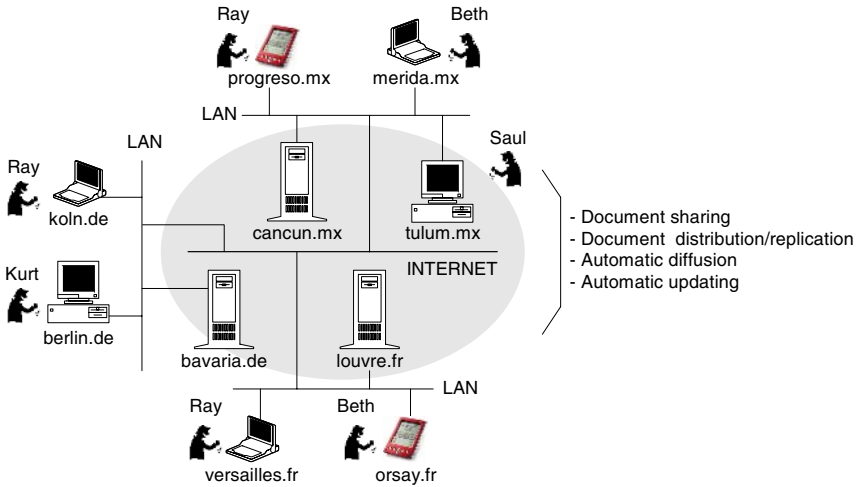


Fig. 2. A typical arrangement of co-author sites

3.3 Characterization of Document Distribution

A document is private and belongs to its creator at creation time. However, a document could be distributed across several sites, in order to make it available during a working session and to persistently store it in a reliable way. To formally define the distribution scope of a document, let DR and DP respectively denote the sets of document replica and proxy representations. As we previously defined, D designates the set of documents. Allow R to be a bijection from D to DR such that, for every document $d \in D$, the $R(d)$ element represents the document replica. Similarly, let P denote a bijection from D to DP such that, for every document $d \in D$, the $P(d)$ element represents the document proxy.

On the other hand, let a and d respectively denote an author and a document, such that $a \in A$ and $d \in D$. Also, suppose that $d \in B(a)$, i.e. author a creates the d document. In this way, the d document belongs to the $B(a)$ documentation base available to author a . As the document owner, author a has all the access rights on the d document, while any other author a' , such that $a' \in A$ and $a' \neq a$, has no access rights. Finally, consider that author a has the $S(a)$ set of storage sites and the $W(a)$ set of working sites at his disposal.

Based on the previous definitions, a snapshot of the document after distribution can be expressed as follows. The $\{R(d)\} \times S(a)$ Cartesian product denotes the distribution scope of the d document across the storage sites of author a . The $(R(d), s)$ pair, such that $(R(d), s) \in \{R(d)\} \times S(a)$, indicates a $R(d)$ replica that is persistently created at the storage site $s \in S(a)$. Similarly, the $\{R(d)\} \times W(a)$ Cartesian product denotes the distribution scope of the d document across the working sites of author a . The $(R(d), w)$ pair, such that $(R(d), w) \in \{R(d)\} \times W(a)$, designates a $R(d)$ copy that is temporarily created at the working site $w \in W(a)$.

To exemplify these expressions, allow us to consider a scenario in which Ray produces the *report* document on the *progreso.mx* working site. As mentioned earlier,

Ray has the *cancun.mx*, *bavaria.de* and *louvre.fr* storage sites and the *progreso.mx*, *koln.de* and *versailles.fr* working sites at his disposal. According to the $\{R(d)\} \times S(a)$ and $\{R(d)\} \times W(a)$ products, we define the distribution scope of the *report* document across *Ray*'s sites as follows²:

- $\{R(\textit{paper})\} \times S(\textit{Ray}) = \{(\mathbf{R}(\textit{paper}), \textit{cancun.mx}), (\mathbf{R}(\textit{paper}), \textit{louvre.fr}),$
 $(\mathbf{R}(\textit{paper}), \textit{bavaria.de})\}$
- $\{R(\textit{paper})\} \times W(\textit{Ray}) = \{(\mathbf{R}(\textit{paper}), \textit{progreso.mx}), (R(\textit{paper}), \textit{versailles.fr}),$
 $(R(\textit{paper}), \textit{koln.de})\}$

Thus, a $R(\textit{report})$ replica is permanently created at each *Ray*'s storage site. Such an organization provides *Ray* with high availability of his documents. Additionally, a working copy may be temporarily created in cache at his current working site, e.g. *progreso.mx*, in order to allow *Ray* starting management or authoring tasks. In this way, rapid feedback at the user interface level is achieved. Working copies also can be loaded to the *koln.de* and *versailles.fr* working sites, whenever *Ray* aims at consulting or modifying the document contents.

3.4 Phases of the Distribution Protocol

Identification and location of co-author sites are important issues when designing a distribution protocol. To overcome them, we use the *author definition* entity [2]. For every author a , such that $a \in A$, it holds the $S(a)$ and $W(a)$ sets of storage and working sites available to him/her and the T_a set of relations among these sites. The *author definition* for author a is replicated in all sites contained in the $S(a)$ and $W(a)$ sets. Following the evolution of this entity (i.e. site creation, modification or destruction), the replicas are steadily updated.

Each time an author starts a session at one of his/her working sites, the distribution support determines from his/her local *author definition* the storage sites from where documents can be downloaded. During this connection phase, the *author definition* can be updated at the current working site. Also, the distribution support relies on a two phase protocol to carry out document distribution across co-author sites. The first phase concerns the interaction between a working site and a storage site, while the second phase involves the interaction among storage sites. Formally, we describe the phases of the distribution protocol as follows.

Let a and d respectively denote an author and a document, such that $a \in A$ and $d \in D$. Consider that $d \in B(a)$, i.e. author a creates the d document. Let w and s respectively represent a working site and a storage site available to author a , such that $w \in W(a)$ and $s \in S(a)$. Finally, suppose that $s \in T_a(w)$, i.e. the s site can handle requests from author a working at the w site.

Phase 1. Request for creating a document. An application at the w site sends the system at the s site a request for creating the d document. Then, the system creates a $R(d)$ replica of the d document at the s site. Before starting Phase 2 to distribute the d document across the $S(a) \setminus \{s\}$ storage sites of author a , the system at the s site sends an acknowledgement to the application at the w site. If needed, a working

² Bold caption means actual distribution while normal caption signifies potential distribution.

copy may be temporarily created at the w site. In this way, author a is allowed to immediately work on the d document without concerning him/her about distribution issues.

Phase 2. Distribution of a document entity. The system at the s site takes charge of distributing the d document across the involved sites. Thus, it obtains two pieces of information from the $R(d)$ replica at the s site. The first piece consists of the $S(a)\setminus\{s\}$ storage sites the d document would be distributed to. The second piece holds the type of representation (i.e. replica or proxy) to be created in those sites. By default, a temporal working copy (i.e. valid for a working session) is created at a working site, while either a persistent replica or proxy can be created at a storage site. Using these information, the system at the s site asks its pairs at the $S(a)\setminus\{s\}$ sites to create a replica of the d document. Then, they create a $R(d)$ replica at their respective sites and send an acknowledgement to the system at the s site.

Even if every storage site can carry out the distribution of document state changes, the $R(d)$ persistent replica at the s site does not have to explicitly refer to the other persistent replicas $\{R(d)\}\times(S(a)\setminus\{s\})$. Similarly, as the w working site is related to the $T_a(w)$ set of storage sites, the temporal working copy at the w site does not have to keep references to the persistent replicas $\{R(d)\}\times(S(a)\cap T_a(w))$. In contrast, the distribution protocol relies on the notion of *author definition* to locate both working copies and persistent replicas. In this way, every copy or replica points to the *author definition* held in the local site. Furthermore, every w' working site, such that $w' \in W(a)$ and $w' \neq w$, will know about the new document d when author a sets-up an application at this site. At set-up time, the application can ask the system at the site $s' \in T_a(w')$ information about the d document and create a working copy at the w' site if needed.

To illustrate the distribution protocol phases, let us use a scenario concerning the creation of the *report* document previously mentioned. In Phase 1, author *Ray* working at the *progreso.mx* site asks the groupware application to create the *report* document (see Fig. 3). Subsequently, the application inspects the local author definition to identify the storage site(s) that can handle its requests (step 1). According to the T_{Ray} function, the *progreso.mx* working site is related to the *cancun.mx* storage site. In this way, the application forwards the request to the system at the *cancun.mx* site (step 2). In turn, it creates a replica ($R(report)$, *cancun.mx*) and sends an acknowledgment to the application (step 3). Then, a working copy is temporarily created in *Ray*'s cache to allow him immediately managing or working on the document (step 4). In Phase 2, the system at the *cancun.mx* site is responsible for distributing the *report* document. Thus, it first examines the local author definition, belonging to *Ray*, in order to determine the storage sites the document is to be distributed to (step 5). According to the S function, the $S(Ray)$ set contains the storage sites available to *Ray*. The system at the *cancun.mx* site asks its peers at the *bavaria.de* and *louvre.fr* sites to create a replica of the *report* document (step 6). Finally, they create a replica at their corresponding sites and then send an acknowledgment to their peer at the *cancun.mx* site (step 7).

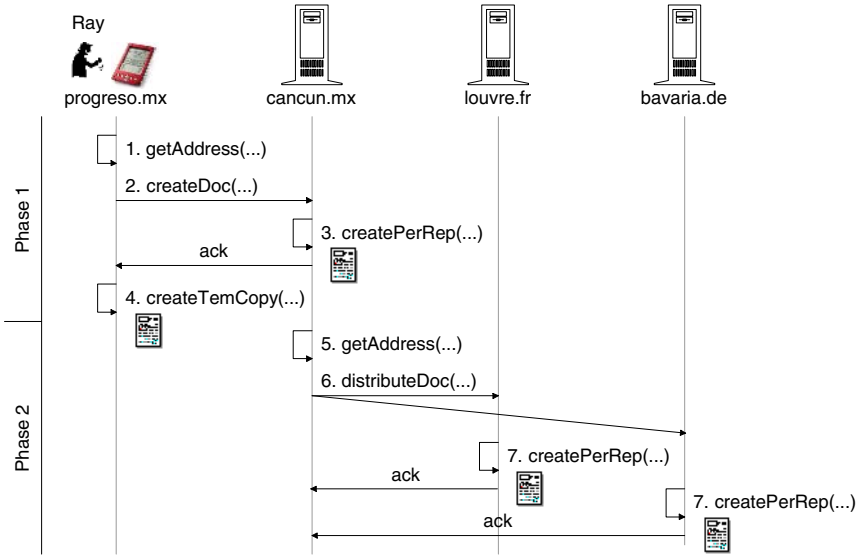


Fig. 3. Phases of the distribution protocol

The replica ($R(report)$, *louvre.fr*) does not have to keep an explicit reference to the replica ($R(report)$, *bavaria.de*) and vice-versa. Similarly, they do not need to know about the replica ($R(report)$, *cancun.mx*) and vice-versa, as the local author definition holds the set of storage sites where persistent replicas can be maintained. Like persistent replicas, the temporal working copy at the *progreso.mx* site refers to the local author definition, instead of holding a reference to the persistent replica at the *cancun.mx* site. In this way, if the network connection between the *progreso.mx* and *cancun.mx* sites fails or the *cancun.mx* site fails, then the *louvre.fr* site can handle requests concerning the *report* document.

3.5 Sharing a Web Document Entity

In order to allow several authors to concurrently work on a shared document, our approach relies on document fragmentation and role granting. A fragment can be recursively divided to form new fragments, according to the document logical structure. The number of fragments can vary from one document to another, depending on the requirements of the authoring group. In this way, authors can access document fragments in accordance with their access rights. AllianceWeb [2] is a PIÑAS-based application that supports both document fragmentation and access rights granting by means of roles.

To share a document with other authors, the document owner has to form an authoring group. Formally, group composition can be defined as follows. Let suppose that author $a \in A$ decides to share his document $d \in B(a)$ with some of his colleagues. The C_a set contains the colleagues of author a , such that $C_a \subseteq (A \setminus \{a\})$. Thus, author a can form the authoring group G from his group of colleagues C_a . He can be also included in the authoring group. Hence, the expression $G \setminus C_a = \{a\}$ defines the relation

between the authoring group G , including author a , and his group of colleagues C_a . Otherwise, the expression $G \subseteq C_a$ defines such a relation.

Allow to suppose that author $a \in G$ breaks up document $d \in B(a)$ into multiple fragments. Let R denote the set of roles. Author a grants a role $r \in R$ on every fragment $f \in F(d)$ to the members of the authoring group G . The MI_d function, such that $MI_d \in F(d) \rightarrow \mathbb{P}(G \times R)$, maps a fragment to a set of *author-role* pairs. Therefore, for every f fragment, the $MI_d(f)$ set contains *author-role* pairs. The (a, r) pair, such that $(a, r) \in MI_d(f)$, indicates that author a can play the r role on the f fragment.

To exemplify these concepts, suppose that *Beth*, *Kurt* and *Saul* are *Ray*'s colleagues. As *Ray* decides to share the *report* document with *Beth* and *Saul*, he creates the corresponding authoring group. Then, he breaks up the *report* document in three fragments, f_1 , f_2 and f_3 , and grants roles to the group members. According to the M_d function, *Ray* defines the following access control sets for each fragment of the *report* document:

- $M_{report}(f_1) = \{(Beth, reader), (Saul, reader), (Ray, writer)\}$
- $M_{report}(f_2) = \{(Beth, reader), (Saul, writer), (Ray, reader)\}$
- $M_{report}(f_3) = \{(Beth, reviewer), (Saul, null), (Ray, writer)\}$

Beth and *Saul* are allowed to read fragment f_1 and *Ray* may write it. Concerning fragment f_2 , *Saul* is permitted to write it, while *Beth* and *Ray* may read it. Finally, *Saul* does not have any access rights on fragment f_3 , while *Beth* and *Ray* may review and write it, respectively.

As mentioned in Section 3.1, a fragment can refer to multimedia resources by means of hyperlinks. Thus, the referenced resources can by default inherit the access control definition of the referencing fragment. However, the access rights on its resources can be redefined in order to increase the co-authoring concurrency on a fragment. Let us formally define these concepts as follows. Let $E(f)$ denote the set of multimedia resources referenced by the f fragment. The N_f function, such that $N_f \in E(f) \rightarrow \{M_d(f)\}$, maps a resource to the access control set $M_d(f)$, which contains the *author-role* pairs for the f fragment. Hence, the $N_f(e)$ set denotes the access control definition for every resource $e \in E(f)$. To illustrate these concepts, suppose that some fragments of the *report* document refer to multimedia resources. Fragment f_1 references the *image* and *audio* resources, while fragment f_3 includes a *video* resource. As for fragment f_2 , it does not refer to any. In this way, the *image* and *audio* resources inherit the access control definition of f_1 (i.e. the $M_{report}(f_1)$ set) while the *video* inherits that of f_3 (i.e. the $M_{report}(f_3)$ set). In accordance with the N_f function, the access control definitions of these multimedia resources are defined as follows:

- $N_{f_1}(image) = \{(Beth, reader), (Saul, reader), (Ray, writer)\}$
- $N_{f_1}(audio) = \{(Beth, reader), (Saul, reader), (Ray, writer)\}$
- $N_{f_3}(video) = \{(Beth, reviewer), (Saul, null), (Ray, writer)\}$

According to the access control sets, $M_{report}(f_1)$, $N_{f_1}(image)$ and $N_{f_1}(audio)$, only *Ray* is authorized to modify the f_1 fragment as well as the *image* and *audio* resources, while *Beth* and *Saul* may read and visualize them. As for the f_3 fragment and the *video* resource, *Beth* and *Ray* can respectively review and modify them, while *Saul* does not have any access rights in accordance to the access control sets, $M_{report}(f_3)$ and

$N_{j\beta}(\text{video})$. However, authoring concurrency on the f_1 fragment can be increased by redefining the sets $N_{j\beta}(\text{image})$ and $N_{j\beta}(\text{audio})$. For instance, *Ray* still might modify the f_1 fragment, while *Beth* would be allowed to change the *image* resource and *Saul* might modify the *audio* resource. Even more, *Ray* might not have any access rights on the *audio* resource.

3.6 Distribution of Multimedia Resources

An initialization phase has to be performed among the group members, before starting redistribution of the *report* document across the group sites. During this phase, the author definition of each member is exchanged and persistently stored in the working and storage sites of the group members. Replicas of *Ray*'s author definition are distributed across the sites of *Beth* and *Saul* and vice-versa. In this way, each member can have an alternative connection to another site and work with shared and distributed documents in a dynamic and transparent way.

As soon as distribution of the *report* document is finished, it is available at *Ray*'s storage sites (i.e. *cancun.mx*, *bavaria.de* and *louvre.fr*) as well as at those of *Beth* (i.e. *cancun.mx* and *louvre.fr*) and *Saul* (i.e. *cancun.mx* and *tulum.mx*). Although co-authors share common storage sites (i.e. *cancun.mx*) just one replica of the *report* document is maintained in those sites. In order to collaboratively co-author the *report* document, a working copy is stored in *Ray*'s cache at his working sites (i.e. *progreso.mx*, *koln.de* and *versailles.fr*) and at those sites available to *Beth* (i.e. *merida.mx* and *orsay.fr*) and *Saul* (i.e. *tulum.mx*).

Working copies and replicas of multimedia resources also are distributed across the group storage and working sites. However, proxies can be created at those sites in order to save local storage space. Decisions about creating either a replica or a proxy depend on the access rights granted to the corresponding author on the resource. Let us formally define these concepts as follows. As previously mentioned, the S and W functions respectively map an author to a set of storage sites and to a set of working sites, while the G set denotes the authoring group. Therefore, the $S[G]$ and $W[G]$ sets respectively represent the group's storage and working sites from/to where the d document can be loaded/distributed. On the other hand, let SP_e and SR_e respectively denote two subsets of the group's storage sites $S[G]$, where the e resource can be created as proxy and as replica depending on the access control set $N_j(e)$. If pair $(a, r) \in N_j(e)$ specifies that author $a \in G$ is allowed to play a role $r \in R$, which completely denies access to the e resource, then a proxy will be distributed across his storage sites $S(a)$. Otherwise, a replica will be created at those sites.

In order to determine the distribution scope of proxies and replicas, let us suppose that all access rights on the e resource are disabled for the r role. Expressions (1) and (2) define the sets of storage sites, SP_e and SR_e , where proxies and replicas of the e resource respectively can be created. Further, let us to consider that two authors share a common storage site and one of them has access rights on a given resource, while the other does not. Such situation causes a representation ambiguity, as the resource must be distributed across the storage sites both in replica and proxy forms. To solve the ambiguity, a replica rather than a proxy will be created to ensure availability:

$$SP_e = S [G \cap N_f(e)^{-1} [\{r\}]] \tag{1}$$

$$SR_e = S [G \cap N_f(e)^{-1} [R \setminus \{r\}]] \tag{2}$$

To illustrate the expressions (1) and (2), let us use a scenario (see Fig. 4). Concerning the *image* resource, replicas are distributed across the group storage sites (i.e. *cancun.mx*, *tulum.mx*, *louvre.fr* and *bavaria.de*), as the group members have the right to access them. Conversely, proxies for the *audio* resource would be distributed across *Ray*'s sites (i.e. *cancun.mx*, *bavaria.de* and *louvre.fr*) because he does not have any access rights. However, as *Beth* shares the *cancun.mx* and *louvre.fr* storage sites with *Ray* and she does have access rights on the *audio* resource, replicas rather than proxies are created at those sites. Similarly, proxies for the *video* resource would be distributed across *Saul*'s storage sites (i.e. *cancun.mx* and *tulum.mx*) as he does not have any access rights. Nevertheless, as *Beth* and *Ray* share the *cancun.mx* storage site with *Saul* and they do have access rights on the *video* resource, a replica rather than a proxy is created at this site. The distribution scope of the *image*, *audio* and *video* resources across the group's site can be determined as follows:

- $SP_{image} = \{\emptyset\}$
- $SR_{image} = \{cancun.mx, tulum.mx, louvre.fr, bavaria.de\}$
- $SP_{audio} = \{bavaria.de\}$
- $SR_{audio} = \{cancun.mx, tulum.mx, louvre.fr\}$
- $SP_{video} = \{tulum.mx\}$
- $SR_{video} = \{cancun.mx, louvre.fr, bavaria.de\}$

By distributing resources according to the author access rights, replicas of possible costly multimedia resources are created at the storage sites, whose users have access rights on them. Otherwise, a proxy is persistently kept in order to save storage space.

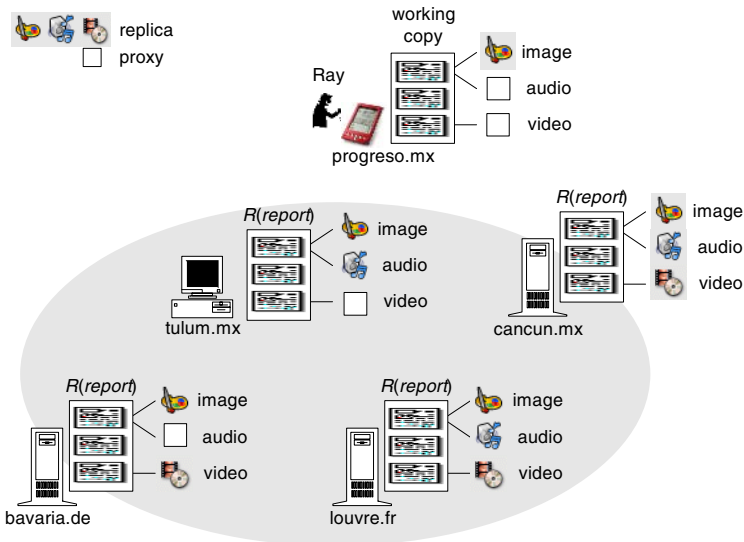


Fig. 4. Distribution of shared multimedia resources

Additionally, working copies or proxies for resources can be distributed across working sites. A copy can be temporarily created at his current working site, if the author rights authorize him/her to access the resource. In this way, the author can be provided with rapid feedback at the user interface. Instead, a proxy is created. Moreover, when an author (e.g. *Ray*) has access rights on costly resources (e.g. *video*) but he can not actually access them due to device constraints (e.g. a PDA with limited storage space), temporal proxies rather than copies are created in the cache at his current working site (e.g. *progreso.mx*), even if replicas are held in his storage sites (e.g. *cancun.mx*, *bavaria.de* and *louvre.fr*).

3.7 The PIÑAS Platform

The PIÑAS platform [2] provides support to collaboratively and consistently produce shared documents in the Web environment. It is based on a client/server architecture, whose server side consists of three main layers (see bottom block of Fig. 5). The bottom layer offers a set of HTTP-based proprietary communication functions. The middle layer contains the collaborative entities offered by the platform, as well as their management services. And finally, the top layer provides a synchronous API (*Application Program Interface*) to access these services.

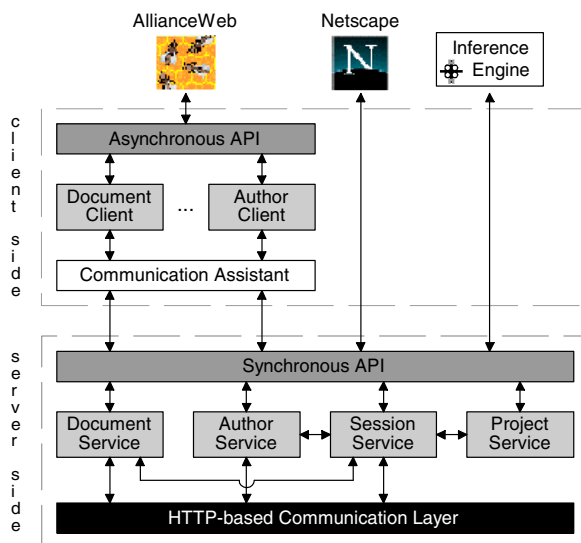


Fig. 5. Architecture of the PIÑAS platform

To support collaboration among co-authors, the PIÑAS platform manages different types of entities, such as *Author*, *Document*, *Project*, *Application* and *Session*. *Author* entity manages information allowing to identify a collaborator at the system and social levels. Author identification allows to grant access rights to a collaborator, to determine his/her contributions and to coordinate his/her actions. *Document* entity

represents a Web document shared by an authoring group. It can contain other entities such as fragments and multimedia resources. *Project* entity maintains the *Author* and *Document* entities involved in a collaborative endeavor. *Application* entity allows authors to know the status (e.g. presence, availability, etc.) of other authors/documents and to collaboratively work on a document. Finally, *Session* entity handles information about the *Author* and *Document* entities (i.e. Project entity) associated to an *Application* entity.

The top block of Figure 5 separates the applications interacting with the platform through the synchronous API (server side) from those interacting by means of the asynchronous one (client side). Applications use the asynchronous API when they lack of communication support to interact with the platform services. The assistant module carries out all communications with the synchronous API, while the entity modules at the client side accomplish function calls of the corresponding modules at the server side. The PIÑAS platform supports three types of applications: PIÑAS-based groupware applications (e.g. AllianceWeb), standard Web browsers (e.g. Netscape) and dedicated applications (e.g. inference engine tool). Particularly, standard Web browsers have limited access to the platform entities and services.

4 Conclusions and Future Work

Being the most used work environment of the Internet, the Web provides a potential as a viable technology for groupware development, deployment and evaluation. In order to support an efficient use of costly resources, an adaptive support for managing them is required. Most Web-based platforms offers only a distribution strategy which applies to all objects of the groupware application. Out of the Web environment, only few platforms allowing synchronous work propose flexible distribution supports. To overcome these limitations, we proposed an adaptive approach for distributing shared Web documents and resources across the involved sites. Distribution is carried out according to the current arrangement of co-author sites, the site storage capabilities and the co-author roles. Documents and resources can be represented both in replica and proxy form to achieve an efficient use of costly resources. In order to support collaborative work in disconnected mode, our approach is based on the multi-site author principle, which relates an author with several working and storage sites. Additionally, associations among working and storage sites can be established to provide a first level support for nomadic work. In this way, authors can dynamically and transparently transfer their working environment from one place to another.

There are still some open issues. We aim at providing flexible support for managing consistency of shared document bases. Our current approach allows two or more authors to potentially play roles that authorize them to concurrently modify a single resource contents. For controlling concurrency, a pessimistic strategy based on one master and many slave copies is used. However, when resources are shared among several documents produced by different authoring groups, this strategy seems restrictive. We also aim to support automatic and dynamic determination of neighboring, reliable and efficient storage sites. Thus, a working site can get documents from the closest, more efficient and reliable storage site of those sites at its disposal. Another

important issue concerns the adaptive support for updating and propagating a document state changes, by automatic integration of all contributions.

Acknowledgments. This work is supported by ECOS/ANUIES project M03M01.

References

1. W. Applelt, WWW Based Collaboration with the BSCW System, In Proc. of the 26th Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'99), Springer, LNCS 1725, pp. 66-78, Milovy (Czech Republic), Nov. 26-Dec.4 1999.
2. D. Decouchant, J. Favela, and A. M. Martínez, PIÑAS: A Middleware for Web Distributed Cooperative Authoring, In Proc. of the 2001 Symposium on Applications and the Internet (SAINT'2001), pp. 187-194, IEEE Computer Society, San Diego, CA (USA), January 2001.
3. P. Dewan and R. Choudhary, A High-Level and Flexible Framework for Implementing Multi-user Interfaces, ACM Transactions on Information Systems, 10(4):345-380, 1992.
4. P. Dourish, Using Metalevel Techniques in a Flexible Toolkit for CSCW applications, ACM Transactions on Computer-Human Interaction, 5(2):109-155, June 1998.
5. W. K. Edwards, E. D. Mynatt, K. Petersen, M. J. Spreitzer, D. B. Terry and M. M. Theimer, Designing and Implementing Asynchronous Collaborative Applications with Bayou, In Proc. of the Tenth ACM Symposium on User Interface Software and Technology (UIST'97), Alberta (Canada), October 1997.
6. Y. Y. Goland, E. J. Whitehead Jr., A. Faizi, S. R. Carter and D. Gensen, HTTP Extensions for Distributed Authoring – WebDAV, RFC 2518, February 1999.
7. R. D. Hill, T. Brinck, S. L. Rohall, J. F. Patterson and W. Wilne, The Rendezvous architecture and language for constructing multi-user applications, ACM Transactions on Computer-Human Interaction, 1(2):81-125, June 1994.
8. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J. M. Loingtier, J. Irwin, Aspect-Oriented Programming, In Proc. of the European Conference on Object-Oriented Programming (ECOOP'97), Springer-Verlag, LNCS 1241, 1997.
9. S. Lukosch, Adaptive and Transparent Data Distribution Support for Synchronous Groupware, In Proc. of the 8th International Workshop on Groupware (CRIWG'2002), Springer-Verlag, LNCS 2440, pp. 255-274, La Serena (Chile), September 2002.
10. S. Mendoza, D. Decouchant, A. M. Martínez and A. L. Morán, Adaptive Resource Management in the PIÑAS Web Cooperative Authoring, In Proc. of the Second International Atlantic Web Intelligence Conference (AWIC'2004), Springer-Verlag, LNAI 3034, pp. 33-43, Cancún (Mexico), May 2004.
11. T. O'Grady, Flexible Data Sharing in a Groupware Toolkit, Master's thesis, University of Calgary, Department of Computer Science, November 1996.
12. N. Preguia, J. Legatheaux, H. Domingos and S. Duarte, Data Management Support for Asynchronous Groupware, In Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW'00), Philadelphia PA (USA), pp. 69-78, December 2000.
13. M. Roseman and S. Greenberg, Building Real-Time Groupware with GroupKit., ACM Transactions on Computer-Human Interaction, 3(1):66-106, 1996.
14. A. S. Tanenbaum and M. Van Steen, Distributed Systems Principles and Paradigms, Prentice Hall, pp. 678-697, USA, 2002.

Appendix: The Set Notation

Syntax	Definition
$\mathcal{P}(W)$	The power-set of a set W is the set whose members are all the sets whose members belong to W .
$U \leftrightarrow V$	A relation from U to V is a set of pairs constructed from elements of U and V .
p^{-1}	The inverse of a relation p is that relation with each pair of the form (u, v) turned into the pair (v, u) .
$\text{dom}(p)$	The domain of a relation p from U to V is the subset of U whose elements are related to at least one element of V under the relation p .
$\text{ran}(p)$	The range of a relation p from U to V is the subset of V whose elements are related to at least one element of U under the relation p .
$p[W]$	The image of a set W under a relation p from U to V is the subset of V whose members are the second elements of those pairs of p whose first elements are members of W .
$U \nrightarrow V$	A partial function from U to V is a relation that does not contain two distinct pairs with the same first element.
$U \rightarrow V$	A total function from U to V is a partial function from U to V whose domain is U .
$U^i \rightarrow V$	A partial injection from U to V is a partial function from U to V whose inverse is a partial function from V to U .
$U \rightarrow V$	A total injection from U to V is a partial injection from U to V and a total function from U to V at the same time.
$U \rightarrow V$	A partial surjection from U to V is a partial function from U to V whose range is exactly V .
$U \rightarrow V$	A total surjection from U to V is a partial surjection from U to V and a total function from U to V at the same time.
$U^i \rightarrow V$	A partial bijection from U to V is a partial injection from U to V and a partial surjection from U to V at the same time.
$U \rightarrow V$	A total bijection from U to V is a total injection from U to V and a total surjection from U to V at the same time.

Agilo: A Highly Flexible Groupware Framework

Axel Guicking, Peter Tandler, and Paris Avgeriou

Fraunhofer IPSI,

Dolivostrasse 15, D-64293 Darmstadt, Germany

{axel.guicking, peter.tandler, paris.avgeriou}@ipsi.fraunhofer.de

Abstract. Today there exist many frameworks for the development of synchronous groupware applications. Although the domain of these applications is very heterogeneous, existing frameworks provide only limited flexibility to integrate diverse groupware applications in a meaningful way. We identify five variation points that a groupware framework needs to offer in a flexible way in order to facilitate the integration of diverse groupware applications. Based on these variation points, we propose a groupware framework called Agilo that tries to overcome the limited flexibility of existing frameworks by offering multiple realizations of these variation points and providing a modular architecture to simplify the integration of applications and the extensibility and adaptability to different application and integration requirements.

1 Introduction

Today there exist many frameworks to support and to simplify the development of applications for synchronous groupware [1]. While the application domain of these applications covers a diverse range from simply structured and inherently conflict-free applications like chats to conflict-rich shared whiteboards and shared knowledge maps with highly structured data models, the combination of diverse applications from this domain requires the integration on different levels: user interface, application logic, and data model.

The difficulties of combining different applications are caused by their use of different concepts and abstractions, such as different object sharing approaches and distribution architectures. While there are many groupware frameworks that provide certain concepts and several frameworks that offer flexibility in some aspects, yet, there is no framework that offers enough flexibility to combine very heterogeneous groupware applications. In addition, different frameworks often use different domain-specific abstractions, making it hard for application developers to learn a new framework and difficult for them to combine different frameworks [2]. The groupware frameworks we developed in the past [3,4] had a focus on cooperative hypermedia systems, e.g. [5]. Although these frameworks provide excellent support for modeling complex object structures they are too heavy-weight to design applications that don't benefit from using shared objects with transaction-based conflict management and replication support (such as

chats or voting tools). However, many groupware systems benefit from the combination of such simple tools with complex ones, making it necessary to combine applications with different requirements.

In this paper we present a new Java-based framework called Agilo that seeks to overcome the shortcomings of existing groupware frameworks with respect to their limited flexibility. Although Agilo supports application integration on all above levels, we concentrate on the two latter levels and describe how it provides the required flexibility by offering multiple realizations of several architectural commonalities of synchronous groupware applications.

The structure of the rest of this paper is as follows: in section 2 we identify several variation points common to synchronous groupware applications and how they are realized in existing groupware frameworks. Section 3 presents how these variation points are realized in the Agilo framework. The final section 4 concludes the paper with a short summary as well as the current status of the framework development and future work.

2 Analysis of Variation Points and Related Work

The diversity of synchronous groupware applications demonstrates that, depending on the specific application, there are many different requirements for the underlying framework. In this section, we identify five *variation points* (also called hot spots [6] or hooks [7]) that represent the aspects of groupware applications which may differ from one to another. The essential requirement for a groupware framework as proposed in this paper is the ability to combine different manifestations of each of the following variation points on the framework level in order to be able to combine different groupware applications (Fig. 1). The identified variation points are a starting point to characterize different types of groupware applications. A more comprehensive analysis whether the identified variation points are sufficient requires further research.

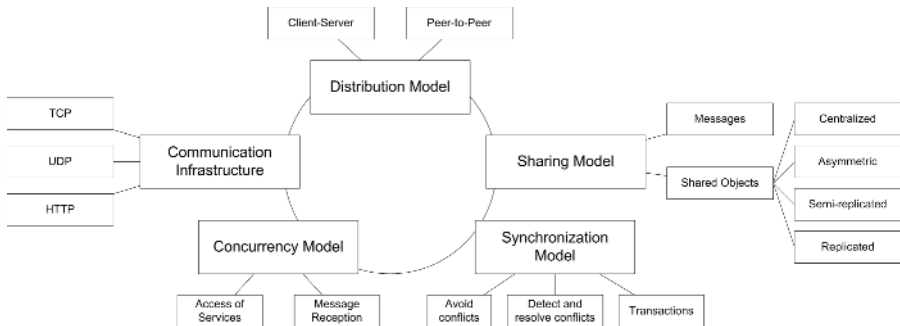


Fig. 1. The variation points and important realizations

Distribution Model. The first aspect that requires variability is the distribution model [1]. The two most common forms are Client-Server and Peer-to-Peer. Most other alternatives can be mapped on a combination of these two approaches [8].

There are many groupware frameworks that support a Client-Server architecture, such as COAST [3] or Rendezvous [9], while others are designed as Peer-to-Peer systems, e.g. GroupKit [10]¹ or DreamTeam [11]. Depending on the usage context, each approach has benefits and liabilities. While Client-Server reduces complexity and simplifies consistency issues as well as support for latecomers [1], Peer-to-Peer avoids having the server as bottleneck and single point of failure [8,11]. Additionally, Client-Server is more appropriate when using handheld devices because of their limited resources – the central server then also plays the role of a storage medium. In some circumstances it might even be better to use a hybrid approach where some clients communicate mediated by a server while others form a Peer-to-Peer subgroup [1].

In order to be able to integrate applications that use different distribution models and to adapt the distribution model of an application to domain-specific and environmental constraints, it is essential that groupware frameworks allow flexibility in choosing and adapting the distribution model accordingly.

Communication Infrastructure. The requirements of groupware applications often influence the selection of the communication infrastructure. This includes support for different protocols and different marshalling.

Multiple *communication protocols* must be supported in different application contexts. For clients running on a LAN, a fast protocol such as TCP or UDP is sufficient. If the communication must cross firewalls, it might be necessary to use HTTP or another protocol that firewalls support. Therefore all recent messaging protocols such as SOAP² and XMPP³ are defined independently of the underlying communication protocols.

Similarly, different contexts and applications introduce different requirements for the data exchange format which we call *marshalling*, i.e. the transformation of messages into machine-independent format appropriate for sending through the network. If large amounts of data have to be transmitted, the marshalling should be designed to reduce the size of the data, which may include a binary encoding and compression. On the other hand, if small or heterogeneous devices are communicating, the marshalling must be designed in a way that allows all devices to support it. While SOAP and XMPP support different protocols, they offer an XML-based marshalling only. The COAST framework [3] allows the use of different marshalling for different clients, but uses a single proprietary protocol based on TCP.

Sharing Model. For some applications such as a simple chat application, it is convenient to use messages to inform other clients about application state

¹ Although GroupKit relies on a central server called “Registrar” the communication between the clients is based on a Peer-to-Peer distribution model.

² <http://www.w3.org/TR/soap/>

³ <http://www.ietf.org/rfc/rfc3920.txt>

changes. However, applications like a shared knowledge map have a complex object structure that can be implemented far more easily on top of a higher-level abstraction than messages, e.g. shared objects.

While messages are transient objects that carry specific semantics such as a text message in a chat session and that are usually sent only once between two nodes in a system, shared objects are long-living and often persistent objects that are manipulated and updated often by different users. Therefore, the access of shared objects needs to be synchronized in order to avoid data corruption and inconsistencies (see below). Furthermore, the shared objects are distributed in the system using a distribution scheme – typical distribution schemes are central, asymmetric, semi-replicated, and replicated [1].

To free the developer from the burden of implementing concurrency control strategies and distribution schemes as part of the application the framework needs to provide an object-sharing abstraction that includes these two aspects. However, there are two essential requirements: First, the developer must be able to adapt the different aspects of the sharing model in order to optimize the use of available resources such as network traffic and performance. Second, the developer must not be required to use this shared data abstraction at all to avoid potential performance overhead. Besides, for applications where no conflicts can occur (such as chats) or that rely on concepts that do not fit to a shared data abstraction, using shared data is of less value.

Several groupware frameworks directly support sharing of information, such as COAST [3], Rendezvous [9], and DyCE [4]. However, these systems force the developer to use the sharing abstraction. GroupKit [10] allows the combination of both approaches, messages and shared objects.

Concurrency Model. Due to the nature of the domain of synchronous groupware each such application has to deal with concurrency issues. To let application developers concentrate on the application logic, groupware frameworks need to make use of an efficient concurrency behavior. This includes the potentially concurrent access of services by different clients as well as the combined use of asynchronous and synchronous application components without degrading non-functional quality requirements, such as performance, robustness, and scalability. With respect to the sharing model the framework has to correctly resolve the concurrent reception of messages and concurrent manipulation and access of shared objects, respectively.

For example, DreamTeam provides support for interweaving synchronous and asynchronous communication using the Half-Sync/Half-Async pattern [11]. The COAST server uses the Active Object pattern to process incoming messages [3].

Synchronization Model. When concurrent processes access shared resources, synchronization is necessary in order to ensure consistency of data in case of concurrent modification. There are two principal approaches to ensure consistency: *Avoid* conflicts by locking data before modification or to *detect and resolve* conflicts. Common locking mechanisms include mutexes and semaphores. Common conflict resolution mechanisms include transactions and protocols for updating

shared data. Both approaches can be implemented in many different ways. Locking is appropriate if, e.g., changes are hard to detect or complicated to resolve. However, locking reduces performance, as the application has to wait for the lock before being able to continue, which affects the usability of interactive systems.

Depending on application requirements, both strategies can be appropriate and therefore groupware frameworks need to provide enough flexibility in this respect. For example, DyCE [4] offers optimistic transactions, whereas COAST [3] additionally offers pessimistic transactions, both with automatic conflict detection and rollback.

3 Framework Design

The design of the Agilo framework directly addresses the variation points described in the previous section. It is based on experiences with groupware frameworks we developed in the past [3,4]. Its flexibility is increased by using design patterns from the domain of distributed and concurrent computing. This leads to an extensible and flexible groupware architecture that allows the integration of heterogeneous groupware applications while giving developers enough freedom in choosing abstractions that fit best to the applications they are building.

Before describing how the different variation points are realized in the Agilo framework, the core concepts of the framework are described next.

The Agilo framework is designed around two key concepts: *Modules* and *Messages*. Modules are software components that are either located on the framework level or on the application level. An Agilo groupware application consists of several modules each running on a node of the system. Messages are application-specific data chunks that are sent between nodes. Incoming messages at a node are processed sequentially and are “forwarded” to one or more application modules which usually send messages to one or more modules running on other nodes as result of processing an incoming message. Providing this message-based communication concept the framework allows the development of groupware applications with a very simple need for communication support such as chats and voting tools, while more sophisticated communication needs can be built easily on top of the message communication (see below).

The framework core is designed to be as small as possible whereas most of the functionality is implemented as separate modules. This approach reveals two advantages: first, the knowledge about the framework required to build applications is kept very small and it can be extended successively as needed. Second, many parts of the framework can be configured independently, leading to a high adaptability and flexibility of the framework.

The remainder of this section elaborates on how exactly the framework provides this flexibility by offering alternative realizations at the different variation points described in the previous section.

Distribution Model. The distribution model of Agilo has been designed to accommodate both the Client-Server and Peer-to-Peer distribution architectures.

In order to be able to establish a Client-Server distribution, the framework consists of three parts: A client-side part, a server-side part, and a common part that is required by both client and server. The Peer-to-Peer distribution is achieved by making each participating node a combined client and server, i.e. by deploying client and server components together in each node. Additionally, both distribution architectures require specific configuration settings in order to adapt the server and client functionality to work in the respective distribution type.

Communication Infrastructure. The Communication Infrastructure of Agilo allows the use of different transport and data protocols. It is realized by following the Client-Dispatcher-Server pattern [12]. The communication between client and server or among peers can be customized on two levels: On the lower level, Agilo supports different transport protocols, such as TCP or HTTP. Protocol-specific implementations accomplish sending and receiving messages while hiding implementation details such as fault-tolerance and native resource handling. The upper level provides different marshalling behavior to support different data-exchange protocols (e.g. SOAP, XMPP). The customizable marshalling behavior especially allows the integration of heterogeneous clients, such as PDAs and smartphones.

Sharing Model. Besides the core concept of “low-level” messages, the Agilo framework offers support for “high-level” shared objects that are implemented on top of the two core concepts of Agilo, messages and modules.

Agilo provides a concrete interface for objects that need to be shared while the distribution scheme is implemented in a separate ObjectManager module. The data itself and its distribution scheme are thus decoupled, allowing the use of different distribution schemes such as centralized, semi-replicated, or replicated shared objects. Application-specific objects that need to be shared have to implement a specific interface in order to be managed by the ObjectManager.

Concurrency Model. The Concurrency Model of Agilo makes provision how multiple concurrent threads can simultaneously work together in the context of the groupware application. Specifically, clients can interweave synchronous and asynchronous messages following the Half-Sync/Half-Async pattern [13].

Another concurrency concern is the processing of incoming messages on the nodes of the system. Messages are received by the node’s ConnectionHandler module following the Reactor pattern [13]. The incoming messages are unmarshalled and enqueued in the node’s MessageHandler module. The messages are dequeued by a single-threaded active object [13], called MessageRouter that is responsible for notification of the node’s application modules. A different implementation of the ConnectionHandler uses the more performant Proactor pattern [13]. Furthermore, instead of the naive single-threaded MessageRouter, a multi-threaded implementation using the Leaders/Followers pattern [13] can be used for module notification if there is no need for a *globally* consistent order of messages.

In case of a Peer-to-Peer distribution model concurrency issues arise because the order of incoming messages is no longer guaranteed to be the same on all

peers. The handling of these problems when using a Peer-to-Peer setting is part of the communication infrastructure.

Synchronization Model. The Synchronization Model of Agilo uses different locking mechanisms, such as semaphores and mutexes as well as Java's built-in synchronization mechanisms to enforce controlled access to shared data. Additionally, it allows for detection and resolution of conflicts. The framework supports the use of transactions to combine multiple actions of a client into an atomic action. When a client commits a transaction, a single message containing the manipulations of the affected shared objects is sent to the server where it is processed like any other message. Since the MessageHandler module processes messages sequentially in the order they arrived, the processing of incoming messages uses an implicit transaction management. In the case of a Peer-to-Peer distribution model, the same concurrency issues arise as described in the previous subsection.

4 Conclusions

In this paper we identified the limited flexibility of existing frameworks for synchronous groupware applications as a significant shortcoming in order to combine heterogeneous groupware applications in a reasonable way. This paper focused on the integration of groupware applications on the application logic and data model levels that were partitioned into five different variation points. Underlying frameworks need to support these points of synchronous groupware applications in a flexible and configurable way. Since existing groupware frameworks lack the required flexibility we proposed a new groupware framework called Agilo that seeks to overcome this shortcoming by providing enough flexibility and extensibility with respect to all identified variation points. By providing a very modular architecture that clearly separates different concerns it offers the required flexibility to be applicable for a wide variety of groupware applications.

Since this paper presents work in progress some of the features described above are not yet fully implemented in the Agilo framework. The Client-Server and a rudimentary Peer-to-Peer distribution model, the communication infrastructure, the concurrency model as well as parts of the sharing and synchronization models are already implemented as described in the previous section. However, several essential parts are still missing, such as different distribution schemes of shared objects and a transaction-based synchronization model.

Although the framework is not yet completely implemented, experience derived from its use in a large commercial meeting support system⁴ has already proved that the architecture of the framework greatly simplifies the development of synchronous groupware applications. In order to integrate the meeting support system with support for distributed meetings we used an existing chat application framework⁵. In this framework we ported the lower communication

⁴ <http://www.ipsi.fraunhofer.de/digital-moderation>

⁵ <http://www.ipsi.fraunhofer.de/concertchat>

level to Agilo which made it easy to access the generated meeting documents from the chat and provide a tight integration of the two systems.

Besides the implementation of the missing parts mentioned above, the next steps concerning the proposed framework include more case studies, i.e. implementing other diverse groupware applications. Furthermore, the evaluation of the framework concepts and how these support application developers as well as how different combinations of variation point alternatives influence quality requirements such as scalability and performance remain open issues.

References

1. Phillips, W.G.: Architectures for synchronous groupware. Technical Report 1999-425, Queen's University (1999)
2. Lukosch, S., Schümmer, T.: Communicating design knowledge with groupware technology patterns. In: Proc. CRIWG 2004. LNCS, Springer (2004) 223–237
3. Schuckmann, C. et al.: Designing object-oriented synchronous groupware with COAST. In: Proc. CSCW'96, ACM Press (1996) 30–38
4. Tietze, D.: A Framework for Developing Component-based Co-operative Applications. PhD thesis, Darmstadt University of Technology, Germany (2001)
5. Streitz, N.A. et al.: DOLPHIN: Integrated meeting support across local and remote desktop environments and liveboards. In: Proc. CSCW'94, ACM Press (1994) 345–358
6. Schmid, H.A.: Systematic framework design by generalization. *Communications of the ACM* **40** (1997) 48–51
7. Froehlich, G. et al.: Reusing hooks. In Fayad, M.E. et al., ed.: *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. John Wiley & Sons (1999) 219–236
8. Roth, J.: A taxonomy for synchronous groupware architectures. In: Workshop “Which Architecture for What” of CSCW'00. (2000)
9. Hill, R.D. et al.: The Rendezvous architecture and language for constructing multiuser applications. *ACM ToCHI* **1** (1994) 81–125
10. Roseman, M., Greenberg, S.: Building real time groupware with GroupKit, a groupware toolkit. *ACM ToCHI* **3** (1996) 66–106
11. Roth, J.: ‘DreamTeam’: A platform for synchronous collaborative applications. *AI & Society* **14** (2000) 98–119
12. Buschmann, F. et al.: *Pattern-oriented Software Architecture. A System of Patterns*. Volume 1. John Wiley & Sons Ltd (1996)
13. Schmidt, D.C. et al.: *Pattern-oriented Software Architecture. Patterns for Concurrent and Distributed Objects*. Volume 2. John Wiley & Sons Ltd (2000)

Autonomous and Self-sufficient Groups: Ad Hoc Collaborative Environments

Joan Manuel Marquès^{1,2} and Leandro Navarro²

¹ Universitat Oberta de Catalunya, Departament of Computer Sciences,
Av. Tibidabo, 39-43, 08035 Barcelona, Catalonia, Spain
jmarquesp@uoc.edu

² Universitat Politècnica de Catalunya, Department of Computer Architecture,
Jordi Girona, 1-3, D5-105, Campus Nord, Barcelona, Catalonia, Spain
{marques, leandro}@ac.upc.edu

Abstract. Asynchronous collaborative applications and systems have to deal with complexities associated with interaction nature, idiosyncrasy of groups and technical and administrative issues of real settings. Existing solutions address asynchronous collaboration via simplified and centralized models. In this paper we present LaCOLLA, a fully decentralized middleware for building collaborative applications that provides general purpose collaborative functionality without requiring anyone to provide resources for the whole group. This helps applications to incorporate collaboration support and deal with most complexities derived from groups and its members. The implementation of LaCOLLA follows the peer-to-peer paradigm and pays special attention to the autonomy of its members and to the self-organization of its components. Resources (e.g. storage, task execution) and services (e.g. authorization) are provided by its members, avoiding dependency from third party agents or servers. This work was first validated by simulation. Then we built the middleware and adapted some collaborative applications.

1 Introduction

One of the most significant benefits of the Internet has been the improvement on people's interactions and communication. E-mail, Usenet News, Web and Instant Messaging are four of the most well-known and successful examples of this. Internet has allowed the creation of asynchronous virtual communities where members interact in a many-to-many basis. Many-to-many interaction, uncommon in the physical world, has transformed the way people learn, do work together, find others with common interests and share information among them, etc. After a decade of great excitement, the pace of this transformation is slowing down because collaboration is much more than these tools, because the Internet is designed for one-to-one interaction (the Internet transport is designed for the communication between two hosts) and that applications with collaborative necessities have to deal with complexities derived from:

- *Interaction nature:* participants are dispersed, many-to-many collaboration, people participate in the collaboration at different times, the same person connecting from different locations at different times of the day (home, work, mobile).

- *Idiosyncrasy of groups*: variety of issues such as flexibility, dynamism, decentralization, autonomy of its participants, different kinds of groups (task oriented, long-term, weak commitment groups, etc), groups exist while its members participate in group activities and provide necessary resources, etc.
- *Technical and administrative issues*: guarantees for the availability of information generated in the group, interoperability among applications, security aspects (authorization, access rights, firewalls), participants belonging to different organizations or departments with different authorities that impose rules and limits to facilitate administration, internal work and individual use, etc. [1]

Development of applications that take into account all those requirements are too complex and costly, therefore collaborative applications focus only in a few key aspects while neglecting others. In that way, most of the solutions resort to simpler client/server centralized models using resources administrated by a third party (a service provider). Client/server solutions –or more generally speaking, all solutions that require some sort of centralization– impose technical, administrative and economic restrictions that interfere with the interaction nature and idiosyncrasy of groups.

In contrast, Peer-to-Peer (P2P) systems or networks are distributed systems formed only by the networked PCs of the participants. All machines share their resources: computation, storage and communication. They all act both as servers and as clients. P2P systems are self-sufficient and self-organizing, applying protocols in a decentralized way to perform search and location, and sharing the burden of object transfers. As resource provision and coordination is not assigned to a central authority, all participants have similar functionalities and there is no strict dependency to any single participant. P2P networks may be robust and attain tolerance to failures, disconnections and attacks. [2]

In this paper we present LaCOLLA, a fully decentralized P2P middleware for building collaborative applications that provides general purpose collaborative functionalities based on the resources provided by group participants only. The provision of these functionalities will avoid applications deal with most of complexities derived from groups, members working across organizational boundaries and requiring additional resources. This simplification (transparency) will help include collaborative aspects into applications in an ad-hoc manner.

LaCOLLA began as a middleware implemented following the peer-to-peer paradigm paying special attention to the autonomy of its members and to self-organization of its components. Another key aspect was that resources (e.g. storage) and services (e.g. authorization) were provided by its members (avoiding dependency from third party agents). At this first stage, it provided support to: storage, awareness, groups, members, instant messaging and location transparency. Now we are incorporating the ability to execute tasks using computational resources provided to group. With that ability, groups will definitely evolve to become entities per se, not only gatherings or collections of members.

Having groups as units of organization and use of resources would help to change to a view of the Internet as a collection of communities: groups of individuals sharing resources among them (an individual may belong to different groups and a resource may belong to different groups). As an example, in virtual learning environments

students may need to do activities in groups using some kind of software. It will be useful that any member of the group could install the software by deploying it using the computational resources available to the group. After that, any member of the group could use that software and the results will be stored on storage resources belonging and available to group.

This ability of pooling resources belonging to groups has been strongly influenced by grid systems. Grids are large-scale geographically distributed hardware and software infrastructures composed by heterogeneous networked resources owned and shared by multiple administrative organizations which are coordinated to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications [3]. In contrast, our focus is on the ad-hoc creation of groups based solely on the resources provided by the participants, independently of underlying administrative organizations or external service providers.

Groove (<http://www.groove.net>) is a platform that partially covers some of the ideas behind LaCOLLA approach. In [4] Groove is defined as a system that lets users create shared workspaces on their local PCs, collaborating freely across corporate boundaries and firewalls, without the permission, assistance, or knowledge of any central authority or support groups. Groove allows transparent synchronization among workspaces, but depends on relay servers to provide offline queuing, awareness, fan-out and transparency (to overcome firewall and NAT problems). Those relay servers are provided by third parties. The main differences between Groove and LaCOLLA are that groove emphasizes transparent synchronization of collaborating PCs, along with direct communication among them. Also the fact that provides third party relay servers. Whereas LaCOLLA emphasizes on self-organization as a group and uses only resources provided by its participants (no dependency on third parties). Participants are not obliged to provide resources to group, but group works only with resources provided by its members. All resources connected to group are synchronized transparently and are used to articulate collaboration.

The rest of the paper is organized as follows: Section 2 presents the requirements that should satisfy an asynchronous collaborative middleware. Section 3 describes the functionalities and architectural aspects of LaCOLLA, with emphasis on what we call virtual synchronism: virtually immediate access to changes and latest versions of objects, along with the API offered to applications and an overview of internal mechanisms behavior. Section 4 presents experimental results from a simulator, concluding in Section 5.

2 Requirements for an Asynchronous Collaborative Middleware

As mentioned previously, asynchronous collaborative applications have to deal with many aspects to support collaboration. The basic requirements a middleware should satisfy to facilitate the development of this kind of applications are [5]:

- *Decentralization*: no component is responsible of coordinating other components. No information is associated to a single component. Centralization leads to simple solutions, but with critical components conditioning the autonomy of participants.
- *Self-organization of the system*: the system should have the capability to function in an automatic manner without requiring external intervention. This requires the

ability of reorganizing its components in a spontaneous manner in presence of failures or dynamism (connection, disconnection, or mobility).

- *Oriented to groups*: group is the unit of organization.
 - *Group availability*: capability of a group to continue operating with some malfunctioning or not available components. Replication (of objects, resources or services) can be used to improve availability and quality of service.
 - *Individual autonomy*: members of a group freely decide which actions perform, which resources and services provide, and when connect or disconnect.
 - *Group's self-sufficiency*: a group must be able to operate with resources provided by its members (ideally) or with resources obtained externally (public, rent, interchange with other groups, ...)
 - *Allow sharing*: information belonging to a group (e.g. events, objects, presence information, etc.) can be used by several applications.
 - *Security of group*: guarantee the identity and the selective and limited access to shared information (protection of information, authentication).
 - *Availability of resources*: provide mechanisms to use resources (storage, computational, etc.) belonging to other groups (public, rented, interchange between groups to improve availability, etc.)
- *Internet-scale system*: formed by several components (distributed). Members and components can be at any location (dispersion).
 - *Scalability*: in number of groups, guaranteed because each group uses its own resources.
- *Universal and transparent access*: participants can connect from any computer or digital device, with a connection independent view (e.g. as a web browser).
 - *Transparency of location of objects and members*: applications don't have to worry about where are the objects or members of the group. Applications use a location independent identifier and may access to different instances as people move, peers join and leave, or any other conditions change.
 - *Support disconnected operational mode*: work without being connected to the group. Very useful for portable devices.

3 LaCOLLA

LaCOLLA is a middleware that follows the requirements presented in the previous section. Four main abstractions have inspired the design process of LaCOLLA: oriented to groups, all members know what is happening in the group, all members have access to latest versions of objects, and tasks can be executed using the computational resources belonging to the group. These abstractions take shape in the following functionality.

3.1 Functionality

LaCOLLA provides to applications the following general purpose functionality [5]:

- *Communication by “immediate” and consistent dissemination of events*: information about what is occurring in the group is spread among members of the

group as events. All connected members receive this information right after it occurs. Disconnected members receive it during the re-connection process. This immediate and consistent dissemination of events helps applications to provide awareness to members.

- *Virtually strong consistency in the storage of objects*: components connected to a group obtain access to latest version of any object. Objects are replicated in a weak-consistent optimistic manner. Therefore, when an object is modified, different replicas of the object will be inconsistent for a while. However, LaCOLLA guarantees that, when an object is accessed, the last version will always be provided (given that events are disseminated immediately).
- *Execution of tasks*: members of a group (or the applications these members use) can submit tasks to be executed using computational resources belonging (or available) to the group. In the present version, tasks are Java classes executed locally that perform computational activities. In future versions we want to be able to deploy services that would provide services at group level. Examples of this kind of services could be a service to coordinate some dynamic and volatile aspect in a synchronous collaborative activity, a session group-level awareness service, or any other service that can provide an added value to groups and that the fact of being deployed in a centralized manner (using only computational and storage resources belonging to group) doesn't affect the decentralization, autonomy and self-sufficiency of the group.
- *Presence*: know which components and members are connected to the group.
- *Location transparency*: applications don't have to know the location (IP address) of objects or members. LaCOLLA resolves them internally (similar to domain name services like DNS).
- *Instant messaging*: send a message to a subgroup of members of the group.
- *Management of groups and members*: administrate groups and members: add, delete or modify information about members or groups.
- *Disconnected mode*: allow applications operate offline. During re-connection, the middleware automatically propagates the changes and synchronizes them.

3.2 Architecture

The architecture of LaCOLLA [5] is organized in five kinds of components (figure 1). Each component behaves autonomously. Each member decides to instantiate any number of the following components in the peer is using:

- *User Agent (UA)*: interacts with applications (see section 3.4 for a more detailed explanation). Through this interaction, it represents users (members of the group) in LaCOLLA.
- *Repository Agent (RA)*: stores objects and events generated inside the group in a persistent manner.
- *Group Administration and Presence Agent (GAPA)*: in charge of the administration and management of information about groups and their members. It is also in charge of the authentication of members.

- *Task Dispatcher Agent (TDA)*: distributes tasks to executors. In case that all executors were busy, the TDAs would queue tasks. Also guarantees that tasks will be executed even though the UA and the member disconnects.
- *Executor Agent (EA)*: Executes tasks.

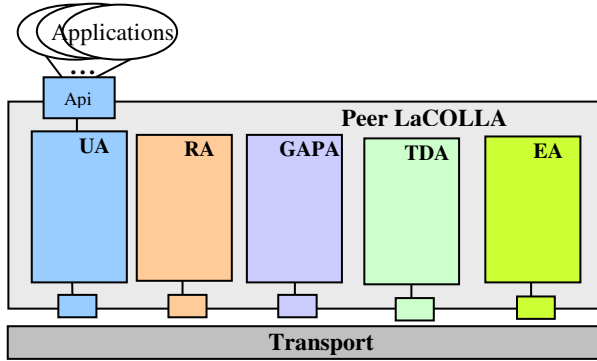


Fig. 1. Peer LaCOLLA

Components interact one to each other in an autonomous manner. The coordination among the components connected to a group is achieved through internal mechanisms. Internal mechanisms [5] have been grouped in: events, objects, tasks, presence, location, groups, members and instant messaging. They are implemented using weak-consistency optimistic protocols [6, 7] and random decision techniques [8]. Table 1 describes which components are involved in each category of mechanisms. More details about presence, events and objects mechanisms are provided in section 3.4.

Table 1. Categories of mechanisms implemented by each kind of component

Categories of Mechanisms	UA	RA	GAPA	TDA	EA
Events	X	X	-	-	-
Objects	X	X	-	-	X
Tasks	X	-	-	X	X
Presence	X	X	X	X	X
Location	X	X	X	X	X
Instant Messaging	X	-	X	-	-
Groups	X	X	X	X	X
Members	X	-	X	-	-
Security	X	X	X	X	X
Disconnected operational mode	X	-	-	-	-

Components and mechanisms related to tasks are based on the ideas used to design JNGI [9], a decentralized and dynamic framework for large-scale computations for problems that feature coarse-grained parallelization. While the components of JNGI communicate using JXTA [10], we use the communication facilities of LaCOLLA. Among the aspects that characterize LaCOLLA one that deserves special attention is what we have named virtual synchronism.

3.2.1 Virtual Synchronism

LaCOLLA guarantees to applications that all events delivered to LaCOLLA will be received almost immediately (i.e. immediately or just after reconnection) by the rest of connected members. This guarantee provides the feeling of knowing what is happening in the group while it is occurring. Disconnected members will receive the events during the re-connection process.

LaCOLLA also guarantees that the last version of all objects (based on the previous guarantee) belonging to group will be available immediately for all members.

The sum of both guarantees is what we have named virtual synchronism. Apart from the up-to-date perception that members of the group have at any moment, virtual synchronism has an interesting side effect. This side effect is very useful in an autonomous, decentralized and dynamic storage system: since all components know the location of all objects (and their replicas), components access them directly (without a resolver that informs about location of last version of objects). This allows LaCOLLA to have an autonomous and decentralized policy to handle objects and their replicas at the same time that guarantees immediate access to last versions.

3.3 Example of LaCOLLA Group

Figure 2 is a snapshot of a collaborative group that uses applications connected to LaCOLLA. Each member belonging to group provides to it the resources that she/he wants. As we have said, that decision depends on the capacity and connectivity of the computer the member is using and on the degree of involvement that she/he has in the group. In this example, two members (C and D) provide all possible components (RA, GAPA, EA and TDA). Other two members (B and F) provide all components except execution components (provide RA and GAPA). Three of the members (A, E and G) provide no resources to group.

The members of the group use several applications to perform the collaborative tasks. At the moment the picture was taken, they were using an asynchronous forum, a file sharing tool and an instant messaging application. Not all members use all applications at same time.

Those applications share presence, members and group information. On the one hand, this prevents users to register to each application and also provides presence information even though they are using different applications. On the other hand, application developers don't have to worry about where the necessary information is located. LaCOLLA middleware also facilitates the sharing of information among applications (if compatible formats are used) due the fact that information, events and objects are stored in LaCOLLA storing resources (RA).

Member D (represented by discontinuous lines) is not connected to group at this moment. Even though, her/his peer is connected to group, providing all its resources to it. That means that all generated events and some of the objects would be stored in her/his peer LaCOLLA (RA), tasks would be executed or planned using its resources (EA, TDA), or that users would be authenticated by her/his peer (GAPA), information of members and groups would be also stored in it.

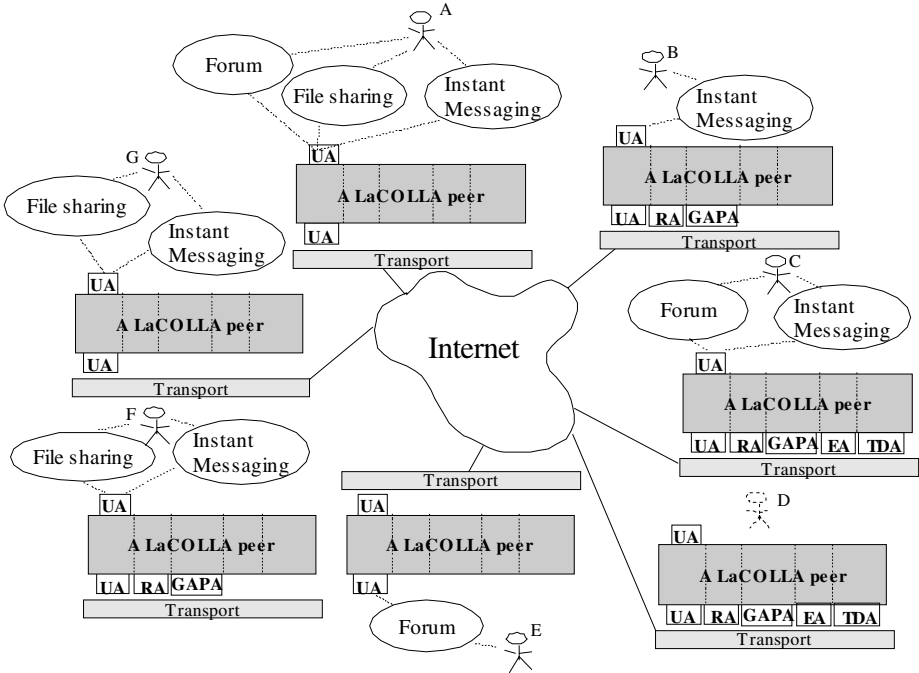


Fig. 2. Snapshot of a collaborative group that uses applications connected to LaCOLLA

An example of a group like the one presented in figure 2 could be a collaborative group doing a collaborative learning practice in a virtual university (as is UOC - Universitat Oberta de Catalunya). The learning practice could be a software development project or a case study. In those cases, a member of the group initiates the group (providing at least one RA and one GAPA components) and invites other members (who contribute with more resources and components to the group). From that point on, the group operates using the resources provided by its members. Although any member disconnects its resources or is removed as member of the group, the group will be operative. And most important, nothing will happen if the initiator of the group disconnects its resources or is removed from the group. As long as members provide resources to the group, it will exist. Whenever no member provides resources, the group would extinguish.

LaCOLLA is independent of the applications that use its functionalities. Many applications (not only the kind of applications presented in the figure) involved in a

collaborative task could benefit from the general purpose collaborative functionalities that LaCOLLA provides. These applications could range from applications that only share generated information to sophisticated collaborative applications exploiting awareness information and coordinating actions (as events) of participants in the collaboration.

3.4 LaCOLLA Middleware

At the moment of writing this paper, we had the first beta version of LaCOLLA middleware. This version can be found at: <http://lacolla.uoc.edu/lacolla/>. It includes the source code, some basic instructions on how to use LaCOLLA, and installation procedures. LaCOLLA middleware has an open source license and is written in the Java language, what makes it independent from the underlying platform.

From both building collaborative applications that use LaCOLLA and from using the applications we developed, we obtained valuable ideas and improvements to introduce in the second beta version. The new version will pay special attention to security issues, which are at its minimum expression in the first version. We are also planning to introduce new components and mechanisms that will allow mobile devices (PDA, mobile phone, sensors, etc.) become LaCOLLA peers.

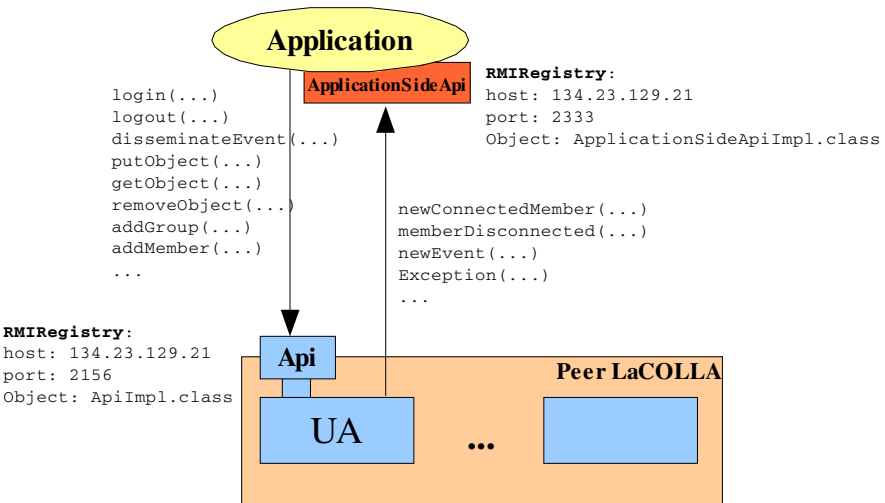


Fig. 3. LaCOLLA API. It has two parts. Applications use UA’s API to ask LaCOLLA to perform some action. The other API is provided by the applications to the UA were they are connected, that API is used by LaCOLLA to notify events or information to applications.

3.4.1 LaCOLLA API

LaCOLLA provides a powerful API that can be easily used by any application. As can be seen in figure 3, the API of LaCOLLA is divided in two parts. The first part is the API provided by LaCOLLA (through its UA) to applications. The detail of functions that an UA provides to applications is listed on table 2.

The second part of the API is used by an UA to notify events or information coming from LaCOLLA to applications connected to the UA. Table 3 lists the functions. As can be seen in figure 3, UAs invoke functions at `ApplicationSideApi` class. This class is provided with LaCOLLA middleware and must be extended by any application that wants to use LaCOLLA.

Java RMI is used to publish and invoke each part of the API. In the example of figure 3, UA's API is published at host 134.23.129.21 and port 2156. When an

Table 2. API functions that User Agents offer to applications

Category	Function	Description
Presence	login	Connects user to group.
	logout	Disconnects user from group.
	whosConnected	Which members are connected to the group?
Events	disseminateEvent	Sends an event to all applications belonging to group.
	eventsRelatedTo	Which events have occurred to a specific object?
Objects	putObject	Stores an object in LaCOLLA.
	getObject	Obtains an object stored into LaCOLLA.
	removeObject	Removes an object stored in LaCOLLA.
Tasks	submitTask	Submits a task to be executed by computational resources belonging to group.
	stopTask	Stops a task.
	getTaskState	In which state is the task?
Instant Messaging	sendInstantMessage	Sends a message to specified members of the group.
Groups	addGroup	Creates a new group.
	removeGroup	Removes a group.
	modifyGroup	Modifies the properties of a group.
	getGroupInfo	Gets information about the properties of a group. (Look at <code>groupInfo</code> function)
	getGroupInfoSync	Gets information about the properties of a group in a synchronous manner. This function does not return until the operation is completed and a result is available.
Members	addMember	Creates a new member.
	removeMember	Removes a member.
	modifyMember	Modifies the properties of a member.
	getMemberInfo	Gets information about the properties of a member.

Table 3. API functions that UA invokes on applications

Category	Function	Description
Presence	newConnectedMember	Notifies that a new member has been connected.
	memberDisconnected	Notifies that a member has been disconnected.
Events	newEvent	Reception of an event occurred in the group.
Tasks	taskStopped	Notifies that the task has been stopped nicely.
	taskEnded	Notifies the ending of a task.
Instant Messaging	newInstantMessage	Reception of a new instant message.
Groups	groupInfo	Reception of the group information.
Other functions	exception	Notifies that an internal exception or anomalous situation has occurred.
	appsAlive	UA queries the state of the application. Used to know if application is alive and connected to group.

application wants to login, logout, send an event, put an object, get an object, etc. it has to invoke the API function at this location.

The same thing happens with the API provided by each application to LaCOLLA. In that case, each application extends `ApplicationSideApi` and publishes it. In the example, application is at host 134.23.129.21 and at port 2333. This API allows UAs notify to applications that a member has connected, that a member has disconnected, that there is a new event, an exception, etc.

It is also interesting to notice that all applications connected to a LaCOLLA peer will use the API provided by its UA, but that each application will have its own API and that the UA will notify to each application individually.

If the application is written in Java the integration with LaCOLLA is very easy. It has to use Java rmi to invoke the API of LaCOLLA at UA. The application also has to extent `ApplicationSideApi` class provided along with the LaCOLLA middleware. This makes very easy to adapt applications done in Java to benefit from LaCOLLA.

If the application is written in other programming languages, the developer has to build a module to be able to use the API of LaCOLLA. This module is very easy to build because it only has to translate parameters and results to and from Java. For instance, if the application is written in C/C++, JNI (Java Native Interface) can be used. The module will encapsulate both sides of the API: the invocations from application to UA and the notifications from UA to application.

3.4.2 Components and Internal Mechanisms

Components are implemented in Java and behave according to its local information (autonomy). Coordination among components connected to a group is achieved by internal mechanisms, which allow components learn new information and synchronize its local information with other components. Internal mechanisms behave in a decentralized and autonomous manner. Components communicate by message passing. Messages are serialized Java objects sent using TCP sockets.

There are 10 categories of internal mechanisms divided in several sub-mechanisms. Each sub-mechanism performs different actions depending on the kind of component. Is out of the scope of this paper to detail how the decentralized and self-organized behavior of LaCOLLA is achieved. A fully and detailed description can be found at [5, 11]. Alternately we are going to explain the general behavior of some LaCOLLA's internal mechanisms and the key aspects to understand its philosophy.

LaCOLLA middleware is based on the presence mechanism. To guarantee the consistency and a good performance of LaCOLLA it is required that each component connected to group knows which other components are connected to the group. The other key mechanism is event dissemination. Presence is the basis for the peer-to-peer behavior (decentralization, autonomy, self-organization and self-sufficiency). Event mechanism provides immediateness and consistency of view. In the next paragraphs more detail of both categories of mechanisms will be provided. Prior to that, is important to understand that LaCOLLA is a middleware intended to support asynchronous group collaboration and some sorts of synchronous collaboration. Therefore, groups are considered to have a small number of members and components (as is stated in validation section, LaCOLLA can deal with groups formed by 100 or

more components, but groups, to be realistic collaborative groups, should typically have 5, 10 or 20 members, not more). LaCOLLA was not created to support communities of members sharing or performing some weak-collaborative task.

Presence sub-mechanisms are: connection of a component, disconnection of a component, consistency of connected information, and detection that a component is no more connected. When a component wants to connect a group, sends its authentication information to a GAPA. If authentication is ok, GAPA answers with information about which components the GAPA knows that are connected to group. Then the new component sends a message to all components he knows that are connected to the group (those that GAPA has informed him) informing about its connection to the group. Prior to an ordered disconnection, the disconnecting component informs other components about its disconnection. To synchronize information about connected components, two techniques are used: a) every time a component sends a message to another component it includes the information about the components the sender knows that are connected. This allows the receiver to learn about connected components he didn't know that were connected. b) Time to time, a component randomly selects N^1 components and performs a consistency session with them. During a consistency session between A and B, A tells B which components A knows that are connected to group; B tells A which components knows that are connected to group. The last sub-mechanism related to presence refers to detection of components that are no longer connected to group: when a component (A) hasn't received any message from B for a long period of time², A tries to contact B. If A can't reach B, A removes B from its connected components list.

When an action occurs, an event is generated to inform about the action. Actions can be: new document, new member, document read, or any action that an application wants to disseminate to all members. As was explained in virtual synchronism part of the section 3.2, events are used to provide awareness information to members, but are also used to guarantee the internal consistency of the system. The dissemination of events mechanism guarantees that all connected components have all generated events in a time that users perceive as immediate.

When a new event is created, the component where the event was created sends it to all components the component knows that are connected. As can be seen, the performance of this mechanism is strongly related with presence mechanism. All RA store in a persistent manner all received events. Components not connected to the group or components that the sender of the event doesn't know that are connected to the group will not receive the event. To overcome this, a consistency sub-mechanism is implemented. Event's consistency mechanism is based on an adaptation of Golding's Time-Stamped Anti-Entropy algorithm [6] and is performed between an UA or RA and an RA. Consistency sessions among RA are as follows: time to time, an RA (RA1) randomly selects another RA (RA2) among the RA that knows are connected to group. Then, RA1 sends to RA2 a summary of all events that RA1 has received. RA2 sends to RA1 all events that RA2 has and that RA1 doesn't have along

¹ $\text{Max}(2, \log_2(\text{numberConnectedComponents})+1)$. This number was adjusted by simulation.

² This period of time is a parameter that can be adjusted. Component A can also know about component B through some other component (C). In that case, either by presence sub-mechanism a) or b) C has informed A that B was still connected.

with the summary of all events that RA2 has. Finally, RA1 sends to RA2 all events that RA1 has that RA2 doesn't have. Similarly, in the case of consistency sessions between UAs and RAs, the UA asks an RA for events that the UA doesn't have but UA never provides new information to RAs.

Events mechanism doesn't provide any order guarantee. Events mechanism provides immediateness (events are sent by originator to all other connected components right after the event is created) and consistency (consistency sessions guarantee that not connected components or components that haven't received the event will eventually receive it³). In the case some ordering guarantees were required, applications should provide them. As future work we are considering to implement some event's ordering policies in top of LaCOLLA and provide them to applications.

Events and presence mechanisms are in the basis of all other mechanisms. For example, objects mechanism is in charge of storing, retrieving, removing and guaranteeing the availability of objects stored in LaCOLLA. When an object is stored in LaCOLLA, the object is sent to any RA and an event is disseminated to all components to inform about the new object and its location. The event will be used by any component to know where is located the object. When an UA wants to retrieve an object, the UA knows where the object is located (some moment in the past, the UA received an event informing about the location of the object). Consequently, it obtains the object from any of its locations. To guarantee the availability of objects, they are automatically replicated in a decentralized manner. Every time a new replica is created, an event is disseminated to inform about the availability of new replica and its location.

Other mechanisms also combine push, pull and autonomous decision behaviors as it has been explained for presence and events mechanisms. Even though the push behavior is frequently used, neither components nor the network are saturated because groups are usually small.

This combination of autonomy of components and direct communication among them (in a peer-to-peer manner) along with the common ownership of resources provides a flexibility that suits the idiosyncrasy of our groups.

4 Validation

As said in section 3.4, LaCOLLA middleware implements the functionalities presented in this paper. We also adapted and implemented some collaborative applications (an instant messaging tool, an asynchronous forum, and a document sharing tool) that benefit from LaCOLLA. These realistic applications helped us to improve the architecture and implementation of LaCOLLA. We have done limited tests with a number of ad-hoc users. All these tests confirm the usefulness of LaCOLLA. The next step is going to be to extend the functionalities of the applications we developed and use them in regular university courses at UOC.

Before implementing LaCOLLA middleware, a simulator was implemented to validate the proposed architecture under several realistic scenarios. The simulator

³ Implemented variant of TSAE algorithm used in events' consistency sessions [5, 6] ensures that.

used J-Sim [12] as network simulator and implemented the UA, RA and GAPA components, virtual synchronism and the internal mechanisms necessary to prove that LaCOLLA behaves, as expected, in an autonomous, decentralized and self-sufficient manner.

Several experiments were done with synthetic workloads with different degrees of dynamism (failures, connections, disconnections or mobility), with different sizes of groups (from 5 to 100 members) and with different degrees of replication (number of RA and GAPA). All components were affected by dynamism.

Simulations had two phases. The first phase simulated a realistic situation. In that phase all internal mechanisms were operative. During this phase members' activity was simulated and components connected, disconnected, moved or failed. The second phase was called repair phase and only internal mechanisms were active. This second phase was used to evaluate how long LaCOLLA required achieving: a) self-organization: all connected components have consistent the information about all internal mechanisms, b) virtual synchronism: all connected components have all events and have consistent the information about available objects, c) presence and location: all connected components have consistent the information about presence and location.

Experiments showed that, in spite of the dynamism and the autonomous and decentralized behavior of components, LaCOLLA required short amount of time (with respect to the rate of changes) to update the information referring to internal mechanisms in all components. Experiments also showed that members knew what was happening in the group and that they had access to the latest versions of objects in a time they perceived as immediate [5].

Figure 4 shows the time required by LaCOLLA (depending on group size) to be self-organized, to provide virtual synchronism, and to have consistent information about presence and location. Note that, for groups of typical size (10 members),

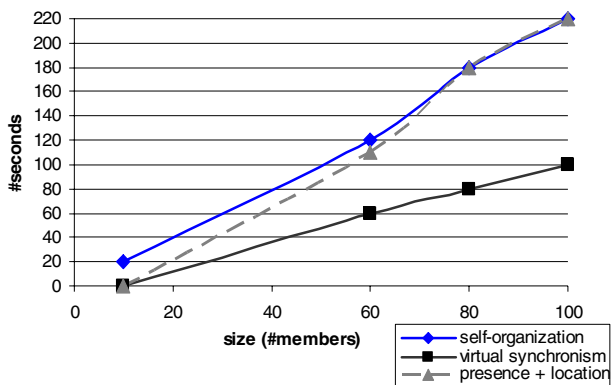


Fig. 4. Simulation results. The figure shows the time required a) to be self-organized, b) to have consistent all information related to virtual synchronism (events and objects) and c) to have consistent the information related to presence (presence and location).

LaCOLLA has good performance: it requires 20 seconds to self-organize, and less than 10 seconds to provide virtual synchronism. It deserves special attention the fact that, even though all components don't have all consistent information about internal mechanisms (self-organization), connected members know all what is happening in the group and have access to the last version of objects (virtual synchronism) in a time that they perceive as immediate. This is due to the decentralized implementation of internal mechanisms and to the fact that non-key mechanisms have long-term consistency policies. In this figure it is also plotted the time required to have consistent presence and location mechanisms because they have a great influence in the achievement of self-organization.

When the size of groups increases, the required time grows, but still maintains low enough values for asynchronous collaboration (e.g. with 60 members: self-organization takes 2 minutes, providing virtual synchronism in 1 minute). This also proves that LaCOLLA can be used in situations where quite large groups require asynchronous sharing capabilities. These values will be further adjusted based on experience with real users using the current middleware with specific applications.

5 Conclusions and Future Work

Asynchronous collaborative applications have to adapt to group idiosyncrasy and interaction style and support the formation of ad hoc collaborative environments for people willing to cooperate using only their own computers, without any additional computing resources (i.e. servers). This requires the autonomy and self-sufficiency that peer-to-peer networks can only offer. We have identified groups as units of resource sharing, by which several individuals dispersed through Internet may spontaneously start to collaborate by just sharing their own computers to form an independent ad hoc community.

In this paper we have described the general characteristics and properties of LaCOLLA, a decentralized, autonomous and self-organized middleware for building collaborative applications that operates with resources provided by their members, that adapts to the idiosyncrasy and to the interaction nature of human groups, and that allows execution of tasks using resources belonging to the group. We also presented the details of current LaCOLLA middleware implementation, paying special attention to its API.

From both building collaborative applications that use LaCOLLA and from using the developed applications we obtained valuable ideas and improvements to introduce in the next versions of LaCOLLA. These new versions will pay special attention to security issues, which are at its minimum expression in the first version; and to introduce new components and mechanisms that will allow mobile devices (PDA, mobile phone, sensors, etc.) become LaCOLLA peers.

We are also planning to use LaCOLLA in real collaborative settings. In that sense, we are planning to use collaborative applications that use LaCOLLA middleware in some collaborative learning practices at UOC. UOC is a virtual university that mediates all relations between students and lecturers through Internet. We think that this kind of collaborative environments where participants never physically meet one to each other will benefit from approaches like the one provided by LaCOLLA,

specially for the degrees of autonomy and self-sufficiency that can be achieved. These real experiences will be of great value for us to further refine the architecture and adjust the implementation of the middleware.

Acknowledgements

Work partially supported by MCYT-TIC2002-04258-C03-03.

References

1. Foster, I.; Kesselman, C.; Tuecke, S. (2001). The Anatomy of the Grid Enabling Scalable Virtual Organizations. Lecture Notes in Computer Science.
2. Navarro L., Marquès J.M., Freitag F. (2004). On distributed Systems and CSCL. The First International Workshop on Collaborative Learning Applications of Grid Technology (CLAG 2004). Held in conjunction with the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2004). April 19 - 22, 2004, Chicago, Illinois, USA. <http://www.ccgird.org/ccgrid2004>.
3. Bote, M., Dimitriadis, Y., Gómez-Sánchez, E. (2003) Grid uses and characteristics: a grid definition. In Proceedings of First European Across Grids Conference, 2003.
4. Hurwicz, M. (2001). Groove Networks: Think Globally, Store Locally. Network Magazine. May 2001.
5. Marquès, J.M. (2003). LaCOLLA: una infraestructura autònoma i autoorganitzada per facilitar la col•laboració. Ph.D. thesis, <<http://people.ac.upc.es/marques/LaCOLLA-tesiJM.pdf>>
6. Golding, R.A. (1992). Weak-consistency group communication and membership. Doctoral Thesis, University of California, Santa Cruz.
7. Saito, Y.; Shapiro, M. (2002). Replication: Optimistic Approaches. Technical Report HPL-2002-33, HP Laboratories, 2002. <<http://www.hpl.hp.com/techreports/2002/HPL-2002-33.html/>>.
8. Carter R. L. (1995). Dynamic server selection in the Internet. In Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95).
9. Verbeke, J.; Nadgir, N.; Ruetsch, G.; Sharapov, I. (2002) Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment. Manish Parashar (Ed.): Grid Computing, Third International Workshop, Baltimore, USA. LNCS 2536 Springer 2002, ISBN 3-540-00133-6. <<http://jngi.jxta.org/>>
10. JXTA: <http://www.jxta.org/>. An overview paper: L. Gong. Project JXTA: A Technology Overview, 2001. <<http://www.jxta.org/project/www/docs/TechOverview.pdf>>.
11. LaCOLLA: <http://lacolla.uoc.edu/lacolla>
12. J-Sim: <http://www.j-sim.org>

Empowering End-Users: A Pattern-Centered Groupware Development Process

Till Schümmer¹, Stephan Lukosch¹, and Robert Slagter²

¹ Fern Universität in Hagen, Computer Science Department, Germany
{till.schuemmer, stephan.lukosch}@fernuni-hagen.de

² Telematica Instituut, The Netherlands
robert.slagter@telin.nl

Abstract. When developing groupware satisfying user requirements is even more difficult than in the context of single-user application development; not only the interaction with the application itself but also the interaction between group members must be respected. Current design methodologies insufficiently focus the designers' attention to this aspect. Therefore, we propose the *Oregon Software Development Process (OSDP)* that fosters end-user participation, structures the interaction between end-users and developers, and emphasizes the use of a shared language between users and developers.

1 Introduction

The lack of end-user participation when designing software can be the source of invalid requirements resulting in low end-user acceptance and inadequate systems; especially for groupware design. In single-user software, the interaction of a user with the application and the domain data has to be supported. Compared to these, groupware systems add a level of complexity, as also the interaction between group members has to be supported.

We argue that this complexity can only be resolved by fostering interaction between end-users and developers in the development process. For an efficient interaction, end-users have to understand the different design alternatives. Additionally, developers and end-users need a shared language for expressing the core concepts of the developed system. Current groupware development process models do not completely solve this problem.

In this paper, we describe the *Oregon Software Development Process (OSDP)* that is based on observations made during *The Oregon Experiment* [1]. The OSDP fosters end-user participation, pattern-oriented transfer of design knowledge, piecemeal growth in form of short iterations, and frequent diagnosis and reflection for improving the application. End-users are empowered to better understand development possibilities for groupware and to act as an expert. A previous publication [2] introduced the core practices of the OSDP. Now, we focus on the relation of OSDP with groupware design.

In the following sections, we will exactly define the requirements for a groupware design process model, analyze existing approaches with regard to the requirements, and show how the OSDP solves these requirements. We will also

discuss experiences that were gathered while developing a collaborative learning environment using the OSDP.

2 Design Theories That Focus on Users

Important contemporary influences of design theories start with Heidegger's discussion of *situated tool use*. According to Heidegger [3], the design of a tool can only take place *in situ* as users are in most cases not aware of the tool. When using, e.g., a car, the driver is focussing on the context, namely the road and the destination. She starts to become aware of the car (as a tool), when her expectations of the car do not match the real behavior, e.g. because of an unexpected noise. Heidegger calls such a situation a *break*. For the case of user-centered design, we can draw three important conclusions from breaks:

1. The unexpected behavior pushes the, up to then unnoticed, tool in the users' attention. Users start to *reflect* on their tool usage and make their expectations explicit.
2. The reflection has the goal of understanding different *requirements* or *forces*. Users make their current expectations explicit and compare these with the current tool behavior. Since the tool does not behave in the expected way, users detect conflicting forces.
3. The context of the break is important since it provides full access to all conflicting and related forces that result in the unexpected behavior.

The understanding of *interrelating forces* has been the most important ground for another influential work in contemporary design: In *Synthesis of Form* [4], Alexander proposed a mathematical approach for defining design problems. He observed that all design problems have the goal of creating an artifact matching the users' expectations regarding form and function in a specific context. Alexander outlined a way for relating the different requirements of a design problem to the requirements which influence each other. He proposed to organize these requirements in a tree structure in which the leaf nodes represent the requirements that have mutual influences.

A designer should address the leafs first and compose solutions from the leafs so that recursively the whole tree of requirements is satisfied. Idealistically, all strong connected sets of requirements (the leafs of the tree structure of problem decomposition) can be solved independently.

An often ignored warning of Alexander was the difficulty to understand all requirements upfront. Satisfying a requirement can in turn be the trigger for new requirements. This problem of changing and evolving requirements was further discussed by Rittel and Webber [5]. They defined a problem where the solution has an impact on the requirements as a *wicked problem*. For such problems, it is difficult to design a solution. Instead, it is important to constantly improve the solution to reduce the number of conflicting forces.

A second warning of Alexander was that the tree decomposition always leads to ignored forces. In *A city is not a tree* [6], Alexander (1965) clarified that

decomposition always needs to investigate the context of the unit of densely connected requirements. In *The timeless way of building* [7], Alexander proposed a new way of addressing connected requirements that puts an emphasis on the context of the problem – the *pattern*. A pattern is a morphological law that explains how to design an artifact in order to resolve a problem in a specific context. Patterns have several properties that inform the design:

- They make the problem explicit instead of just stating a solution. The reader of a pattern (ranging from lay persons to experts) is provided with a description of the problem that helps him to compare the implicitly perceived problem with other known problems.
- They state the conflicting forces and explain how these forces change when applying the solution.

In their book *The Oregon Experiment* [1], Alexander et al. describe how patterns were used in a participatory design process. The approach combined different aspects, that were later separated: *diagnosis and repair*, *piecemeal growth*, and *end-user involvement*.

The basic idea propagated by the Oregon Experiment was to establish a user group that steers the design process of a university campus. To empower users to act as designers the user group agreed on a set of patterns that were considered as relevant given their problem space. Changes to campus buildings were from then on initiated by the users: First the users were encouraged to reflect on their actions and diagnose what features of the building were obstacles for a specific action according to Heidegger’s notion of breaks. Then, users were asked to select relevant patterns that address the observed problem and sketch an improved design. Together with architects, this sketch was refined and implemented if the user group agreed on the proposal. Alexander called this the process of *diagnosis and repair* which is comparable to Schön’s theory of *reflection-in-action* [8].

End-user involvement is the main goal of participatory design [9]. It has its origins in the debate between Scandinavian industry and trade unions regarding the relationship of work and democracy [10]. The goal was to amplify the user’s voice and thereby improve working life. However, although the end-users are in tight interaction with designers, the designers typically do all design activities. Participatory design stresses communication with users throughout the design process, but assumes that the user can play the role of skilled worker. Transfer of design knowledge to users was not in the main focus of participatory design.

2.1 Summary of Requirements

From the above discussion of design theories, we derive the following requirements on a groupware design process model:

R1: The process should encourage users to reflect on their activities and adapt their group environment so that the group task can be better supported (in line with Heidegger’s theory of breaks, Alexander’s theory of diagnosis and repair, and Schön’s theory of reflection in action).

- R2:** The process needs to support iterative extension and modification of functionality so that it can adapt to changing group processes (in line with Heidegger’s and Alexander’s understanding of design as well as the theory of Wicked Problems).
- R3:** The process must involve end-users who shape their own environment since they are the key to conflicting forces (in line with the Oregon Experiment and the school of participatory design).
- R4:** The process needs means to make end-users and developers eloquent enough to express and exchange their personal needs, group needs, or design views (one of the main goals of the pattern approach).
- R5:** The latter requirement implies that end-users have to be empowered to understand and assess the different design decisions in a way that the benefits and drawbacks become clear to the user (which helps them to better understand the wickedness of the problem).
- R6:** The solutions should be adaptable to the socio-technical context like the group structure, the programming language, the communication protocol, or the system environment. This satisfies Alexander’s vision of a design that is deeply situated in the user’s context.
- R7:** Wherever possible, users should be empowered to perform these adaptations on their own. This empowers the user to act in breaks immediately without the need of getting an expert involved.
- R8:** Successful adaptations and new designs have to be made available for all users so that they can be used for solving future problems (in line with the Oregon Experiment).

In the next section, we will review existing software and groupware design approaches and show that none of them supports all these requirements.

3 Existing Approaches

Related work can be grouped in two classes: (1) generic software processes for general software development and (2) the application and appropriation of these processes for the context of groupware development.

3.1 Design of Software

Alexander’s theory of decomposition was adapted by the software development community leading back to the NATO conference on software engineering [11]. It led to *Top-Down* and *Bottom-Up* design which in turn influenced the *Waterfall* method [12]. The waterfall method arranges software projects in different phases. It starts with the design of the requirements and finally reaches the designed and implemented product. The problem with waterfall approaches is that all requirements are collected at the beginning of the project and changes to the requirements should be avoided later on. As Alexander discussed it, this is a problematic understanding of design. Especially, it is in conflict with R2.

Currently, it is widely accepted that iterative development processes are a good means for dealing with changing or uncertain requirements, e.g. the *Spiral Model* [13] that implements *piecemeal growth* (R2). It can, e.g., be found in the *Unified Process* [14] and the *eXtreme Programming* methodology [15]. The latter bridges the gap to end-user involvement by arguing for an on-site customer who interacts with the development team.

End-user involvement and the process of empowering end-users to act as designers has further been discussed in the areas of *participatory design* of software systems, *end-user development*, and *tailorable software*.

In participatory design for software systems, the end-users closely interact with the developers to shape the system [16]. The development however stays in the hands of the software developers (violating R7). As in other contexts, the transfer of knowledge from experts to end-users is not supported. In this sense, we can state that participatory design of software systems requires additional means for educating end-users. It thus satisfies R3 but violates R4.

End-user development extends participatory design with the respect that end-users should be empowered to modify their software on their own (satisfying R7). One technology for end-user development is the provision of tailorable software that allows end-users to make adaptations to the software, while they are using it [17] (satisfying R2, R3, R7).

The challenge is to provide opportunities for tailoring that are appropriate for the people who need to perform the changes. As with other design problems, tailoring requires insight in the problem at hand, the various conflicting forces and possible solutions that is provided by the process. The tailoring approach is thus in conflict with R4 and R6.

To support learning and communication (R4), Gamma et al. [18] applied the idea of design patterns to the application domain of software design. These patterns capture frequently used software patterns for object-oriented design. This initial publication triggered pattern miners from various application domains including human computer interaction [19], hypermedia design [20], or pedagogy [21] and while the first patterns of the gang of four were focussing on designers only, later applications of the pattern idea stated patterns in a way so that they are understandable by developers and end-users and thus can serve as a *Lingua Franca* for design [22] that helps end-users and developers in communication. In this sense, patterns satisfy the requirements for a shared language (R4) and the support for understanding best practices in design (R5).

3.2 Groupware Design

There are only a few groupware specific processes. Dewan proposed a waterfall-like development process for the design and the evaluation of groupware [23]. He focused on the development of research groupware prototypes and their evaluation using self-studies, lab-studies, and field-studies. Although it could be used in an iterative setting, the process brings with it all problems discussed in the previous section, namely incomplete and uncertain requirements (violating R2).

The problem of incomplete requirements is especially relevant for groupware settings, as in addition to the interaction with a set of artifacts the interaction between users has to be supported. Since human interaction should not be understood in a mechanistic way, it is very likely that each user's action will affect other users' expectations of next steps and his own actions. In this sense, Fitzpatrick [24] argued that groupware development is a wicked problem, which implies that waterfall approaches are not suitable for groupware development.

Participatory design methods have become quite prominent in groupware design, e.g. [25], [26], or [27]. They argue that groupware systems have to be designed for modifications to suit evolution of use (satisfying R2 and R3). An important issue of tailoring groupware is the impact of tailoring operations: if a tailoring operation impacts other participants, they may as a group perform the tailoring. In that case, the users need a method to decide on a common tailoring style and a process, how tailored groupware systems are reintegrated in the group [28] and thus encourage sharing of best practices (R5). Most tailoring environments do not consider this.

Tailoring, as well as end-user involvement, has been identified as an important factor in groupware development processes. *Extended STEPS* [29], *OTD* [29] [30], and *SER* [31] represent the state of the art in this area.

Extended STEPS is an iterative participatory development approach initially proposed by Floyd et al. [32] and adapted to the context of tailorable collaborative applications by Wulf and Rohde [29]. The main idea is that end-users collaborate with developers to prepare the embedding of the developed software in the organizational context (R3, R6). This helps to better situate the software in the user's work context. During system use, developers maintain the system and end-users adapt it by means of tailoring (R1, R2). But STEPS does not structure knowledge transfer between the different stakeholders and capturing of knowledge (violating R4, R5, R8). The patterns that fulfilled this role in the Oregon Experiment are not present in STEPS.

OTD is an evolutionary approach to appropriate group processes and technology to support the processes during system use. It fosters reflection (R1) and iterative design (R2). The end-user is involved throughout the process (R3) and performs modifications through tailoring (R7). Finally, qualification for participation is considered as an integral part of the process, which satisfies (R4). Open issues are the sharing of best practices, the ways how changed artifacts are embedded in the environment (violating R6).

SER addresses mutual learning of end-users and developers (R4). It starts with a seeding iteration to capture domain knowledge and requirements. This knowledge is used to perform evolutionary growth (comparable to piecemeal growth) where the software is built (R2). Solutions found in this process are fed back to the knowledge repository (R5). Finally, if the initial set of requirements and captured knowledge does no longer provide answers to current problems, the project moves in a phase of reseeded, where new requirements are gathered and new initial knowledge is captured. SER is close to the process in the Oregon Experiment, but there are important differences: SER does not discuss how

knowledge transfer over projects takes place; it does not discuss how the problems and solutions from previous projects should be captured for reuse. Additionally, it does not make the process of diagnosis and reflection explicit (violating R1).

In summary, we observe that existing processes have focussed on parts of the requirements. But no process for software development or specifically groupware development satisfies all requirements that we consider as needed for an informed and empowered end-user involvement.

4 The Oregon Software Development Process

The Oregon Software Development Process (OSDP) is based on four principles which help to address our requirements:

1. Fostering end-user participation (R1, R3, R7),
2. Pattern-oriented transfer and capturing of design knowledge (R4, R5, R8),
3. Piecemeal growth via short design and development iterations (R2),
4. Frequent diagnosis and reflection on working practices and how the application supports them (R1, R6).

An essential part of this development process is the use of patterns. The patterns used in OSDP follow the pattern structure outlined in [2]. Our evolving collection of groupware patterns currently contains more than 70 patterns at different levels of abstraction (e.g., [33], [34]). Patterns are used as a means to capture and represent design knowledge, but also as a means of communication between end-users, designers and other stakeholders. Given these different types of use, OSDP distinguishes two types of patterns:

1. high-level patterns targeted at end-users, and
2. low-level patterns targeted at software developers.

Low-level patterns deal, e.g., with class structures, control flow, or network communication. High-level patterns focus on the system behavior, as it is perceived by the end-user and empower the end-users to tailor their groupware application to meet their requirements. In the extreme, high-level patterns describe how end-users can compose off-the-shelf components and embed them in their work process. In that case developers no longer need to assist end-users in implementing a pattern which of course requires that developers work according to the OSDP.

4.1 OSDP Iterations

Figure 1 illustrates the three types of iterations that OSDP advocates in a design process (denoted by the three concentric circles). The following paragraphs explain these iterations. As OSDP is an iterative approach, each iteration may actually be performed multiple times; in fact, designers and users may even choose to go back to a previous iteration type.

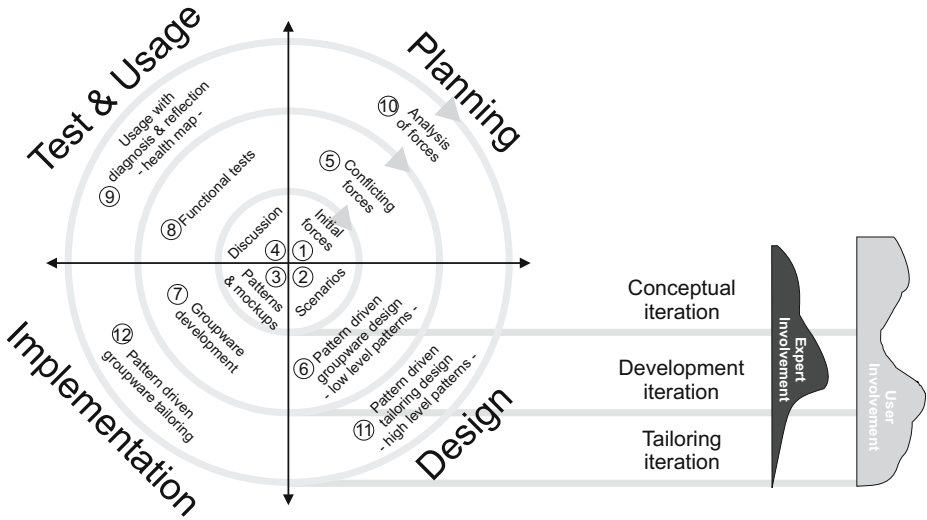


Fig. 1. The Oregon Software Development Process

The purpose of the innermost *conceptual iterations* of OSDP is twofold: first it helps to get an idea of the group processes that need to be supported by the system and thereby defines the scope for selecting potential technical solutions. Second, it helps to form scenario interest groups that later on collaborate on a shared scenario. At the beginning of the iteration, potential users of the groupware system gather and exchange stories of system use (1). This helps them to capture their experience and situate the system under development in their context. There is a major difference between the development of groupware solutions for a well-known group and groupware that should be shipped to a large user group (off-the-shelf groupware): In the first case, the user group for the conceptual iteration emerges naturally from a specific organizational context. In the latter case, a user group has to be carefully composed by analyzing the target market.

Together with a development team, the stories are translated to scenarios of future system use (2). In this design phase of the conceptual iteration, users envision a future system and relate the different parts of the scenarios to groupware patterns. The patterns help to describe the future system by means of high-level prototypes (e.g., paper-based Mock-Ups) (3). They also help to explore related problem areas that are often difficult to identify from outside the context (according to Heidegger and Schön). The whole user group examines the prototypes created by the *scenario interest groups (SIGs)* (4). They decide which scenarios are most relevant for the future development. The result of the conceptual iterations is an ordered list of scenarios that describes what parts of the system should be built. A second result is that SIGs have formed that will tightly interact with the developers in the development iterations.

To check the requirements and a solution direction, the OSDP advocates using short *development iterations*. They are made up from the detection of conflicting forces (5), a pattern-driven design phase (6), the implementation of this design (for instance using development technologies such as frameworks or object-oriented components) (7), and functional tests (8).

Again, the users collaborate in SIGs. They examine their scenario and create task cards describing tasks for the developers (according to the eXtreme Programming planning game). The developer helps the users in breaking down the scenario into tasks that are approximately of equal duration. The description of the tasks is based on high- and low-level patterns. End-users use the pattern catalogue (from the conceptual iteration) to acquire knowledge about the process of design in this stage (6). The pattern language can be extended if new conflicting forces (5) cannot be mapped to any patterns from the current pattern language. This can mean that patterns have been forgotten in the conceptual iterations or that the configuration of forces has not yet been encountered before.

Since more than one SIG can work on the scenarios in parallel, one group member plays the role of the *gardener* (as proposed in [35]). She collects all task cards and sorts them according to their relevance (indicated by the users).

In the *tailoring iterations* end-users collaborate with the developed groupware system. They are encouraged to reflect on their system use (reflection in action). Whenever they encounter a break situation (9), they analyze the conflicting forces (10). A pattern collection helps them by providing recurring problems together with the appropriate solutions (11). If the patterns describe solutions at a high level and if the groupware system supports users in tailoring the system, users can apply the pattern's solution by tailoring (12). Otherwise, they describe the problem using a task card (6) and passing this card on to the gardener. The gardener will in turn resort the collection of task cards in order to provide all users and developers with an updated plan for the system development.

Again, one has to take the focus of the groupware system into account. If the groupware is used in one specific organizational context, it is easy to maintain contacts between users, scenario interest groups, and, if needed, the development team. Off-the-shelf groupware requires a more sophisticated process of collecting and sharing user-made appropriations of the groupware system. In an ideal case, users of the groupware would interact using a repository of best practices for groupware appropriation. However, establishing a community that contributes to and makes use of such a repository exceeds the scope of this paper.

A second role that is important in tailoring iterations is the *pattern scout*. She observes the users and looks for best practices. These best practices are captured as new high-level patterns and added to the group's pattern collection. By that way, the pattern collection evolves to a group memory of design and tailoring knowledge for the specific groupware application.

During a project life-cycle, the different types of iterations will have different importance, as indicated in figure 2. At the project start, the team will start with the conceptual iterations. When first requirements emerge, the development iterations will start, while the conceptual iterations become less important. When

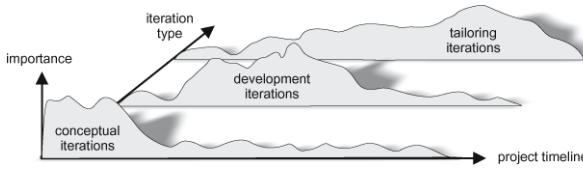


Fig. 2. Frequency of iterations during the project

the development has resulted in a product that end-users can actually use, the tailoring iterations can be started as well. The figure also shows that product development typically decreases over time (for instance, since bugs are fixed) and that the majority of tailoring iterations take place in the first period after the groupware system is introduced. Finally, the tailoring iterations end when the product is replaced.

5 Experiences

Setting. The OSDP described in this article has been applied in the interdisciplinary development project CURE [36]. It was a two year project with the goal of creating a collaborative learning environment for the FernUniversität in Hagen to support different collaborative learning scenarios, e.g. collaborative exercises, tutor-guided groups with collaborative exercises, virtual seminar, virtual lab, and collaborative exam preparation. Instead of providing a different learning environment to each scenario, it was decided to develop a single collaborative learning platform supporting all scenarios and tailorable by the teachers and students to the needs of their actual course.

The project team brought together end-users from different schools, e.g. computer science, psychology, and mathematics, with the software development team. The development team consisted out of four senior employees and three students who contributed to the development. The end-users were university teachers who wanted to use one or more of the above learning scenarios in their teaching and students with the goal of improving their learning conditions.

Applying OSDP. For the *conceptual iterations*, all project members were invited to develop scenarios of system use. The university invited teaching staff and students to share their stories of traditional education and distance education. At this stage, the project members consisted of 12 end-users and 4 developers (R3). After the stories were shared in the whole group, initial scenarios were selected (relating to the stories) and SIGs formed for each scenario. The SIGs were equipped with a set of groupware patterns. For time reasons, the patterns were communicated in an oral presentation. Users could request the printed version as well. Users and developers refined five different learning scenarios that should be supported by the learning environment. The steps of the different scenarios were related to patterns from the pattern collection (R4, R6). The pattern helped to consider related forces (R5).

In the *development iterations*, the SIGs developed more detailed task cards together with the software developers. Patterns served as metaphors for talking about the system (R4) and helped the users to focus on one aspect at a time for each task card (R2). The users were asked to *shop* cards that were considered as most important. This led to a ranking of cards for each scenario. The gardener merged the cards from the different scenarios and the developers implemented the tasks described on the cards by adapting the patterns to their groupware context (R6).

After the first development iteration (approx. 2 weeks of development), teachers started to use the system and entered *tailoring iterations*. They did functional tests of the first prototypes and requested new functionality (R2). The latter was supported by their knowledge of system design that was based on groupware design patterns (R4). They also started to reflect on their activities following the principle of diagnosis (R1). In the early phases of the project, these requests were escalated to the developers. In the later phases where tailoring mechanisms had evolved, they appropriated the system on their own (R7), including the composition of communication and collaboration technology as well as the tailoring of collaboration spaces. The pattern scout started to observe the users at this phase of the project. He looked for best practices and asked the users to share these practices with other users (R8). After the first major release, approx. 300 students started to use the system and were also asked to participate in development iterations and create task cards (R1, R3).

In all phases of the development, the patterns (and especially the stories from the patterns) were very helpful for the participating developers and the developers observed that users started to use the pattern names in their communication. In cases, where the users were familiar with the patterns, they actively proposed implementing specific patterns (R4).

In summary, the application of the OSDP fulfilled all of our requirements. End-users reflected on their activities (R1). Due to the short development iterations end-users early tested a first prototype and requested additional functionality (R2). R3 was fulfilled, as in the beginning of the project three times more end-users than developers actively participated in the requirements iterations. The used pattern language served as a shared language for developers and end-users to discuss the system and to request new functionality (R4 and R5). The developers used the patterns as guideline to implement the identified story cards (R6). R7 was satisfied as the end-users could tailor the resulting system to their own needs. Finally, R8 was satisfied as the pattern scout identified several patterns that were added to the pattern collection.

A Pattern Sequence in the CURE Project. In this section we report on a sequence of patterns as it was used during the iterations of the OSDP when developing the CURE system. Fig. 3 shows parts of the used pattern collection, relations between the patterns (as arrows) and the pattern sequence (visualized by a thick gray line). The numbers show which pattern was used at which point in the process.

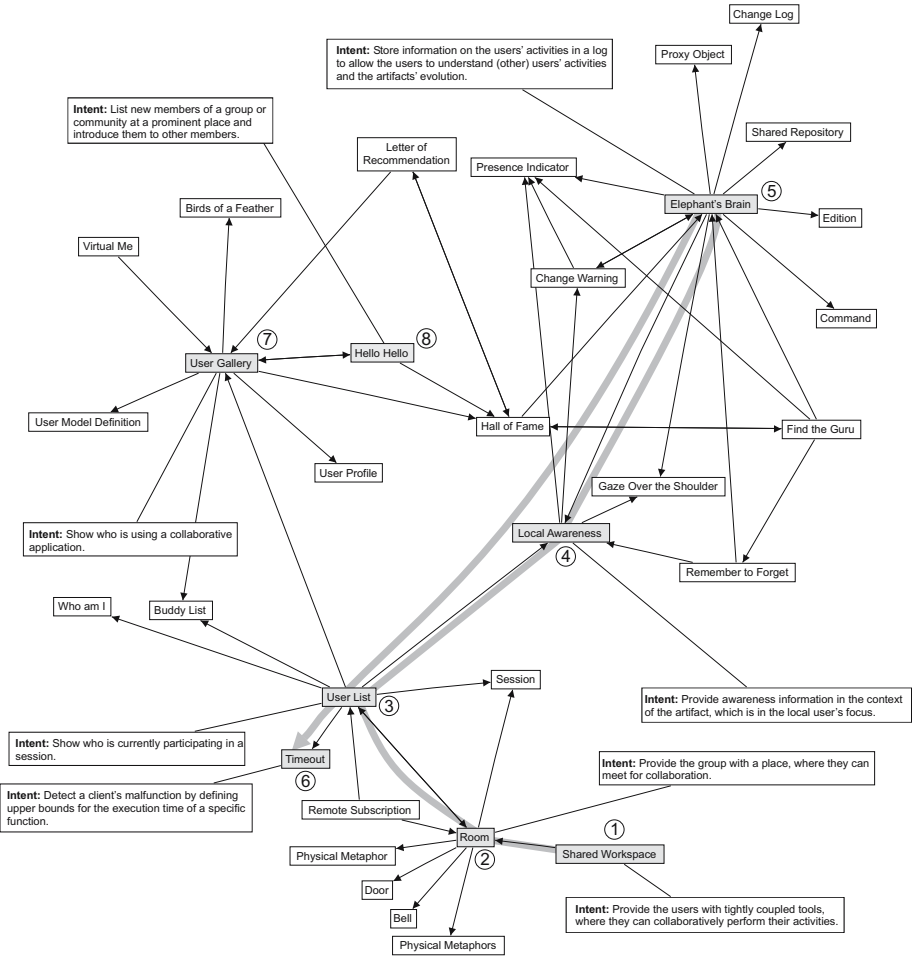


Fig. 3. A part of the pattern collection used in the CURE project

The initial metaphor for CURE was that of a SHARED WORKSPACE ①. Users should be allowed to access, modify, and store pages (the learning material) in a folder structure. The access to the pages should be managed for each folder. Pages may either be directly edited using a simple WIKI-like syntax, or they may contain binary documents or artifacts. In particular, the syntax supports links to other pages, external URLs, or mail addresses. The server stores all artifacts to support collaborative access.

The shared workspace metaphor soon showed to be insufficient for the CURE system. The users and the development team had problems in distinguishing mechanisms for user access management, document storage, and learning material (as content of the documents) and structuring their interaction.

The problems of the shared workspace metaphor were addressed with the ROOM ② pattern. By modeling the learning environment with rooms, users could understand the difference between structuring the virtual learning environment and interacting within the environment. Users, who are in the same room can access all pages that are contained in the room. Changes of these pages are visible to all members in the room. The ROOM ② pattern served as the most important metaphor during the following development. Four months after the development of the first room, a mailbox was added to allow asynchronous communication. Another three months later, the rooms were equipped with a persistent chat that was always visible to users present in the room.

The ELEPHANT'S BRAIN ⑤ was demanded by the developers, not by the end-users. Nevertheless, the forces that led the developers to the Elephant's Brain were brought up by the users. Users wanted to have a USER LIST ③ that visualizes LOCAL AWARENESS ④. To provide local awareness, the CURE system needed means to track a user's current position. Activities performed by a user (read accesses and changes to pages) were stored in a central repository, the ELEPHANT'S BRAIN ⑤.

As mentioned above, users regarded it as important to know who else is currently present in the same room. While discussing the LOCAL AWARENESS ④ pattern, it became clear that users considered it as more important to see the users, who are currently in the same room. The confocal users calculated by the LOCAL AWARENESS ④ pattern were shown in a user list. To keep the data of the USER LIST ③ up to date, the developers brought up the use of the TIMEOUT ⑥ which allows to detect a client's malfunction. In a first iteration, this list was a textual list of user names. In later iterations, the user names were replaced by the user's picture.

This sequence only reports on a small number of patterns used during the different iterations of the OSDP. However, it shows how patterns provide a shared language between developers and end-users to identify requirements and express the core concepts of the developed system.

A Close Look on Tailoring. The effect of tailoring iterations can be illustrated by looking at the user group of the psychology department that used CURE for virtual seminars. They expressed two new needs while using CURE: A better support for familiarizing with their seminar participants and a way to introduce new users to other students. In the following, we present how the psychologists approached these issues during several tailoring iterations.

For supporting the familiarization between seminar participants, the psychologists implemented the USER GALLERY ⑦ by using tailored pages in the group rooms. These pages included standardized sections in which the users were asked to introduce themselves. In a later version, a global USER GALLERY was added to the system by the developers. This example shows how users applied the patterns to shape their group rooms (R7), which then matured to a system component.

For the second issue, the psychologists followed the HELLO, HELLO pattern ⑧. Therefore, they had to define a special welcome area in which the introduction

could take place. The welcome area was defined both by its time (in the early phases of the seminar) and by its location (a special initial page on which the participants should introduce themselves). By providing the users with a location for introducing themselves and with a phase of the seminar, the social process (how to interact in CURE) was adapted to the specific goal (that of the HELLO, HELLO pattern).

6 Conclusions

The lack of end-user participation when designing groupware can lead to invalid requirements as well as a low end-user acceptance of the resulting groupware system. In this paper, we analyzed design theories and their impact in terms of requirements on a groupware design process. We compared these requirements with existing process models for groupware design and showed that none of the existing process models completely fulfills these requirements.

Based on this observation, we proposed the application of the Oregon Software Development Process (OSDP) in the context of groupware development. It fosters end-user participation, pattern-oriented transfer of design knowledge, piecemeal growth in form of short iterations, and frequent diagnosis or reflection that leads to an improved application.

Finally, we showed how OSDP fulfills all identified requirements by reporting on its use in an interdisciplinary groupware development project. During this project, end-users actively participated in the design of the groupware system by communicating their requirements to the developers and reflecting their own context. Especially, during the tailoring iterations the end-users applied patterns from the pattern language in order to adapt their groupware to their emerging needs. The resulting groupware has a high end-user acceptance with currently about 1100 active users.

As mentioned in the description of the process, the development of off-the-shelf groupware complicates the selection and involvement of end-users. We have briefly proposed ways how this could be done. These proposals should be tested in the context of off-the-shelf groupware development.

Additional plans for future work include the application of OSDP in industrial settings. We are also experimenting with the process's applicability in educational settings. Both settings will help us to further investigate the process' impact on educating end-users and novice developers in groupware development.

References

1. Alexander, C., Silverstein, M., Angel, S., Ishikawa, S., Abrams, D.: *The Oregon Experiment*. Oxford University Press, New York (1980)
2. Schümmer, T., Slagter, R.: *The oregon software development process*. In: *Proceedings of XP2004*. (2004)
3. Heidegger, M.: *Sein und Zeit*. 17 (1993) edn. Niemeyer, Tübingen (1927)
4. Alexander, C.: *Notes on the Synthesis of Form*. 7 (2002) edn. Harvard University Press, Cambridge, Massachusetts (1964)

5. Rittel, H.W.J., Webber, M.M.: Dilemmas in a general theory of planning. *Policy Sciences* **4** (1973) 155–169
6. Alexander, C.: A city is not a tree. *ARCHITECTURAL FORUM* **122** (1965) 58–62
7. Alexander, C.: *The timeless way of building*. Oxford University Press, New York (1979)
8. Schön, D.A.: *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York (1983)
9. Schuler, D., Namioka, A.: *Participatory design: Principles and Practices*. Erlbaum, Hillsdale N.J. (1993)
10. Bjerknes, G., Bratteteig, T.: User participation and democracy: a discussion of scandinavian research on systems development. *Scand. J. Inf. Syst.* **7** (1995) 73–98
11. Naur, P., Randell, B., eds.: *Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, Brussels (1969)*, Scientific Affairs Division (1968)
12. Boehm, B.W.: Software engineering. *IEEE Trans. Computers* (1976) 1226–1241
13. Boehm, B.W.: A spiral model of software development and enhancement. *IEEE Computer* **21** (1988) 61–72
14. Jacobson, I., Booch, G., Rumbaugh, J.: *Unified Software Development Process*. Addison-Wesley (1999)
15. Beck, K.: *eXtreme Programming Explained*. Addison Wesley, Reading, MA, USA (1999)
16. Muller, M.J., Kuhn, S.: Participatory design. *Communications of the ACM* **36** (1993) 24–28
17. Kahler, H., Mørch, A., Stiemerling, O., Wulf, V.: Tailorable systems and cooperative work (introduction). *Special Issue of Computer Supported Cooperative Work* **9** (2000)
18. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA (1995)
19. Borchers, J.: *A Pattern Approach to Interaction Design*. John Wiley and Sons Ltd. (2001)
20. Rossi, G., Garrido, A., Carvalho, S.: Design patterns for object-oriented hypermedia applications. In Vlissides, J.M., Coplien, J.O., Kerth, N.L., eds.: *Pattern Languages of Program Design 2*. Addison-Wesley Addison-Wesley, Reading, MA, USA (1995) 177–191
21. Eckstein, J.: Workshop report on the pedagogical patterns project: Successes in teaching object technology. In: *Proceedings of OOPSLA'99 Educator's Symposium, Denver (1999)*
22. Erickson, T.: Lingua francas for design: sacred places and pattern languages. In: *Proceedings of the conference on Designing interactive systems, ACM Press (2000)* 357–368
23. Dewan, P.: An integrated approach to designing and evaluating collaborative applications and infrastructures. *Computer Supported Cooperative Work* **10** (2001) 75–111
24. Fitzpatrick, G.A.: *The Locales Framework: Understanding and Designing for Cooperative Work*. PhD thesis, Department of Computer Science and Electrical Engineering, The University of Queensland (1998)
25. Grudin, J.: Groupware and social dynamics: eight challenges for developers. *Communications of the ACM* **37** (1994) 92–105

26. Slagter, R., Biemans, M., Ter Hofte, G.H.: Evolution in use of groupware: Facilitating tailoring to the extreme. In: Proceedings of the CRIWG Seventh international Workshop on Groupware, Darmstadt, IEEE Computer Society Press (2001) 68–73
27. Teege, G.: Users as composers: Parts and features as a basis for tailorability in cscw systems. *Computer Supported Cooperative Work (CSCW)* **9** (2000) 101–122
28. Fernandez, A., Haake, J.M., Goldberg, A.: Tailoring group work. In Haake, J.M., Pino, J.A., eds.: Proceedings of the CRIWG'2002 – 8th International Workshop on Groupware. LNCS 2440, La Serena, Chile, Springer-Verlag Berlin Heidelberg (2002) 232–242
29. Wulf, V., Rohde, M.: Towards an integrated organization and technology development. In: DIS '95: Proceedings of the conference on Designing interactive systems, ACM Press (1995) 55–64
30. Wulf, V., Krings, M., Stiemerling, O., Iacucci, G., Fuchs-Fronhofen, P., Hinrichs, J., Maidhof, M., Nett, B., Peters, R.: Improving inter-organizational processes with integrated organization and technology development. *Journal of Universal Computer Science* **5** (1999) 339 – 365
31. Fischer, G., Grudin, J., McCall, R., Ostwald, J., Redmiles, D., Reeves, B., Shipman, F.: Seeding, evolutionary growth and reseeded: The incremental development of collaborative design environments. In Olson, G., Malone, T., Smith, J., eds.: *Coordination Theory and Collaboration Technology*, Lawrence Erlbaum Associates (2001) 447–472
32. Floyd, C., Reisin, F.M., Schmidt, G.: Steps to software development with users. In: ESEC '89: Proceedings of the 2nd European Software Engineering Conference, Springer-Verlag (1989) 48–64
33. Lukosch, S., Schümmer, T.: Patterns for managing shared objects in groupware systems. In: Proceedings of the 9th European Conference on Pattern Languages and Programs, Irsee, Germany (2004) 333–378
34. Schümmer, T.: Patterns for building communities in collaborative systems. In: Proceedings of the 9th European Conference on Pattern Languages of Programs (EuroPLoP'04), Irsee, Germany, UVK, Konstanz, Germany (2004) 379–440
35. Rittenbruch, M., McEwan, G., Ward, N., Mansfield, T., Bartenstein., D.: Extreme participation - moving extreme programming towards participatory design. In Binder, T., Gregory, J., , Wagner, I., eds.: *Participation and Design: Inquiring Into the Politics, Contexts and Practices of Collaborative Design Work – PDC 2002 Proceedings of the Participatory Design Conference*, Malmo, Sweden (2002)
36. Haake, J.M., Haake, A., Schümmer, T., Bourimi, M., Landgraf, B.: End-user controlled group formation and access rights management in a shared workspace system. In: CSCW'04: Proceedings of the 2004 ACM conference on Computer supported cooperative work, Chicago, Illinois, USA, ACM Press (2004) 554–563

Integrating Synchronous and Asynchronous Interactions in Groupware Applications*

Nuno Preguiça, J. Legatheaux Martins, Henrique Domingos, and Sérgio Duarte

CITI/DI, FCT, Universidade Nova de Lisboa,
Quinta da Torre, 2845 Monte da Caparica, Portugal

Abstract. It is common that, in a long-term asynchronous collaborative activity, groups of users engage in occasional synchronous sessions. In this paper, we discuss the data management requirements for supporting this common work practice. As users interact in different ways in each setting, requirements and solutions often need to be different. We present a data management system that allows to integrate a synchronous session in the context of a long-term asynchronous interaction, using the suitable data sharing techniques in each setting and an automatic mechanism to convert the long sequence of small updates produced in a synchronous session into a large asynchronous contribution. We exemplify the use of our approach with two multi-synchronous applications.

1 Introduction

Groupware applications are commonly classified as synchronous or asynchronous depending on the type of interaction they support. Synchronous applications support closely-coupled interactions where multiple users synchronously manipulate the shared data. In synchronous sessions, all users are *immediately* notified about the updates produced by other users. At the data management level, it is usually necessary to maintain multiple copies of the data synchronized in realtime, merging all concurrent updates produced by the users. Several general-purpose systems have been implemented [25,28,26].

Asynchronous applications support loosely-coupled interactions where users modify the shared data without having *immediate* knowledge of the updates produced by other users. At the data management level, it is common to support a model of temporary divergence among multiple, simultaneous streams of activity [4] and to provide some mechanism to automatically merge these streams of activity. Some general-purpose (e.g. [18,5]) and application-specific (e.g. [17] for document editors) systems have been implemented.

A common work practice among groups of individuals seeking a common goal is to alternate periods of closely-coupled interaction with periods of loosely-coupled work. During the periods of closely-coupled interaction, group elements can coordinate and create joint contributions. Between two periods of close interaction, individuals tend to produce their individual contributions in isolation.

* This work was partially supported by FCT/MCTES through POSI/FEDER.

In this paper, we address the data management problems of supporting this type of work practice in groupware applications, dubbed as multi-synchronous applications. We describe the three main mechanisms we have used to add support for synchronous sessions in the DOORS system [21], a replicated storage system designed to support asynchronous groupware.

First, a mechanism to allow applications to synchronously manipulate the data stored in the data management system. Second, a mechanism that allows to use different reconciliation and awareness techniques in each setting, as needed by some applications (e.g.: text editing systems tend to use operational transformation [8] in synchronous settings, and versioning [2,3] in asynchronous settings). Finally, a mechanism to automatically convert long sequences of synchronous operations into a small sequence of asynchronous operations. This mechanism is needed to accommodate the difference of granularity in the operations used in each setting (e.g. in text editing systems, *insert/remove character* operations are used in synchronous settings, and *update text line/paragraph/section* operations are usually used in asynchronous settings).

The remainder of this paper is organized as follows. Section 2 discusses the requirements and presents the design choices used for supporting applications in synchronous and asynchronous settings. Section 3 present the DOORS system, detailing the integration of synchronous and asynchronous interactions. Section 4 presents multi-synchronous applications implemented in our system. Section 5 discusses related work and Sect. 6 concludes the paper with some final remarks.

2 Design Options

In this section we present the design options used to integrate synchronous interactions in an object-based system designed to support the development of asynchronous groupware applications. We start by reviewing the basic requirements that must be addressed to support each type of interaction independently.

2.1 Basic Requirements and Design Options

Synchronous Interaction: In synchronous applications, users access and modify the shared data in realtime. To this end, a common approach is to allow several applications running on different machines to maintain replicas of the shared data. When an update is executed in any replica, it must be immediately propagated to all other replicas. To achieve this requirement, our support for synchronous replication lies on top of a group-communication infrastructure and includes support for latecomers, as it is usual in synchronous groupware.

The user interface of the synchronous application must be updated not only when the local user updates the shared data, but also whenever any remote user executes an update. To this end, our system allows applications to register callbacks for being notified of changes in the shared data. These callbacks are used to update the GUI of the application. This approach allows a synchronous application to be implemented using the popular model-control-view pattern, with the model replicated in all participants.

Asynchronous Interaction: In asynchronous interactions, users collaborate by accessing and modifying shared data. To maximize the chance for collaboration, it is usually important to allow users to access and modify the shared data without restrictions (besides access control restrictions). To provide high data availability, our system combines two main techniques. First, it replicates data in a set of servers to mask network and server failures. Second, it partially caches data in mobile clients to mask disconnections. High read and write availability is achieved using a “read any/write any” model of data access that allows any clients to modify the data independently.

This optimistic approach leads to the need of handling divergent streams of activity (caused by independent concurrent updates executed by different users). Several reconciliation techniques have been proposed in different situations (e.g. the use of undo-redo [15], versioning [3], operational transformation [8,30,31], searching the best solution relying on semantic information [23]) but no single technique seems appropriate for all problems. Instead, different groups of applications call for different strategies. Thus, unlike most systems [7,3,18] that implement a single customizable strategy, our system allows different applications to use different reconciliation techniques.

Awareness has been identified as important for the success of collaborative activities because individual contributions may be improved by the understanding of the activities of the whole group [6,12]. Our system includes an integrated mechanism for handling awareness information relative to the evolution of the shared data. Different strategies can be used in different applications, either relying on explicit notification, using a shared feedback approach [6], or combining both styles. Further details on the requirements and design choices for asynchronous groupware in mobile computing environments are presented elsewhere [21].

2.2 Integrating Synchronous and Asynchronous Interactions

An asynchronous groupware activity tends to span over a long period of time. During this period, each participant can produce his contributions independently. Groups of participants can engage in synchronous interactions to produce a joint contribution. Thus, it seems natural to consider the result of a synchronous interaction as a contribution in the context of the long-term collaborative process. We address the specific requirements for implementing this strategy in our object-based system in the remaining of this section.

Updates with Different Granularities: Some applications use operations with different granularities in synchronous and asynchronous settings. For example, consider collaborative editing systems¹. Synchronous editors (e.g. Grove [8], REDUCE [32]) allow multiple users to modify a shared document by executing operations to insert or remove a single character. These operations are immediately propagated and executed in all users’ replicas. In contrast, systems for

¹ Similar situations occur for other applications (e.g. conferencing systems, graphical editors), as discussed in [22].

asynchronous settings (e.g.: CVS [3], Iris [17]) tend to use a copy-modify-merge paradigm, where reconciliation of divergent replicas is achieved by considering updates on large regions (e.g.: lines in CVS and document elements in Iris).

One reason for this situation is the difference in the level of expected awareness. In synchronous settings, users expect to have immediate knowledge about all other users' updates. Thus, all update operations must be propagated. In asynchronous settings, users are expected to work in isolation without having immediate knowledge of the modifications produced by other users. Therefore, coarse-grain updates can be propagated when a user finishes a working session.

Two additional reasons exist. The first is related with the reconciliation techniques used in each setting and it will be discussed later. The second reason is related with the technical difficulty of managing a very large number of small operations. For each operation, an excessive amount of data is created (including the type and parameters of the operation and information to order and trace dependencies among operations – the problem of reducing the information needed to trace dependencies is only partially addressed in [27]). This poses problems in terms of storage, network bandwidth and complexity of the reconciliation process.

The above reasons suggest that the granularity of operations used in each setting should be different: small for synchronous settings and large for asynchronous settings. To this end, our system includes a mechanism to compress the log of *small* operations executed by users. During a synchronous interaction, the *small* operations are incrementally converted and compressed in a small sequence of *large* operations in background. This sequence of *large* operations is the result of the synchronous session and it is integrated in the asynchronous collaborative process as any contribution produced by a single user.

Different Reconciliation and Awareness Techniques: In some applications, different reconciliation and awareness techniques are used in synchronous and asynchronous settings. For example, in collaborative text editors, operational transformation [8,30,14] has become the reconciliation technique of choice in synchronous mode while versioning [3,18,2] is used in asynchronous mode. To understand the reason for this difference, it is important to understand the limitations of each technique and how users interact to overcome such limitations.

It is known that operational transformation can lead to semantic inconsistencies [30,19] when concurrent updates are executed. The following example illustrates the problem. Suppose that a document contains: *There will be student here*. In this text there is a grammatical error that can be corrected by replacing “student” by “a student” or “students”. If two users concurrently execute these different changes, operational transformation leads to: *There will be a students here*. The result is semantically incorrect, as it contains a new error. Moreover, the merged version does not represent any of the users' solutions and it is likely that it does not satisfy any of the users.

In synchronous settings, this problem can be easily solved as users immediately observe all concurrent modifications. Thus, users can coordinate themselves and immediately agree on the preferred change. This is only possible because

users have strong and fine-grain awareness information about the changes produced by other users. In this case, the automatic creation of multiple versions to solve conflicts would involve unnecessary complexity. Moreover, it is not clear which user interface widgets to use for presenting these multiple versions.

In asynchronous settings, updates are not immediately merged and each contribution tends to be large. Thus, as users have no (strong) awareness information about the updates produced by other users, it is likely that using operational transformation to merge concurrent updates to the same semantic unit would lead to many semantic inconsistencies. This is the main reason for not using this technique in asynchronous editing systems: it seems preferable to maintain multiple semantically correct versions and let users merge them later, instead of a single semantically incorrect version that does not satisfy anyone.

Regarding awareness, the difference in the used techniques is an immediate consequence of the coupling degree. In synchronous settings, users must have immediate feedback about other users' actions. Thus, very accurate and detailed information must be constantly disseminated and presented to users. In asynchronous settings, it is common that users only need to know what changes have been produced recently (and what users may be editing the document). Thus, it is often sufficient to maintain with each document a log that describes the changes produced by each user in each isolated working-session (e.g. CVS [3]).

A system that supports synchronous and asynchronous interactions should accommodate different reconciliation and awareness techniques for each settings. To this end, we structure data objects used in collaborative applications according to an object framework that includes independent components to handle most aspects related with data sharing, including reconciliation and awareness management. Thus, for each data type, the programmer may specify a different technique (component) to be used in each setting.

As discussed earlier, our system allows to use operations with different granularities in each setting by automatically converting the operations. This approach is important for reconciliation as the techniques used in each setting expect operations with different granularities. It is also important for awareness support, as the granularity of awareness information needed in each setting is closely related with the granularity of operations. In our system, the awareness component handles the awareness information produced when an operation is executed.

3 DOORS

In this section, we start by briefly presenting the DOORS system architecture and the DOORS object framework. A more detailed description, discussing support for asynchronous groupware, can be found in [21]. Then, we detail the integration of synchronous sessions in the overall asynchronous activity.

3.1 Architecture

DOORS is a distributed object store based on an “extended client/replicated server” architecture. It manages coobjects: objects structured according to the

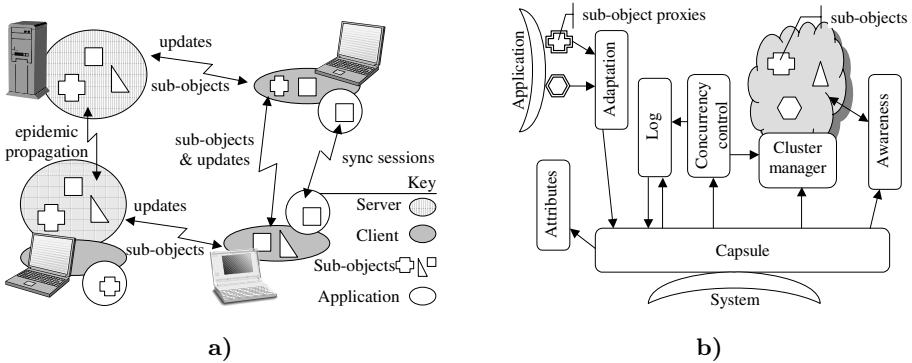


Fig. 1. DOORS architecture (a) with four computers with different configurations. Coobjects are replicated by servers, partially cached by clients and manipulated by applications. Coobjects are structured according to the DOORS object framework (b).

DOORS object framework. A coobject represents a data type designed to be shared by multiple users, such as a structured document or a shared calendar. A coobject is designed as a cluster of sub-objects, each one representing part of the complete data type (e.g. a structured document can be composed by one sub-object that maintains the structure of the document and one sub-object for each element of the structure). Each sub-object may still represent a complex data structure and it may be implemented as an arbitrary composition of common objects. Besides the cluster of sub-objects, a coobject contains several components that manage the operational aspects of data sharing — Fig. 1.b depicts the approach (we describe each component and how they work together later).

Figure. 1.a depicts the DOORS architecture, composed by servers and clients. Servers replicate workspaces composed by sets of related coobjects to mask network failures/partitions and server failures. Server replicas are synchronized during pair-wise epidemic synchronization sessions. Clients partially cache key coobjects to allow users to continue their work while disconnected. A partial copy of a coobject includes only a subset of the sub-objects (and the operational components needed to instantiate the coobject). Clients can obtain partial replicas directly from a server or from other clients.

Applications run on client machines and access data using a “get/modify locally/put changes” model. First, the application obtains a private copy of the coobject (from the DOORS client). Second, it invokes sub-objects’ methods to query and modify its state – update operations are transparently logged in the coobject. Finally, if the user chooses to save her changes, the logged sequence of operations is (asynchronously) propagated to a server.

When a server receives operations from a client, it delivers the operations to the local replica of the coobject. It is up to the coobject replica to store and process these operations. Servers synchronize coobject replicas by exchanging unknown operations during pairwise epidemic synchronization sessions.

3.2 DOORS Object Framework

As outlined above, the DOORS system core executes minimal services and it delegates on the coobjects most of the aspects related with data sharing, including reconciliation and the handling of awareness information. To help programmers to create new applications reusing *good* solutions, we have defined an object framework that decomposes a coobject in several components that handle different operational aspects (see Fig. 1.b). We now outline this object framework, introducing each component in the context of the local execution of an operation.

Each coobject is composed by a set of *sub-objects* that may reference each other using sub-object proxies. These sub-objects store the internal state and define the operations of the implemented data-type. The *cluster manager* is responsible to manage the sub-objects that belong to the coobject.

Applications always manipulate a coobject using sub-objects' proxies. When an application invokes a method on a *sub-object proxy*, the proxy encodes the method invocation (into an object that includes all needed information) and hands it over to the adaptation component. The *adaptation component* is responsible for interactions with remote replicas. The most common adaptation component executes operations locally.

The *capsule component* controls local execution of operations. Queries are immediately executed in the respective sub-object and the result is returned to the application. Updates are logged in the *log component*. When an operation is logged, the capsule calls the concurrency control component to execute it.

The *concurrency control/reconciliation* component is responsible to execute the operations stored in the log. In the client, operations are usually executed immediately. The result of this execution is tentative [7]. An update only affects the *official* state of a coobject when it is finally executed in the servers. In [21], we have discussed extensively how to use different reconciliation strategies (components) in the context of asynchronous groupware applications.

The execution of an operation may produce some awareness information. The *awareness component* immediately processes this information (e.g. by storing it to be later presented in applications and/or propagating it to the users).

Besides controlling operation execution, the capsule defines the coobject's composition. The composition described in this subsection represents a common coobject, but different compositions can be defined. The capsule implements the interface used by the system to access the coobject. The *attributes component* stores the system and type-specific properties of the coobject.

To create a new data-type (coobject) the programmer must do the following. First, he must define the sub-objects that will store the data state and define the operations (methods) to query and to change that state. From the sub-objects' code, a pre-processor generates the code of sub-object proxies and factories used to create new sub-objects, handling the tedious details automatically. Second, he must define the coobject composition, selecting the suitable pre-defined components (or defining new ones if necessary). Different components can be specified for use in the server and in the client during private and shared (synchronous) access. Different data-sharing semantics are obtained using different components.

3.3 Integration of Synchronous Sessions

In this subsection we detail the integration of synchronous sessions in the overall asynchronous activity.

Manipulate Coobjects in Synchronous Sessions: Each site that participates in a synchronous session usually maintains its own copy of the shared data. To this end, we need to maintain several copies of a coobject synchronously synchronized.

To achieve this goal, we use the synchronous adaptation component that propagates updates executed in any replica to all replicas. This component relies on a group communication sub-system (GCSS) – JGroups [1] in the current implementation – for managing communications among session participants.

An application (user) may *start a synchronous session* in a client when it loads a coobject from the data storage. In this case, the coobject is instantiated with the components specified for shared access in the client. In particular, a version of the synchronous adaptation component must be used. This component creates a new group (in the GCSS) for the synchronous session.

When a new user wants to *join a synchronous session*, the user's application has to join the group for the synchronous session (using the name of the session and the name of one computer that participates in the session). During this process, the application receives the current state of the coobject (relying on the state transfer mechanism of the GCSS) and creates a private copy of the coobject. Any user is allowed to *leave the synchronous session* at any moment.

In each group there is a designated primary (that can change during the group lifetime). Besides being responsible to save the result of the synchronous session, the primary plays an important role in the instantiation of sub-objects. When the cluster manager of any replica needs to instantiate a new sub-object, it asks the primary to send the initial state of the sub-object (as obtained from the DOORS client) to all replicas. This approach guarantees that all replicas instantiate all sub-objects in a coherent way.

Applications manipulate coobjects by executing operations in sub-objects' proxies, as usual. The proxy encodes the operation and delivers it to the adaptation component for processing. Query operations are processed locally as usual. For an update operation, the adaptation component propagates the operation to all elements of the synchronous session using the GCSS (step 2 of Fig. 2).

The GCSS may deliver operations in the same total order or in FIFO order to all replicas. When the operation is received in (the adaptation component of) a replica, including the replica where it has been initially executed, its execution proceeds as usual (by handing the operation to the capsule for local execution, as explained in Sect. 3.2). When total order is used, replicas are kept consistent by simply executing all operations by the order they are received. When FIFO order is used, no delay is imposed on local operations, but replicas receive operation in different order. Thus, it is usually necessary to use an operational transformation reconciliation component to guarantee replica convergence.

To *update the application GUI*, an application may register callbacks in the adaptation component to be notified when sub-objects are modified due to op-

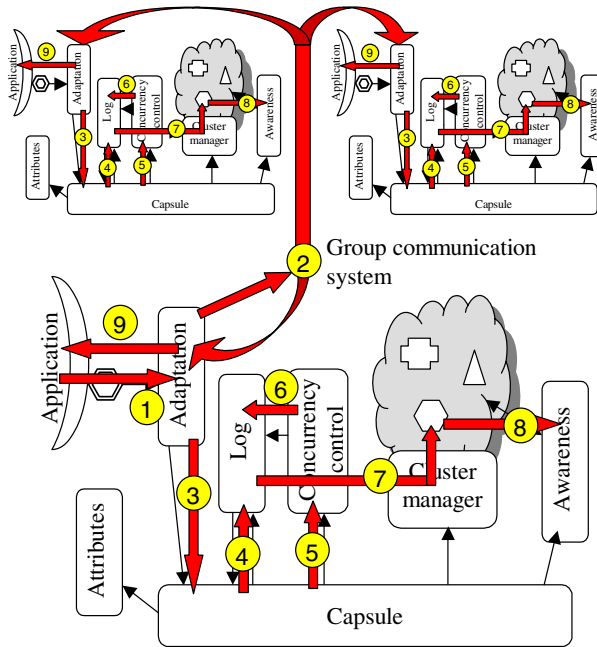


Fig. 2. Synchronous processing of an update operation in three replicas of a coobject

erations executed by remote users (or local users). These callbacks are called by the adaptation component when the execution of an operation ends (step 9).

The DOORS approach to manage synchronous interactions, described in this subsection, does not imply any contact with the servers. An application running on a DOORS client can participate in a synchronous session if it can communicate with other participants using the underlying GCSS. Thus, a group of mobile clients, disconnected from all servers, may engage in a synchronous interaction even when they are connected using an ad hoc wireless network.

Saving the Result of a Synchronous Interaction as an Asynchronous Contribution: As discussed in Sect. 2.2, some applications need to convert the *small* operations used in synchronous mode into the *large* operations used in asynchronous mode.

In the DOORS system, this is achieved by the log compression mechanism implemented by the log component. As described in Sect. 3.2, all update operations executed in a synchronous session are stored in the log before being executed. Besides the full sequence of operations, the log component also maintains a compressed version of this sequence. An operation is added to the compressed sequence after being stably executed (and after the reconciliation component executes the last undo or transformation to the operation) using the algorithm presented in Fig. 3. This process is executed in background to have minimal impact on the performance of the synchronous session.

```

Compress (seqOps: list, newOp: operation) =
  FOR i:= seqOps.size - 1 TO 0 DO
    IF Compress( seqOps, i, newOp) THEN RETURN seqOps
    ELSE IF NOT Commute( seqOps.get(i), newOp) THEN BREAK
  END FOR
  seqOps.add( ConvertToLarge( newOp))
  RETURN seqOps

```

Fig. 3. Algorithm used for log-compression

The basic idea of the algorithm is to find out an operation already in the log that can compress the new operation (e.g. an insert/remove operation in a text element can be integrated into an operation that sets a new value to the text element by changing the value of the text). If no such operation exists, the new operation is converted into an asynchronous operation and logged (e.g. an insert/remove operation can be converted into an operation that sets a new value to the text element – the value of the text after being modified).

To use this approach, the coobject must define the following methods of the compression algorithm: *Compress*, for merging two operations; *Commute*, for testing if the result of executing two operations does not depend on the execution order; *ConvertToLarge*, for converting a small *synchronous* operation into a large *asynchronous* operation. The examples presented in the next section show that these methods are usually simple to write.

The result of the synchronous session is the compressed sequence of operations. Only the designated primary can save the result of the session. In respect to the overall evolution of the coobject, the sequence of operations is handled in the same way as the updates executed asynchronously by a single user. Thus, the sequence of operations is propagated to the servers, where it is integrated according to the reconciliation policy that the coobject uses in the server.

Using Different Reconciliation and Awareness Strategies: As discussed in Sect. 2.1, some applications need to use different reconciliation and awareness techniques during synchronous and asynchronous interactions. In our system, different techniques can be used by specifying that a coobject is composed by different components in the server and during shared access in the client.

The reconciliation and awareness components, defined for use during shared access, control data evolution and awareness in the synchronous session. The reconciliation and awareness components, defined for use in the servers, control behavior during asynchronous interactions, i.e., how stable replicas stored in the servers evolve and what awareness information is maintained.

4 Applications

In this section, we present two applications that exemplify our approach to integrate synchronous and asynchronous interactions. These applications and the DOORS prototype have been implemented in Java.

4.1 Multi-synchronous Document Editor

The multi-synchronous document editor allows users to produce structured documents collaboratively — these documents are represented as coobjects. For example, users may use a synchronous session to discuss and create the outline of the document and to edit controversial parts. Each user may, after that, asynchronously produce his contributions editing the sections he is responsible.

A document is a hierarchical composition of containers and leaves. Containers are sequences of other containers and leaves. A single sub-object stores the complete structure of a document, including all containers. Leaves represent atomic units of data that may have multiple versions and different data types. A sub-object that extends the multi-version sub-object stores each leaf.

For example, a LaTeX document has a root container with text leaves and scope containers. A scope container may also contain text leaves and scope containers. Scope containers can encapsulate the document structure but they have no direct association with LaTeX commands. For example, a paper can be represented as a sequence of scope elements, one for each section (see Fig. 4). The file to be processed by LaTeX is generated by serializing the document structure.

Asynchronous Edition: During asynchronous edition, users can modify the same elements independently. The coobject maintains syntactic consistency automatically, as follows. Concurrent updates to the same text leaf are merged using the pre-defined strategy defined in its super-class: two versions are created if the same version is concurrently modified; a remove version is ignored if that version has been concurrently modified; otherwise, both updates are considered. Users should merge multiple versions later. Concurrent changes to the same container are merged by executing all updates in a consistent way in all replicas (using an optimistic total order reconciliation component in the server).

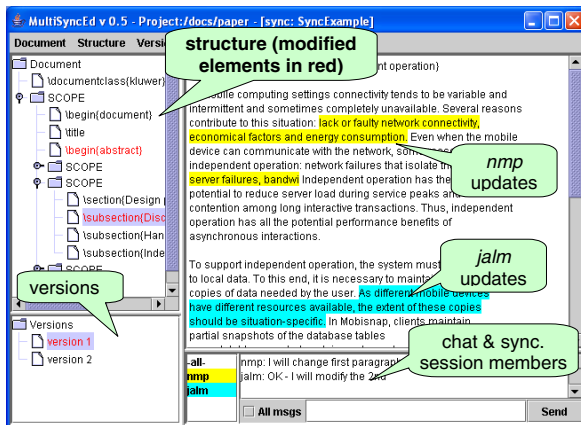


Fig. 4. Multi-synchronous document editor with a LaTeX document, while synchronously editing one section

Synchronous Edition: The multi-synchronous editor allows multiple users to synchronously edit a document. To this end, a document coobject is maintained synchronously synchronized using the synchronous adaptation component that immediately executes operations locally. Thus, users observe their operations without any delay. For handling reconciliation during a synchronous session, a reconciliation component that implements the GOTO operational transformation algorithm [30] is used.

For supporting synchronous edition, a text element also implements operations to insert/remove a string in a given version. These operations are submitted when the user writes something in the keyboard or executes a cut or paste operation. Remote changes are reflected in the editor's interface using the callback mechanism provided by the adaptation component. For example, Fig. 4 shows a synchronous session with two users. The selected text version presents updates from each user with a different color. In the structure and versions windows, elements that have been modified in the current session are presented in red.

For converting *synchronous* operations into *asynchronous* operations, the following rules are used. Operations *commute* if they act upon different structure elements or different versions. Otherwise, they do not commute. The update version operation *compresses* insert/remove string operations — the new value of the version is updated to reflect the insert/remove operations. No other compression rule is needed for converting a synchronous session into an asynchronous contribution². An insert/remove operation can be *converted to a large* update version operation, where the new value of the version is the result of applying the given operation to the current state of the version.

4.2 Multi-synchronous Conferencing Tool

In this section we describe a *conferencing* tool that allows to integrate discussions produced in a chat tool as posts in a message board, thus allowing to maintain an integrated repository of synchronous and asynchronous messaging interactions produced in the context of some workgroup.

This application maintains a newsgroup-like shared space where users can post messages asynchronously. A shared space is used to discuss some topic and it may include multiple threads of discussion. A shared space is represented as a coobject and each thread is stored in a single sub-object. In each shared space, there is an additional sub-object that indexes all threads of discussion.

Two operations are defined: create a new thread of discussion with an initial message and post a (reply) message to an existing thread. The following reconciliation strategy is used in the servers: all updates are executed in all replicas using a causal order. This approach guarantees that all *reply* messages are stored in all replicas before the original message, but it does not guarantee that all messages are stored in the same order – this is usually considered sufficient in this context.

² Additional compression rules are applied as part of the normal log compression mechanism: create/delete version pairs are removed; add/remove element pairs are removed; an update version replaces a previous update version.

Our tool also allows users to maintain several replicas of a shared space synchronously synchronized. This is achieved using the synchronous adaptation component, as before. The reconciliation component executes all operations immediately in a causal order (as in the servers). During synchronous interaction, users can engage in synchronous discussions that are added to the shared space as a single reply to the original post — replies are created using a chat tool.

The thread sub-object defines an additional operation for synchronous interactions: add a message to a previous message. When the user decides to start a new discussion, it issues a *post message*. This initial *post message* operation compresses all following *add message* operations issued in the synchronous discussion (by including the new messages). In this case, the other rules needed for log compression are very simple: two operations, *a* and *b*, commute if they neither modify the same message nor *b* posts a reply to the message posted by *a*, or vice-versa; no rule is need for converting operations as all add messages are compressed into the initial post message.

5 Related Work

Several systems have been designed or used to support the development of asynchronous groupware applications in large-scale distributed settings (e.g. Lotus Notes [18], Bayou [7], BSCW [2], Prospero [5], Sync [20], Groove [11]). Our basic system shares goals and approaches with some of these systems but it presents two distinctive characteristics. First, the object framework not only helps programmers in the creation of new applications but it also allows them to use different data-management strategies in different applications (while most of those systems only allow the customization of a single strategy). Second, unlike our system and BSCW, all other systems handle the reconciliation problem but do not address awareness support. From these systems, three can provide some integration between synchronous and asynchronous interactions.

In Prospero [5], it is possible to use the concept of streams (that log executed operations) to implement multi-synchronous applications (by varying the frequency of stream synchronization). This approach cannot support application that need to use different operations or different reconciliation strategies.

In Bayou, a replicated database system, the authors claim that it is “possible to support a fluid transition between synchronous and asynchronous mode of operation” [7] by connecting to the same server. However, without a notification mechanism that allows applications to easily update their interface and relying on a single replica, it is difficult to support synchronous interactions efficiently.

In Groove [11], some applications can be used in synchronous and asynchronous (off-line) modes. In Sketchpad, the same reconciliation strategy seems to be used (execute all updates by some coherent order, using a *last-writer wins* strategy). This may lead to undesired results in asynchronous interactions as the overwritten work may be large and important. In this case, it is not acceptable to arbitrarily discard (or overwrite) the contribution produced by some user, and the creation of multiple versions seems preferable [29,16].

Other groupware systems support multi-synchronous interactions. In [10], the authors define the notion of a room, where users can store objects persistently and run applications. Users work in synchronous mode if they are inside the room at the same time. Otherwise, they work asynchronously. In [13], the authors present a multi-synchronous hypertext authoring system. A tightly coupled synchronous session, with shared views, can be established to allow multiple users to modify the same node or link simultaneously. In [24], the authors describe a distance-learning environment that combines synchronous and asynchronous work. Data manipulated during synchronous sessions is obtained from the asynchronous repository, using a simple locking or check-in/check-out model.

Unlike DOORS, these systems lack support for asynchronous groupware in mobile computing environments, as they do not support disconnected operation (they all require access to a central server). Furthermore, either they do not support divergent streams of activity to occur during asynchronous edition or they use a single reconciliation solution (versioning). Our solution is more general, allowing to use the appropriate reconciliation solutions for each setting.

In [27], the authors propose a general notification system that supports multi-synchronous interactions by using different strategies to propagate updates. They also present a specific solution for text editors that implements an operational transformation (OT) algorithm that solves some technical problems for using OT in asynchronous settings. However, as discussed in Sect. 2.2, in asynchronous settings, OT may lead to unexpected results that do not satisfy any user – creating multiple version seems preferable. Our approach, allowing the use of a different reconciliation technique in each setting, can address this problem.

In [19], the authors present a brief overview of SAMS, an environment that supports multi-synchronous interactions using an OT algorithm extended with a constraint-based mechanism to guarantee semantic consistency. The proposed approach seems difficult to use and, as the previous one, it does not allow to use different operations or reconciliation techniques in each setting (as it is important for supporting some applications).

In [9], the authors present a system that supports both synchronous and asynchronous collaboration using a peer-to-peer architecture to replicate shared objects. In this system, replica consistency is achieved in both settings by executing all operations in the same order – an optimistic algorithm using roll back/roll forward is used. Again, this approach does not address the need of using different operations and different reconciliation strategies in each setting.

6 Final Remarks

In this paper, we have presented a model to integrate synchronous and asynchronous interactions in mobile computing environments. Our approach is built on top of the DOORS replicated object store, that supports asynchronous groupware relying on optimistic server replication and client caching.

To integrate synchronous sessions in the overall asynchronous activity we address the three main problems identified as important in the discussion of

Sect. 2. First, our system maintains multiple replicas of the data objects stored in the DOORS repository synchronized in realtime. To this end, we rely on a group communication infrastructure to propagate all operations to all replicas.

Second, our system addresses the problem of using different reconciliation and awareness strategies in different settings. To this end, the programmer may use an extension to the DOORS object framework that allows to use different reconciliation and awareness components in each setting.

Finally, it addresses the problem of using operations with different granularities for propagating updates in synchronous and asynchronous settings. To this end, it integrates a compression algorithm that converts a long sequence of *small* operations used in synchronous settings into a small sequence of *large* operations.

The combination of these mechanisms allows our system to provide support for multi-synchronous applications – the applications presented in Sect. 4 exemplify the use of the proposed approach. To our knowledge, our system is the only one to provide an integrated solution for all those problems in a replicated architecture that supports disconnected operation. More information about the DOORS system is available from <http://asc.di.fct.unl.pt/dagora/>. DOORS code is available on request.

References

1. JGroups. <http://www.jgroups.org>.
2. R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkell, J. Trevor, and G. Woetzel. Basic Support for Cooperative Work on the World Wide Web. *Int. Journal of Human Computer Studies*, 46(6):827–856, 1997.
3. P. Cederqvist, R. Pesch, et al. Version Management with CVS. <http://www.cvshome.org/docs/manual>.
4. P. Dourish. The parting of the ways: Divergence, data management and collaborative work. In *Proc. of the European Conf. on Computer-Supported Cooperative Work (ECSCW'95)*, 1995.
5. P. Dourish. Using metalevel techniques in a flexible toolkit for CSCW applications. *ACM Trans. on Computer-Human Interaction (TOCHI)*, 5(2):109–155, 1998.
6. P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proc. of the 1992 ACM Conf. on Computer-supported cooperative work*, 1992.
7. W. Edwards, E. Mynatt, K. Petersen, M. Spreitzer, D. Terry, and M. Theimer. Designing and implementing asynchronous collaborative applications with Bayou. In *Proc. of the ACM Symp. on User interface software and technology*, 1997.
8. C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In *Proc. of the 1989 ACM SIGMOD Int. Conf. on Management of data*, 1989.
9. W. Geyer, J. Vogel, L.-T. Cheng, and M. Muller. Supporting activity-centric collaboration through peer-to-peer shared objects. In *Proc. of the 2003 ACM Conf. on Supporting group work (GROUP '03)*, 2003.
10. S. Greenberg and M. Roseman. Using a room metaphor to ease transitions in groupware. Tech. Report 98/611/02, Dep. Comp. Science, Univ. of Calgary, 1998.
11. Groove. Groove Workspace v. 2.5. <http://www.groove.net>.
12. C. Gutwin and S. Greenberg. Effects of awareness support on groupware usability. In *Proc. of the Conf. on Human factors in computing systems*, 1998.

13. J. Haake and B. Wilson. Supporting collaborative writing of hyperdocuments in SEPIA. In *Proc. of the ACM Conf. on Computer-supported cooperative work*, 1992.
14. A. Imine, P. Molli, G. Oster, and M. Rusinowitch. Proving correctness of transformation functions in real-time groupware. In *Proc. of the 8th European Conf. on Computer-Supported Cooperative Work (ECSCW'03)*, Sept. 2003.
15. A. Karsenty and M. Beaudouin-Lafon. An algorithm for distributed groupware applications. In *Proc. of the 13th Int. Conf. on Dist. Computing Systems*, 1993.
16. R. H. Katz. Toward a unified framework for version modeling in engineering databases. *ACM Comput. Surv.*, 22(4):375–409, 1990.
17. M. Koch. Design issues and model for a distributed multi-user editor. *Computer Supported Cooperative Work*, 3(3-4):359–378, 1995.
18. Lotus. Ibm lotus notes. <http://www.lotus.com/notes>.
19. P. Molli, H. Skaf-Molli, G. Oster, and S. Jourdain. SAMS: Synchronous, Asynchronous, Multi-Synchronous Environments. In *Proc. of the 2002 ACM Conf. on Computer supported cooperative work in design*, 2002.
20. J. P. Munson and P. Dewan. Sync: A java framework for mobile collaborative applications. *IEEE Computer*, 30(6):59–66, June 1997.
21. N. Preguiça, J. L. Martins, H. Domingos, and S. Duarte. Data management support for asynchronous groupware. In *Proc. of the 2000 ACM Conf. on Computer supported cooperative work*, 2000.
22. N. Preguiça, J. L. Martins, H. Domingos, and S. Duarte. Integrating synchronous and asynchronous interactions in groupware applications. Technical Report TR-01-2005 DI-FCT, Univ. Nova de Lisboa, 2005.
23. N. Preguiça, M. Shapiro, and C. Matheson. Semantic-based reconciliation for collaboration in mobile environments. In *Proc. of the 11th Conf. on Cooperative Information Systems (CoopIS) - LNCS 2888*. Springer, 2003.
24. C. Qu and W. Nejdl. Constructing a web-based asynchronous and synchronous collaboration environment using webdav and lotus sametime. In *Proc. of the 29th ACM SIGUCCS Conf. on User services*, 2001.
25. M. Roseman and S. Greenberg. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Trans. on Computer-Human Interaction (TOCHI)*, 3(1):66–106, 1996.
26. C. Schuckmann, L. Kirchner, J. Schümmer, and J. M. Haake. Designing object-oriented synchronous groupware with coast. In *Proc. of the 1996 ACM Conference on Computer supported cooperative work*, 1996.
27. H. Shen and C. Sun. Flexible notification for collaborative systems. In *Proc. of the 2002 ACM Conf. on Computer supported cooperative work*, 2002.
28. H. S. Shim, R. W. Hall, A. Prakash, and F. Jahanian. Providing flexible services for managing shared state in collaborative systems. In *Proc. of the 5th European Conf. on Computer Supported Cooperative Work (ECSCW'97)*, 1997.
29. C. Sun and D. Chen. Consistency maintenance in real-time collaborative graphics editing systems. *ACM Trans. on Comp.-Human Interaction (TOCHI)*, 9(1), 2002.
30. C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Trans. on Comp.-Human Interaction (TOCHI)*, 5(1), 1998.
31. N. Vidot, M. Cart, J. Ferrié, and M. Suleiman. Copies convergence in a distributed real-time collaborative environment. In *Proc. of the 2000 ACM Conf. on Computer supported cooperative work*, 2000.
32. Y. Yang, C. Sun, Y. Zhang, and X. Jia. Real-time cooperative editing on the internet. *IEEE Internet Computing*, 4(3):18–25, 2000.

An Architectural Model for Component Groupware

Cléver R.G. de Farias^{1,2}, Carlos E. Gonçalves², Marta C. Rosatelli²,
Luís Ferreira Pires³, and Marten van Sinderen³

¹Departamento de Física e Matemática,
Faculdade de Filosofia Ciências e Letras de Ribeirão Preto (FFCLRP/USP),
Av. Bandeirantes, 3900, 14040-901 – Ribeirão Preto (SP), Brazil
farias@ffclrp.usp.br

²Programa de Mestrado em Informática, Universidade Católica de Santos,
Rua Dr. Carvalho de Mendonça, 144, 11070-906 – Santos (SP), Brazil
{cleverfarias, ceg-elus, rosatelli}@unisantos.edu.br

³Centre for Telematics and Information Technology, University of Twente,
P.O. Box 217, 7500 AE, Enschede, The Netherlands
{pires, sinderen}@cs.utwente.nl

Abstract. This paper proposes an architectural model to facilitate the design of component-based groupware systems. This architectural model has been defined based on (1) three pre-defined component types, (2) a refinement strategy that relies on these component types, (3) the identification of layers of collaboration concerns, and (4) rules for the coupling and distribution of the components that implement these concerns. Our architectural model is beneficial for controlling the complexity of the development process, since it gives concrete guidance on the concerns to be considered and decomposition disciplines to be applied in each development step. The paper illustrates the application of this architectural model with an example of an electronic voting system.

1 Introduction

The technological advances of the last decade have brought many changes into our society. Computers have become essential working and entertainment tools. Yet, most of the computer systems are targeted to single users, although most of our working tasks are likely to involve a group of people. Systems that provide support for groups of people engaged in a common task are called *groupware systems*.

The development of groupware systems poses many different challenges. Apart from the social aspects of groupware, developers are faced with problems typical of both distributed systems and cooperative work. Problems pertaining to distributed systems are, amongst others, the need for adequate levels of transparency, reliability, security, and heterogeneity support. Problems related to cooperative work are mainly the need for flexibility, integration, and tailorability in groupware systems [5].

The use of component-based technologies contributes to solve these problems [2, 8, 11, 19, 21, 22]. Component-based development aims at constructing software artefacts by assembling prefabricated, configurable and independently evolving building blocks called components. A component is a binary piece of software, self-contained, customisable and composable, with well-defined interfaces and dependencies.

Components are deployed on top of distributed platforms, contributing to solve many of the distribution-related problems of groupware systems. Components can also be configured, replaced and combined on-the-fly, which enhances the degree of flexibility, integration and tailorability provided by a system.

One of the biggest challenges in system development is the definition of the system architecture. The architecture of a (computing) system can be defined as the structure (or structures) of the system in terms of software components, the externally visible parts of those components and the relationships among them [3]. In this way, the architecture can be seen as the top-level decomposition of a system into major components, together with a characterisation of how these components interact [23].

A proper definition of the architecture of a system facilitates not only the system design as a whole but also the development and reuse of components for a family of similar systems, the so-called product line development. Thus, our work proposes an architectural model to help groupware developers tackling the design of component-based groupware systems. Our architectural model defines different types of components that serve as basis for system and component refinements. Our model also defines different types of collaboration concerns to help the identification of the different types of components and the assignment of functionality to components.

The remainder of this work is structured as follows: section 2 discusses the refinement strategy adopted in this work; section 3 identifies component types to be applied in the (component-based) development process of groupware systems; section 4 proposes a set of consecutive layers, one for each specific collaboration aspect of a cooperative work process; section 5 introduces the concept of collaboration coupling between these layers and discusses some related distribution aspects; section 6 illustrates the application of our architectural model with a case study related to an electronic voting system; finally, section 7 presents some conclusions.

2 Refinement Strategy

In the design of a groupware system, we use of the concept of functional entity as an abstraction for an entity in the real world (e.g., a system, a system user or a system component) capable of executing behavior. A functional entity executes behavior by itself or in cooperation with other functional entities, which form the environment of this entity.

2.1 Refinement Principle

There are two main approaches to tackle the refinement of a functional entity in general: (1) to refine the interactions between the functional entity and its environment without changing the granularity of the functional entity itself, i.e., without decomposing the functional entity into smaller parts, or (2) to decompose the functional entity into smaller parts and allocate the functional entity interactions to these parts without changing these interactions, except for the introduction of new (internal) interactions between the smaller parts. The first approach is called *interaction refinement*, while the second one is called *entity refinement* [16].

Fig. 1 illustrates the difference between the interaction refinement and entity refinement in the refinement of a system into system parts. Fig. 1 also shows that these approaches can be combined in successive refinement steps to produce some design, in which both the system is decomposed into smaller parts and the interactions are refined into more detailed interactions.

In the context of this work, we consider that a refinement process is carried out only according to the entity refinement approach. Consequently, we assume that the interactions between a functional entity and its environment are preserved as we refine the system into a set of interrelated components. Therefore, unless explicitly mentioned, we use the term *refinement* or *decomposition* to denote entity refinement.

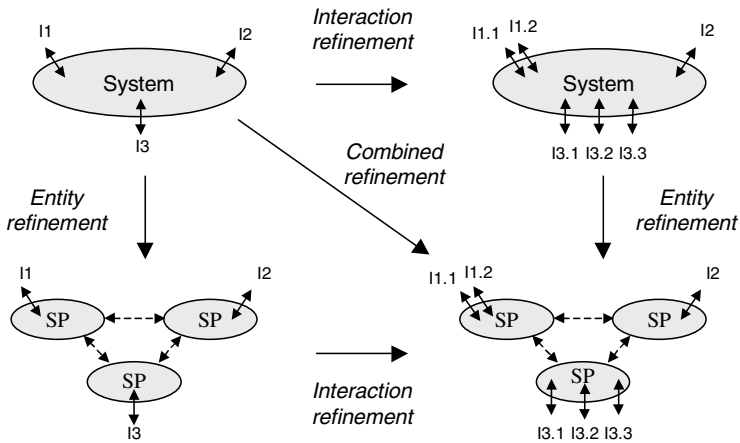


Fig. 1. Alternative refinement approaches

2.2 Component Decomposition

We can identify two slightly different approaches regarding the decomposition of a system into components: the *continuous recursion* approach [4] and the *discrete recursion* approach [7].

In the continuous recursion approach, the system is continuously refined into finer-grained components, until components of a desired granularity level or complexity are identified. Since in this case no specific component types are defined beforehand, this approach can only provide general guidelines for reducing the complexity of a component.

In the discrete recursion approach, the system is systematically refined into components of different types, which are pre-defined according to, for examples, different objectives or milestones identified throughout the design trajectory. A component type defines a number of characteristics common to a number of components. Different component types can be made to correspond to different component granularities, although this is not always necessarily the case. This decomposition approach is capable of providing both general and more specific refinement guidelines for each component type.

Fig. 2 illustrates the difference between the two approaches. Fig. 2a shows the continuous recursion refinement approach, in which all components defined in the consecutive decomposition steps are of the same type ('grey' components). Fig. 2b shows the discrete recursion refinement approach, in which two component types are defined beforehand (grey and white components). Furthermore, Fig. 2b shows a decomposition discipline in which only a single component type (either grey or white) is used at a certain level of granularity.

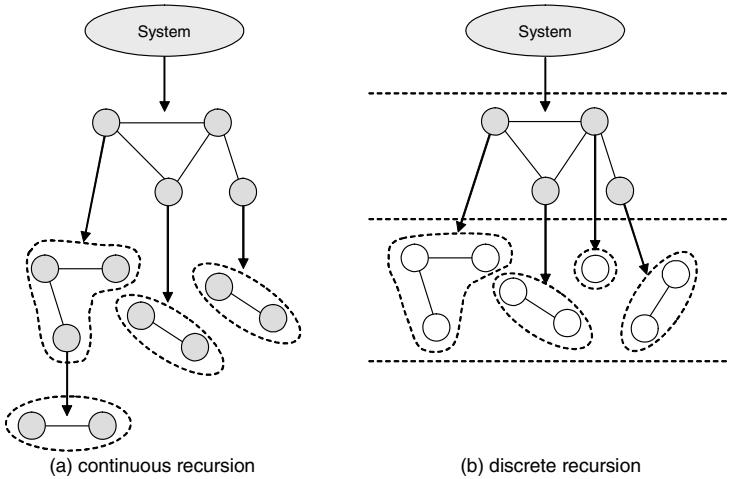


Fig. 2. Alternative component decomposition approaches

Fig. 2 also shows that these approaches are not fundamentally different (it is all about successive decomposition), and that discrete recursion could even be seen as a specialisation of continuous recursion.

3 Component Types

In this work we use the discrete recursion approach for component decomposition, because this approach allows us to tailor the component types according to different (sets of) concerns. Thus, we identify three different types of components inspired by [7]: *basic components*, *groupware components*, and *application components*.

A basic component is the most basic unit of design, implementation, and deployment. A basic component is not further refined into other components. Additionally, the behaviour of a basic component is carried out by binary code. Therefore, an instance of a basic component runs on a single machine, which is usually part of a distributed environment.

A groupware component consists of a set of basic components that cooperate in order to provide a mostly self-contained set of groupware functions. A groupware component embodies the behaviour corresponding to an independent collaborative concept or feature that can be reused to build larger groupware components and systems.

The self-containment of a groupware component does not imply that this component is isolated from other components. On the contrary, a groupware component should be composable, i.e., one should be able to compose different groupware components, and these components should be able to interact with each other. However, such a component should have minimal dependencies in order to maximize its reuse.

A groupware component also encapsulates distribution aspects. Since a groupware component consists internally of basic components, and basic components can be distributed individually across a network, the distribution aspects normally required by a groupware component are consequently addressed by the composition of (distributed) basic components. However, basic components can be used to address not only the physical distribution aspects, but also the distribution of concerns and responsibilities that form a groupware component.

An application component corresponds to a groupware application, i.e., an independent application that can be used separately or integrated into another groupware system. Any groupware system under development can be considered an example of an application component. However, groupware components can also be used as building blocks for larger application components. In most cases, an application component consists of a set of interrelated groupware components that cooperate in order to provide some application-level functionality.

In order to illustrate this component hierarchy, we take a videoconferencing system as an example. This system can be seen as a composition of individual applications, such as videoconferencing, chat, and shared whiteboard applications. A videoconferencing application can be decomposed into separate groupware components, which provide, e.g., audio support, video support, and attendance support. An audio support component can be decomposed into separate basic components to handle the connection establishment, coding, decoding, transmission, and so on.

Nevertheless, this classification scheme is flexible and subject to the designer's choice and interpretation. For example, in the videoconferencing system above, one could alternatively assign a separate groupware component to handle each of the videoconferencing, chat, and shared whiteboard concerns. In this alternative, the system is seen as a composition of individual groupware components, instead of a composition of application components.

4 Collaboration Concerns

In order to structure groupware systems we have identified layers of collaboration concerns that have to be handled by these systems. We have also identified rules for configuring these layers so that a meaningful groupware system can be obtained.

4.1 Collaboration Concern Layers

Suppose a particular groupware component manages the editing of a shared document. This component is responsible for maintaining the consistency of the document, allowing multiple users to change the document simultaneously. Initially, a user may choose to have a different view of the document. For example, a user may choose an "outline" view, as opposed to another user who uses a "normal" view at the same

time. The question is whether this particular choice of the first user should affect the way in which the second user views the document or the component should allow different users to have different views of the document simultaneously.

Consider that this particular component is also responsible for keeping the users informed about changes in the document. Another question is whether a user should be notified of every change in the document or any action of another user, or the component should only notify the user when a change affects the part of the document this specific user is currently working on.

These are typical issues that have to be dealt with by a groupware component. In order to provide flexibility to deal with these and other issues, we identify a number of so-called *collaboration concern layers*, on which different aspects of the functionality of a groupware component can be positioned.

We have identified four separate layers: *interface*, *user*, *collaboration*, and *resource*. Each layer uses the functionality provided by the layer below in order to provide some functionality that is used by the layer above. Fig. 3 depicts the collaboration concern layers identified in this work and their relationships.

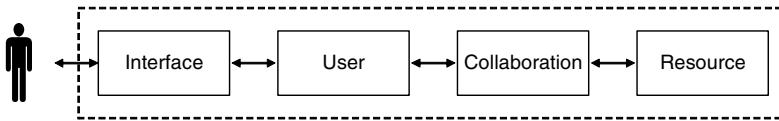


Fig. 3. Collaboration concern layers

The interface layer is concerned with providing a suitable interface between a human user and the groupware component. Considering a drawing component of a shared whiteboard application, the interface layer should enable a user to create new drawings and change or delete existing drawings by means of graphical commands. The interface layer should also enable the user to visualize the drawing itself through the interpretation of drawing commands. Therefore, the interface layer handles all the direct communication with the users via a graphical user interface.

The user layer is concerned with the local support for the activities performed by a single user. The user layer addresses the local issues with respect to each individual user that do not affect the collaboration as a whole. For example, suppose that our drawing component enables a user to make changes to a local copy of a shared drawing, without changing the shared drawing immediately. This allows the user to incorporate changes to the shared drawing only when this user is absolutely satisfied with these changes. Therefore, the user layer maintains a user's perception of the collaboration. The user layer also supports the interface layer, by relating it with the collaboration layer.

The collaboration layer is concerned with collaboration issues of multiple users. The logic involved in different collaboration aspects, such as communication, coordination and cooperation functionalities [6, 9], are mainly tackled at this layer. Considering our drawing component, the collaboration layer should be able to handle the drawing contributions of multiple users, relating them as necessary. Therefore, this layer is responsible for the implementation of the core aspects of the collaboration and for relating the user layer to the resource layer.

The resource layer is concerned with the access to shared collaboration information (resources), which could be, for example, kept persistently in a database. In our drawing component, the resource layer should be able to store the drawing and drawing commands, saving and loading them as necessary. The resource layer is only accessible through the collaboration layer.

4.2 Implementation of Concern Layers

Each collaboration concern layer can be implemented by one or more basic components. An interface component implements the interface collaboration layer. Similarly, user, collaboration, and resource components implement the user, the collaboration, and the resource collaboration layers, respectively. Nevertheless, it is not uncommon to someone implement more than one layer using a single component, in case the functionality provided by these layers is simple enough to be implemented by a single component.

We distinguish between three different types of functionality interfaces that a component can support based on the purpose and visibility (scope) of the interface: *graphical user interfaces*, *internal interfaces* and *external interfaces*.

A graphical user interface (GUI) supports the interactions between a human user and an interface component. An internal interface supports the interactions between the components of a single groupware component. Such an interface has internal visibility with respect to a groupware component, i.e., a groupware component cannot interact with another groupware component using internal interfaces. An external interface supports the interactions between groupware components. Such an interface has external visibility with respect to groupware components.

Interface components and resource components usually do not have external interfaces. Therefore, interactions between groupware components are normally only achieved via the user and collaboration components.

A groupware component does not need to have all four layers. For example, it is only meaningful to have a resource layer if some shared information has to be stored persistently. Similarly, an interface layer is only meaningful if the component interacts with a human user.

Nevertheless, if a groupware component has more than one layer, these layers should be strictly hierarchically related. For example, the interface layer should not access the collaboration layer or the resource layer directly, nor should the user layer access the resource layer directly. Thus, a single groupware component should consist of at least one and up to four ordered collaboration concern layers.

Fig. 4 illustrates some examples of groupware components that conform to the rules given above. Each layer is represented by a corresponding basic component. A groupware component is represented by a dashed rectangle, while a basic component is represented as a solid rectangle. An interface is represented by a T-bar attached to a component, while an operation invocation is represented by an arrow leading to an interface. An external interface is represented by a solid T-bar that crosses the boundary of the groupware component, while an internal interface is represented by a dashed T-bar inside the groupware component. Graphical interfaces are not explicitly represented.

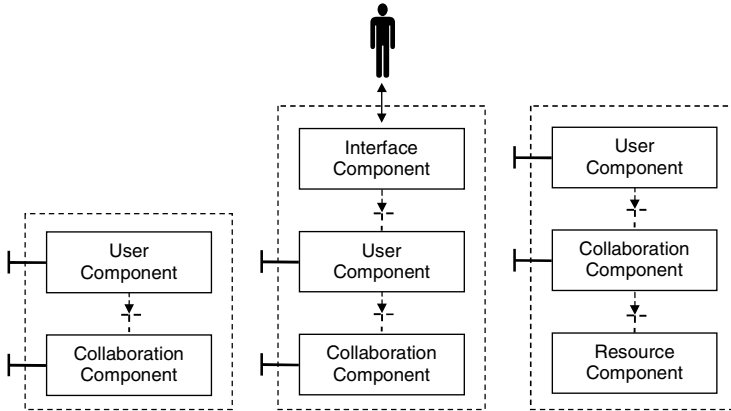


Fig. 4. Valid layering distributions

4.3 Related Work

Our collaboration concern layers have been identified based on a number of developments, such as Patterson’s state levels [15], Herzum and Sims’ distribution tiers [7] and Wilson’s architectural layers [24]. These developments address similar issues although with different terminology.

Patterson [15] identifies four state levels in which a synchronous groupware application can be structured, namely display, view, model, and file. The display state contains the information that drives the user display. The view state contains the information that relates the user display to the underlying information in the application, which is the model state. The file state consists of a persistent representation of the application information. Based on Patterson’s state levels, Ter Hofte proposes a four-level collaborative architecture known as the zipper architecture [22].

Herzum and Sims [7] propose a distribution architecture for business components that consists of four tiers, namely user, workspace, enterprise, and resource. These tiers are roughly equivalent to our collaboration concern layers. However, this architecture emphasizes distribution aspects, instead of the collaboration aspects that we emphasize in our work.

Table 1. Collaboration concern layers and related approaches

Collaboration Concern Layers	Patterson’s State Levels	Herzum and Sims’ Distribution Tiers	Wilson’s Architectural Layers
Interface	Display	User	View
User	View	Workspace	Application-Model
Collaboration	Model	Enterprise	Domain
Resource	File	Resource	Persistence

Wilson [24] proposes the development of a distributed application according to four architectural layers, namely view, application-model, domain and persistence. The view layer deals with user interface issues, the application-model layer deals with application-specific logic, the domain layer deals with domain-specific logic, and the persistence layer deals with the storage of information in a persistent format.

Table 1 shows the correspondence between our collaboration concern layers, Patterson's state levels, Herzum and Sims distribution tiers, and Wilson's architectural layers.

5 Collaboration Coupling

The coupling of collaboration concerns and their logical or physical distribution are two important aspects to be considered in the design of groupware systems.

5.1 Coupling Levels

So far we have discussed the collaboration concern layers of a groupware component in the context of a single user. However, the whole purpose of groupware systems is to support interactions involving multiple users.

In the context of a groupware system, and particularly in the context of a groupware component, two or more users may or may not share the same perception of the ongoing collaboration supported by the system. For example, in the case of the groupware component that manages the editing of a shared document, if one user chooses an outline view of the document instead of a normal view, this decision could possibly affect another user's view of the document. In case all the component users share the same perception, the outline view would replace the normal view for all users. Otherwise, the other users do not share the same perception of the collaboration, and only that particular user would have an outline view of the document.

We can apply the same reasoning to each collaboration concern layer of a groupware component. As a consequence, collaboration concern layers can be coupled or uncoupled. *Coupling* was introduced as a general mechanism for uniting the interaction contributions of different users, such that users might share the same view or state of a collaboration [22].

A collaboration concern layer of a groupware component is coupled if all users have the same perception of the information present in the layer and how this information changes. Therefore, a collaboration concern layer across multiple users can be coupled or uncoupled. An important property of coupling is downwards transitivity, which means that if a layer is coupled, the layers below, from the interface layer down to the resource layer, must be coupled as well in order to ensure consistency.

Four levels of coupling can be established based on the collaboration layers defined in this work: *interface*, *user*, *collaboration*, and *resource* coupling.

The interface coupling level represents the tightest coupling level. All the component users have the same perception of the collaboration, starting at the user interface layer. This level corresponds to the collaboration style known as What You See Is What I See (WYSIWIS) [20].

The user coupling level offers more freedom (independence of use) to the component user than the interface coupling level. All the component users have the same perception of the collaboration starting at the user layer, i.e., the information at the user layer is shared by all users, but their interface layers are kept separate.

The collaboration coupling level goes a step further and offers more freedom than the user coupling level. All the component users have the same perception of the collaboration starting at the collaboration layer, i.e., the information at the collaboration layer is shared by all users, but their interface and user layers are kept separate.

The resource coupling level offers the loosest coupling level. All component users have the same perception of the collaboration only at the resource layer, i.e., the information at the resource layer is shared by all users, but their interface, user and collaboration layers are kept separate.

Fig. 5 depicts the collaboration coupling levels defined in this work for two users. A large rectangle labelled with the layer initial indicates that the layer it represents is coupled, while a small rectangle also labelled with the layer initial indicates that the layer it represents is uncoupled.

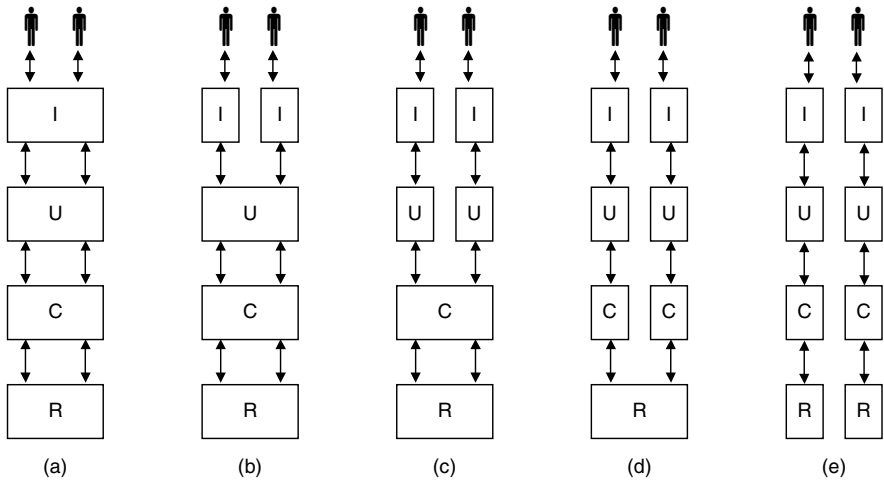


Fig. 5. Collaboration coupling levels

Fig. 5a to Fig. 5d show the interface, user, collaboration and resource coupling levels, respectively. Fig. 5e depicts the absence of coupling at all levels, i.e., in this figure the two instances of the groupware component operate independently from each other (offline collaboration).

In a truly flexible groupware system, the users of this system should be able to choose between the different levels of coupling, changing it at run-time based on the characteristics and requirements of the task at hand. They should also be able to choose a temporary absence of coupling, i.e., the users may decide to work independently for a while,

resuming their coupling status sometime later. In this case, additional mechanisms to guarantee the consistency of the collaboration afterwards have to be implemented.

5.2 Distribution Issues

There are basically two ways to achieve collaboration coupling in a given layer: using a centralised architecture or using a replicated architecture with synchronization mechanisms.

In a centralized architecture, a single copy of the collaboration state, i.e., all the information contained in a layer, is maintained and shared by the users of the layer. Concurrency control mechanisms should be used to avoid inconsistencies if needed.

In a replicated architecture, multiple copies of the collaboration state are maintained, one for each user of the layer. In this case, synchronization mechanisms (protocols) are used to maintain the consistency across the replicated copies of the collaboration.

The collaboration state of an uncoupled layer is non-centralised by definition, i.e., each user maintain its own copy of the collaboration state. However, the collaboration state of a given coupled layer can be either centralised or replicated. Thus, different architectures for each coupling level can be applied in a single groupware system.

The resource layer can only be coupled otherwise we are actually talking about distinct instances of a collaboration in place of a single one (see example in Fig. 5e). This implies that the resource coupling level can only be centralised or replicated. Each layer above can be coupled or uncoupled; in case the layer is coupled, the coupling level can be again centralised or replicated. A fully centralised architecture has only centralised layers, a fully replicated architecture has only replicated layers, and a hybrid architecture combines centralised and replicated layers. However, once a coupled layer is implemented using a centralised architecture, all layers below should also be implemented using a centralised architecture. This is because it makes little sense to centralise some information in order to assure consistency, and then to replicate some other information upon which the first one depends on.

Although possible in principle, it is very unlikely that interface coupling is achieved using a centralized architecture because of the complexity involved and response time requirements. Interface coupling is usually implemented based on shared window systems (see [1, 10, 20]).

Fig. 6 illustrates three possible combinations of centralised and replicated architectures to achieve coupling at the collaboration layer for two users. In Fig. 6 a rectangle represents an instance of a basic component, which is labelled with the initial of the layer it implements. A large rectangle indicates a component in a centralized architecture, while two small rectangles connected by a synchronization bar (a double-edged horizontal arrow) indicate a component in a replicated architecture. Small rectangles without a synchronization bar indicates that the layer they represent are uncoupled. Fig. 6a depicts a fully centralised architecture (both the resource and the collaboration layers are centralised), Fig. 6b depicts a hybrid architecture (the resource layer is centralised while the coordination layer is replicated), and Fig. 6c depicts a fully replicated architecture (both the resource and the collaboration layers are replicated).

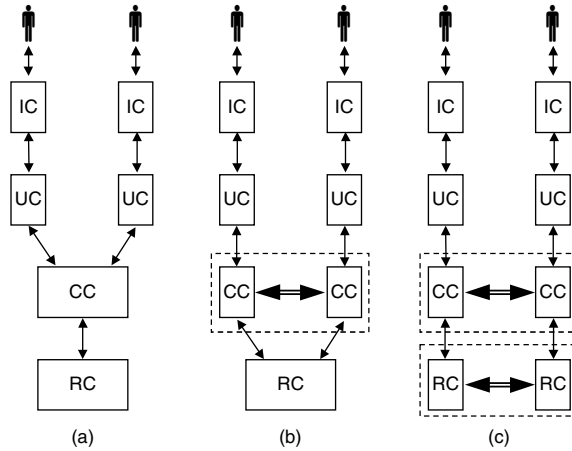


Fig. 6. Centralised, hybrid, and replicated architectures

The choice between a centralised architecture and a replicated architecture is mainly related to implementation issues [22]. A centralised architecture is indicated whenever a given coupling level either requires some specialised or excessive processing power that prevents replication or changes from a coupled state to an uncoupled state at this level are unlikely. A replicated architecture is indicated whenever changes from a coupled state to an uncoupled state are desired, thus improving the flexibility of the component.

A discussion on the benefits and drawbacks of centralised versus replicated architectures, as well as the mechanisms used to achieve consistency in both architectures, falls outside the scope of this work. For detailed discussions on these issues we refer to [1, 10, 14, 17, 18, 22].

6 Design of an Electronic Voting System

In order to exemplify our architectural model we have applied it in the development an Electronic Voting System (EVS).

The EVS basically enables its users to create polls and/or vote on them. To use the EVS, any user is required to register first. Registered users can then log in and out of the system and update their personal profile. Any registered user can open a poll, defining its subject, voting options and eligible participants. Once a poll is registered, it will be available for voting to all selected participants. A registered user can visualize a list of all the polls available in the system and cast a single vote to any open poll in which she participates. Additionally, the results of a closed poll should be available for consultation by all users.

The design of the EVS system was carried out in a number of design steps according to the guidelines provided in [5]. In this work, we used UML 2.0 [12, 13] as our modelling language. Fig. 7 depicts the high level architectural model of the EVS using UML component diagram.

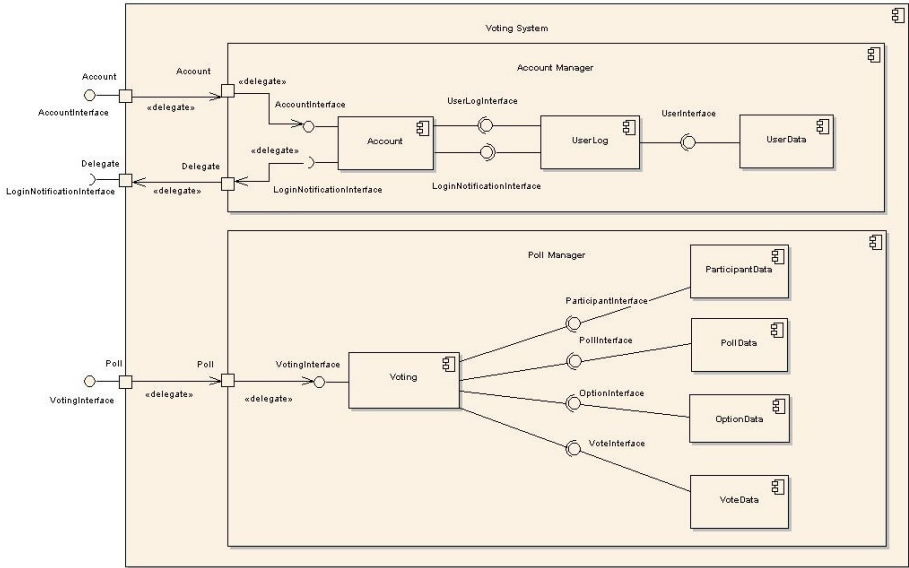


Fig. 7. EVS high level architecture

In the first step the EVS service was specified. At this point the EVS was seen as an application component, called Voting System. In the second step, this component was decomposed into two groupware components:

- Account Manager, which is responsible for the registration of users, their logging on and off the system, and provision of user awareness;
- Poll Manager, which is responsible for the control over the creation of polls, as well as the cast of votes;

In the third step, each groupware component was then refined into a number of simple components. In this case study we did not consider the interface layer as part of the groupware component, but considered it as part of the client application.

The Account Manager component was refined into the basic components Account, UserLog and UserData, which implement the user, collaboration and resource layers respectively. The identified components are coupled at the collaboration coupling level, using a fully centralised architecture for the coupled layers. Since the user layer is uncoupled, a separate instance of the component Account should be created to support each separate user of this component.

The Poll Manager component was refined into the basic components Voting, ParticipantData, PollData, OptionData and VoteData. The component Voting implements the user and collaboration layers, while the remaining components implement the resource layer. The identified components are coupled at the resource coupling level, using a fully centralised architecture for this layer. Since the user/collaboration layer is uncoupled, a separate instance of the component Voting should be created to support each separate user of this component.

The EVS components were implemented as Enterprise Java Beans (EJB) components. As an implementation infrastructure, we used the JBOSS 3.2 application server, the Hypersonic database, and JAAS for authentication. The client was developed using SWING and a communication library of the JBOSS server.

The components Account and Voting were implemented as two stateless session beans, while the component UserLog was implemented as a statefull session bean. The remaining components were implemented entity beans, using bean managed Persistence. Notification across components was implemented using the Java Message Service (JMS).

Fig. 8 depicts some screenshots of the voting system user interface. Fig. 8a shows the system main interface. This interface shows the list of current logged users. Fig. 8b shows the poll registration interface. Fig. 8c shows the poll voting interface. Fig. 8d shows the poll results interface.

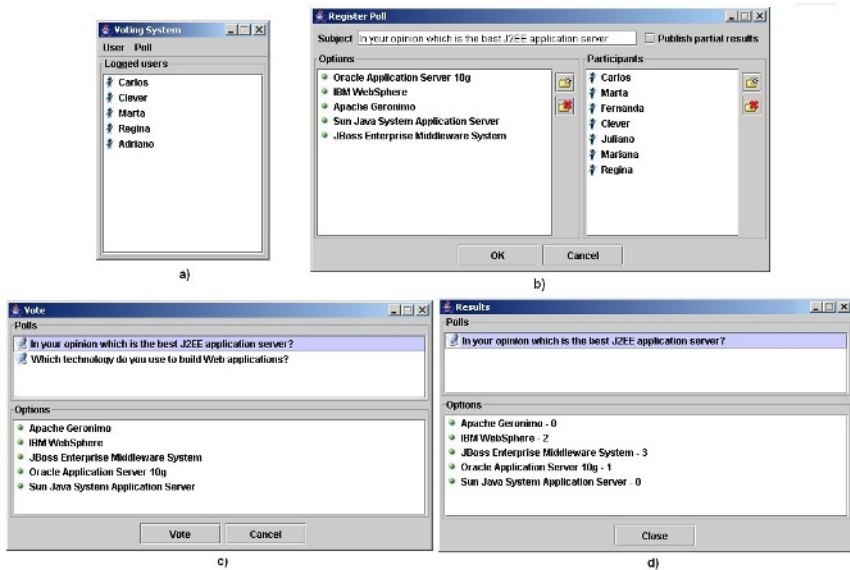


Fig. 8. Voting system user interface

7 Conclusion

This paper proposes an architectural model for the development of component-based groupware systems. According to this model, a groupware system is recursively decomposed into a number of interrelated components, such that the service provided by the groupware system is provided by the composed individual services of these components.

The decomposition process is carried out according to a discrete recursion approach based on pre-defined component types. We believe that the use of specific component types facilitates the decomposition process if compared to a decomposition approach

that does not make such distinction. Additionally, the use of pre-defined component types facilitates the identification and reuse of existing components.

We have discussed the use of collaboration concern layers and collaboration coupling to structure the different collaboration aspects within the scope of a groupware component. We believe that the use of these layers and guidelines facilitates the logical and physical distribution of these aspects and the assignment of functionality to components thereafter thus improving reuse and speeding up the development process.

We have illustrated the application of the architectural model proposed in this work by means of a simple case study describing the development of an electronic voting system, which is being developed in the scope of the project TIDIA-Ae¹.

Acknowledgements

This work has been partially supported by FAPESP under project number 2003/08279-2.

References

1. Ahuja, S.R., Ensor, J.R. and Lucco, S.E.: A comparison of application sharing mechanisms in real-time desktop conferencing systems. In *Proceedings of the 1990 ACM Conference on Office Information Systems (COIS'90)*, pp. 238-248, 1990.
2. Banavar, G., Doddapaneti, S., Miller, K. and Mukherjee, B.: Rapidly Building Synchronous Collaborative Applications by Direct Manipulation. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW'98)*, pp. 139-148, 1998.
3. Bass, L., Clements, P. and Kazman, R.: *Software Architecture in Practice*. Addison-Wesley, 1997.
4. D'Souza, D. F. and Wills, A. C.: *Objects, Components and Frameworks with UML: the Catalysis Approach*. Addison Wesley, 1999.
5. de Farias, C. R. G.: *Architectural Design of Groupware Systems: a Component-Based Approach*. PhD Thesis, University of Twente, the Netherlands, 2002.
6. Fuks, H., Raposo, A. B., Gerosa, M. A. and Lucena, C. J. P.: Applying the 3C Model to Groupware Development. In *International Journal of Cooperative Information Systems (IJCIS)*, 14(2-3), pp. 299-328, 2005.
7. Herzum, P. and Sims, O.: *Business component factory: a comprehensive overview of component-based development for the enterprise*. John Wiley & Sons, 2000.
8. Hummes, J. and Merialdo, B.: Design of Extensible Component-Based Groupware. In *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, 9 (1), pp. 53-74, 2000.
9. Laurillau, Y. and Nigay, L.: Clover Architecture for Groupware. In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW'02)*, pp. 236-246, 2002.
10. Lauwers, J.C., Joseph, T.A., Lantz, K.A. and Romanow, A.L.: Replicated architectures for shared window systems: a critique. In *Proceedings of the 1990 ACM Conference on Office Information Systems (COIS'90)*, pp. 249-260, 1990.

¹ <http://tidia-ae.incubadora.fapesp.br/novo/>

11. Litiu, R. and Prakash, A.: Developing adaptive groupware applications using a mobile component framework. In *Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work (CSCW'00)*, pp. 107-116, 2000.
12. OMG: *UML 2.0 Infrastructure Specification*. Adopted Specification, Object Management Group, 2003.
13. OMG: *UML 2.0 Superstructure Specification*. Revised Final Adopted Specification, Object Management Group, 2004.
14. Patterson, J.F., Day, M. and Kucan, J.: Notification servers for synchronous groupware. In *Proceedings of ACM 1996 Conference on Computer Supported Cooperative Work (CSCW'96)*, pp. 122-129, 1996.
15. Patterson, J.F.: A taxonomy of architectures for synchronous groupware applications. *SIGOIS Bulletin*, 15 (3), pp. 27-29, 1995.
16. Quartel, D., Ferreira Pires, L. and Sinderen, M.: On Architectural Support for Behaviour Refinement in Distributed Systems Design. In *Journal of Integrated Design and Process Science*, 6 (1), pp. 1-30, 2002.
17. Roth, J. and Unger, C.: An extensible classification model for distribution architectures of synchronous groupware. In *Designing Cooperative Systems: the Use of Theories and Models, Proceedings of the 5th International Conference on the Design of Cooperative Systems (COOP'00)*, pp. 113-127, 2000.
18. Schuckmann, C., Kirchner, L., Schümmer, J. and Haake, J.M.: Designing object-oriented synchronous groupware with COAST, In *Proceedings of ACM 1996 Conference on Computer Supported Cooperative Work (CSCW'96)*, pp. 30-38, 1996.
19. Slagter, R. J.: *Dynamic Groupware Services: Modular design of tailorable groupware*. PhD Thesis, University of Twente, the Netherlands, 2004.
20. Stefik, M., Bobrow, D.G., Foster, G., Lanning, S. and Tatar, D.: WYSIWIS revised: early experiences with multiuser interfaces. *ACM Transactions on Office Information Systems*, 5(2), pp. 147-167, 1987.
21. Teege, G.: Users as Composers: Parts and Features as a Basis for Tailorability in CSCW Systems. In *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, 9 (1), pp. 101-122, 2000.
22. ter Hofte, G. H.: *Working Apart Together: Foundations for component groupware*. PhD Thesis, Telematics Institute, the Netherlands, 1998.
23. van Vliet, H.: *Software Engineering: Principles and Practice*. John Wiley & Sons, USA, 2000.
24. Wilson, C.: Application Architectures with Enterprise JavaBeans. *Component Strategies*, 2(2), pp. 25-34, 1999.

An Architecture for Collaborative Geomodeling

Luciano P. Reis^{1,2}, Alberto B. Raposo³, Jean-Claude Paul⁴,
and Fabien Bosquet⁵

¹ Petrobras Research Center, Brazil

² Institut National Polytechnique de Lorraine, France
luciano.reis@petrobras.com.br

³ Tecgraf, Computer Science Dept., PUC-Rio, Brazil
abraposo@tecgraf.puc-rio.br

⁴ ISA / LORIA, France
Jean-Claude.Paul@loria.fr

⁵ Earth Decisions Sciences, France
bosquet@earthdecision.com

Abstract. This paper presents an architecture for distributed synchronous collaborative visualization and modeling applied to the geosciences. Our goal is to facilitate the creation of heterogeneous collaboration sessions, in which participants may use different versions of a core CAD application, configured with specific functionalities and multimedia user interfaces, through the composition of run-time plugins. We describe the domain requirements, the architectural concepts that facilitate the integration of our collaboration plugins with the core application, and the management of communication channels to allow the definition of role-based control policies adapted to specific types of sessions.

1 Introduction

Geomodeling, the computer-aided design of geological objects and their properties [17], involves a large spectrum of skills spread over different domains: geophysics, geology and reservoir engineering. A numerical earth model is shared by people with different types of specializations and evolves continuously through a team effort. In the oil and gas industry, during the exploration and production of a reservoir, new data is constantly acquired, and the model needs to be frequently updated as new decisions have to be taken based on the most up-to-date information.

Effective geomodeling is strongly graphics-based. High-performance graphics make it possible for the professionals involved in the process to interactively visualize and edit the integrated three-dimensional models. Visualization is used as a powerful tool for data understanding and insight, as a support for interactive modeling, and as a common language for communication and collaboration within multi-disciplinary teams. Currently, virtual-reality applications are starting to be used to enhance comprehension and to improve precision in some modeling tasks [11, 14].

Given the geographical dispersion of operations and professionals in the industry, and the increasing availability of computing, graphics and networking resources, remote collaboration offers great potential to improve distributed cooperation and

decision-making. The scarcity of geomodeling experts also makes this technology very important for consulting and training. However, currently only a few commercial modeling applications are starting to offer some collaboration functionalities, mainly the synchronization of points-of-view among remote users .

A key problem faced by the companies is how to integrate different types of tools, legacy and new, to compose a comprehensive software solution that provides a coherent and efficient environment for remote collaboration.

Therefore, the objective of this work is the development of an architecture for remote synchronous collaborative modeling and visualization applied to oil and gas exploration and production. This application domain is closely related to other areas, like collaborative engineering and collaborative scientific visualization. However, some specific requirements influence the design of the proposed architecture. In particular we are interested in facilitating the coordinated use of heterogeneous user interfaces and interaction paradigms by participants in multi-disciplinary collaborative sessions with the support of multimedia communication, taking into account the different types of collaborative activities to be performed, the roles of the participants and the communication and cooperation channels involved.

Unlike related collaboration solutions [1, 19, 25], we do not expect the development of tools compliant to a predefined architecture. Instead, our purpose is to facilitate the transformation of a well-designed operational application, which follows established design principles [9] and is extensible by run-time plugins, into a collaborative system through the introduction of cooperation and communication mechanisms. Our implementation is based on Gocad [10], a pioneering geological modeling application.

Along with the core application and its functional plugins, the proposed collaboration solution involves the development and integration of:

- A collaboration plugin responsible for providing session management services and for supporting synchronous collaboration through the creation of “communication channels” (commands, camera, telepointers, 3D annotations, avatars, model distribution, audio and video) among distributed instances of the application, with broadcasting and floor control mechanisms;
- A custom tool for multimedia communication, also integrated as a plugin, providing audio and video channels subject to the defined control policies;
- A virtual-reality plugin compatible with the collaboration functionality provided;
- A real-time data-acquisition plugin, used for automatic integration into the model of data arriving from remote well-drilling sites [2].

In this article, after an overview of some application scenarios, we emphasize the description of the coordination mechanisms proposed for dealing with the different channels used for cooperation and communication.

2 Collaborative Geomodeling

Before we proceed to the description of the proposed architecture, let us discuss some typical application scenarios and related requirements.

2.1 Collaboration Scenarios

Consulting

User *A* is building a model, and faces a problem on how to perform a certain task. *A* then creates a collaboration session (a conference), invites user *B* to join from a remote site, transfers the model and explains the problem, turning the model around, zooming in the area where the problem occurs, pointing to things and making annotations that *B* can see in a synchronized way, as if both were looking at the same display. *B* may also manipulate the camera and make annotations directly on the three-dimensional model, which are seen by *A* (Figure 1). For the two participants only, conversation could take place over the phone. However, integrated videoconferencing, automatically established among the participants of a conference, simplifies and improves the communication.

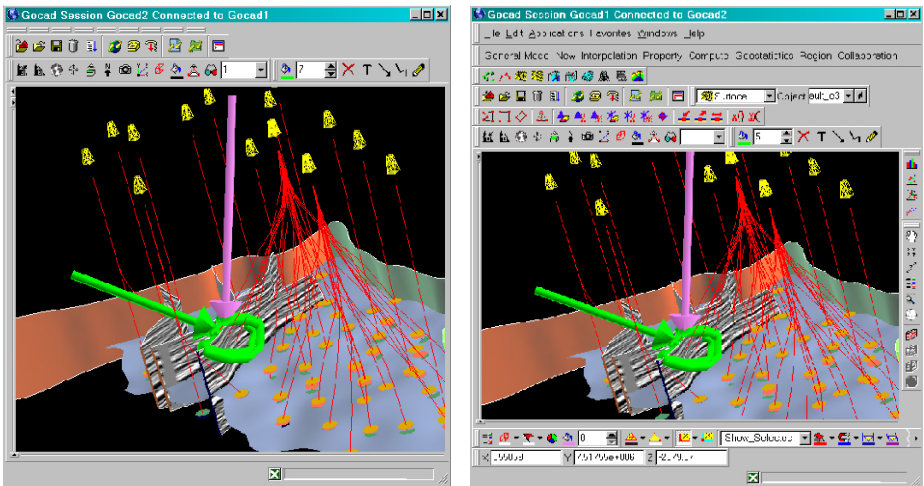


Fig. 1. Screenshots of two users discussing a model, using 3D cursors and annotations (differentiated by colors)

Conflicting camera movements are detected immediately, and a direct negotiation for the control can be done during the discussion over the audio channel, without the need for explicit passing of the camera control. However, if more than two participants are involved, negotiation of control through a verbal protocol becomes awkward, interfering with the main discussion, and a turn-taking mechanism becomes necessary. If both participants need to edit the model, floor passing is also used to regulate the transfer of the control of modeling commands, avoiding possible inconsistencies due to conflicting operations.

Well Planning

An important collaboration scenario is the design and real-time steering of oil wells. While a directional (non-vertical) well is drilled, data acquired at the well bit in the subsurface needs to be received at the office from the remote drilling location and

incorporated into the model, shared by local and remote geoscientists, in nearly real time to support collaborative decisions about the steering of the well trajectory.

This activity involves at least two sites: one or more modeling experts working together at the office, using a desktop version of the application loaded with a well-planning plugin with specific functionalities, and an operator at the drilling site, typically using a less powerful computer and a lightweight version of the application to visualize the shared model, with restricted editing rights. All participants use the plugin for real-time data acquisition. Usually very limited network bandwidth is available at the drilling site, requiring strict control of the use of audio and video. Our system has been frequently applied to this scenario for the survey of actual operations.

Virtual-reality (VR) applications are now starting to be used for the design and steering of complex horizontal wells [14, 13]. However, while some modeling tasks like the design of the well path or a local modification of a surface may be facilitated by the immersive interface, other global modeling tasks become more difficult with an “inside the model” point of view. Also, the immersive interface usually needs to provide restricted functionalities, for ergonomic reasons. We propose that the use of VR as part of heterogeneous collaboration sessions can bring in the benefits but avoid the shortcomings. In this case, a participant in the session uses the VR plugin in an immersive virtual environment, collaborating with other desktop participants. Among other usability issues, camera position events cannot be exchanged between the desktop and the VR users, since their navigation metaphors are totally different.

2.2 Requirements and Consequences

We have observed and taken part in operational collaboration sessions in the scenarios described above. These ethnographic observations have suggested some specific requirements for the proposed architecture, both from the users’ and from the system developers’ points of view, which guided our design choices described below.

Small groups: the activities considered involve a small number of participants. Therefore, scalability is not an issue in this domain. This allows the adoption of a centralized coordination scheme, simplifying the solution.

Graphic performance: visualization and modeling require very responsive interaction (maintenance of high frame rates), demanding an appropriate treatment of the graphics-related communication channels. This implies that the implementation of coordination policies cannot degrade performance. For this reason, for graphic interaction events, we propose a floor control mechanism based on the setting of input and output switches for each channel in the host application, thus avoiding the processing of the real-time events by the external coordination logic.

Integrated multimedia communication: audio and video are fundamental components of a synchronous collaborative modeling system. In some systems this is provided through external tools, requiring separate session management and controls. We consider that audio and video need to be provided as integrated communication channels. As the use of multiple audio and video streams poses strong requirements over bandwidth and processing consumption, conference participants need to be able to control individual connections, subject to the role-based policies defined for the conference.

Reuse of single-user functionality: application deployment is one of the most challenging aspects of groupware development [7]. Our target applications require the integration of a very large set of tools and multidisciplinary modeling functionalities, typically developed over many years. Therefore, the objective of the architecture proposed is to allow the flexible extension of an operational single-user application into a collaborative system through the incorporation of plugins, not requiring recompilation and redistribution.

Concurrency control: geological modeling involves the construction of diverse types of mathematical objects (surfaces, meshes, solid models), with complex geometry and topology, interdependency relationships, and a very large number of elements (a single version of some objects may easily approach the available memory size). Currently it is not possible to assume that modeling operations may be always rolled back efficiently. In this context, effective consistency maintenance guaranteeing high responsiveness and concurrency is still not feasible [23].

Therefore, the approach presently adopted for concurrency management in modeling is to actually avoid conflict through the use of mutual exclusion by the floor-control setting for the modeling commands (or, optionally, to rely on a social protocol). As the mechanism proposed is extensible, more sophisticated floor strategies [5, 6] and concurrency control [23, 8, 21] can be used in the future.

Awareness: for effective collaboration, group awareness needs to be provided by different means [21]. The list of participants in a session, their roles and control rights over the different communication channels need to be easily accessible. Annotations, telepointers and avatars need to be associated to the participants with visual cues, like labels and colors.

Heterogeneous operation conditions: networking resources for distributed participants are typically very heterogeneous. To guarantee the required responsiveness we have opted for a replicated architecture [4], in which the basic cooperation mechanism is the broadcasting of commands and interaction events over the communication channels, requiring low network bandwidth.

3 Collaboration Architecture

The proposed architecture was designed in the context of the scenarios and requirements described above. A collaboration plugin (NetGocad), loaded at runtime, allows the transformation of the single-user application (Gocad) into a distributed collaborative system through its extension with broadcasting and control capabilities, by means of the incorporation of CORBA objects and of a configurable coordination component.

The separation of coordination policies from computation, as proposed elsewhere [15, 3], and the definition of collaboration types, participant roles and floor control over the channels through an object-oriented scheme, implemented with a simple and powerful interpreted extension language (Lua), greatly simplifies the evolution of the system.

3.1 Gocad

Gocad (Geological Objects Computer Aided Design) is a CAD software, originally developed by an international research consortium [10], that allows the construction of earth models for geophysics, geology and reservoir engineering applications. Its architecture is based on the systematic use of Design Patterns [9] such as: Abstract Factory, Builder, Chain of Responsibility, Command, Composite, Factory Method, Interpreter, Iterator, Observer, Proxy, and Singleton. It also provides a flexible development framework to allow the creation of plugins, which can be dynamically loaded at runtime inside the application shell.

The implementation of NetGocad, described in the next section, is therefore facilitated by this framework, in particular by the use of the Command, Abstract Factory and Observer Patterns.

Gocad uses the Command Pattern to isolate the processing of operations from their invocation at the user interface. All menu operations generate string commands that are then executed, facilitating the creation of a mechanism to broadcast them to remote servers. The use of Abstract Factories and Observers allows the run-time redefinition of classes of the main application by the plugin. This mechanism is used to create new observers for broadcasting commands and graphic events.

3.2 NetGocad

Figure 2 shows a simplified diagram with NetGocad's main classes, responsible for the collaboration functionality. Most of them are implemented in CORBA, chosen as the distribution middleware because it is language-independent and multiplatform, and facilitates the integration with interoperability solutions used in the industry [18].

The central class in this architecture is the Participant, which acts as the main server for remote client requests, invoking operations in the local instance of the application. The Conference, instantiated by the Manager, keeps track of a group of Participants. It provides the same interface for channel events (section 3.4) as Participants do but broadcasts them to its members, either directly or through LuaConference, as discussed below. The abstract class Partner acts as a superclass for both Participants and Conference.

The main CORBA classes are described below:

Partner: Declares the interface for the methods that treat the communication channel events, actually implemented by Participants and Conferences. Clients keep references to a Partner.

Participant: Implements methods for treating communication channel events by invoking the appropriate operations at the host application (command execution, setting of the camera position, etc.).

Conference: Manages a group of Participants and broadcasts channel events to them. Also keeps track of floors: if a communication channel is controlled, the participant controlling the floor is the only one allowed to send events through this channel, as described in the next sections.

Manager: Manages the overall conferencing system. For a given domain (a company, for instance), it is responsible for registering and listing participants, creating and listing conferences, and allowing clients to connect to remote participants and conferences. It is the only published object, instantiated by a daemon, and serves as the entry point for the creation of collaboration sessions.

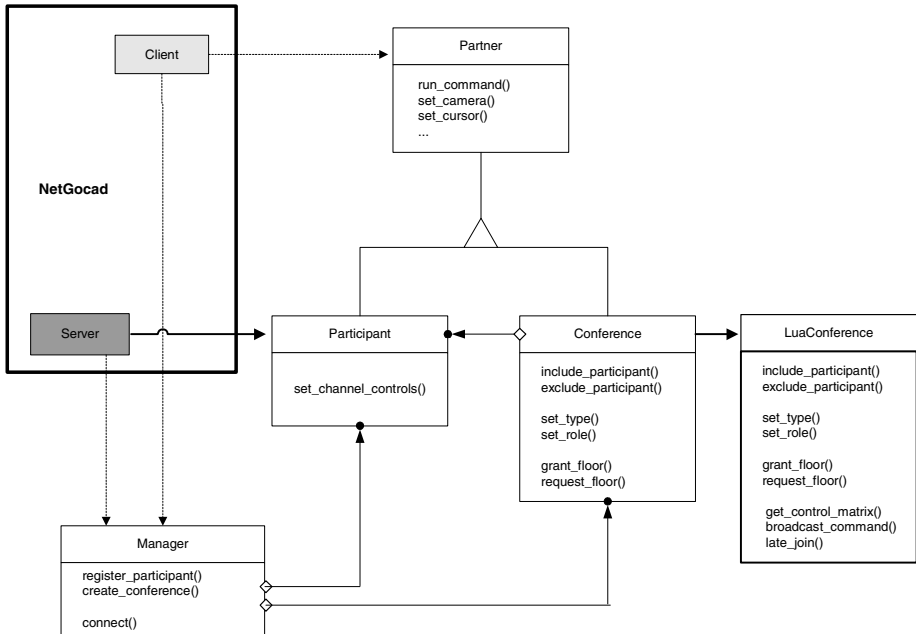


Fig. 2. NetGocad main classes (simplified)

Client, Server and LuaConference are not CORBA classes:

Client: Singleton class [9] implemented in C++. Instantiated by the application, provides methods for connecting to the Manager and to Partners, keeping references to them. When a user performs an operation locally (executes a command, a camera movement, etc.), a corresponding *observer* (redefined by the plugin for the application) will use the Client instance to invoke the same operation on the Partner it is connected to.

Server: Singleton class implemented in C++ used by the application to instantiate a Participant and to register it with the Manager.

LuaConference, a module implemented in the Lua extension language [12, 16], is created by each Conference at runtime and becomes responsible for the coordination policies. It loads the definitions of collaboration types and participant roles (section 3.5), and interacts with the Conference to provide configurable services (session management, floor control, definition of conference types and participant roles).

3.3 Operation

For a collaboration session to take place, a daemon instantiating a Manager needs to be running, responsible for registering and listing available Participants and Conferences, and for providing connection services.

When a group of people wants to collaborate, first somebody creates a conference, choosing a collaboration type among the ones loaded by LuaConference, and automatically becomes the conference owner. Then the other participants call the conference, choosing one of the roles specified for the current conference type. If the owner accepts a call, the caller is allowed to join and receives rights over the communication channels according to the chosen role (Figure 3). Afterwards, whenever a participant sends an event to the conference over a communication channel, it is broadcasted to all members enabled to receive it.

If somebody starts audio or video communication, the appropriate streams are created, according to the policy in place, through the videoconferencing tool (section 4).

3.4 Communication Channels

We define a “communication channel” as an abstract path for conveying information among instances of the application (commands, camera positions, telepointers, etc.), subject to a control policy. To each channel corresponds a method responsible for treating events, declared in the Partner class and implemented by Participants (except for audio and video, which are treated separately).

For the definition of the control logic in LuaConference, all channels are treated as elements of a homogeneous array, for which control matrices are defined. However, for the implementation of broadcasting mechanisms, we subdivide channels into *interaction channels*, *command channels* and *streaming channels*.

Interaction channels carry events that are generated by direct manipulations inside the 3D camera (camera movements, telepointers, avatars), that need to be processed at high rates for smooth graphic interactivity. Three-dimensional cursors (telepointers) identifying their users by color or label (Figure 1) allow participants to point to shared objects. Avatars display the position of the camera and the frustum of a participant, in conferences involving participants with non-synchronized points of view.

These events are not passed through the conference virtual machine (LuaConference). Instead, their broadcasting is done by the CORBA Conference and controlled by the switches (control matrices) set for the channel, as described below. If all participants are free to send and to receive the events, broadcasting is done through CORBA’s Event Service. Otherwise, this service cannot be used, because it provides no event filtering. The Notification Service (an extension of the Event Service) does allow event filtering and independent quality of service (QoS) control for each channel – an important feature, since different channels have different performance requirements. In this case, filtering would be based on the connection matrix, but currently this service is not employed in our implementation because it is not available in the CORBA version used [22].

Command channels carry the application commands, directly sent by the Conference to LuaConference, which is then responsible for broadcasting. The host application

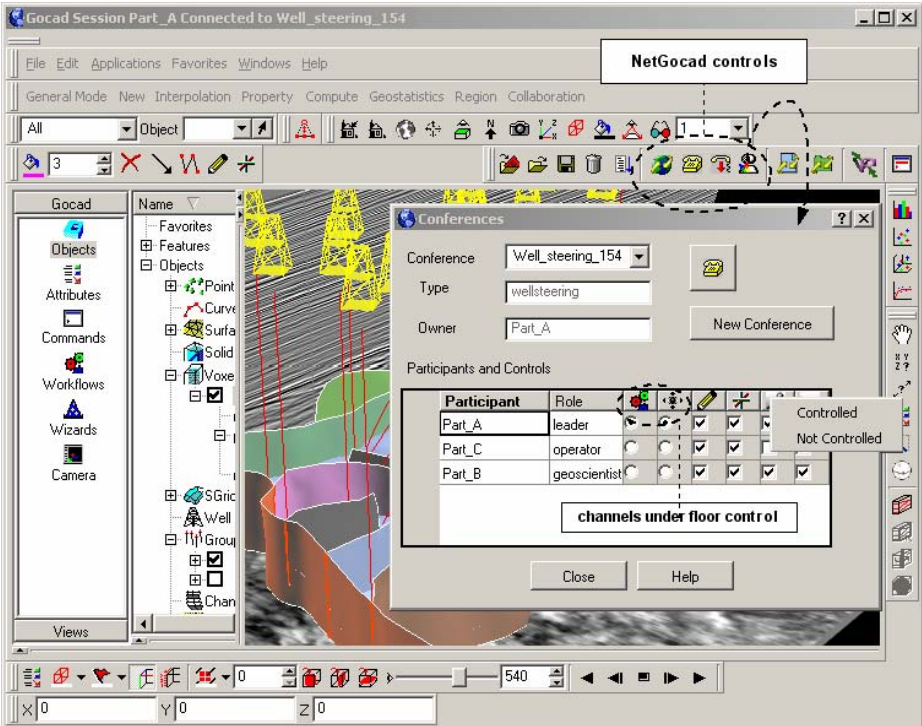


Fig. 3. Conference user interface

commands can be classified in a resource file associating lists of commands to labels. These lists are defined for the main application and for each plugin in a configuration file, and are then obtained by LuaConference and used during broadcasting for parsing and filtering commands. A mandatory class is “donot_broadcast”, defining all the commands that cannot be broadcasted (registration, connection, exit, model transfer, creation of interaction tools, etc.).

This mechanism can be used for the logical subdivision of the command channel into separate channels, so that each can be subject to independent floor control. This is currently done with three-dimensional annotations (freehand drawing, polygonal lines and arrow symbols) which, although similar in effect to other graphic interaction channels, are in fact generated by the application as commands only when the interactive creation of a primitive is complete.

The same type of command processing could be used for the specification of floors on objects and tools through the control of the commands that manipulate them.

All the commands broadcasted since the beginning of a conference are logged, to allow late joining (logged commands are sent by the Conference to new participants joining a session).

Streaming channels are subject to the same floor control mechanism but are handled separately (as described in section 4), due to the specific requirements they pose (in particular, audio and video streams are created in a peer-to-peer mode).

Control Matrices

In a conference with n participants, for each channel, $n \times n$ matrices of boolean values specify the connectivity state of each participant. Three types of matrices are used: *Key*, *Intention* and *Connection*. Key matrices specify the rights of each participant to send (K_{out}) and to receive (K_{in}) information to one another, according to the participants' roles in a certain conference type. Intention matrices specify the instantaneous intentions of participants to send (I_{out}) and to receive (I_{in}) information (given that they have this right). Connection continuously expresses enabled connections, as the result of the compilation of Key and Intention matrices.

A connection is established when the send and receive Keys and Intentions of correlated participants are true. That is, for each position in the connection matrix,

$$C_{ij} = (I_{out_{ij}} \wedge K_{out_{ij}}) \wedge (I_{in_{ji}} \wedge K_{in_{ji}})$$

The specification of the rights of each *role* to send/receive events to/from each other *role*, for each collaboration type, is done as described below (section 3.5). The $n \times n$ matrices for the participants are then dynamically assembled by LuaConference as they enter and leave the session, based on their roles, and communicated to the host application by the Conference. Intentions are directly specified by the participants through the user interface controls (enabled or not according to the key values).

3.5 Collaboration Specification

The control model adopted is inspired in COCA (Collaborative Objects Coordination Architecture) [15], but with some essential differences:

- as graphic performance is a main concern, our floor control mechanism changes the scheme defined in COCA based on the processing of all events through a virtual machine, for the control of input and output switches (tested by the event observers) at the host application by the plugin;
- instead of using logic-based rules (as in COCA) or declarative programs (as in DCWPL [3]) to define control policies – which are powerful and flexible but can become quite difficult to write and adapt by non-skilled programmers – we use some elegant mechanisms provided by the Lua language (tables in particular) to create a simple object-oriented syntax for the specification of collaborations, which are interpreted by the LuaConference virtual machine at runtime.

This scheme allows the definition of the different types of *Collaborations* that a conference can assume. Collaborations contain *Roles* (associated to the participants) and communication *channels*, for which *floors* can be defined. Collaborations specify an extensible set of default functions to provide floor services, session management services and regulation services (specification of conference types and participant roles), which can be redefined for particular collaboration types.

Communication channel names are associated to the host application's channel indexes. For each channel of a collaboration type the rights of each role to send/receive to other roles are specified (Key matrices). By default all channels are open (true values in the matrix do not need to be specified, only blocked connections need to be specified with false values).

Intention matrices by default are also filled with true values. False entries are used to specify connections initially blocked (for instance, for audio and video channels, so that no more than the desired streams are created in the beginning of a videoconference).

The controllable channels (the ones for which floors can be established by the conference owner during the conference) are defined; all others will remain free (with no floor control, although participants may still opt not to send or receive over the channel). Default roles for the conference owner (the participant who started the conference) and for new participants are also defined.

The communication between LuaConference and the Conference class in the host application follows the conventional mechanisms for Lua.

As an example, let us show part of a simplified definition for a well steering session, with some very basic roles and policies. In this example, VR users cannot send or receive camera events to/from anybody, and operators do not receive or send video streams at the beginning of the session (but can receive the floor afterwards).

```

-- correlate channel names to application indexes
appl_channels = {cmd=0, cam=1, annot=2, cursor=3, audio=4, video=5, avatar=6}

WellSteering = Collaboration {
    type = "wellsteering", -- define a collaboration type
    channels = {"cmd", "cam", "annot", "cursor", "audio", "video"}, -- controllable channels
    floors = {"cmd", "cam"}, -- channels initially under floor control
    LeadGeoscientist = Role {type = "leader"}, -- define roles
    Geoscientist = Role {type = "geoscientist"},
    VrGeoscientist = Role {type = "vr"},
    OnSiteOperator = Role {type = "operator"},
    K_in = { ["cam"]= {"vr:all"]=false }, -- define role-based matrices for
    K_out = { ["cam"]= {"all:vr"]=false }, -- each channel (all other role pairs,
    L_in= { ["video"]= {"operator:all"]=false } }, -- for all channels, are true by default)
    L_out= { ["video"]= {"operator:all"]=false } },
    default_owner_role = "leader", -- default role assumed by the owner
    default_participant_role = "geoscientist", -- default role for other participants
}
-- Obs: K_out, K_in, L_out, L_in are true for all other role pairs, for all channels

```

Collaboration methods can then be redefined in Lua, in an object-oriented way (for instance, *ClassRoom:grant_floor*).

LuaConference Interface

The interface provided by LuaConference to the host application implements the collaboration services (session management, floor control, etc.). Below is an extract of some of the main methods:

```

-- Session Management
function include_participant(participant, role)
function exclude_participant(participant)

-- Floor
function grant_floor(ch, participant, requester)
function request_floor(ch, requester)
function get_floor_controller(ch)

-- Regulation
function set_type(type) -- set collaboration type
function get_type()

```

```

function get_collaboration_types()
function init_channels()
function set_role(participant, role)
function get_role(participant)
function get_collaboration_roles(collab_type) -- return list of roles for the collaboration

// Broadcast commands
function broadcast_command(command, sender)

// Channels control
function get_control_matrix(M, ch) -- assembles the nxn participants matrix for the
-- channel, given the role-based control matrix M
-- (one of K_out, K_in, I_out, I_in)

// Late join
function late_join(participant) -- send all logged commands to the new participant
    
```

4 Videoconferencing

Audio and video communication channels are provided in NetGocad through the use of a custom multiplatform videoconferencing tool, CSVTool (Collaboration Supported by Video) [20]. This allows for a tight integration of this service, with no duplication of session management functionalities, and the direct control of audio and video streams according to the coordination policies defined.

The tool is integrated into the system through a separate plugin (gCSV) to avoid the establishment of a dependency. If the plugin is not present, the commands relative to videoconferencing are simply ignored.

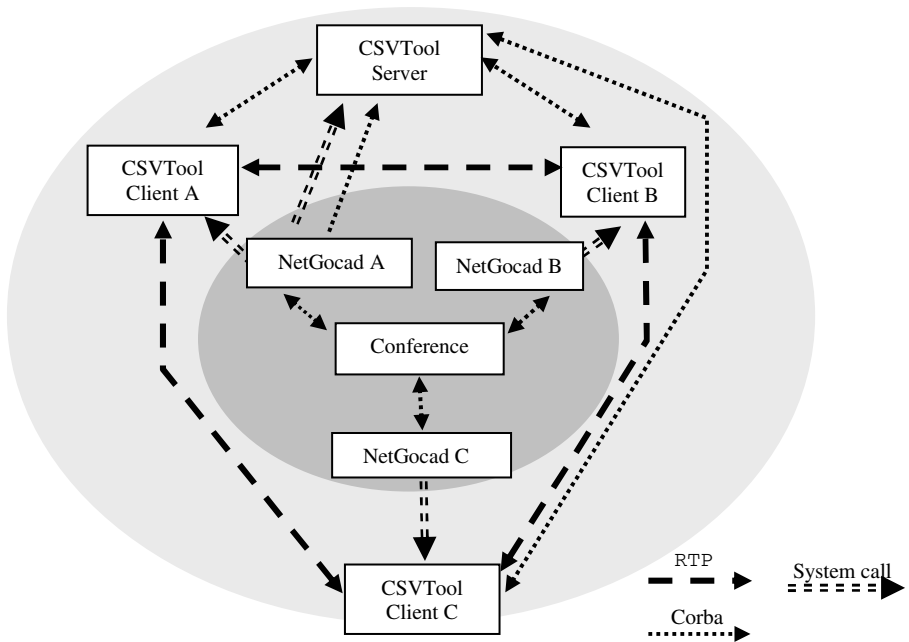


Fig. 4. CSV with NetGocad

CSVTool is implemented with JMF (Java Media Framework) [24] and can be used in two modes: integrated into a collaborative application or as standalone videoconferencing tool. Independently of the operation mode, it is divided into two modules, the server and the client. The server is responsible for the management of the participants and videoconferences, as described in the next section.

All the information exchanged among clients, except the audio and video streams, passes through the server. The most common messages are addition and removal of participants, which imply the creation or removal of streams. The server is not prone to traffic overburden because it does not receive the “heavy traffic” – the streams – which is transmitted directly between clients, in a peer-to-peer fashion.

The client/server communication is implemented in CORBA, and the communication among clients for stream transmission is made in RTP (Real- Time Protocol).

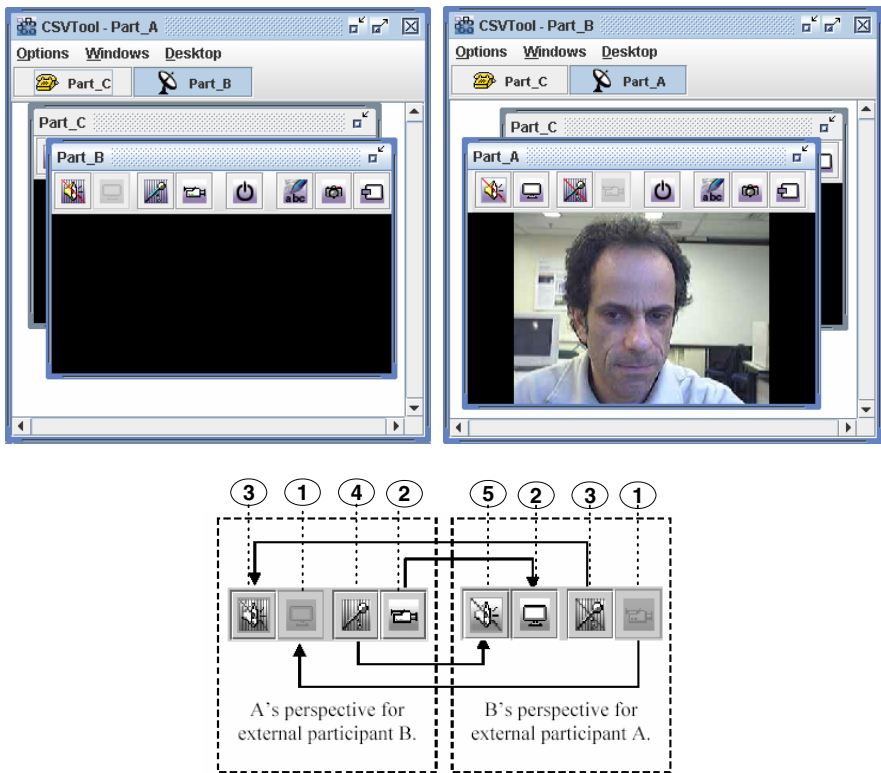


Fig. 5. CSVTool interface. Each audio/video control button may assume five different configurations: 1.Disabled: the respective stream cannot be activated because the capture device is unavailable or is disabled due to the participant’s role in the current conference type – the button becomes gray; 2. Active: the connection is active; 3. Off: both participants do not want to activate the stream –the button is dashed and hachured; 4. Waiting: the local participant wants to activate the streams, but the remote participant does not – the background of the button is hachured; 5. External: the remote participant wants to activate the streams, but the local participant does not – the button is dashed. Other buttons activate additional resources such as textual chat and snapshots.

CSV Operation with NetGocad

When a NetGocad participant starts a videoconference, the Conference launches a CSV server through its owner (Figure 4) and informs all Participants to start CSV clients, which automatically connect to the server.

The CSV server is then responsible for managing the clients and audio and video streams. All the communication between the NetGocad Conference and the CSV Server is done in CORBA, through the conference owner.

Control matrices move between the CSV clients and the server, whenever necessary, by means of message exchange. Key and Intention matrices are sent by the NetGocad Conference to CSV, which uses them to set the connections' state.

The CSV tool updates the received Key matrices to indicate the availability of capture devices at each participant's machine, tested at start time. Conceptually, disabling a connection due to a participant's role in a certain conference type is equivalent to switching off the device needed to send or to receive the related information.

One interesting feature provided is that the video stream sent by each participant can be switched from the image captured by the camera to the captured screen, so the tool can also be used for the remote display of a user's desktop. This is very useful for explanations about the operation of the application, or for consistency checks.

Complementary to the connection matrix, an External Intention matrix is computed in CSV to reflect connections that are not established because just one side decided not enable the stream. This information is used for the selection of the appropriate icon to represent the state of the connection at the user interface (Figure 5). As users have individual windows relative to one another, the intentions for each channel direction can be explicitly indicated. For the other channels, the setting of controls is done in a one-to-all basis (Figure 3).

5 Conclusion and Future Work

In this paper we presented an architecture that enables the transformation of a single-user application into a collaborative system by means of the integration of runtime plugins. We developed a control mechanism suited to the requirements of the application domain (collaborative three-dimensional geomodeling and visualization) and to the multimedia communication channels employed, based on the use of matrices associated to the channels, defining role-based rights and the dynamic intentions of the participants

The control scheme proposed, integrated into the application through a simple and flexible scripting language, generalizes the treatment of the different channels considered taking into account their specific performance demands. This scheme can be adapted to any application that follows some established design principles, particularly the use of the Command, Abstract Factory and Observer patterns.

We have concluded the integration of the tools and control mechanisms described, and are currently developing specific control policies for the scenarios discussed and others.

There are many issues that we would like to consider next. We want to assess the usability of the system under heterogeneous configurations, especially with the use of the VR plugin. This raises many usability issues that will require further development

of the immersive interface. We also need to conduct evaluations using an adequate methodology tailored for groupware, an issue now being studied.

Other difficult CSCW concepts we want to explore in the future are the use of private views, concurrency control and asynchronous collaboration supported by the workflow engine used by the core application.

We expect that in the long term the integration of these features will create new collaboration tools which will certainly have a strategic importance in the industry.

References

1. Anupan, V. and Bajaj C.: SHASTRA: An Architecture for Development of Collaborative Applications. *IEEE Multimedia*, Vol. 1, Number 2 (1994) 39-49
2. Campos, J.L.E.: Real-Time Well Drilling Monitoring using gOcad. 22nd GOCAD Meeting (2002) web site: www.ensg.inpl-nancy.fr/GOCAD/meetings/Nancy2002/
3. Cortez, M. and Mishra, P.: DCWPL: A Programming Language for Describing Collaboration Work. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (1996) 21-29.
4. Dewan, P.: Architectures for Collaborative Applications. In: Beaudouin-Lafon, M. (ed.): *Computer Supported Co-operative Work*, Vol. 7 of *Trends in Software*. John Wiley & Sons (1999) 169-193.
5. Dommel, H.P. and Garcia-Luna-Aceves, J.J.: Floor Control for Multimedia Conferencing and Collaboration. *Multimedia Systems*, Vol. 5, Number 1 (1997) 23-38
6. Edwards, W. K.: Policies and Roles in Collaborative Applications. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (1996) 11-20.
7. Ehrlich, K.: Designing Groupware Applications. In: Beaudouin-Lafon, M. (ed.): *Computer Supported Co-operative Work*, Vol. 7, *Trends in Software*. John Wiley & Sons (1999) 1-28.
8. Ellis, C.A and Gibbs, S.J.: Concurrency Control in Groupware Systems, *SIGMOD Conference*, Vol. 18 (1989) 399-407
9. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley (1995)
10. GOCAD Research Consortium, web site: www.gocad.org
11. Gruchalla, K.: Immersive Well-Path Editing: Investigating the Added Value of Immersion, In: *Proceedings of IEEE Virtual Reality* (2004) 157-164
12. Ierusalimschy, R, *Programming in Lua*, Lua Org (2003)
13. Inside Reality, web site: www.oilfield.slb.com/content/services/software/virtual/index.asp
14. Leikness, S., Osvoll, I.: Success Factors in Troll Geosteering. *Offshore Technology Conference* (2005)
15. Li, D., Muntz, R.: Coca: Collaborative Objects Coordination Architecture. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (1998) 178-188
16. Lua, web site: www.lua.org
17. Mallet, J.L., *Geomodeling*, Oxford University Press (2002)
18. Open Spirit, web site: www.openspirit.com
19. Pang A., Wittenbrink, C.M., Goodman T.: CSPray: A Collaborative Scientific Visualization Application. In: *Proceedings Multimedia Computing and Networking*, Vol. 2417 (1995) 317-326
20. Pozzer, C. et al :A Multi-user Videoconference-Based Collaboration Tool: Design and Implementation Issues, In: *Proceedings of the 9th International Conference on CSCW in Design* (2005) 547-552

21. Prakash, A.: Group Editors. In: Beaudouin-Lafon, M. (ed.): Computer Supported Co-operative Work, Vol. 7 of Trends in Software. John Wiley & Sons (1999) 103-133
22. Puder, A., Romer, K.: Mico: An Open Source CORBA Implementation", Morgan Kaufmann (2000)
23. Sun, C. and Chen, D.: Consistency Maintenance in Real-Time Collaborative Graphics Editing Systems. ACM Transactions on Computer-Human Interaction ,Vol. 9, Issue (2002) 1-41
24. Sun Microsystems, Java Media Framework Home Page, web site: java.sun.com/products/java-media/jmf/
25. Tay, F.E.H., Roy, A: CyberCAD: A Collaborative Approach in 3D-CAD Technology in a Multimedia-Supported Environment. Computers in Industry, Vol. 52, Number 2 (2003) 127-145

Remote Control Point Motion Prediction in Internet-Based Real-Time Collaborative Graphics Editing Systems

Bo Jiang^{1,2}, Jiajun Bu¹, Chun Chen¹, and Jianxv Yang¹

¹ College of Computer Science, Zhejiang University, Hangzhou, P.R. China

² College of Computer and Information Engineering,
Zhejiang Gongshang University, Hangzhou, P.R. China
{nancybjiang, bjj, chenc}@zju.edu.cn,
yangjianxv@yahoo.com.cn

Abstract. Monitoring the remote motion of objects or the control points of objects is one of the most important ways to promote awareness in Internet-based real-time collaborative graphics editing systems. However, such kind of remote control point motion is usually influenced by network jitter which leads to halting and jumping presence. Although motion prediction has been proved effective to complement the negative effect of jitter, the low accuracy of prediction remains a problem. In this paper, we present novel algorithms that can improve the accuracy to restore the remote motion smoothly and immediately. The prediction algorithms have been implemented in CoDesign - a prototype system of collaborative graphics editing. Experiments were carried out to test the effectiveness of the algorithms and the results show that by applying effective remote motion prediction the usability of the system can be greatly enhanced.

1 Introduction

As a special class of synchronous groupware systems, Internet-based real-time collaborative graphics editing systems (GES) [1, 2, 3, 4] allow several users geographically distributed to simultaneously view and edit the same graphics document. To achieve high responsiveness, a replicated architecture is usually adopted in which the shared documents are replicated and scattered at collaborating sites. Supporting good awareness [5] makes cooperators understand others' up-to-the-moment activities in a shared workspace, which becomes a crucial part of a successful collaborative GES with replicated architecture.

Monitoring the state of artifacts in a shared working setting is one of the most effective ways to gain awareness information. It is usually necessary for people to trace the movement of the artifacts which operated by cooperators especially in closely-coupled collaborative work. However, displaying the procedure of remote moving or dragging operations on certain artifacts often seems jumpy for the traits of delay and jitter of Internet. Incorrect representations of the movement of artifacts can lead to misunderstanding and even operation conflicting in collaborative graphics design. Therefore, ameliorate the effects of delay-jitter to display the remote motion in time

and smoothly becomes one of the most significant challenges in the designing and implementation of a successful GES with fine awareness.

The obvious solution to minimizing the effects of jitter is to introduce a buffer at the receiver to smooth the arrival process [6]. By applying such scheme, the buffered stream starts later but plays smoothly. Buffering eliminates jitter by increasing overall latency. However, delayed replay of collaborators' work will make great negative impact on the effect of awareness in real-time collaborative GES. Tracing [7] the visual embodiments of groupware system enhances the visual representation of collaborators' motion and complements the problem of jitter. As tracing can hardly maintain the immediacy or the naturalness of the original embodiments' motion, the technique is limited. Telepointer (one of the embodiments in collaborative workspace) prediction [8] is tested to be an effective way to present immediate and natural remote interaction. Yet little has been done to improve the accuracy of prediction.

In this paper we investigate the use of motion prediction as a technique for eliminating jitter effect on presence of remote motion. We first introduce Dead-reckoning algorithms into our system. We then present the Machine Learning algorithm that improves the prediction accuracy in certain degree. Adapting to the editing habit of users, prediction algorithms with Changeable Scale is proposed which shows better prediction performance in GES. Experiments of prediction algorithms are used to test the effectiveness of the prediction and the corresponding results are reported.

The structure of this paper is as follows: Section 2 describes the main characteristics of a collaborative GES with high usability and depicts the jitter effect on Control Point Based MOVE Operation. Section 3 presents three prediction algorithms and the corresponding experiments that test the effectiveness of the algorithms. Section 4 proposes comparison to related work. Finally, Section 5 concludes the paper.

2 Background

The goal of real-time distributed collaborative GES is to allow people in different places to edit shared graphics documents, as naturally and simply as they do in face-to-face collaboration. The critical issue in designing and implementing real-time collaborative GES with high usability in non-deterministic communication latency network environment is to meet the following requirements:

1. *high concurrency and relative unconstrained*: multiple users should be allowed to edit any part of the document at any time freely and users concentrate on certain parts of workspace that related to the assigned design work in most cases;
2. *high responsiveness and good awareness*: the response to local user actions must be quick and users can obtain detailed knowledge of remote users' activities.

2.1 Control Point Based MOVE Operations

Among various operations that can be acted on objects in object-based GES[4, 10], there's a special class of operations that related to the movement of a particular

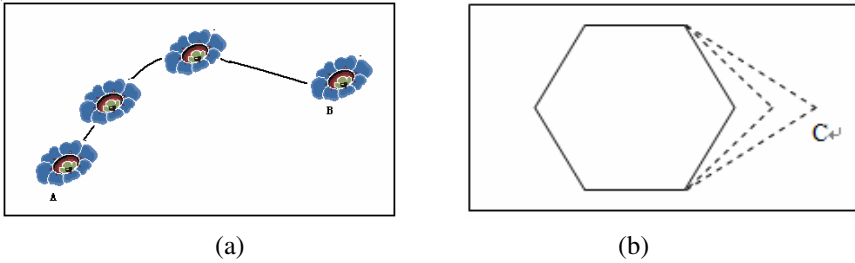


Fig. 1. Examples of CPB MOVE Op

control point, which is called Control Point Based MOVE Operation (CPB MOVE Op). In Fig. 1a operation MOVE changes the flower's coordinate from position A to position B. In Fig. 1b the shape of the polygon is modified by dragging Point C. Control points are the centre of the flower and Point C respectively in these examples.

It is always required that a continuous transmission of XY locations of control points to be sent over a network to provide collaborators the awareness information of the control point motion related to CPB MOVE Ops.

2.2 Jitter Effect on Control Point Based MOVE Operations

Distributed operation introduces communication delays between the collaborative sites on the Internet. Often, networks exhibit variability in delay, called jitter which can result in a jerky presentation of remote participant's actions. Continuous CPB MOVE Ops streams are sensitive to jitter that is due to the transmission over the Internet, such as transmission delay, propagation delay and queuing delay.

Tracing the continuous changing effect of the remote control point is aimed at conveying a sense of natural presence of others' activity, and the motion should be smooth. The control point's position must be updated with the pace of the movement of original motion. However, jitter over wide area network may lead to the following problems in closely-coupled collaborative graphics editing:

1. *semantic misunderstanding*: users may have an ambiguous idea of the semantic intention of remote user by monitoring the jerky display;
2. *operation conflicting*[9]: as it can be hardly to specify whether the halt of the motion is caused by jitter or not, users may take it for granted that the remote CPB MOVE Op task has been fulfilled and move the object that conflicts with remote CPB MOVE Op.

It is obvious that late-received control point stream and discontinuous presentation of others' motion that caused by jitter may decrease the usability of the system. In the following paragraphs, we present the scheme of control point prediction that can complement the effect of jitter.

3 Prediction Algorithms

Predicting the next location of control point based on past positions to simulate the actual track of the motion can eliminate the negative effect that brought by jitter. In the flowing paragraphs we present prediction algorithms used in our prototype system. We carried out corresponding experiments to evaluate the effectiveness.

3.1 Basic Dead-Reckoning Algorithm

(1) Algorithm

The first algorithm we used is a basic Dead-reckoning scheme. Dead-reckoning has been widely and successfully used in Internet game where Dead-reckoning improved player's interaction with distributed objects [8, 10, 11].

Here we indicate the next position of the control point by (X_{next}, Y_{next}) while the current position of the control point by $(X_{current}, Y_{current})$. The next position is calculated according to the following formula:

$$\begin{aligned} X_{next} &= X_{current} + aveVelocity_X + aveAcceleration_X, \\ Y_{next} &= Y_{current} + aveVelocity_Y + aveAcceleration_Y. \end{aligned}$$

where, $aveVelocity_X$ and $aveVelocity_Y$ indicate current average velocities of the control point in x and y axes. $aveAcceleration_X$ and $aveAcceleration_Y$ indicate current average acceleration of control point in x and y axes.

(2) Effectiveness

We carried out an experiment to test the effectiveness of the algorithm. Ten volunteers (5 male, 5 female) from local University and pattern designing company were invited to our lab. The experiment was conducted on Dell PC running CoDesign system application, using a 17-inch monitor set to 1024x768 resolution, 256M memory and 2.4G CPU. While volunteers were drawing and moving some graphics objects from one position to another, jitters were generated by a simulation application to simulate unstable network. Our prediction system application adopts the Dead-reckoning prediction and Machine Learning prediction respectively. At the end of jitter, system calculates the error of prediction, difference between the last predicted position and corresponding true position extracted from received package at the end of jitter. The testing result is shown in Fig.2.

3.2 Machine Learning Algorithm

(1) Algorithm

As it is shown in Fig. 2 mean error of the basic Dead-reckoning algorithm is unbearable with the increase of network jitter period. To solve the problem, the basic Dead-reckoning was improved to better smooth the prediction trace of control point. The new prediction system, which is called Machine Learning Algorithm, is able to adjust the prediction algorithm dynamically according to the former prediction error in the practical environment. The predicted position is calculated as follows:

$$\begin{aligned} X'_{next} &= X_{current} + aveVelocity_X + aveAcceleration_X + \Delta x = X_{next} + \Delta x, \\ Y'_{next} &= Y_{current} + aveVelocity_Y + aveAcceleration_Y + \Delta y = Y_{next} + \Delta y. \end{aligned}$$

where, Δx and Δy are variables to correct the values of X_{next} and Y_{next} respectively. Δx is related to: (1) the difference between last predicted X and first true X received at the end of jitter in the previous prediction process. (2) Current jitter lapse. (3) Jitter period in the previous prediction process. If it is the first time to predict, we did not consider Δx and Δy . Calculation of Δy is similar to that of Δx . To illustrate the algorithm:

$$\Delta x_2 = (X_{1'last} - X_{1true}) * JitterLapse_2 * 0.5 / JitterPeriod_1,$$

where, subscript 1 indicates the previous prediction process, while subscript 2 indicates the current prediction process. $X_{1'last}$ is the last predicted position in the previous prediction process. X_{1true} is the blocked last true value by the previous network jitter in the previous prediction process.

(2) Effectiveness

We carried out the similar experiment as above. The testing result of Machine Learning is shown in Fig.2.

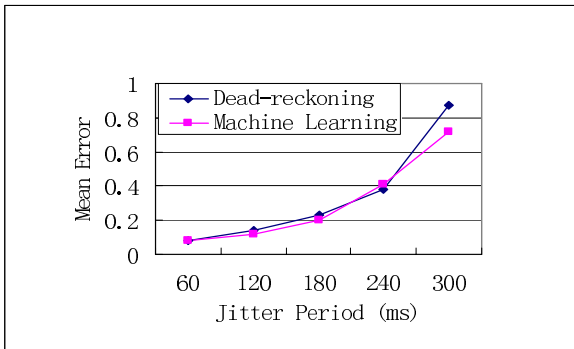


Fig. 2. Mean Error for 5 jitter periods with dead-reckoning and Machine Learning. The mean error is a difference divided by a unit length. If the mean error > 1 , we record it as 1, for it is beyond the prediction region too far. The difference is distance between the last predicted position and first position extracted from received package at the end of jitter. The unit length indicates a local area with user's most frequent activities. In our prediction system, the unit length = 10% of diagonal of canvas.

Generally speaking, Machine Learning is better than Dead-reckoning in terms of prediction accuracy, especially at the higher jitter periods. However, as illustrated, Dead-reckoning is better than that of Machine Learning at 240 ms.

3.3 Changeable Scale Algorithm

(1) Algorithm

As we know, in a certain period of time user usually concentrates on his/her nearby editing areas. When a control point is moved to somewhere else, new position will not

be far from the original one. The improved prediction algorithm, Changeable Scale, adapts to human habit in order to enhance the accuracy of the prediction in GES.

The main idea of the algorithm is to adjust prediction power according to the distance between the position user begin to drag and predicted position by original algorithm, such as dead-reckoning or machine learning. The longer the distance between the two positions is, the lower the next prediction power could be. The algorithm is as follows:

$$\begin{cases} X'_{next} = X_{next} * \alpha \\ Y'_{next} = Y_{next} * \alpha \end{cases}, \text{ where } \begin{cases} X_{next} = X_{current} + aveVelocity_x + aveAcceleration_x \\ Y_{next} = Y_{current} + aveVelocity_y + aveAcceleration_y \end{cases}$$

Where,

$$\alpha = \begin{cases} 1 & , Length \leq UnitLength \\ UnitLength / Length & , Length > UnitLength \end{cases}$$

where, *Length* is the distance between the position user begin to drag and predicted position by original algorithm. *UnitLength* is a unit length defined by system, which was described above.

Therefore, there is a circle centered by the position user begin to drag, with a radius of *UnitLength*. If the predicted position by original algorithm is within the circle, we

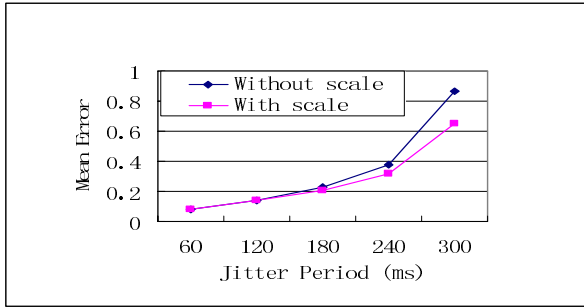


Fig. 3. Mean Error for Dead-reckoning with and without changeable scale

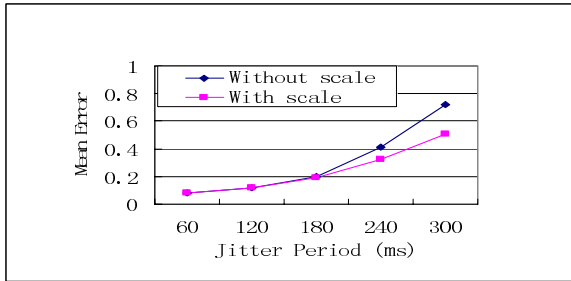


Fig. 4. Mean Error for Machine Learning with and without changeable scale

take the dead-reckoning algorithm unchanged for we get $\alpha = 1$. While the predicted position by original algorithm is without the circle, we take $UnitLength / Length$.

(2) Effectiveness

According to the experiment, comparison of the prediction with changeable scale to the prediction without changeable scale is shown in Fig. 3 and Fig. 4. As shown in both figures, prediction with changeable scale is better than that without changeable scale in terms of prediction accuracy, especially when jitter period increases.

4 Comparison to Related Work

Some work has been done in smoothing the display of remote motion of embodiments. [5, 8] applied Dead-reckoning to improve player's interaction with distributed objects in games. However, Dead-reckoning only presents good performance in predict the motion of objects that force-based and strong inertial properties. [9] Dead-reckoning prediction was applied to reduce the effects of jitter on telepointer motion. Experiments were carried out and suggest that prediction can increase the immediacy and naturalness of remote interaction in groupware system. Yet the accuracy of prediction remains a problem.

In our work, motion prediction of control point of objects in GES is studied. Machine Learning Algorithm improves the predicting accuracy of Dead-reckoning by adjusting the prediction algorithm dynamically according to the former prediction error. The performance is further enhanced by integrate Changeable Scale with the former algorithms. The control point movement will rarely be random because control point motion is bound with certain designing artifact. The intended motion in GES helps to attain better prediction performance. Experiments show that the improved prediction algorithms maintain the performance of prediction accuracy at usable levels.

5 Conclusions

Supporting awareness is an idea that holds promise for improving the usability of real-time collaborative graphics editing system. Restoring the consistent and immediate motion track of artifacts that controlled by remote user is proved to be effective to promote awareness. However, motion display performs poorly on Internet for delay jitter. In this paper, algorithms that used to predict the motion of control points in Internet-based real-time GES are presented. Machine Learning algorithm that improves prediction accuracy by taking the former prediction error into account. Changeable Scale algorithm that adapts to users' editing habits presents better prediction performance. Experiments are explored to report and compare the effectiveness of each algorithm. The study shows that the algorithms we proposed can limit the impact of jitter and provide collaborators with smooth and accurate motion awareness information, furthermore, dramatically improve the usability of GES.

We are pursuing our work in several directions. One is to develop adaptive motion prediction schemes suitable for various jitter periods and different graphics editing operations. Another one is to consider more sophisticated prediction scheme based on the characteristics of GES.

Acknowledgements

This paper is supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. Z603231. Thanks to the volunteers for taking part in the experiments.

References

1. C. Sun and D. Chen: Consistency Maintenance in Real-Time Collaborative Graphics Editing Systems. *ACM Transactions on Computer-Human Interaction*. 9,1 (2002) 1-41
2. X. Wang, et al. Achieving Undo in Bitmap-Based Collaborative Graphics Editing Systems. In: *Proceedings of 2002 ACM Conference on Computer Supported Cooperative Work (CSCW'02)*, November 16-20. New Orleans, Louisiana, USA (2002) 68-76
3. Xianghua Xu, Chun Chen, Jiajun Bu and Yong Li: Distributed Dynamic-locking in Real-time Collaborative Editing Systems. In: *Proceedings of 10th International Conference on Groupware (2004)* 271-279
4. Bo Jiang, Chun Chen, Jiajun Bu: CoDesign-A Collaborative Pattern Design System Based on Agent. In: *Proceedings of the Sixth International Conference on Computer Supported Cooperative Work in Design, Canada (2001)* 319-323
5. Gutwin, C., and Greenberg, S.: The Importance of Awareness for Team Cognition in Distributed Collaboration. In: E. Salas and S. M. Fiore (Editors) *Team Cognition: Understanding the Factors that Drive Process and Performance (2004)* 177-201
6. A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, R. Tewari: Buffering and Caching in Large-Scale Video Servers. In: *Proceedings of the 40th IEEE Computer Society International Conference, Washington, DC, USA (1995)* 217-224
7. Gutwin, C. Traces: Visualizing the Immediate Past to Support Group Interaction. In: *Proceedings of Graphics Interface 2002, Calgary (2002)* 43-50
8. Gutwin, C., Dyck, J., Burkitt, J.: Using Cursor Prediction to Smooth Telepointer Jitter. In: *Proceedings of Group 2003 (2003)* 294-301
9. Sun, C. and Chen, D.: A Multi-version Approach to Conflict Resolution in Distributed Groupware Systems. In: *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems. Taipei, Taiwan (2000)* 316-325
10. Capin, T., Pandzic, I., Thalmann, D., Magnenat-Thalmann, N.: A Dead-Reckoning Algorithm for Virtual Human Figures. In: *Proceedings of VRAIS'97 (IEEE Press), Albuquerque, USA (1997)* 161-168
11. Durbach, C. and Fournneau, J-M: Performance Evaluation of a Dead Reckoning Mechanism. In: *Proceedings of the Second International Workshop on Distributed Interactive Simulation and Real-Time Applications, IEEE Press, Montreal, Canada (1998)* 23-32

Synchronization Contexts as a Means to Support Collaborative Modeling

Niels Pinkwart

University of Duisburg-Essen, Faculty of Engineering,
47048 Duisburg, Germany
pinkwart@collide.info

Abstract. This paper presents an approach to support collaborative modeling with graph based representations. In particular, the problem of partially shared models with associated semantics is addressed, and an architectural solution to enable flexible modes of partial application synchronization under the constraint of retaining a common semantics in the shared model parts is presented.

1 Introduction

In science, the term *model* refers to a schematic, simplified and idealized representation of an object or a domain in which the relations and functions of the elements of the objects are made explicit. There is an analogy between the model and the object it describes in the sense that these two are structurally identical. Modeling is understood as the activity of creating, manipulating and using models. As models are a simplified and manageable means of understanding complex real phenomena, the importance of modeling in a number of professional and educational usage scenarios is evident [6].

A general function that computers can have in the domain of modeling is that they can serve as tools that execute models or run simulations based on models. Both is possible for many rather formal modeling languages like, e.g., Petri Nets [10] or System Dynamics [3]. Current networked computer systems are technically able to go beyond this. Archival and retrieval functions can, e.g., foster the exchange and re-use of modeling material. Networking also principally enables a cooperative synchronous use of modeling tools.

Among the variety of representations that can serve as a means for modeling, this paper concentrates on graph based ones - i.e., models consisting of visually explicit objects and their relationships. Several studies [8,12] indicate that this representational type (as opposed to, eg., textual forms or formulas) has certain advantages - including aspects like guidance for the users, the explicit structure of the model, and the fact that graphs seem to be inviting to users to "try out" creative solutions, which is an important aspect in modeling.

In addition, graph based representations are widely used, and the variety of modeling languages that rely on graph based representations (or that can, as one alternative, be represented in such a notation) is impressive. More formal

languages with exactly defined semantics are, e.g., Petri Nets [10], System Dynamics [3], or Entity-Relationship diagrams [2]. Languages with an intermediary level of formality (i.e., some parts of the expressions allow for an automatic interpretation and simulation, while others do not) include, e.g., most diagram types of the unified modeling language UML [1]. Finally, there are also a lot of qualitative modeling techniques that make use of graph based representations. In those, the object and link *types* can usually be exactly distinguished and are possible subject of interpretation, the *content* of these objects and links however is usually not accessible to computer based interpretation techniques. Examples of this category include mapping techniques like concept mapping [9].

This paper describes an approach that, retaining openness concerning the range of supported graph based modeling languages, focuses on the critical question of how to gain flexibly while (partially) sharing models that have associated formal or semiformal semantics.

2 Synchronization Contexts in Graph Based Modeling

2.1 Basics

Attempting to formalize shared graph based models, it is reasonable to start with a description of visual graph models:

Definition 1. *Let \mathcal{N} be a set of elements called node types, and let N be a set of nodes. Then a mapping $dom : N \mapsto \mathcal{N}$ is called a node type mapping. The image of dom , written $dom(N)$, is called node domain of N . Edge type mappings and edge domains are defined in analogy. If $dom_N : N \mapsto \mathcal{N}$ and $dom_E : E \mapsto \mathcal{E}$ are node type and edge type mappings, then a graph $G=(N,E)$ is called a typed graph over $(\mathcal{N},\mathcal{E})$.*

The following definition adds visual information to the concept of typed graphs:

Definition 2. *Given two sets V_N and V_E , called visual node attributes and visual edge attributes, then a pair $L=(\lambda_N,\lambda_E)$ of mappings with $\lambda_N : N \mapsto V_N$ and $\lambda_E : E \mapsto V_E$ is called a layout of a Graph $G=(N,E)$.*

A typed graph with associated visual attributes is referred to as a *visual typed graph*. This definition of a layout for a graph is abstract in the sense that it does not prescribe concrete sets V_N and V_E . Typical parameters would, e.g., be cartesian coordinates. Yet, a variety of alternatives (like, e.g., explored in visual language theory), are possible here.

To enhance visual typed graphs to "real" models, two more ingredients are necessary: syntax and semantics. While syntactic issues can be addressed using constraints over the visual typed graphs, the following definition for semantics is based on the formal notion of computational model semantics [5]:

Definition 3. *Let \mathcal{G} be a set of visual typed graphs. Then a couple (D, Ip) , consisting of a semantic domain D and a semantic mapping $Ip : \mathcal{G} \mapsto D$ is called a graph semantics for \mathcal{G} .*

This notion of graph semantics is independent of node and edge domains. Though this may appear unusual, as typically the semantics of graph based models are tightly bound to a modeling language and its primitives, the approach of partially decoupling semantics from concrete node and edge domains can contribute to model interoperability: it allows for defining semantic mappings that bridge gaps between modeling languages.

Although the semantic mapping is defined on the graph level, a lot of modeling languages allow for a decomposition of the semantic mapping in the sense of concrete node and edge semantics: here, notations like $Ip(n_G)$ (n being a node in the graph G) and, correspondingly, $Ip(e_G)$ make sense. Two examples:

- The semantics of a node in a calculation tree can be defined as the result of computing the value of the subtree.
- In the field of Petri Nets, the semantics of a place node can be identified as the number of tokens that the place contains.

2.2 Synchronization Contexts

One of the distinguishing factors between the present work and comparable approaches in the domains of metamodeling and visual languages is the explicit support for collaborative usage scenarios with flexibly shared representations. Though typically the concrete support for these mechanisms will be done on the concrete implementation level, there is a foundational issue that can be dealt with well already on the conceptual level: sharing graph structures, there is a possible discrepancy between flexibility of synchronization and coherence (or closure) requirements of models. The logical consequence of aiming at maximum degree of flexibility in sharing graph structures is to allow for a synchronization of arbitrary substructures (i.e. subgraphs) to the extreme case of having only single nodes synchronized. These partially shared structures have interesting application areas and allow for flexible work modes. An example for this is the following: with partially synchronized graphs, it is possible for users to privately work on the construction of a model and to "publish" only selected parts of it, e.g., a subgraph that contains some explicitly marked result elements. Insight into the way that these results were elaborated does not necessarily have to be granted to the group. There are also many ways of orchestrating these private/shared collaboration scenarios with partially shared models, as exemplified in the domain of mathematics [7].

While this degree of flexibility sounds attractive, there are also situations in which partially shared models may be problematic. Apart from the general question how edges could be coupled without also sharing the nodes that an edge connects, a critical point is that partially sharing models may lead to discrepancies between the semantics of the shared model parts. This is due to the

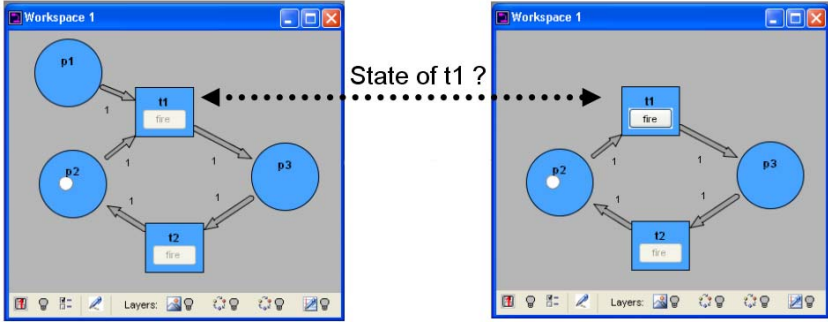


Fig. 1. The problems of partial synchronization

fact that the semantic mapping function is, in general, not context-free. It is not only the general graph semantics that varies, but also that of single nodes which are contained in both of the partially shared models. Figure 1 illustrates this problem with the example domain of Petri Nets. The two workspaces are partially synchronized and differ only in the presence of one single place (p1) and its connection to the transition t1. This causes the semantics of t1 to change, and in particular also affects the semantics of the whole graph: the left net is dead, whereas the right one is non-terminating.

Any general attempt to retain a common semantics between only partially shared (and therefore non-identical) models has to face one problem: either one single global semantics is preserved in the system (and the result is a mismatch between local representation and global semantics), or the semantics is only related to the respective local models. In the latter case, the problem is (as shown in figure 1) the non-existence of a common result.

One possible strategy to deal with this problem is to restrict the degree of flexibility concerning sharing entities. If the semantic mapping of a node does not depend on other entities, then it is reasonable to allow this node to be coupled independently of any other elements in the model graph. Otherwise, the (recursively determined) set of needed model elements has to be included in the set of shared elements:

Definition 4. Let $G=(N,E)$ be a visual typed graph with a semantics $Ip(G)$, and let $n \in N$ be a node of G . If n has an associated semantic value (i.e., the semantic mapping $Ip(n_G)$ of n in G is defined), then a synchronization context of n in G , denoted by $Sync(n_G)$, is a subgraph of G containing n so that $Ip(n_G) = Ip(n_{Sync(n_G)})$. A function $S : N \mapsto \mathcal{P}(G)$ so that each node is mapped to a corresponding synchronization context is called synchronization context mapping. A function $\mathcal{S} : N \times G \mapsto \mathcal{P}(G)$ which accepts a node and a graph (containing that node) as input and returns a subgraph which is a synchronization context of the node in the graph is called a generic synchronization context mapping.

Definition 5. A synchronization context $Sync(n_G)$ is called minimal if no real subgraph of $Sync(n_G)$ fulfills the synchronization context condition for n in G .

Proposition 1. Let $G=(N,E)$ be a visual typed graph with a semantics $Ip(G)$, and let $n \in N$ be a node of G with defined semantic mapping $Ip(n_G)$. Then a minimal synchronization context of n in G exists but is, in general, not unique.

Proof. A trivial synchronization context of n in G is obviously G itself, so that the existence is shown. The fact that $Sync(n_G)$ is in general not unique can be shown with a counterexample: a calculation tree consisting of the root node n_1 of type "×", and three child nodes n_2, n_3, n_4 of n_1 that are all of type "number" with $Ip(n_2) = Ip(n_3) = 0$ and $Ip(n_4) = 1$. Here, two different minimal synchronization contexts of n_1 in G are spanned by the node sets $N_1 = \{n_1, n_2\}$ and $N_2 = \{n_1, n_3\}$.

The proof of proposition 1 shows that the minimal synchronization context of a node in a graph can depend on the values of semantic attributes. This means that upon a change in semantics (e.g., caused by a model simulation step), the minimal synchronization context may change. Using synchronization contexts as foundations for partially coupled models, this has to be taken into account: in collaborative work contexts, a non-minimal but stable synchronization context may be superior to a minimal but frequently changing one.

For a number of modeling languages, minimal synchronization contexts can be defined easily, as the following example illustrates for the case of Petri Nets:

Example 1. Petri Nets are visual typed graphs that have the node type set $\mathcal{N} = \{place, transition\}$. For a visual typed graph $G=(N,E)$, a minimal synchronization context mapping S is as follows (for reasons of simplicity, only the nodes that span the synchronization context graph are given):

$$S(n) := \begin{cases} \{n\} & \text{if type}(n) = \text{place} \\ \{n\} \cup \{m \in G : (m, n) \in E \vee (n, m) \in E\} & \text{else} \end{cases}$$

This expresses that places can be synchronized node-wise, whereas the activation state and therefore the semantics of transitions depends on their input and output places, and thus on their complete neighborhood.

A strict consideration of synchronization contexts solves the dilemma between coupling flexibility versus coherence of models. If minimal synchronization contexts are used, the solution is even optimal in the sense that only the "absolutely required" information is shared. Yet, even apart from the dynamics of the minimal synchronization contexts (which may be a serious problem for collaborative work scenarios), one problem remains: there is no generic calculation algorithm for a minimal synchronization contexts. Especially in the case of modeling languages with non-formal semantics (like, e.g., concept maps), it is even unclear what such an algorithm should calculate. The next section of this paper shows an approach to solve at least some of these challenges.

3 Reference Frame Synchronization

Typically, all the concepts presented in the previous section have to be combined in order to express the characteristics of a certain modeling language. E.g., a specific constraint mapping usually belongs to a particular set of node and edge types, and a graph semantics may in turn rely on certain syntactic integrity constraints. For this reason, a central concept that bundles together all the ingredients makes sense. This can be conceived as the formalized abstraction of a visual modeling language:

Definition 6. *Let \mathcal{N} denote a set of node types and \mathcal{E} a set of edge types, and let V_N and V_E be visual node and edge attributes. For a set C of constraint mappings, a semantic domain D , a semantic mapping Ip , and a generic synchronization context mapping S , the tuple $\mathcal{R} = \langle \mathcal{N}, \mathcal{E}, V_N, V_E, C, D, Ip, S \rangle$ is called a Reference Frame.*

Based on this conceptual notion, a system architecture that allows for dynamically plugging in Reference Frames has been implemented in Java [11]. Details about type definitions, constraint mappings, and model semantics are beyond the scope of this paper - focusing on the group work support, we concentrate on the description of the synchronization contexts in the following:

The synchronization context mapping has an explicit representation in the `ReferenceFrame` interface: a method `synchronizeContext(Node, JGraph)` accepts a node and a visual typed graph as parameters. In conformance with definition 4, the policy for implementations of this method is that it calculates a synchronization context of the parameter node in the parameter graph, and couples the whole context instead of only the node. This way, an attempt of synchronizing a node in a graph with other graph instances can be processed *locally*, and lead to a coherently synchronized subgraph.

An alternative to this, which disburdens the developer from implementing the (non-trivial) method, would be the *automatic* calculation of (in the best case minimal) synchronization contexts. However, such a calculation is practically not reasonable unless restricting the semantics calculation severely. In particular, the following three steps would be required: (1) a way to make explicit all the variables in nodes, edges, graph, and the Reference Frame itself which belong to the semantics, (2) a method to compare these variables with partially synchronized applications, and (3) a technique to build the minimal synchronization context based on the results of the comparison. In particular the third point is the problematic one:

- Straightforward algorithms that simply test which subgraph is a minimal synchronization context are no real option due to their complexity, especially taking into account that (in step 2) this is a distributed algorithm.
- The idea of relying on the comparison results (step 2 in the algorithm) does not work either: even if the nodes and edges with varying semantics are known, the step of determining which elements of the graph must minimally be synchronized in order to "repair" the inconsistency cannot be derived easily due to the (required!) openness of the semantic mapping function.

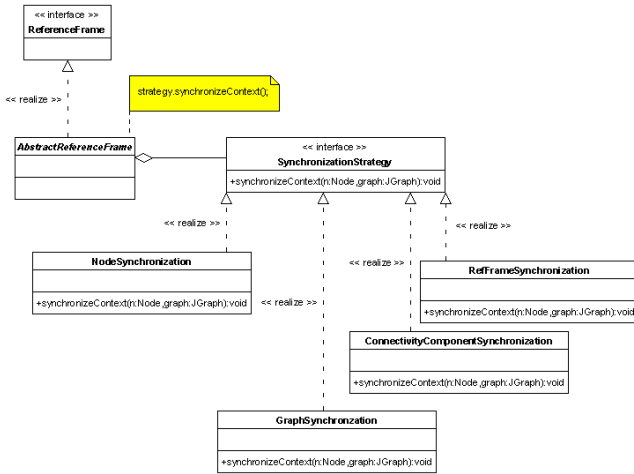


Fig. 2. Synchronization contexts as strategies for Reference Frames

- Finally, algorithms that go into detail about the interdependencies and are able to calculate a suitable synchronization context based on the comparison results are very similar (and not less complex!) than the ones required for implementations of `synchronizeContext(Node, JGraph)`.

In order to assist the developer in the task of defining suitable synchronization contexts, a number of typical algorithms that are applicable for a variety of modeling languages can be pre-defined and implemented in form of a Strategy pattern [4]:

Single Node. This simple implementation does not add any nodes to be additionally synchronized. This algorithm makes sense if the semantics of a node does not depend on the surrounding context in the graph.

Whole Graph. The second trivial case always synchronizes the whole graph upon the attempt to synchronize one node. This strategy guarantees a synchronization context, but obviously in most cases synchronizes too much.

Connectivity Component. In modeling languages where the graph structure plays an important role, the semantics may often be retained if the connectivity component that a node belongs to is synchronized along with the node.

Subgraph induced by Reference Frame. Typically, for "closed" (non-interoperable) modeling languages the subgraph of the graph which consists only of types known by the Reference Frame is a good candidate for a synchronization context.

4 Conclusions and Outlook

This paper introduced a means for adding a degree of flexibility to shared workspace systems that make use of graphs as primary representations: us-

ing synchronization contexts, also semantically rich structures can be partially shared - the method ensures that the joint parts in all the involved applications have the same "local interpretation". Current versions of the Cool Modes software [11] support the synchronization contexts as presented in this paper.

Ongoing research deals with the question if the minimal synchronization contexts, though formally providing shared semantically rich artifacts for collaborative work, are sufficient in the sense of producing a shared *understanding* in the involved user group - or if, on the other hand, there are certain situations in which a full model sharing is more suitable than partial sharing with even well-designed synchronization contexts and appropriate awareness mechanisms.

References

1. G. Booch, I. Jacobson, and J. Rumbaugh. *The Unified Modeling Language User Guide*. Addison Wesley Professional, Boston, MA (USA), 1998.
2. P. P.-S. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
3. J. W. Forrester. *Principles of Systems*. Pegasus Communications, Waltham, MA (USA), 1968.
4. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of reusable Object-Oriented Software*. Addison-Wesley Professional, Boston, MA (USA), 1995.
5. D. Harel and B. Rumpe. Meaningful modeling: What's the semantics of "semantics"? *Computer*, 37(10):64–72, 2004.
6. D. H. Jonassen. *Computers as Mindtools for Schools*. Prentice Hall, Upper Saddle River, NJ (USA), 2000.
7. M. Kuhn, H. U. Hoppe, A. Lingnau, and M. Fendrich. Evaluation of exploratory approaches in learning probability based on computational modelling and simulation. In *Proceedings of the IADIS conference of Cognition and Exploratory Learning in Digital Age (CELDA)*, pages 83–90, Lisbon, Portugal, 2004. IADIS Press.
8. S. Löhner, W. R. van Joolingen, and E. R. Savelsbergh. The effect of external representation on constructing computer models of complex phenomena. *Instructional Science*, 31:395–418, 2003.
9. J. D. Novak and D. B. Gowin. *Learning How to Learn*. Cambridge University Press, Cambridge, England, 1984.
10. C. A. Petri. *Kommunikation mit Automaten (communication with automata)*. Schriften des Rheinisch-Westfälischen Instituts für Instrumentelle Mathematik, Bonn, Germany, 1962.
11. N. Pinkwart. A plug-in architecture for graph based collaborative modeling systems. In *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AI-ED)*, pages 535–536, Amsterdam, The Netherlands, 2003. IOS Press.
12. D. D. Suthers and C. D. Hundhausen. An experimental study of the effects of representational guidance on collaborative learning processes. *Journal of the Learning Sciences*, 12(2):183–219, 2003.

Tailoring Infrastructures: Supporting Cooperative Work with Configurable Email Filters

Volkmar Pipek¹, Markus Won², Roman Englert³, and Volker Wulf⁴

- ¹ IISI - International Institute for Socio-Informatics, Heerstr. 148, 53111 Bonn
volkmar.pipek@iisi.de
Laboratory of Human-computer Interaction and Group Technology,
Department of Information Processing Science, University of Oulu Linnanmaa,
FIN-90570 Oulu, Finland
volkmar.pipek@tol.oulu.fi
- ² Conet AG Theodor-Heuss-Allee 19, 53773 Hennef, Germany
mwon@conet.de
- ³ Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
roman.englert@telekom.de
- ⁴ Institute for Information Sciences, University of Siegen
Hölderlinstr. 3, 57068 Siegen, Germany
wulf@fb5.uni-siegen.de
Fraunhofer Institute for Applied Information Technology (FhG-FIT)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
volker.wulf@fit.fraunhofer.de

Abstract. In fragmented work settings like network organizations or virtual organizations, monolithic approaches to implement support for collaboration would require the actors involved to agree on the usage of the approach or tool under consideration. As the autonomy of actors in these settings makes this hard to achieve, we suggest an exploration and an increase in the end-user tailorability of basic software infrastructures to enable even actors in these settings to tailor their collaboration support to their needs. An example for this strategy is illustrated by using email as a basic groupware technology. We use server-based email filters to improve the coordination of work processes and increase group awareness in these settings, and focus on making it easy for end users to understand and tailor the technology according to their needs. We use and enhance concepts from the discussion on the "tailorability" of CSCW systems (a visual filter composition language, a component-based architecture and additional support for exploration and documentation) to implement and evaluate our prototype.

1 Introduction

Computer support for cooperative work has not only become a key success factor for organizations, but also the enabling technology for new forms of inter-organizational cooperation and distributed work. Virtual organizations ([21];[16]) evolved as a new form of joining core competencies of different actors (individuals and organizations) to offer products and services beyond the skill and knowledge of the individual actors.

Comparing the technologies and tools used in virtual organizations to those in traditional forms of organizations we find heterogeneity rather than homogeneity, different tools, technologies and usages rather than a uniform, standardized groupware platform. Cooperation usually takes place in a heterogeneous infrastructure with some shared standards and basic technologies, but with many different tools and usages (cf. [26],[29]). Like in other collaborative settings, there still remains the necessity to be able to flexibly configure the infrastructure that is collaboratively used according to new or changed work contexts [9]. This characteristic of "Tailorability" has up to now mostly been explored for monolithic CSCW tools or frameworks (e.g. [14], Wang and Haake 2000). But there is the opportunity as well as the need to transfer tailorability to the domain of heterogeneous software infrastructures. From a user's perspective on their collaboration infrastructure as a whole, the tailoring support should mark "Group Tailoring Hot-Spots" of the infrastructure.

As a consequence of the technological heterogeneity encountered in virtual organizations, and a lack of motivation among actors to agree on shared standards, it is often email that remains most common denominator of cooperation technology. Important groupware functionality, like awareness support of workflows or cooperation structures is then usually not available in these settings.

In our work we explore the idea of fostering group support in heterogeneous software infrastructures by using server-side email filters to implement a lightweight technological support for group collaboration in these settings. Using standard email protocols and standard web technology we try to avoid interference with the technological infrastructure of the users.

Rather than providing as much groupware functionality as possible we focused on providing end-user friendly means to tailor the functionality to the given requirements of a specific collaborative setting. Here, we applied and enhanced concepts known from earlier research on tailoring and end-user development [23].

2 Related Work

2.1 Exploring Infrastructure Technology for Collaboration Support

There are various experiences to consider in the context of our work. A number of publications have recently paid more attention to the fact that support of collaborative work does not often operate on a stand alone basis, but rather is embedded in a technological infrastructure ([3], [5], [8], [4], [23]).

Dourish ([3],[5]) elaborated on the problems that the 'layeredness' of infrastructures, particularly the necessity to rely on lower layers of infrastructure when constructing a tool infrastructure for collaborative work, causes in the context of the development of collaborative systems. He also presented a concept of collaboration support that intertwines with the file system of an operating system instead of providing a separate tool [4]. Hanseth and Lundberg [8] addressed the role of standards in software infrastructures. On the one hand, standards are necessary to provide the compatibility among software tools, but on the other hand, standards also define structures of in- and exclusion of information and resources in collaborative

systems. Pipek and Kahler [23] addressed similar issues, and discussed the problems and opportunities for tailoring support by categorizing existing approaches to support tailoring in CSCW tools.

2.2 Email Technology

There are several problems for getting optimal support from this technology. There is no automatic support e.g. for different email categories (meeting request, note, reminder, etc.) except using the subject line, send mails as CCs, etc. If there is such functionality, it is usually client-based and therefore user-specific. So, some email clients (i.e. Microsoft Outlook) allow for sending meeting arrangements, or they create automatic answers. The disadvantage is that nobody knows which types of email clients are used by others and how they are configured. Even though the overall design of email clients has been discussed extensively lately [6], there has been not much progress regarding email usage on the group level.

As described, many approaches for email-based group support rely on client-based technologies. There are server-side email configuration systems as Procmail [7] but they are very difficult to use for non-programmers since they are script-based. Users need access to the server to configure such systems. To overcome this technological obstacle, web interfaces for these filter system (e.g. Websieve) have been developed. But these do not support a group-oriented approach (e.g. configurations always relate to one user). In practice, often only simple support, e.g. automatic replies in case of absence, is actually being used. Email-based tools for supporting collaboration have been developed earlier (e.g. [13], [1]). Using filters to support group cooperation is a new idea in that context. Using an email system with enhanced capabilities (group support, etc.) has also never been explored with sufficient support for end users so far.

2.3 Support for Tailoring

Tailorability today is a well-accepted property of information and communication systems designed to support group work. To fit the changing needs of group work in organizations, the technological infrastructure (i.e. the software tools) has to be flexible, and the flexibility has to be manageable for the users. The core question in the tailoring discussion is what can be done to adapt tools and related work practice in a use context to each other to support cooperative work in an optimal way (i.e. [9]). It has also been discussed how these aspects change the basic design of the software artifacts which are to be tailored.

The “architectural” perspective explored tailorability to develop concepts and examples of very flexible software systems, which could be adapted to their use scenarios ([12],[14]). Object-Oriented and Component-Based Systems [28] have been explored to increase the flexibility of software artifacts designed to support group work, other approaches addressed issues of analyzing, separating and composing tailoring entities along the typical functionality of CSCW systems [14].

The “user-interface” perspective explored how tailorable software should present itself to the tailors. Henderson and Kyng (1991) addressed the question, who the tailor is, and distinguished three levels of tailoring (choosing between predefined alternatives, constructing new artifacts from existing pieces, and reprogramming the

artifact) which require different levels of expertise regarding the supporting technology.

While the general discussion on tailorability was associated with CSCW settings, the discussion of these issues is now continued under the label of end-user development and considering a wider scope of tools ([23],[25]).

2.4 Component-Based Tailorability

Component-based architectures are very common in modern software development, and allow for better reusability of software components. Thus, software development becomes faster (and cheaper) as well as the quality can be increased (by using tested and well-known components). Not only do developers benefit by this technique, end-users too avail themselves of it if component-based architectures are used to build highly tailorable and manageable (for end-users) applications [15]. The most striking advantages are:

- The component concept is easy to understand: An application or composition consists of several components. Each of those components has its own function and they all work together as one system. The communication between components is done by sending messages them.
- The component approach allows for very powerful tailoring mechanisms: Components can be chosen, they can be parameterized and they can be bound together. Thus, the tailoring language only consists of three basic but very powerful operations.
- The visualization and the tailoring operations (visual tailoring language) can be figured out easily: Based on the first two points composing an application can be done by “drawing” it [27].

If we regard an email filter (or a set of email filters working together) as a component-based application then we can use the same techniques as mentioned above.

Several user studies have shown that those concepts can be easily understood by end-users [30]. In several thinking-aloud tests [18] and different applications as well as in a field study this was evaluated [30]. Here we found out that the critical point is to understand the semantics of a component (a single one as well as a compound component).

2.5 Tailoring Interfaces

Obviously, ordinary groupware users can not be expected to acquire programming skills to be able to tailor an artifact accordingly. Several approaches, some inspired by Nardi’s [17] work on end-user programming which aim at developing tailoring environments which provide simple concepts and interfaces for end-user (i.e. [14]).

There are several techniques that support using an application or learning its functionality. As mentioned above, in our case we first have to provide simple ways to understand single components. Furthermore, the workflow (or event flow) of a composition has to be taken into account.

In general, looking at features that support learning of tailoring languages we can draw on experiences concerning ordinary functions in single user applications.

Tailoring environments for users without programming skills should be designed consistently with ordinary functionality (cf. [12];[17]).

Features that encourage learning of single user applications allow structuring, describing, experimenting with and exemplifying the usage of the functionality (e.g. [2]; [20]). These features are provided by programmers for the users.

In the following, we address several concepts mentioned above that can be used to ease the learning of how to tailor or compose own filter as well as understand “pre”-configured email-filters (i.e. filters composed and used by other users).

Experimenting and Exploration: Mackay [11] as well as Oppermann and Simm [19] found that experimentation plays a major role in learning tailoring functions. Nevertheless, Mackay [11] reports that the fear to break something is a barrier to tailoring. Oppermann and Simm [19] found that the effects resulting from experimenting with tailoring functions are difficult to perceive. “Undo function”, “freezing points”, “experimental data”, and “neutral mode” are features which support users in carrying out experiments with a system’s function. Especially in the context of email or groupware in general experimental data can lower the barrier of tailoring as testing of system changes can affect other users.

Exemplifying: Examples provided by other users are an important trigger to tailoring [31]. Animation machines present a recorded sequence of interaction. Such animation gives an example on how users can apply certain functions.

Describing: Mackay [11] found that the lack of documentation of respective functions is a barrier to tailoring. Manuals and help texts are typical means to describe the functionality of applications. A description that is provided by the software vendor informs users about the state transition within the software system in which the execution of a function results.

3 Pre-study: Using Email in Virtual Organizations

Virtual organizations usually comprise of actors with a high level of autonomy that usually work in distributed setting for a limited time period to achieve a shared goal. Email technology plays an important role in connecting members of virtual organizations with each other [21]. More sophisticated and more powerful groupware technologies usually are available, but only in the form of complex products that have to be bought (costly), installed (time-consuming) and administrated (both). Additionally, users have to be trained. Users in a virtual organization often are very skeptical whether the benefit associated with the use of a groupware platform is worth the investments necessary to install, maintain and learn about the groupware platform. Thus, they usually fail to agree on one technology to use [29].

Email often remains as the only common technology to use for cooperation. In an explorative user study we further explored how email is being used as a collaboration technology in virtual teams. 22 interviewees from 7 organizations, working in distributed work settings and using heterogeneous infrastructures were asked to describe their current work contexts and how team members and external partners collaborate. Some communication scenarios were discussed regarding the specific problems that occur. We learned that usually the subject and the sender of an email

were used to identify the topic or the importance of an email. They should give a first insight how server-based email filtering could ease collaboration. Finally the participants were asked to outline their own scenarios. The resulting key scenarios for email usage were:

- Meeting arrangements: Those emails often include more than two persons. Answers are sent to all recipients then. This procedure requires many emails to be sent and read.
- Replying to external inquiries: sometimes produced by an email form integrated in a web site have to be answered. Sometimes these inquiries are archived (especially the included addresses) and the answer has to be checked by a colleague.
- Exchanging documents: The most common way to exchange document is to send them by email. Whereas working documents are only passed between team members, camera-ready document have to be stored at a special place. Furthermore, at some stage the team leader has to be informed about current work progress.

Those three scenarios show how email is used to coordinate work or collaboration. In many cases copies are generated and sent to another team member (superior, secretary, archive). Often this can be done semi-automatically depending on email properties such as subject line, recipient, including attachment (i.e. name contains "final version"), etc.

Looking at different types of emails we found that,

- recipients distinguished between personal mails (having one recipient) or "information mails" (more recipients or addressed to a group email alias),
- recipients distinguished external and internal (colleagues) senders, and
- in general, most emails were sorted only by checking the subject line and the sender's name, not by reading the mail itself.

These observations informed the design of email-based group support. On the level of collaboration, the following functions were assessed as helpful in a group scenario: Group-related configurations: Relating filtering rules not only to a person but to a group is an important step for establishing a shared notion of email-based collaboration understanding of email usage.

Transparency of configurations: In collaborative settings it is important to understand other individual's or other group's handling of the shared technology. Occasionally transparency is also necessary to understand one self's sorting schemes in order to find an email.

Awareness support (e.g. [24]: If users operate with email-based concepts to coordinate their work, a peripheral awareness of the email correspondence and the flow of work can be useful. Emails can be automatically sent to other's (interested people) as carbon copy (CC) or they can be forwarded.

To bring our ideas into practice, we estimated appropriate means for configuration and tailoring as more important than supporting more sophisticated concepts.

4 Concept and Architecture

Similar to Dourish's [4] way of enhancing a file system, the basic idea to introduce group functionality for collaboratively using email in fragmented work settings is to

use tailoring opportunities on ‘deeper’ layers of the shared infrastructure. The use of server-side email filters has the advantage of implementing groupware functionality at a location that can be shared by users even in heterogeneous infrastructures. Additionally, we exploited ideas from the discussions on tailorability to guarantee the system’s usability for end users.

4.1 Using a Component-Oriented Approach

Emails can be seen as messages that are passed between sender and receivers via a chain of servers. Server-sided email filters can provide additional navigation components (and connections between them) between individual mailboxes or mail servers. As described above, traditionally filter systems are to be configured by scripting languages, which are not suitable for most end users. We decided to describe the filter system as a net of components. Aside from using component-based visualization concepts, we also used component-based technology for transforming our component-based filter descriptions into a script that can be understood by the server (and vice versa). In using components on both levels, our system remains scalable for further extensions.

4.2 Useful and Manageable Filter Criteria

Our filter concept uses the MIME email properties (cf. MIME standard) to process emails. In addition to the MIME standard, it is possible to introduce new X-Tags (self-defined mail attributes) that can be used for filtering. The prototype presented here does not integrate all technical possibilities but aims at checking out which of the possible properties of emails and the resulting filters are relevant. Most of the scenarios we encountered in our pre-study can be automated with the help of filters that analyze the standard mail tags in various ways. The prototype should integrate the following kinds of filter techniques:

- Checking the sender’s name (on equality)
- Checking the subject (subject contains one or a set of words)
- Checking the recipient list.

As described above, additional filter types can be added easily (date, time stamp, visited server, etc.). Regarding the possible actions that are being performed based on filter results, our prototype needs to be able to

- generate copies of existing mails, and
- create notification mails (about what was send or received).

4.3 Architecture

Figure 1 shows an architectural view of our system. Client and Server are not modified by our enhancements. With our component-based approach we only changed the visualization of the filters, and not their functionality. There is also a local storage for the component-based mail filters which are converted into the script-based filter files every time the filter configuration changes. The filter administration environment accesses the filter configuration storage and changes its content. It

merely serves as an editor of the filter configurations that are stored on the server. The email client has access to the email server the same way as if there was no filter system.

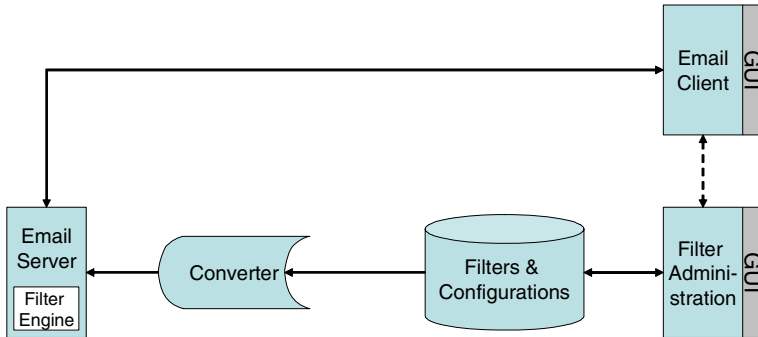


Fig. 1. Architectural Model

4.4 The Tailoring Client

As described above, our component-based approach not only allows for very flexible applications but is also easier to handle for non-programmers. Filters or little ‘filter workflows’ can be designed as a component net. In the visual tailoring language, we distinguish four kinds of components:

- **Inbox (before filtering):** This is where an email first arrives.
- **Mailbox (after filtering):** The final box the remaining emails fall into after all filters have been executed properly.
- **Several conditions:** Conditions (i.e. “contains subject ‘Project A?’”) always can be answered with a “yes” or a “no”. According to the answer something happens to the mail.
- **Created emails:** Those are the actions of the email filter system. After a condition is true a new email can be created. This can be the original message or a new (automatically generated) message. As discussed above, the email filter system also uses email as an awareness mechanism. For example, emails are created and sent automatically if something happens that should be considered as important (depending on the filter configuration).

All those components (switches with conditions, new email, etc.) are bound together in a complex component and represent one filter configuration. At the tool’s front end the filters are visualized so that changes to the composition can be done easily by adding a component (drag it from a toolbox) and connecting it (drawing lines between components).

To enable users to understand the whole system it is necessary to guarantee that all filters within the group are public and can be looked at. For this reason it is possible to also open other persons’ email filters. Another reason for looking at someone else’s filter configurations is to learn and understand how filters can be used. Filters can be used as examples that can be copied into the own filter configuration and customized.

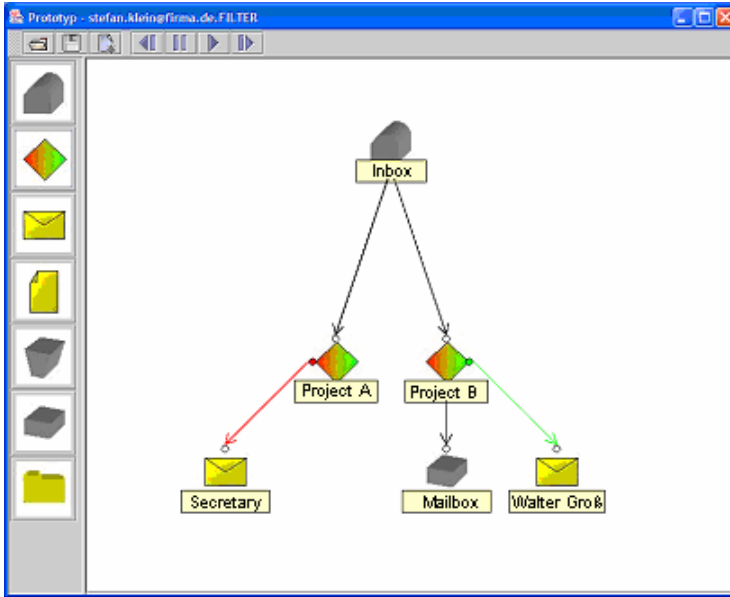


Fig. 2. Screenshot, analyzing filter configuration

Exploration mechanisms have been added to ease the tailoring for end users. Contrary to single-user applications like text processors, groupware services like email have the specific problem that they usually can not be ‘tried out’ because other users may be disrupted. Additionally, ‘learning by doing’, undo buttons, etc. are mechanisms that do not work in an email-based system since emails once sent can not be taken back.

Consequently, we had to integrate possibilities that ease learning and understanding, as well as, to provide means for testing or exploring tailored filters. Users can integrate annotations to whole filter nets as well as to parameterized components and their meanings within a filter. For the description of the exploration mode, see also the descriptions below.

5 Using the Prototype

First, we illustrate an example of its application, and explain the aspects of our prototype. Figure 2 depicts a screenshot of the filter administration tool mentioned in figure 1. Email filters are connected to processes (or ‘workflows’). Normally the inbox of the filter system is on the top of the screen and marks the beginning of the process. Arrows mark possible directions emails can take according to special conditions. In figure 2 we see a filter which first doubles incoming mails. After that “the left one” is checked if it belongs to “Project A”. If not so (red), it is forwarded to the typing pool (secretary). Then the “right mail” is checked on belonging to “Project B”. In this case it is forwarded to “Walter”. Additionally, it is put into the mail client’s inbox. Changing the configuration can be done easily by deleting the arrows

or adding filter components (see toolbox on the right side) using ‘drag and drop’ functions. The configuration of the switches can be done by clicking on it and editing the conditions. Several filter properties are also visualized. Those conditions are very simple boolean expressions as they have to be understood and expressed by the users themselves (i.e. Recipient = walter@noorg.de or Subject CONTAINS “important”). Combinations of expressions are allowed (using AND). Furthermore, every expression can be described in own words (comment) to ease understanding.

5.1 Additional Tailoring Support

To provide the necessary end-user orientation we integrated three mechanisms:

- Tool tips: Explanations are given by textual tool tips to each filter element. For example, as shown in figure 2, the condition “belongs the mail to Project A” is described as “Project A?”. So, inexperienced users can understand different filters and learn their functionality only by reading the annotations (some of which are generated by the system, some enhanced by the users themselves).
- Visualizing other users’ filters: Other group members’ filters can also be loaded into the filter administration tool, although they can not be changed. This eases the understanding how filters can be used efficiently and how they work. For privacy reasons, of course, the emails themselves remain invisible.
- Exploration mode: Users can easily check the configured filter with a test mail. The user then can take an existing mail (or a newly generated one), drag the email from the email client, and drop it onto the inbox of the filter administration client. From this stage, the mail is passed on sequentially through the filter. Newly generated mails (notifications) are visualized the moment they enter the system. If the email is forwarded to another user and other filters are being activated, a second filter window appears. So the user (filter configurator) can test the filter settings with “real” emails without actually sending them through the mail system, and therefore without disturbing other users.

All three techniques ease the understanding of the system. More experienced users learn the functionality of the current filter settings by analyzing the graphical structure of the filter and the additional text information. More complex settings – especially several users’ filter settings that have to be taken into account for understanding the group’s joint working space can be learned by exploring the system’s functionality using emails with the according properties (tags, headers, etc.).

5.2 Learning by Copy and Tailor

As described above the transparency mode can be used to learn about other users’ filter settings. If such filters are displayed they can be stored as filter components and integrated into own filter settings. By using this feature less experienced users can tailor their own filter settings by simply copying and adapting existing ones.

Furthermore, local administrators or super users can use this feature to design filter components for regular collaboration scenarios within their organization.

The quality of copied filter settings here can only be assured by organizational rules. For instance, administrators may store predefined filters at a special directory. If

users include some of those filter settings they know that they were designed by experts and have been tested.

5.3 Awareness and Transparency Functionality

Our email-based approach provides not only for awareness support, but also static transparency of others' filter configurations.

The filter techniques allow establishing awareness services during normal use of email. Notification emails can be generated and sent to colleagues if email with certain properties are being sent or received. For example, emails of important customers (identified by "sender") will be doubled and notification mails will be sent to group members.

By using the exploration mode that allow for investigating of all the users' email filter settings, work processes can be understood more easily. This is particularly important in virtual organizations that usually have no centralized organizational structure or transparent processes that could ease cooperation. Both awareness concepts allow for more transparency within the distributed organization. In the first case, the prototype was built without taking into account any privacy issues. Thus, all users within the group are allowed to browse through all email filters which leads to maximum transparency and learning effect. In the future, privacy issues have to be taken into account more seriously as filter settings may be a sensitive part of self-organization (for individuals as well as to groups).

5.4 Integration into Existing Infrastructures

The main goal of our approach was to allow for enhanced collaboration scenarios in virtual organizations where not all the members use the same technical infrastructure. Thus, what we needed was a technological basis that is available for all potential members without much technical overhead, and which allows greater end-user management flexibility. Email servers nowadays are very easy to handle and to administer (i.e. integration of new users, connecting to the email clients) and there are many providers who offer own email servers on their machines. Most of them are able to interpret script-based filter languages like procmail. This 'deeper' infrastructural level provides an access point for group-oriented functionality.

In the implementation of our prototype, the visualization and configuration of those filters can be done by using our filter administration client which is completely Java-based and therefore executable on many different platforms. The configuration files (XML) have to be stored in a highly accessible storage location.¹ This could be a web server for example. Simple access rights ensure that only the group members are allowed to read and change the contents. Using these technologies, that are the common denominator in today's internet-based infrastructures, it is possible to embed our concept into almost any given work infrastructure.

¹ The "filters and configurations" server is a centralized one. Instead of, different mail servers can be accessed by it.

On the filter server all email accounts of all participating users have to be accessible as the account information is needed to transfer the visually built filter configuration to the email server.²

5.5 User Experiences

To get some information about the feasibility of our concept in practice, we tested our prototype regarding its usability with a small number of users. Our evaluation was based on heuristic evaluation [18]. Three types of persons with different experiences and technical background were interviewed: “normal” users, professional users with programming skills but no administration experience, and one administrator. All of them are working in the field of IT consulting. Thus, they all have been working in virtualized and distributed working scenarios for a long time. First, they were presented with the idea of the software, and later the functionality was presented. After this step, they were asked to answer several questions concerning existing filter configuration. Finally, they were asked to configure own filters. The main results were:

- Exploration mode supports easy and exemplified learning: During the phase in which the interviewees should explain existing filter configurations the two more experienced users interpreted the annotations well, whereas the less experienced one used the exploration mode and generated different mails to understand the functionality of the filter.
- Building by exploration: In the third phase, the test persons had to configure their own filters. Both the programmer and the administrator designed their filters quickly by ‘dragging and dropping’ the filter components and binding them. The third person interactively developed parts of the filter, tested them by using the exploration mode and repeatedly changed parameters within the components. Interestingly the resulting filters differed in design but not in functionality.

The graphical tool was very helpful even for experienced users. Especially when filter configurations become more complex the graphical view is easier to understand compared to textual descriptions of filters.

Especially less experienced users felt very comfortable when using the exploration mode. It helped the understanding of the system’s behavior. The possibility to generate emails using the familiar email client and using them for testing the configuration increases the understanding not only of the filter system but also of email messaging in general.

6 Conclusions

The new forms of organization and collaboration like virtual or networked organizations produce an increased the level of fragmentation of work settings. To also support these settings with groupware technology, it becomes necessary to explore deeper levels of the technological infrastructure at hand, and those levels that

² In fact, the email server then receives procmail scripts which are based on the email configuration.

represent the most common denominator of the technologies used by the potential collaborators. Dourish [4] described such an approach at the level of file systems. We have now described such an approach using standard email technology to provide groupware functionality (awareness support, modeling of small workflows, additional notification services). To make these infrastructures adaptable for end users, we not only have to find "tailoring hot spots" within the shared technology and exploit them for group tailoring support, but we also have to provide the flexibility offered by the software infrastructure in a way that makes it manageable by end users. Therefore, a significant amount of our conceptual work went into the application of tailoring support concepts from standard CSCW tools for these 'infrastructural' technologies of email and web servers. In a small evaluation study, we were able to show the suitability of our concept for end users. Still, there are many open questions around this field of research. Following the use and the further development of email filters over time, especially in those cases where the mutual filter settings are known by all users of a group. The installation of an open community support concept around the configuration of the filters, as e.g. described in the concept of the 'Use Discourse Environments' by Pipek [22] can help foster the appropriation of the technology on a social level. That way, in the long run, it would be possible to find collaboration patterns based on email communication that could then be supported more specifically. Elaborating on earlier work on semi-structured messages [14], a decentralized and standardized support of message filtering could be a powerful infrastructure for the information and knowledge management of the enterprises of the future.

Acknowledgements

We would like to thank Radhakrishnan Subramaniam, Matthias Betz, and the anonymous CRIWG reviewers for their comments on earlier versions of the paper. The research presented here has been partially funded by the German Ministry for Education and Research within the 'Olvio' project under the reference number 01HG8890.

References

1. Camino, B. M., Milewski, A. E., Millen, D. R., & Smith, T. M. (1998). Replying to email with structured responses. *Int. Journal on Human-Computer Studies*, 48, pp. 763-776.
2. Carroll, J. M.: "Five Gambits for the advisory Interfacs Dilemma", in: Frese, M., Ulich, E., and Dzida, W. (eds.), *Psychological Issues of Human Computer Interaction in the Work Place*, Amsterdam, 1987, pp. 257-274.
3. Dourish, P. *Software Infrastructures*. in Beaudouin-Lafon, M. ed. *Computer Supported Cooperative Work*, John Wiley & Sons, 1999, pp. 195-219.
4. Dourish, P. The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents. *Computer Supported Cooperative Work (CSCW) - The Journal of Collaborative Computing*, 12 (4). 2003, pp. 465-490.
5. Dourish, P. and Edwards, W.K. A tale of two toolkits: Relating Infrastructure and Use in Flexible CSCW Toolkits. *Computer-Supported Cooperative Work (CSCW)*, 9 (1), 2000, pp. 33-51.

6. Gruen, D., Rohall, S.L., Minassian, S., Kerr, B., Moody, P., Stachel, B., Wattenberg, M. and Wilcox, E., Lessons from the reMail prototypes. in Proceedings of the 2004 ACM conference on Computer supported cooperative work, (Chicago, Illinois, USA, 2004), ACM Press, pp. 152-161.
7. Hampton, C.A.: "Getting Started With Procmal", <http://www.spambouncer.org/proctut.shtml>, 8.1.2002.
8. Hanseth, O. and Lundberg, N. Designing Work Oriented Infrastructures. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, 10 (3-4), 2001. Pp. 347-372.
9. Henderson, A. and Kyng M.: "There's No Place Like Home. Continuing Design in Use", in: *Design at Work*, Lawrence Erlbaum Associates, Publishers, 1991, pp. 219-240.
10. Lieberman, H., Paternó, F. and Wulf, V. (eds.). *End User Development*. Kluwer, Dordrecht, NL, in press.
11. Mackay, W. E.: "Users and customizable Software: A Co-Adaptive Phenomenon", MIT, Boston (MA), PhD Thesis, 1990.
12. MacLean, A., Carter, K., Lövstrand, L., and Moran, T., "User-tailorable Systems: Pressing the Issue with Buttons", in: *Proceedings of the Conference on Computer Human Interaction (CHI '90)*, April 1-5, Seattle (Washington), ACM-Press, New York, 1990, pp. 175-182.
13. Malone, T.W., Grant, K.R., Lai, K.-Y., Rao, R. and Rosenblitt, D., *Semistructured Messages are Surprisingly Useful for Computer-Supported Coordination*. in *Proceedings of CSCW 88*, (1988), Morgan-Kaufmann, pp. 311-334.
14. Malone, T.W., Lai, K.-Y. and Fry, C., *Experiments with Oval: A Radically Tailorable Tool for Cooperative Work*. in *Int. Conference on CSCW (CSCW'92)*, Toronto, Canada, 1992, ACM Press, pp.289-297
15. Morch, A., Stevens, G., Won, M., Klann, M., Dittrich, Y. and Wulf, V. *Component-based technologies for end-user development*. *Communications of the ACM*, 47 (9). 2004, pp. 59-62.
16. Mowshowitz, A: *Virtual Organization*. In: *Communications of the ACM*, 40-9, 1997, pp. 30-37
17. Nardi, B. A., "A Small Matter of Programming - Perspectives on end-user computing", MIT-Press, Cambridge et al., 1993.
18. Nielsen, J.: *Evaluating the Thinking-Aloud Technique for Use by Computer Scientists*. In: *Hortson, H.R., Hix, D. (Hrsg.): Advances in Human Computer Interaction*, Vol. 3, 1992, pp. 69-82. Nielsen, J.: *Usability Engineering*. AP Professional, New York, 1993.
19. Oppermann, R. and Simm, H., "Adaptability: User-Initiated Individualization", in: *Oppermann, R. (ed.): Adaptive User Support – Ergonomic Design of Manually and Automatically Adaptable Software*, Lawrence Erlbaum Ass., Hillsdale, New Jersey, 1994.
20. Paul, H., "Exploratives Agieren", Peter Lang, Frankfurt/M (Germany) 1994.
21. Picot, A., Reichwald, R., Wigant, R.: "Die grenzenlose Unternehmung - Information, Organisation und Management", 3. Aufl., Wiesbaden, Germany, Gabler, 1998.
22. Pipek, V., *From Tailoring to Appropriation Support: Negotiating Groupware Usage*, Faculty of Science, Department of Information Processing Science (ACTA UNIVERSITATIS OULUENSIS A 430), University of Oulu, Oulu, Finland, 2005, 246 p.
23. Pipek, V. and Kahler, H. *Supporting Collaborative Tailoring*. in *Lieberman, H., Paterno, F. and Wulf, V. eds. End-User Development*, Kluwer, Dordrecht, NL, 2005, to be published.
24. Sandor, O., Bogdan, C. and Bowers, J., *Aether: An Awareness Engine For CSCW*. in *5th European Conf. on CSCW (ECSCW'97)*, Kluwer., 1997, pp. 221-236.

25. Sutcliffe, A. and Mehandjiev, N. Special Issue on End-user development: tools that empower users to create their own software solutions. *Communications of the ACM*, 47 (9), 2004.
26. Rittenbruch, M., Kahler, H. and Cremers, A.B., Supporting cooperation in a virtual organization. in *Proceedings of the international conference on Information systems*, (Helsinki, Finland, 1998), Association for Information Systems, 1998, pp. 30-38.
27. Stiemerling, O., Component-based tailorability, Ph.D.-Thesis, Institute for Computer Science III, University of Bonn (Germany), 2000.
28. Stiemerling, O. and Cremers, A.B. The EVOLVE Project: Component-Based Tailorability for CSCW Applications. *AI & Society*, 14. 2000, pp. 120-141.
29. Törpel, B., Pipek, V. and Rittenbruch, M. Creating Heterogeneity - Evolving Use of Groupware in a Network of Freelancers. Special Issue of the *Int. Journal on CSCW on "Evolving Use of Groupware"*, 12 (4). pp. 381-409
30. Wulf, V.: "Let's see your Search-Tool! – On the Collaborative Use of Tailored Artifacts", in: *Proceedings of Group '99*, ACM Press, New York, 1999, pp. 50-60.
31. Wulf, V. and Golombek, B. Direct Activation: A Concept to Encourage Tailoring Activities. *Behaviour & Information Technology*, 20 (4), 2001, pp. 249-263.

A Collaborative Framework for Unexpected Exception Handling

Hernâni Mourão^{1,*} and Pedro Antunes²

¹ Escola Superior de Ciências Empresariais, Instituto Politécnico de Setúbal,
Campus do IPS – Estefanilha, 2914-503 Setúbal, Portugal, and
LASIGE (Laboratory of Large Scale Information Systems)
hmourao@esce.ips.pt

² Faculdade de Ciências, Universidade de Lisboa, Departamento de Informática,
Campo Grande – Edifício C5, 1749-016 Lisboa, Portugal, and
LASIGE (Laboratory of Large Scale Information Systems)
paa@di.fc.ul.pt

Abstract. This paper proposes a collaborative framework handling unexpected exceptions in Workflow Management Systems (WfMS). Unexpected exceptions correspond to unpredicted situations for which the system can not suggest any solutions. We introduce the notion that exception recovery is a collaborative problem solving activity that should be addressed through an intertwined play between several actors performing two types of tasks: (1) diagnosing situations; and (2) planning recovery actions. We propose a set of dimensions to classify the exceptional situations and their relations to recovery strategies. We also discuss the importance of monitoring recovery actions within the scope of diagnosis tasks. The proposed solution is implemented through a dedicated workflow.

1 Introduction

The work processes carried out by organizations in their daily operations have been identified to belong to a continuum ranging from totally unstructured to completely structured [33]. The majority of the available organizational information systems fall close to both sides of the spectrum boundaries [33]. In particular, traditional WfMS fall into the highly structured boundary, usually supporting organizational processes through the execution of work models. In the context of Schmidt's work [32], work models play the role of scripts in formal organizational structures and have a normative engagement. Closer to the other end of the spectrum limits, Suchman [35] proposes the notion of maps, which position and guide actors in a space of available actions, providing environmental information necessary to decision making but avoiding a normative trait.

* The author is deeply grateful to Ulm's University group for their friendship and scientific support during his staying. Their sharing of ideas and great field experience on Workflows was very important to this work.

To support the various organizational needs, WfMS should cope with the whole spectrum of structured and unstructured activities. This requirement has been identified by Ellis and Nutt [18], when they realize that WfMS must be flexible to succeed. Also, Abbot and Sarin [1], based on empirical evidence, claim it is necessary to integrate procedural and nonprocedural work in WfMS. They define nonprocedural work as “unchoreographed interactions between people.”

In the WfMS community nonprocedural work has been designated “exception handling,” encompassing the set of actions aiming to react to a kind of event that is out of the scope of the work model. Exceptions either can not be predicted during the design phase or, although being predictable, are deliberately excluded from the work model to reduce complexity [6; 10; 14; 24; 31]. The Eder and Liebhart’s [14] classification of expected and unexpected exceptions has been widely accepted, since it enables a division between the exceptions that can be predicted in the design phase from those that can not. In our work, we advocate a novel approach to exception classification, assuming a continuum from expected to unexpected exceptions.

This paper is structured in the following way. We start by revising the notion of completeness in an exception handling WfMS. Then, we present a framework to deal with the above mentioned dichotomy: maintain model-based work whenever possible and change to a kind of map guidance whenever the scope is outside the limits under which the work models were designed [2]. A set of guidance mechanisms is proposed to support users dealing with unexpected exceptions, delivered in the form of contextual information about the affected processes. As Bernstein [5] states, emergent actions must be sustained by context information as actors dealing with these environments become overloaded with information.

Our solution is based on a previous developed exception handling workflow [25]. In the present work we expand the exception handling workflow with three functions [31]: detection, diagnosis, and recovery. Most importantly, the recovery phase is now intertwined with the diagnosis phase, as we came to realize that a proper diagnosis is an iterative process requiring harvesting contextual information and collaboration between users.

We also classify exception handling strategies and relate them with exceptional situations. Finally, we present and discuss the implementation details, illustrate the framework usage with an example, and finish with the conclusions and future work.

2 Revising the Completeness Requirement

To be complete, an exception handling system should consent users to carry out recovery activities without restrictions, i.e., the flexibility of the exception handling system should be on par with the flexibility actors have on their daily activities when working without system control. Several impacts of this definition should be taken into consideration.

This definition is based on the notion that people tend to solve their problems with all the available means. If any system restrictions are imposed to the users’ primary objective of reaching a solution, they will overcome the system [21; 34].

The consequences of this open perspective on WfMS are profound. The common restrictions to ad hoc model changes, based on structural and dynamic properties,

must therefore be relaxed. We believe that these restrictions are only applicable if one wants to keep the execution under the specified work models. However, if the objective is, for instance, to graciously abort a workflow instance, no consistency check is necessary. Even further, if the user decides to implement a recovery action that deliberately inserts structural conflicts in the work model, s/he should be advised on potential problems but allowed to carry out that action.

On the other hand, users should not be restricted to the services provided by the exception handling system, since they will use everything needed to overcome the situation. The challenge is to implement a comprehensive set of services that may reduce the user's needs to handle exceptions outside the system scope. The framework described in this paper integrates such services while being open to the organizational environment. Thus, some exception handling activities will be partially outside the WfMS scope. The framework integrates environmental information about external activities, thus guiding actors in the course of actions, but will not assume control of those activities. These environmental monitoring tasks collect information necessary to plan the recovery actions or monitor the evolution of actions implemented out of the framework's scope.

3 Related Work and Scope of the Framework

The main appointed reasons for the lack of flexibility in current WfMS are: 1) complex work models where only predictable events are foreseen (expected exceptions, see below) [6; 7; 11]; 2) inability applying model changes to already running instances [17; 28; 36]; 3) difficulties applying ad hoc changes to cope with very small model variations [28; 36]; 4) tight coupling between modeling and enactment [19; 23]; and 5) formal models currently adopted to represent work are inadequate to flexibility support [13].

Various approaches to flexibility can be found in the literature, as different authors understand differently the properties a WfMS should exhibit to effectively deal with office work. We identify two parallel research streams [20]: meta-model and open-point. Meta-model approaches take into major consideration structural and dynamic constraints to model adaptations, while open-point approaches rely on the users' abilities to assure that no inconsistencies are inserted in the system.

The meta-model approaches fundamentally address expected exceptions, coded in special constructs and invoked whenever a predefined exceptional situation is detected [6; 10; 11; 14] – e.g., Event Condition Action (ECA) rules. Several techniques, such as exception mining [10; 22], case base reasoning [24], conversational case base reasoning [37], and knowledge bases [12] have been proposed to expand the system flexibility handling exceptions. If we consider a continuum from expected exceptions to completely unexpected exceptions, all these systems handle events falling close to expected exceptions limits of the spectrum.

On the meta-model approaches to address unexpected exceptions, we find several solutions [7; 17; 26; 36]. The most important distinction from the previous set, is that these solutions support dynamic changes and ad hoc interventions.

Regarding the open-point approaches in more detail, we find interactive enactment [23] and flexible enactment [19]. These approaches assume work models are

incompletely specified, allowing users to interactively adapt them, e.g., inserting tasks. This increases the degree of freedom on the user side to cope with deviations between the work models and the real world, although in a more structured way than a totally open-point intervention would afford. In any case, users will be able to insert unidentified inconsistencies, and possibly put the WfMS at risk [20], considering that no dynamic or structural checks are made.

Like Agostini and De Michelis [2], we agree with both research streams delineated above and posit that a WfMS should offer both advantages: being able to work under model guidance and adopt an open-point behavior when model guidance is not applicable. However, after open-point operations, the system should support users bringing instances back to model control, while identifying potential flow and data inconsistencies. A complete discussion of the mechanisms necessary to bring the system under model control is out of the scope of the present paper. It is though important to mention that these mechanisms are highly constrained by the meta-model assumptions. We point to [29] on this issue.

Also studying the integration between meta-model and open-point approaches, Bernstein [5] proposes a stepwise solution with four stages. The handling activities incrementally progress from totally unspecified to totally specified control. Contrary to this solution, which relies on AI techniques to support the incremental steps, our approach relies on user collaboration.

In summary, our main focus is on exceptions that can not be handled in an automatic way, i.e., can not be dealt by any of the solutions enlarging the original notion of expected exceptions (thus moving close to the unexpected limit). We assume that users should be able to flexibly move the system behavior from totally defined to unstructured processes, where open-point operations are carried out while meta-model assumptions are used to check system coherence. This will enable the adoption of the best strategy to the exceptional situation and facilitates the identification of user inserted inconsistencies. Finally, the system should also support the user to identify the necessary actions to bring the system back to a coherent state.

4 Exceptions Handling Framework

We distinguish three functions in exception handling [12; 31]:

- exception detection
- situation diagnosis
- exception recovery actions

Exception detection has been extensively studied in previous works [6; 10; 11; 31; 25]. Detection can be manual or automatic. A detailed description of the automatic detection techniques is out of the scope of this paper as it is focused on user perspective. We assume that an exception detection component is tightly integrated in the workflow engine, covering the most common situations, such as data, workflow, and temporal events, non-compliance events and application events. Later in this section we will discuss the integration of the detection component. We distinguish manual and automatic detection as they behave differently from user's perspective.

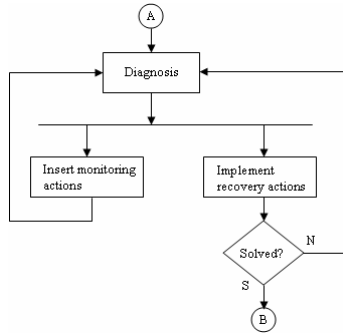


Fig. 1. Exception Handling Cycle

We will rather focus on the other two functions. In our framework we advocate an intertwined play between diagnosis and recovery until the exception is resolved. That is to say, the diagnosis is not considered to be complete on the first approach but rather, through an iterative process where different actors may collaboratively contribute to the solution. We should also stress that both the exceptional situation and perception of the situation may change along this iterative process, as new information is made available. As an example concerning a clinical process, a doctor may decide to insert a new task to collect information on the patient’s status and display this information to everyone involved. Also, if the clinical conditions of the patient change, the diagnosis regarding the exceptional situation may also change and some new objectives and tasks may arise.

After diagnosis, users carry out recovery actions. The open nature of the framework indicates that the recovery actions do not always run in the inner system context, and thus some linking mechanism is necessary to bring environmental information to the system. This issue will be addressed later in more detail.

Besides the detection, diagnosis and recovery functions, we identify a new function addressing monitoring actions necessary to control the progress of the whole exception handling process. These monitoring actions allow users to collect up to date information related to running instances and tasks. Considering again the open nature of the framework, these monitoring actions may also bring environmental information to the system, e.g., establishing a link to a web site with traffic information may help solving the exception of a truck being stuck on a traffic jam. In other cases monitoring actions may require more sophisticated links to external services, e.g., invoking an existing tool to calculate the minimum impact of a machine break down in a lot manufacturing facility. As shown in Figure 1, the exception handling cycle considers monitoring actions running in parallel with recovery actions.

Ellis and Keddara [16] state that a process change is itself a process that can be modeled. Therefore, like Sadik [31], we claim that it is better to cope with unexpected exceptions in work models using a work model. In our framework, the occurrence of an exception starts an exception handling workflow governed according to the exception handling cycle. The workflow is described in the Implementation section.

Figure 2 illustrates the three different levels of the exception handling framework. Dashed lines represent information flows whereas uninterrupted lines represent

control flows. We illustrate the WfMS at the bottom level, including the engine and all running tasks. The mid level represents the system components supporting the exception handling activities.

The top level concerns the users, focused on the two main exception handling functions: diagnosis and recovery/monitoring. The diagnosis and recovery/monitoring functions are carried out by the involved actors with support from the components available in the level below.

Our discussion of completeness in this context requires users not be restricted to the system. Therefore, the “External facilities” component shown in Figure 2 represents what is not under control of the exception handling workflow.

We differentiate two types of activities carried out by the “External facilities”: 1) information gathering, collaboration and decision making; and 2) recovery actions. The former group is related to external communication, coordination, collaboration and decision making tools (e.g., meetings, telephone conversations, or operations research techniques). The later group addresses the external recovery actions necessary to resolve the exception. It is our aim that, for any activity executed outside the scope of the exception handling workflow, some environmental information is inserted in the system for monitoring purposes.

The system interfaces are also identified in Figure 2. Interface 1 (Int. 1) interfaces with the WfMS, while interface 2 (Int. 2 and Int 2’) interfaces with “External facilities” and implements manual exception signaling. Interface 2 is split to maintain simplicity. Interface 1 is used to collect information about the WfMS status, to implement low level recovery actions (launch/suspend tasks, etc), and to signal automatically detected exceptions. Through the connection from interface 2 to “External facilities” environmental information about the operations carried outside the framework’s scope is collected.

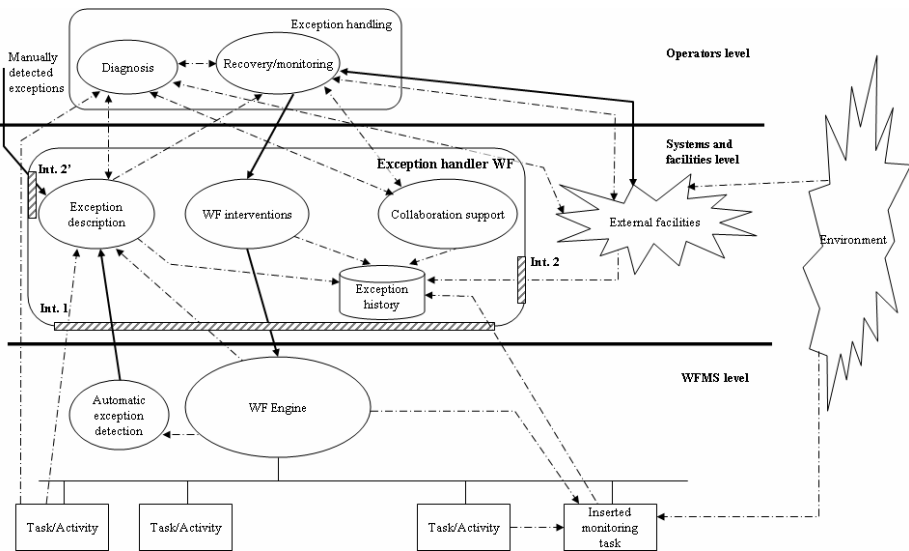


Fig. 2. Operational levels of the exception handling framework

The components *Exception Description*, *WF Interventions*, and *Collaboration support* will be explained later in the Implementation section. Exception detection is also represented in the figure. Manual detected exceptions are inserted by an operator and represented by the uninterrupted line connected to interface 2' on the top left side whereas automatic detection is implemented by the "Automatic exception detection" component located close to the engine. The component uses interface 1 to signal the events to the framework and is located close to the engine because it shares the organization workflow scope.

4.1 Diagnosis

The diagnosis process is mostly dependent on a detailed and accurate assessment of the exceptional event. Using previous classifications [8; 9; 30] and some new added characteristics, we classify exceptional situations using the orthogonal dimensions:

1. Scope – **process specific** when only a small set of instances is affected; or **cross specific** when a large set or different sets of instances are affected. At least one instance must always be associated to the exception;
2. Detection – **automatic** if the exception is automatically detected by the system, or **manually** if the exception is manually triggered;
3. Event type – **data events** related to violation of data rules; **temporal events** when a predefined timestamp occurs; **workflow events** identify special situations at the beginning or ending of tasks or processes, e.g., infinite loops; **external events** are notified by agents or applications external to the WfMS. The assessment of the event type is mandatory, because it directly impacts the handling phase;
4. Impact to organizational goals – **high**, if the particular situation has an important effect on the overall organizational goals; **medium**, if not critical; and **low**, when the organizational goals are not a concern;
5. Organizational impact – **employee**, when only a limited number of employees in the same department are affected by the exception; **group**, when more than one department is affected; and **organizational**, when the overall organization is affected. A responsible person must always be associated to the exception;
6. Difference to the organizational rules – **established exceptions** occur when rules exist in the organization to handle the event but the right ones cannot be found; **otherwise exceptions** occur when the organization has rules to handle the normal event but they do not apply completely to the particular case; and **true exceptions** occur when the organization has no rules to handle the event;
7. Complexity of the solution – **easy**, when the optimal solution can be easily obtained in an acceptable time; **hard**, when the optimal solution is not obtainable within an acceptable time. In this dimension, complexity is not defined as the overall complexity of the handling procedure, but rather an estimation of the possibility to define a cost function based on the available data. Whenever such a function exists, this dimension provides an estimate of the complexity degree to calculate the optimal solution;
8. Reaction time – **quick**, when the reaction to the exception must be as fast as possible; **relaxed**, when the reaction time is not too critical but some decisions must be taken within a time frame imposed by the instance(s); **long**, when the reaction time is not critical. This information is mandatory;

9. Time frame to achieve solution – **quick**, when the situation is expected to be resolved in few working units, normally minutes or hours; **relaxed**, when the time frame is more relaxed, although being a parameter to be taken into consideration, normally measured in working days; and **long** when time is not a critical issue.

Even though some estimates in these dimensions might be available when the event is detected, they can be redefined by users as more information is collected. The old values are kept to maintain an exception history. On the other hand, only the dimensions affected instances, responsible person (organizational impact), event type, and reaction time are mandatory. The user must only insert the most relevant information for the particular situation. This will release the user from inserting information not relevant to handle the concrete situation.

4.2 Recovery

We identified the following dimensions to classify recovery processes:

1. Objective of the intervention – further division presented below;
2. Required type of collaboration – synchronous and asynchronous;
3. Required collaboration level – one person solves the problem; several persons solve the situation in an asynchronous coordinated mode; and several persons solve the situation in a synchronous collaborative mode;
4. External monitoring requirements – there is either enough information to achieve the best solution or additional information must be collected from the environment;
5. Tools to determine the best solution – the solution does not require external decision aids, or there is a need of advanced support to achieve the best solution.

This information is associated to every exception raised. It must be emphasized that, likewise the information necessary to classify the situation, these values may change over time as more information about the exception is obtained. An exception history is kept in the system to be consulted by the involved users.

The *objective of the intervention* is further divided into [3; 10; 15; 27; 31]:

- Abort – further divided in: hard, compensate some tasks, and compensate all tasks;
- Decrease completion time to meet deadline;
- Recover from a system failure condition and replace the system in automatic mode;
- Recover from a task failure and place the system back in automatic mode;
- Recover to achieve the lowest penalty possible, i.e., the exception already impacted negatively on the organizational goals and the objective is to minimize the impact;
- Jump forward to a task in the work model;
- Repeat a previous task that was not executed in the desired way;
- Jump backwards in the work model and compensate some already executed tasks;
- Delay this task. This objective can be useful to release some resources necessary to increase the execution time of another process/instance;
- React to environmental changes. This normally requires a process change.

This classification affords linking the recovery process with a specific set of recovery tasks available at the system level. The *required type of collaboration* expresses how the collaboration support component will interconnect the persons

involved in the recovery process. We differentiate between two types of collaboration: synchronous and asynchronous. In synchronous collaboration all of the persons involved intervene at the same time, while in asynchronous collaboration the persons involved are not engaged in the process at the same time.

Concerning the *required collaboration level*, one has to be aware of concurrent changes made to work models. When ad hoc changes are applied in an asynchronous coordinated mode, every change is seen as an independent change and the resulting work model results from the composition of previous changes. Therefore, the structural and dynamic checks are made on the instance with respect to this new model. However, in the case of concurrent ad hoc changes carried in an asynchronous collaborative mode, the work of Rinderle [28] must be taken into consideration because actions carried out by different users without any agreement – asynchronously – may conflict (if they overlap on the same part of the model).

External monitoring requirements specify if environmental information is necessary to resolve the exception. The need to collect information within the system has already been identified by Basil et al [4]. In this approach we suggest that the recovery process may as well require collecting environmental information, from outside the system, e.g. generate an interface to display traffic information because a truck is stuck on a traffic jam.

The item *tools to calculate the best solution* identifies any additional tools necessary to calculate the best recovery solution. This affords linking the framework with external tools supporting the decision process.

5 Relationships Between Diagnosis and Recovery

Some relationships can be established empirically between the diagnosis of the situation and recovery strategies. Although some field trials should be carried out to validate the relations they seem very intuitive and easy to explain. These relations can be used as information to feed a decision support system that helps users on the selection of the most appropriate strategy given a concrete scenario.

We start by identifying the dimensions of the recovery strategy that do not depend on the classification; and the dimensions of the classification that do not have a clear impact on the recovery strategy. Then, the remaining relationships and respective consequences are explored.

On the side of recovery strategies, the *objective of the intervention* is determined by the external environment and does not depend of any characterization of the situation. It is related to the organizational goals regarding a particular event.

The dimensions *scope*, *detection*, and *event type* do not have any direct impact on the recovery strategy. *Detection* is important to know how the event was identified. Since similar events can be automatic or manually detected, this is of minor importance in determining the recovery strategy. The *scope* dimension determines the number of instances affected by the situation but does not affect the recovery strategy. The same strategy can be applied to all instances, or different strategies may be applied to different instances. Finally, the *event type* dimension does not have a clear relationship to the recovery strategy. For instance, a timeout does not imply that the *reaction time* should be quick. However, the *reaction time* dimension can be used to

increase the context awareness of a particular timed event. For instance, a timeout may be classified as critical in some situations and not critical in others.

Table 1 summarizes the identified relationships. The rows refer to diagnosis and the columns to recovery. The table shows two types of relationships: the first letter is the relation between the diagnosis and the need to adopt a particular recovery strategy (if the impact is high then there is a strong impact from the diagnosis row on the necessity to use the recovery strategy in the column); and the second letter shows the relation between the diagnosis and the particular type of recovery strategy within the class (if the impact is high there is a strong relation between the diagnosis row and the type of recovery strategy within the column). This means that in a particular situation, even though the diagnosis might not indicate the necessity to use a particular recovery strategy, if the users decide to use it then the chosen recovery strategy might depend on the diagnosis. E.g., the time dimension on the diagnosis does not affect the decision to use a collaboration type (L on the first letter), but if the users adopt a collaboration type then the time dimension has an impact on the type of collaboration type to choose (H on the second letter).

Table 1. Relation between event classification and handling strategies

	Collaboration type	Collaboration level	Ext. monitoring	Tools to best solution
Time	L/H	L/H	L/H	L/H
Goals impact	H/L	L/L	M/H	M/H
Organizational impact	H/L	H/L	L/L	L/L
Difference to organizational rules	H/L	L/L	M/H	M/H
Complexity	L/L	L/L	M/H	H/H

Legend – L – low; M – medium; H – high

As the time associated with the exception is usually an independent factor, we start by discussing time relations. Also, it is important to note that time restrictions have a strong impact on the way people deal with problems. We have defined two dimensions related with time: *reaction time* and *time frame to achieve solution*. The former is important to specify how the person responsible should be informed about an exception. Then, upon starting the diagnosis phase, that person can define the *time frame to achieve solution* in a different way than reaction time: e.g., some contention action was implemented but the final solution can be implemented in a more relaxed time frame. Therefore, once the parameter *time frame to achieve solution* is defined, it will have a stronger effect on the decision process than the *reaction time*. The *time* row in the table reflects this effect and is obtained from these two dimensions.

One should not expect any impact from the *time* dimension on the usage of any *collaboration type*, i.e., for any value of *time* nothing can be concluded about collaboration among users (L on the first letter). However, if the *time* is *quick* and the user wants to use collaboration the synchronous type should be the choice. On the other hand, if the *time* is not *quick*, one can expect that an asynchronous *collaboration type* may be the choice (H on the second letter). This shows low impact from the *time* dimension on whether any *collaboration type* should be used, but a strong relationship between the *time* dimension and the *collaboration type* to use. The relation is therefore “L/H”.

The relationship between *time* and *cooperation level* is similar, since *time* does not affect the usage of any *collaboration level*, but if one is to be used then asynchronous *cooperation level* should be the choice on situations that require fast responses as autonomous actors react faster. Synchronicity increases the delay on recovery actions.

Similar relationships are found between *time* and *monitoring*, and between *time* and *tools*. The need for monitoring environmental information and using external tools to calculate the best solution depends on a particular case and not on time; but if they are needed the particular ones to choose will be restricted by the time factor.

The *organizational goals* dimension has implications on the *required collaboration type*, since events with high impact should involve the user(s) responsible for the task(s) and their supervisors, but the type of collaboration is not imposed. The *required collaboration level* is not affected in any sense by the *organizational impact* as there is not an indication to use any of the identified collaboration levels due to the type of organizational impact. Even further, the collaboration level to choose is not influenced by the goals impact. There are situations with high (low) organizational goals impact that can be solved with only one user implementing recovery actions and others where more than one person is necessary.

Regarding *monitoring* and *tools*, even though the necessity to use them depends on the particular context, the usage of these mechanisms should deserve more attention on situations with high impact on the organizational goals. The value in the table M/H reflects these considerations where the M is used to signal that the concrete scenario has a higher relation, but the impact of the organizational goals dimension should also be taken into consideration. On the other hand, if the impact on the organization goals is high special care should be placed on the monitoring actions and on the tools to use.

The *organizational impact* has a strong relationship with the *collaboration level* and *type*. However, there is a small relationship with the type within these dimensions. The relationship with *monitoring requirements* and *tools* is also low.

On the *difference to organizational rules* dimension, it is expected that more users are involved when there are rules but the right ones can not be found, or when there are no rules in the organization to handle the event. The involvement of more users is important to find the right rules or define new ones. There is a high relation with the selection of a *collaboration type*, but no restriction is imposed. The *monitoring requirements* and the usage of *tools* are expected to increase in situations that differ from normal procedures, and the type of adopted mechanism will also depend on the degree of difference. No relation is established with the *collaboration type*.

Finally, for situations where it is possible to use a tool to calculate the best solution and the *complexity* is high, the primer relevance is made on the *external monitoring requirements*. As in previous situations, it is expected that *external monitoring requirements* are mainly influenced by the concrete situation, so we expect a medium relation. Nevertheless, if one is to be chosen, special care must be taken about the right one. In these situations, as it is easily justifiable, there is high relation between usage and type of *tools*. No relation is established with the *collaboration type* and *collaboration level*, as it is expected that the solution, even though complex, can be calculated by only one actor.

6 Implementation

Figure 3 represents the proposed exception handling work model, an extended version of our previous work [25], where two new branches were inserted addressing external monitoring actions and collaboration mechanisms; and some minor changes were done to the *collaboration* component. Further details regarding *association of instances* and *edit exception classification* can be consulted in the cited paper.

In the present work we are not concerned with the specific implementation details, in particular about the WfMS engine or model language used. Our main focus is how exception handling is supported by the proposed framework.

The *collaboration support* component supports users specifically collaborating within the scope of an exceptional event. The tasks implemented by the component (see figure 3) enable the definition of a new responsible, involve more actors, and implement the collaboration mechanism. The *collaborate* task in the model can be synchronous or asynchronous where at any instant the users can choose the type to use. When asynchronous collaboration is being used any involved actor can send a message to any or all of the colleagues using a developed interface. The company email system is used to inform the user that s/he should check the workflow system. Synchronous collaboration support depends on the application domain and environment as it can be implemented by a phone conversation, chat over a computer or even face-to-face conversation. In both cases the exchanged information is stored using the *exception history* component. If it is not possible to store the conversation the users should insert the conclusions and any special comment. Further developments of collaborative components will be subject to future research.

The *wf interventions* component is implemented using two branches: implement recovery actions; and insert monitoring tasks. Recovery actions are a set of atomic interventions that can be carried out on the specific workflow engine, e.g., suspend an instance or insert a task in the original model.

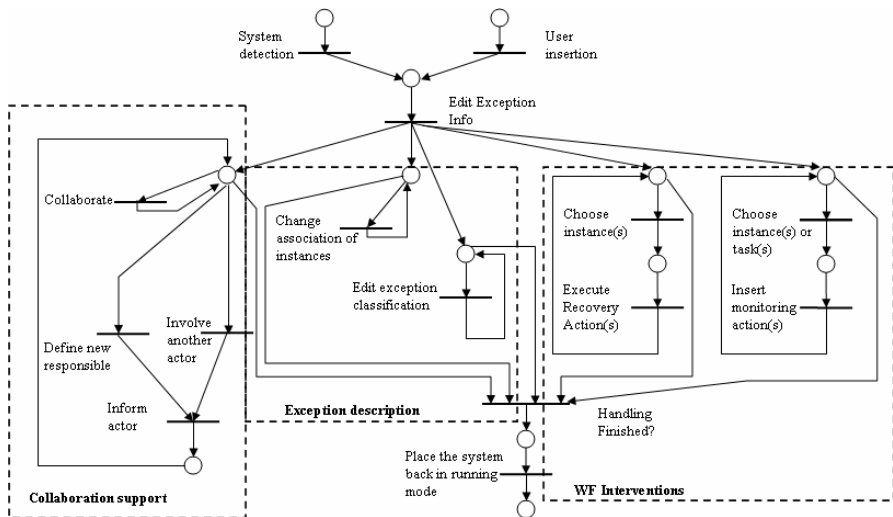


Fig. 3. Exception handling workflow

The *monitoring* branch affords users to insert monitoring tasks that store exception relevant information in *exception history*. Since this information is chronologically stored, the user may monitor the system evolution. Even further, if external environmental information or tools are available, the user may store links, e.g., a link to traffic cameras available through the internet, which may be used to diagnose the situation on a truck that is stuck on a traffic jam.

The detection of a new exception situation is represented in the figure by the first two parallel branches on the top of the figure representing system and manual detection. These tasks will insert all the mandatory information. As mentioned before, a responsible person must always be associated to the exception and will execute the next task “Edit exception info” where the most important information related to the exception is inserted. This task assures that the responsible person is informed on the situation and initiates the exception handling procedure.

The person responsible may then execute any of the actions specified in the six branches of the exception handling workflow. Let us assume that a user decides to involve three more actors in the exceptional event. Then, using the collaboration component (email or chat) s/he informs the other actors that there is an exception to be resolved. The diagnosis phase proceeds using the collaboration component, so the other actors share their views of the present situation. Finally, they decide to insert two monitoring actions in the work model, and two of them will be responsible for the follow up. Once any special event regarding these monitoring actions is triggered, the group is informed and the recovery action may proceed using the execute recovery action branch. The process is repeated until the exceptional situation is overcome.

7 Example

During an operation, the patient’s heart monitoring machine breaks down. As soon as the machine breaks, one nurse connects the patient to an available, but less reliable machine, and continues monitoring the patient’s data. Meanwhile, she manually instantiates the exception recovery workflow and assigns her name as the person responsible. The time frame to reach a solution is set as *quick*, a brief description of the event is inserted and she joins two other persons to the exception handling procedure: one person will try to fix the machine (maintenance operator) and a technical assistant will try to find another compatible machine for backup. As the time frame is set as quick, both departments will be informed by a flashing light and a buzz sound on their coordination room. The respective department coordinators will look at their computers and find this exceptional situation to handle. They will initiate their own recovery tasks and assign them to an employee in their department (using a form requiring the name, the urgency, and a manually inserted description of the task).

As the maintenance employee will go inside the operations room, face-to-face collaboration with the nurse is assured. However, if the situation in the operations room changes, the technical assistant should be informed. A collaborative reporting task is generated where both the technical assistant coordinator and the nurse can read and write information. Both of them write any status changes. If the maintenance operator fixes the machine, the nurse writes this information and the technical coordinator informs the employee. If, on the other hand, a compatible machine is

found, the technical assistant informs his coordinator, who writes down the predicted available time. If the solution is approved by the nurse the maintenance operator will start to prepare the replacement. When the technical assistant arrives with the machine, the maintenance operator replaces it and the exception is closed.

However, one can imagine a dramatic case where all the machines are allocated to patients and someone responsible has to decide whether they can be removed or not. The technical assistant coordinator informs the nurse and they decide to involve the doctor responsible for the area. They insert a new task in the work model supplying information to the doctor with a high priority: again, a signaling system should be available so the doctor is quickly informed. The doctor analyses the task description and decides, given the patients' situations, if she has enough information to make a decision. If not, she initiates a chat session with the nurse (new task). Let us assume the doctor decides to remove a machine from one of her patients but only after a new nurse is assigned to monitor this patient. She therefore affects another instance to the exception (the instance associated to this new patient) and inserts three new tasks: find a new nurse, remove the machine, and replace the machine once the operation is finished (note that the first and second tasks are in parallel to the original operation sequence, while the last is placed after the operation is finished). The first is assigned to the nurse coordinator and the other two to the technical assistant coordinator. Once the new nurse is monitoring the patient, the first task is completed and the technical assistant can take the machine to the operations room. After the operation is finished, the task to move the machine back to the patient is ready to be executed. Only after the machine is replaced in its original position the exception is completed.

8 Conclusions

The major concern addressed by our framework is the support to unexpected exceptions, defined as situations that can not be handled in an automatic way because the system does not have information about them, nor can infer such information from previous analogous situations. Under these circumstances, collaborative user involvement is crucial to determine the most appropriate solution.

Our analysis highlighted a fundamental system requirement: maintain task execution under model guidance during normal operation and change to unstructured behavior when an unexpected exception occurs, supporting users giving the control back to model guidance after the exceptional situation is overcome.

We developed an exception handling process to support this behavior, comprising collaborative diagnosis, recovery and monitoring tasks. Furthermore, we analyzed in detail the characteristics and relationships between the diagnosis and recovery tasks. The diagnosis task is based on a new classification of unexpected exceptions proposed in this paper. Several dimensions characterizing the recovery tasks, as well as relationships with the classification of unexpected exceptions are proposed as well. The resulting framework helps users collaborating to devise appropriate exception handling strategies for unexpected situations.

Bibliography

1. Abbott, K.R., and Sarin, S.K., 1994. Experiences with workflow management: issues for the next generation. Proc. of the 1994 ACM Conference on CSCW. Chapel Hill, North Carolina, United States, pp. 113-120.
2. Agostini, A., and De Michelis, G., 2000. A light workflow management system using simple process models. *CSCW*, 9(3): 335-363.
3. Agostini, A., De Michelis, G., and Loregian, M., 2003. Undo in Workflow Management Systems. *BPM 2003*. Springer-Verlag, Netherlands, pp. 321-335.
4. Bassil, S., Rinderle, S., Keller, R., Kropf, P., and Reichert, M., 2005. Preserving the Context of Interrupted Business Process Activities. 7th ICEIS 2005. USA.
5. Bernstein, A., 2000. How can cooperative work tools support dynamic group process? bridging the specificity frontier. *CSCW '00: Proceedings of the 2000 ACM Conference on CSCW*. ACM Press, Philadelphia, pp. 279-288.
6. Casati, F., 1998. Models, Semantics, and Formal Methods for the Design of Workflows and their Exceptions. PhD Thesis, Politecnico di Milano.
7. Casati, F., Ceri, S., Pernici, B., and Pozzi, G., 1996. Workflow Evolution. *Data and Knowledge Engineering*, 24(3): 211-238.
8. Casati, F., and Pozzi, G., 1999. Modelling exceptional behaviors in commercial workflow management systems. Proc. IFCIS, International Conference on CoopIS, CoopIS '99. IEEE International, Edinburgh, UK, pp. 127-138.
9. Chiu, D.K., 2000. Exception Handling in an Object-oriented Workflow Management System. PhD Thesis, Hong Kong Univ. of Science and Technology.
10. Chiu, D.K., Li, Q., and Karlapalem, K., 2001. WEB Interface-Driven Cooperative Exception Handling in ADOME Workflow Management System. *Information Systems*, 26(2): 93-120. Elsevier Publishers.
11. Dayal, U., Hsu, M., and Ladin, R., 1990. Organizing Long-Running Activities with Triggers and Transactions. *SIGMOD'90*. NJ, USA.
12. Dellarocas, C., and Klein, M., 1998. A Knowledge-based approach for handling exceptions in business processes. *WITS'98*. Helsinki, Finland.
13. Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., and Zbyslaw, A., 1996. Freeflow: mediating between representation and action in workflow systems. Proc. of the 1996 ACM Conference on CSCW. ACM Press, New York.
14. Eder, J., and Liebhart, W., 1995. The Workflow Activity Model WAMO. *Int. Conf. on Cooperative Information Systems*. Vienna, Austria.
15. Eder, J., and Liebhart, W., 1996. Workflow Recovery. 1st IFCIS Intl. Conf. on Cooperative Information Systems (CoopIS'96). IEEE, Belgium, pp. 124 - 134.
16. Ellis, C., and Keddara, K., 2000. A Workflow Change is a Workflow. In: W.D. van der Aalst, J. Oberweis (Editor), *Business Process Management: Models, Techniques, and Empirical Studies*. Springer-Verlag, pp. 201-217.
17. Ellis, C., Keddara, K., and Rozenberg, G., 1995. Dynamic change within workflow systems. *Organizational Computing Systems*., Milpitas, CA, USA.
18. Ellis, C., and Nutt, G.J., 1993. Modeling and enactment of workflow systems. *Application and Theory of Petri Nets*. Springer-Verlag, Illinois, USA, pp. 1-16.
19. Faustmann, G., 2000. Configuration for Adaptation - A Human-centered Approach to Flexible Workflow Enactment. *CSCW*, 9(3): 413-434.
20. Han, Y., Sheth, A.P., and Bussler, C., 1998. A Taxonomy of Adaptive Workflow Management. *Conf. on CSCW - Workshop - Towards Adaptive Workflow Systems*. Seattle, WA, USA.

21. Hayes, N., 2000. Work-arounds and Boundary Crossing in a High Tech Optronics Company: The Role of Co-operative Workflow Technologies. *CSCW*, 9(3): 435-455.
22. Hwang, S.Y., Ho, S.F., and Tang, J., 1999. Mining Exception Instances to Facilitate Workflow Exception Handling. 6th Int. Conf. on Database Systems for Advanced Applications. Hsinchu, Taiwan.
23. Jorgensen, H.D., 2001. Interaction as Framework for Flexible Workflow Modelling. Group '01. ACM Press, Boulder, Colorado, USA.
24. Luo, Z., 2001. Knowledge sharing, Coordinated Exception Handling, and Intelligent Problem Solving for Cross-Organizational Business Processes. PhD Thesis, Dep. of Computer Sciences, University of Georgia.
25. Mourão, H.R., and Antunes, P., 2004. Exception Handling Through a Workflow. *CoopIS 2004*. Springer-Verlag, Agia Napa, Cyprus.
26. Reichert, M., and Dadam, P., 1998. ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10(2): 93-129.
27. Reichert, M., Dadam, P., and Bauer, T., 2003. Dealing with Forward and Backward Jumps in Workflow Management Systems. *Software and Systems Modeling*, 2(1): 37-58. Springer-Verlag.
28. Rinderle, S., 2004. Schema Evolution in Process Management Systems. PhD Thesis, University of Ulm.
29. Rinderle, S., Reichert, M., and Dadam, P., 2003. Evaluation of Correctness Criteria for Dynamic Workflow Changes. *BPM '03*. Netherlands, pp. 41-57.
30. Saastamoinen, H., 1995. On the Handling of Exceptions in Information Systems. PhD Thesis, University of Jyväskylä.
31. Sadiq, S.W., 2000. On Capturing Exceptions in Workflow Process Models. *Proc. of the 4th Int. Conference on Business Information Systems*. Poznan, Poland.
32. Schmidt, K., 1997. Of maps and scripts - the status of formal constructs in cooperative work. *GROUP '97: Proc. of the Int. ACM SIGGROUP Conf. on Supporting Group Work: The Integration Challenge*. United States, pp. 138-147.
33. Sheth, A.P., Georgakopoulos, D., Joosten, S.M., et al, 1996. Report from the NSF workshop on workflow and process automation in information systems. *ACM SIGMOD Record*, 25(4): 55-67. ACM Press.
34. Strong, D.M., and Miller, S.M., 1995. Exceptions and Exception Handling in Computerized Information Systems. *ACM Trans. on Information Systems*, 13(2).
35. Suchman, L.A., 1987. *Plans and Situated Actions*. MIT Press.
36. van der Aalst, W., and Basten, T., 2002. Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science*, 270(1).
37. van der Aalst, W., Basten, T., Verbeek, H., Verkoulen, P., and Voorhoeve, M., 1999. Adaptive Workflow: On the interplay between flexibility and support. *Proceedings of the 1st ICEIS*. Setúbal, Portugal, pp. 353-360.
38. Weber, B., Wild, W., and Breu, R., 2004. CBRFlow: Enabling Adaptive Workflow Management through Conversational Case-Based Reasoning. *European Conf. on Case-Based Reasoning (ECCBR'04)*. Madrid, Spain.

A Workflow Mining Method Through Model Rewriting

Jacques Wainer¹, Kwanghoon Kim², and Clarence A. Ellis³

¹ Institute of Computing State University of Campinas Campinas,
13084-971, SP, Brazil
wainer@ic.unicamp.br

² Collaboration Technology Research Lab., Department of Computer Science,
Kyonggi University, San 94-6 Yui-dong Youngtong-gu Suwon-si Kyonggi-do,
442-760, South Korea
kwang@kyonggi.ac.kr

³ Collaboration Technology Research Group, Department of Computer Science,
University of Colorado at Boulder, Campus Box 430,
Boulder, CO, 80309-0430, USA
skip@cs.colorado.edu

Abstract. This work presents a workflow process mining method that is, at least, as powerful as many others presented in the literature, as measured by the examples presented in the literature. The method is based on a grammar of rewriting expressions, by which a model is adapted to include a new execution trace. We also discuss the intrinsic limits of the mining process, which we believe has not been a topic clearly stated and discussed in the published research.

1 Introduction

In recent times, workflow (business process) and its related technologies have been constantly deployed and so its becoming gradually a hot issue in the IT arena. This atmosphere booming workflows and business processes modeling and re-engineering is becoming a catalyst for triggering emergence of the concept of workflow mining that rediscovers workflows from workflow logs collected at runtime in order to support workflow design and analysis for redesigning and re-engineering workflows and business processes.

Formally, the process mining problem is given a set of logs of workflow execution, called *traces*, reconstruct the workflow model that generated such traces. We are concerned in this paper with exact mining, that is, all traces must be explained or generated by the model. Variations of the process mining problem allows only for some of the traces to be explained by the model, or that not all parts of the traces must be explained.

This paper discusses a model for process mining based on model rewriting. We define a set of grammar rules that modifies a process description (a model) to incorporate a new trace.

Section 2 discusses some of the previous results in the field. Section 3 discusses the intrinsic limits of the mining process, or in other words, we show that the problem is ill-defined and there are an infinite number of solutions to the same problem. We feel that researchers in the field do not have this result or its consequences clear. Section 4

discusses our model rewriting method and its characteristics. Section 5 shows that our model can mine cases that were discussed in the literature as examples of different algorithms for mining, and thus, in a very weak sense, our model subsumes some of these algorithms. Section 6 lists briefly some conclusions.

2 Related Works and Background

So far, there have been several workflow mining related researches and developments in the workflow literature. Some of them have proposed the algorithms [1,2,4,5,6,8,9] for workflow mining functionality, and others have developed the workflow mining systems and tools [3,7,10]. Particularly, as the first industrial application of workflow mining, J. Herbst and D. Karagiannis in [3] presented the most important results of their experimental evaluation and experiences of the InWoLvE workflow mining system. However, almost all of the contribution are still focusing on the development of the basic functionality of workflow mining techniques. W.M.P. van der Aalst's research group, through the papers [1] and [2], proposed the fundamental definitions and the use of workflow mining to support the design of workflows, and described the most challenging problems and some of the workflow mining approaches and algorithms. The problems, which they stated in [2], are the short-loops (one-length loop and two-length loop) problems, invisible task, duplicate task, implicit places, non-free choice, and the synchronization of OR-join place problems. The starting point of this paper is a critical observation and remark on those problems and their solutions proposed in [1] as followings:

- They are looking for the solutions for rediscovering an exact workflow model with the original one from workflow logs.
- The workflow models are based upon the Petri Net modeling methodology that is a quite general-purpose modeling methodology. So, it may make the workflow models more complex and make their solutions much more theoretic, which mean that their problems and solutions may not reasonably reflect the practical workflow models on the real-world's business situation.
- The problems stated in [2] may be not the real workflow mining problems but the phantom problems caused by the Petri Net model, itself.

The work of Schimm [4] has some similarities to ours, in the sense that it uses balances workflow models (what is called in [2] as block-oriented models) and is based on transformation rules on the models. Pinter and Golani [5] use time information of both the start and end of the activities to derive traces that already include information when two activities are parallel. Cook et al. [6] uses a more statistical/probabilistic approach to mining processes, the mined process only probabilistically models the traces. [7,8] are works that do not attempt to mine exact workflow models from traces, but to capture interesting temporal patterns among the activities - more in sync with the goals of data mining itself, which aims at discovering interesting but not necessarily true patterns on the data.

3 The Limits of Mining Processes

A topic that we believe has not been properly discussed in the previous research work is that process mining is an ill-defined problem. There is no single correct or best solution to the problem of given a set of executions traces determine a process model that could have generated them.

There are very large and possibly infinite number of workflow models that satisfy any set of workflow traces. For example, given the following set of traces $T = \{abcde, acbe, abce\}$ let us discuss some of these models.

3.1 Most Generic Models (MGM)

There is one model which we will call the *simple MGM* (sMGM) which is composed of all activities in T in a or-split with a loop back (figure 1). There are also an infinite set of MGM models which extent the simple MGM with any number of *fake activities* (fake activities are activities names not present in T), as illustrated by figure 1.

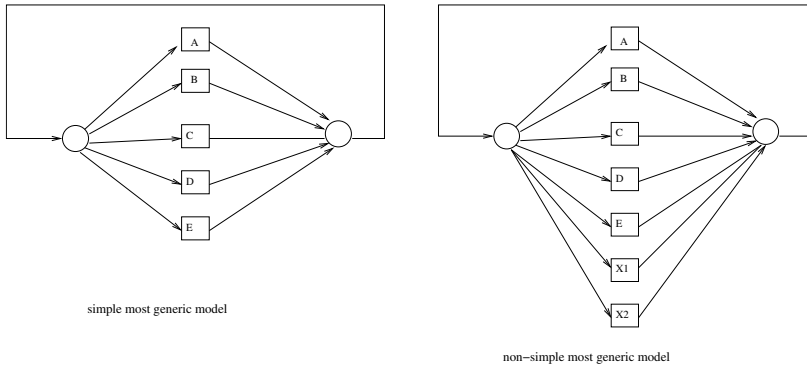


Fig. 1. Most generic models

The MGM correctly and completely model all traces in T and furthermore they model all future execution traces too.

MGM with constraints. One can rightfully claim that MGMs are too generic - if all is allowable, why have a workflow at all? But even if there are constraints that one wishes to enforce, for example, that b cannot happen before a , it is likely that one can create MGMs that satisfy the constraint. For example, a simple MGM that satisfies the constraint that b cannot happen before a is shown in figure 2. But constrained MGMs are not a standard in process mining, because they need a specification of the constraints - no process mining problem specification we are aware of includes constraints. The constraints could be implicitly defined by using *negative traces*, that is traces of executions that should not happen, but again this would make the process mining problem not only much more complex but also unrealistic - one cannot generate a reasonable set of negative traces.

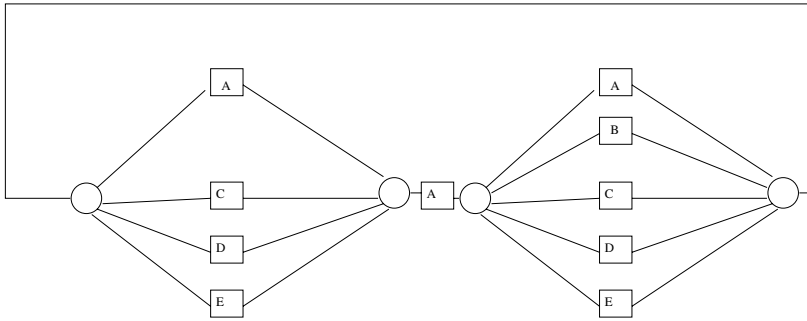


Fig. 2. Constrained MGM

3.2 Most Specific Model (MSM)

In parallel to MGM there is one model that we will call the simple most specific model (sMSM) and there are an infinite number of MSM that extend the sMSM by adding fake activities (or more specific fake traces). The sMSM is just a or-join of all traces in T as show in figure 3. The sMSM will satisfy all past traces, and only them.

3.3 Discussion on the Limits of the Mining Process

This section has demonstrated that there are many and possibly infinite process models (if fake activities are allowed) that could be mined from a set of traces. And some of these models are very easy to compute. This places an epistemic problem on the whole line of process mining research - how to evaluate the different algorithms and techniques proposed.

More specifically, each algorithm will necessary embody an explicit or implicit bias toward particular models among all the possible ones. In other words, mining algorithms must necessary pick one (or more) *reasonable* models among the infinitely many that are possible. But the literature does not discuss what are the criteria that define *reasonable* models.

There seems to be some common sense rules for reasonable models which include, for example no fake activities. Also the MSM seems too “unnatural” - no business will

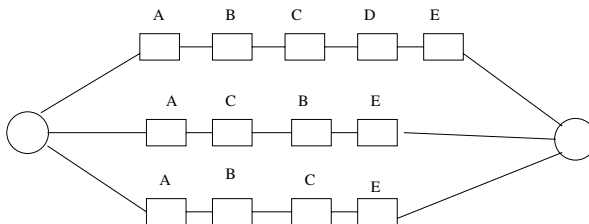


Fig. 3. Simple most specific model

run on such processes. But there is no more deeper discussion in the literature about the rules of reasonable workflow models. We feel that this discussion will be very beneficial to the whole process mining line of research. Unfortunately we are not ready in this paper to advance this issue.

4 A Model Rewrite Method

The central ideas of our approach are:

- the method modifies an existing model to incorporate a new trace. Thus, it is an incremental algorithm: after seeing the first trace the algorithm generate the trivial sMSM for that trace and upon seeing the second trace, adapts the existing model to incorporate the new trace.
- the method is a series of rewrite rules that transform the model plus trace into a new model.

We will follow the following notation

- $\alpha\beta$ represents the sequence of β after α .
- $\alpha + \beta$ represents the or-split/join of α and β
- $\alpha\|\beta$ represents the and-split/join of α and β
- $\alpha\top\beta$ is a loop in which α is in the main branch, and β is in the alternative path
- if M is a model and α is a trace, then $M \oplus \alpha$ is the amalgamation of M and the new trace, but also the or-split/join of the two expressions.
- if α is a model (expression) then α' is a sequence of activities that can be generated from α

Thus, the rewrite rules are expressions of the form $M \oplus \beta \Rightarrow M'$ where M and M' are models (expressions) and β is a trace, that is a sequence of activity names.

4.1 Rewrite Rules

Following the tradition of deduction rules, we classify the rewrite rules in two categories, the *structural* rules and the *introduction* rules. The structural rules distribute the \oplus operator where as introduction rules introduce a workflow operator ($+$, $\|\$, \top) in the appropriate conditions.

The structural rules are:

- S1 - $\alpha \oplus \alpha' \Rightarrow \alpha$ that is, if the new trace can be generated from the model, there is no need to change the model
- S2 - $\alpha\beta \oplus \gamma\delta \Rightarrow (\alpha \oplus \gamma)(\beta \oplus \delta)$

The introduction rules are listed below. The name of the rule is of the form *op*-I where *op* is the operator being introduced.

- or-I $\alpha \oplus \beta \Rightarrow (\alpha + \beta)$,
- and-I $\alpha\beta \oplus \beta'\alpha' \Rightarrow \alpha\|\beta$
- loop-I $\alpha \oplus \alpha'\beta\alpha' \Rightarrow \alpha\top\beta$
- and2-I $(\alpha\|\beta)\gamma\delta \oplus \alpha'\delta' \Rightarrow (\alpha\|(\beta\gamma))\delta$

Furthermore we assume some obvious equivalence rules among model, such as $a + b \equiv b + a$ and $a||b \equiv b||a$, among others.

The rule and2-I is the only case of a *conditional rule*: its application must be checked with previous traces - it may be the case that a previous trace does not agree with the transformation, and thus, it cannot be applied.

4.2 Example

Given the traces $T = \{abcde, acbe, abce\}$ discussed above, the process model rewrite technique would generate the following models:

- $M_1 = abcde$
- $M_2 = abcde \oplus acbe \xrightarrow{S2+S2} a(bcd \oplus cb)e \xrightarrow{S2+andI} a(b||c)(d \oplus \epsilon)e \xrightarrow{orI} a(b||c)(d + \epsilon)e$
- $M_3 = a(b||c)(d + \epsilon)e \oplus abce \xrightarrow{S2} a[(b||c) \oplus bc][(d + \epsilon) \oplus \epsilon]e \xrightarrow{S1+S1} a(b||c)(d + \epsilon)e$

The symbol $\xrightarrow{S2+S2}$ denotes derivation and the superscript are the rules that should be applied.

4.3 Characteristics of the Rewrite Approach

At its current state, our rewrite proposal has three characteristics. First the result *depends on the order* of presentation of traces. If the same set of examples were presented in different order the result can be vastly different models, even though the models correctly generate all traces.

The second characteristic is that it only generates balanced workflow models. Balanced models are workflows models in which or-split/joints are nested within and-split/joint and vice versa.

The more serious limitation of the method is that it is not yet an algorithm. The rewrite rules are non-deterministic and at the moment we do not have a algorithm that will choose which rule to apply and how to apply it.

5 Examples from the Literature

We can show that the rewrite method can at least correctly generate some examples that are discussed in the literature. Thus, at least regarding such examples, our method is equivalent to the one proposed by others.

Example 1 in [2]. The trace T is the following: $T = \{abcd, acbd, abcd, acbd, ef\}$

Given the traces above, the models are:

- $M_1 = abcd$
- $M_2 = abcd \oplus acbd \rightsquigarrow a(b||c)d$
- $M_3 = a(b||c)d \oplus abcd \rightsquigarrow a(b||c)d$
- $M_4 = a(b||c)d \oplus acbd \rightsquigarrow a(b||c)d$
- $M_5 = a(b||c)d \oplus ef \rightsquigarrow (a(b||c)d) + (ef)$

which is exactly the result discussed in [2].

Example 4.2 in [2]. This example is discussed in [2] as the duplicate task. $T = \{ abcd, acbd, abcd, acbd, bf \}$

- $M_1 = abcd$
- $M_2 = abcd \oplus acbd \rightsquigarrow a(b||c)d$
- $M_3 = a(b||c)d \oplus abcd \rightsquigarrow a(b||c)d$
- $M_4 = a(b||c)d \oplus acbd \rightsquigarrow a(b||c)d$
- $M_5 = a(b||c)d \oplus bf \rightsquigarrow (a(b||c)d) + (bf)$

M_5 is the model [2] as the generated model. But here our method shows one of its limitation. Other rewrite rules could be applied at the stage $a(b||c)d \oplus bf$, resulting the complex model $M_{5'} = (a + \epsilon)[(b(\epsilon + f))|(cd + \epsilon)]$. Clearly M_5 is more “reasonable” than $M_{5'}$, but what exactly makes it more reasonable? And how to incorporate this into the selection of rules to be applied so that the resulting model is likely to be reasonable?

As we discussed above, the other challenging problems listed on [2] are more specific to the Petri-net representation of processes that the authors adopt than generic mining problems. In particular examples 4.1 and 4.3 in [2] (hidden tasks and non-choice free) are solved by our method. For example 4.4 (short loops) our method will not generate short sequences of the same task, it will always prefer a loop construction.

Example 5 in [4]. The set of traces is $T = \{ abcdefh, acbdefh, abcdefh, abcdegh, abcdegh, acbdegh, acbdefh, acbdegh \}$. We make a simplification in the set T . In [4], by using both the start and end times of an activity, the author concludes that in the first example, b and c are parallel activities. We represented the traces with parallel execution as two different traces.

- $M_1 = abcdefh$
- $M_2 = abcdefh \oplus acbdefh = a(b||c)defh$
- $M_3 = a(b||c)defh \oplus abcdefh = a(b||c)defh$
- $M_4 = a(b||c)defh \oplus abcdegh = a(b||c)de(f + g)h$
- $M_5 = a(b||c)de(f + g)h \oplus abcdegh = a(b||c)de(f + g)h$
- $M_6 = a(b||c)de(f + g)h \oplus acbdegh = a(b||c)de(f + g)h$
- $M_7 = a(b||c)de(f + g)h \oplus acbdefh = a(b||c)de(f + g)h$
- $M_8 = a(b||c)de(f + g)h \oplus acbdegh = a(b||c)de(f + g)h$

which is the model proposed by the author.

6 Conclusions

This paper presented an ongoing work of defining a process mining method based on model rewrite rules. At the current stage of the research we can only claim that our model correctly deals with examples proposed by other authors as illustration of their own methods and algorithms. So, in some weak sense, it seems that this method is as strong as other algorithms proposed in the literature.

The fact that our method is incremental is, we believe, of particular importance. Our rewrite rules can be used, for example, to incorporate an exceptional execution trace into an existing workflow model. This allows for an interesting methodology to

implement a workflow into an organization - develop a simple workflow model for the most normal processes and use the rewrite rules to adapt the model to the traces of more exceptional cases.

This work also showed that the problem of process mining is ill-defined and only by defining what are “reasonable” models among the infinite number of models that are possible solutions to the mining problem will the field advance in a significant way.

We are currently working in many directions, including a definition of “reasonable” models, and defining a heuristics to select the rewrite rules to be applied so that there the generated models are reasonable.

References

1. W. M. P. van der Aalst, B. F. van Dongena; J. Herbst, L. Marustera, G. Schimm and A. J. M. M. Weijters: Workflow mining: A survey of issues and approaches. *Journal of Data & Knowledge Engineering*, Vol. 47, Issue 2, pp. 237-267, Elsevier, November 2003
2. W. M. P. van der Aalst and A. J. M. M. Weijters: Process mining: a research agenda. *Journal of Computers in Industry*, Vol. 53, Issue 3, Elsevier, April 2004
3. Joachim Herbst, and Dimitris Karagiannis: Workflow mining with InWoLvE. *Journal of Computers in Industry*, Vol. 53, Issue 3, Elsevier, April 2004
4. Guido Schimm: Mining exact models of concurrent workflows. *Journal of Computers in Industry*, Vol. 53, Issue 3, Elsevier, April 2004
5. Shlomit S. Pinter, and Mati Golani: Discovering workflow models from activities’ lifespans. *Journal of Computers in Industry*, Vol. 53, Issue 3, Elsevier, April 2004
6. Jonathan E. Cook, Zhidian Du, Chongbing Liu and Alexander L. Wolf: Discovering models of behavior for concurrent workflows. *Journal of Computers in Industry*, Vol. 53, Issue 3, Elsevier, April 2004
7. Daniela Grigori, Fabio Casati, Malu Castellanos, Umeshwar Dayal, Mehmet Sayal and Ming-Chien Shan: Business Process Intelligence. *Journal of Computers in Industry*, Vol. 53, Issue 3, Elsevier, April 2004
8. San-Yih Hwang, Chih-Ping Wei and Wan-Shiou Yang: Discovery of temporal patterns from process instances. *Journal of Computers in Industry*, Vol. 53, Issue 3, Elsevier, April 2004
9. Kwanghoon Kim and Clarence A. Ellis: Workflow Reduction for Reachable-path Rediscovery. Proceedings of the ICDM2003 WORKSHOP: Foundations and New Directions in Data Mining, Melbourne, Florida, USA, November, 2003
10. Minjae Park and Kwanghoon Kim: An Efficient Workcase Classification Method and Tool in Workflow Mining. Proceedings of the DMIN2005: The International Conference on Data Mining, Monte Carlo Resort, Las Vegas, Nevada, USA, June, 2005

Design of an Object-Oriented Workflow Management System with Reusable and Fine-Grained Components

Gwan-Hwan Hwang¹, Yung-Chuan Lee², and Sheng-Ho Chang¹

¹ Department of Information and Computer Education,
National Taiwan Normal University, Taipei, Taiwan
{ghhwang, shchang}@ice.ntnu.edu.tw

² System Development and Technical Support Department,
Trade-Van Information Service Company, Taipei, Taiwan

Abstract. Languages that support object-oriented programming are now mainstream, and can support software reuse. This study focused on the reusability of components for workflow management systems (WfMSs). Implementing a WfMS in object-oriented programming languages without considering the characteristics of the WfMS does not ensure that all the components will be reusable. We first clarify the reusability of WfMSs and point out the difficulties in constructing reusable components for WfMSs. We then propose an object-oriented model for WfMSs named the “Java-based object-oriented WfMS” (JOO-WfMS), whose components are fine-grained and are classified into a functional stack with three layers. This extends the reusability of objects in developing workflow applications. The resulting architecture can support real-time flow control as well as the dynamic instantiation of objects. Two mechanisms are embedded into the JOO-WfMS to increase the reusability of its components: (1) a workflow failure-handling language, which can increase the reusability of activities when flexible failure recovery is necessary; and (2) the user communication components and their corresponding architecture. The goal of the architecture is to increase the reusability of codes used for communication between the user and the activities in WfMSs.

Keywords: Workflow Management System, Object-Oriented Programming Language, Software Components, Reusability.

1 Introduction

Workflow management systems (WfMSs) are software systems for supporting coordination and cooperation among members of an organization whilst they are performing complex business tasks [1–5]. The business tasks are modeled as *workflow processes*, which are automated by the WfMS. The workflow model (also referred to as the *workflow process definition*) is the computerized representation of the business process. It defines the starting and stopping conditions of the process, the activities in the process, and control and data flows among these activities. An *activity* is a logic

step within a workflow, which includes the information about the starting and stopping conditions, the users who can participate, the tools and/or data needed to complete this activity, and the constraints on how the activity should be completed. Activities are usually organized into a directed graph that defines the order of execution among the activities in the process. Nodes and edges in the graph represent activities and control flow, respectively. A *workflow process instance* is the execution of a workflow process definition by the WfMS. The execution of a workflow process instance is controlled by the *workflow engine*.

Cox introduced the “software IC” concept in 1987 [6], with the aim of software development mirroring hardware development in the use of existing components to generate a new component. The reusability of classes or components is very important to the realization of the software IC. The objects that are instantiations of classes communicate via messages. The fundamental features of object-oriented programming languages that support the reuse of classes include encapsulation, inheritance, polymorphism, and object composition [6,7].

In this paper, we first analyze how to apply the existing technologies in object-oriented programming languages to implement a WfMS with reusable components. We present the design of a Java-based object-oriented WfMS, called a JOO-WfMS. The components comprising the JOO-WfMS are fine-grained, and we aim to reduce the dependence between components to the lowest possible level so as to extend the reusability of components in developing workflow applications. Also, these components are classified into a functional stack with three layers: a base layer that is implemented by the implementer of the JOO-WfMS, and another two layers that are implemented by the application programmer of the workflow system. The resulting architecture can support real-time flow control as well as the dynamic instantiation of objects.

Second, we demonstrate that straightforward application of the encapsulation, inheritance, polymorphism, and object composition limits the reusability of components in a WfMS. Two mechanisms are embedded into the JOO-WfMS to increase the reusability of its components: (1) a workflow failure-handling (WfFH) language, which can increase the reusability of activities when flexible failure recovery is necessary by minimizing the dependence between activities; and (2) the components for implementing user communication and their corresponding architecture. The goal of the architecture is to increase the reusability of codes used to implement communication between the user and the activities.

The remainder of the paper is organized as follows. Section 2 surveys the related work. Section 3 discusses considerations in constructing a WfMS with reusable components. Section 4 presents the functional stack and operational architecture of the JOO-WfMS. Finally, Sections 5 and 6 present the experimental results and conclusion of this paper, respectively.

2 Previous Work

The WooRKS is an object-oriented workflow system designed to assist organizations in defining, executing, coordinating, and monitoring workflow procedures within a shared environment. Its architecture relies on an object-oriented database management

system upon which all the objects are represented and supported by a UNIXTM server, with which clients communicate via the TCP/IP protocol and run Microsoft WindowsTM or OSF/MotifTM [8].

Xsoft's InConcert is an open workflow product realized using object-oriented technology and a client/server architecture with distributed multiplatform and multinetwork support [9]. It allows for dynamic workflow modification ("process mutation") and ad-hoc routing. An application programming interface (API) with C and C++ language bindings facilitates application integration, and provides a full range of application development options for creating custom solutions. Its object-oriented design with tools, associated API, and language bindings provide flexibility for system developers deploying workflow solutions regarding integration with other systems, extendibility, and some degree of reusability. However, its extendibility and reusability apply mainly to process modeling and process implementation, and not to the entire architecture of a WfMS.

TriGS*flow* is a flexible workflow framework supporting frequently changing requirements [10]. Its architecture is based on an object-oriented database (Gem-Stone/S), and employs object-oriented database technology to implement its workflow model. An important feature of TriGS*flow* is the seamless integration of event control action (ECA) rules into an object-oriented model, the flexibility of workflow specification due to rule modeling, and the integration of external applications as part of workflow processing. ECA rules take the follow form:

ON Event IF Condition DO Action

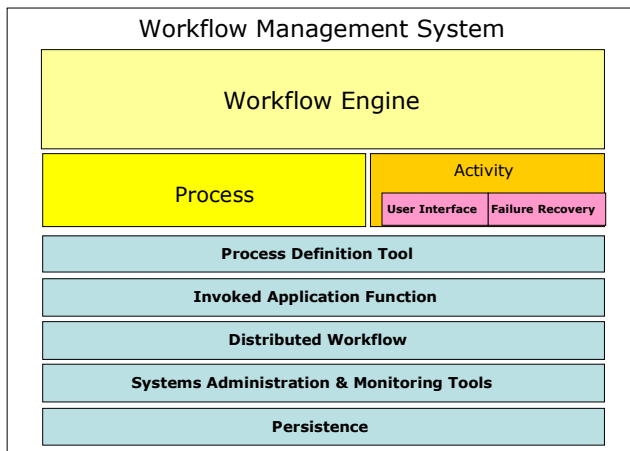
Events represent real-world situations (e.g., a machine breakdown) that result in the invocation of rules. The condition part of a rule defines when to evaluate the condition and a Boolean expression (e.g., "is the machine available?"), possibly based on the result of a database query (e.g., "select all orders that are scheduled on the damaged machine"). The action part (e.g., "schedule the next operation") defines when to execute the action, and a sequence of arbitrary messages. The TriGS*flow* system provides an object-oriented workflow model that allows its users to build workflows by instantiating predefined specialized classes. However, TriGS*flow* focuses on ECA rules, and not on the reuse of components.

The micro-workflow encapsulates workflow features in separate components [11]. At the core of the architecture, several components provide basic workflow functionality, and additional components implement advanced workflow features. Using objects, software developers extend the core only with the features they need. This design localizes the changes required to tailor a workflow feature to the component that implements it. The synchronization-object micro-workflow employs the ECA of TriGS*flow* [10]. Every procedure instance has its own precondition object, which corresponds to the condition part of the ECA model. The *PreconditionManager* is a separate process that monitors preconditions. A procedure begins executing by transferring control to its precondition. The framework initializes the precondition from the workflow runtime, and then places the precondition into the queue of the *PreconditionManager*. In effect, this blocks procedure execution until the condition is fulfilled. The *PreconditionManager* evaluates every queued

precondition continuously. This works well when the number of preconditions is small and when their evaluation is not computationally intensive. As a result, this architecture cannot support real-time flow control.

3 Some Considerations for Constructing Reusable Components for a WfMS

A WfMS comprises many components, including a set of objects. Therefore, the reuse of WfMS components involves the reuse of a *framework* [12], which is the design of a set of objects that collaborate to carry out a set of tasks. Frameworks usually provide the design of interfaces or abstract classes to represent what the developers should implement and the way that its functions are divided among objects. Therefore, to clarify the reusability of components, it is necessary to design a highly reusable WfMS.



Workflow engine, process, and activity are the basic components of the WfMS, and the others are the advanced components.

Fig. 1. Architecture of the WfMS

Based on the WfMC reference model [13] and the OMG workflow management facility [14], we illustrate a generalization of the components of a WfMS in Fig. 1. In addition to the components introduced in the reference model [13], we add two functions that are common in modern workflow systems: failure recovery [15] and persistence. As shown in Fig. 1, the basic components of a WfMS are a workflow engine, process, and activity; a simple WfMS may only comprise these three components. The advanced components are the failure-recovery mechanism, process-definition tool, user interface, application invocation, workflow interoperability, and system administration and monitoring tools. The increasing popularity of the WfMS in different domains increases the importance of these advanced components. In the following, we examine some of these components in order to determine the characteristics they should exhibit in order to be reusable.

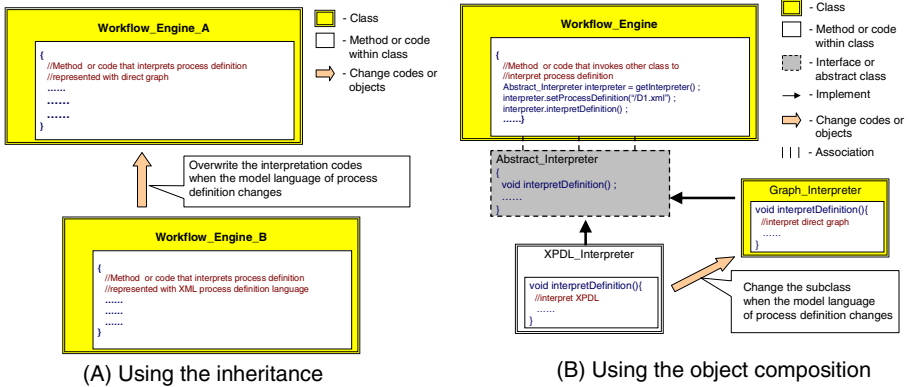


Fig. 2. Changing the interpretation function in the workflow engine

3.1 Workflow Engine

First of all, the workflow engine is the core of a WfMS. It is responsible for various and complex functions and should communicate with the majority of other components in the WfMS. Therefore, its reusability is improved by using a compositional design [11] in which only the basic functions should be implemented within the workflow engine, with the additional functions embedded using objects. For example, one of the important functions for the workflow engine is to interpret the process definition. The interpretation function should be designed according to the employed definition language. Consider the situation that we need to change the definition language from GPSG [16] to XPDL [17]. Fig. 2A illustrates how this can be achieved without changing the design of the workflow engine. The interpretation function is implemented using the `workflow_Engine_A` class. If we want to change the definition model, we need to inherit the `workflow_Engine_A` class and override the method that implements the interpretation function. Although other codes in the methods of the `workflow_Engine_A` class are reused, the `workflow_Engine_A` class cannot be reused directly. A compositional design is shown in Fig. 2B. In this design, we separate the implementation of interpretation function from `workflow_Engine_A` class and implement it in the `Abstract_Interpreter` subclass. The `workflow_Engine_A` class invokes the method defined in the `Abstract_Interpreter` class to interpret process definition.

3.2 Workflow Process

Generally speaking, only one or a small set of activities should be activated during the execution of a process. This is because the lifetime of a process instance is always very long, i.e., several hours or even several days. For the workflow process, it is necessary for the programming language to support a mechanism to load and create the instances of workflow activities dynamically; this can reduce the memory requirements as well as the computation loading of the workflow server. For example, the `forName` and `newInstance` methods in the `Class` class of the Java programming language support loading a specific class and creating an instance, respectively.

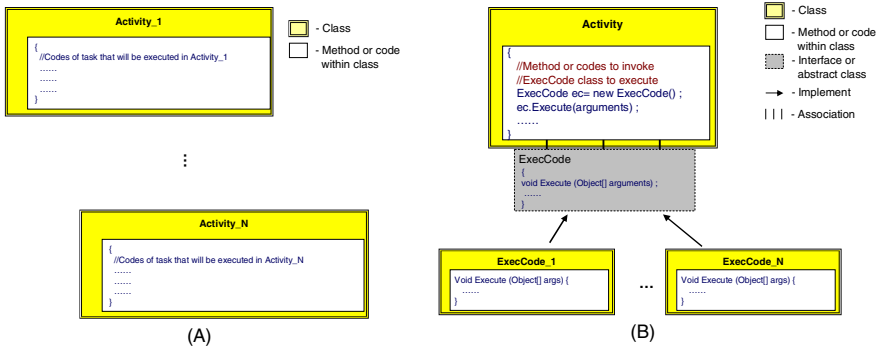


Fig. 3. Changing the execution codes of an activity

Another important feature that the workflow process should support is real-time flow control. Whenever the execution of an activity is finished, the workflow system should invoke the subsequent activity (or activities) instantly. Sometimes the branching action in the flow control requires data access and computation. For example, the workflow system may have to make several database transactions and analyze the results to make the branch decision. Not all workflow systems support real-time flow control. For example, *TriGSflow* [10] and *micro-workflow* [11] employ an ECA rule model to handle flow control. Because many preconditions stored in the queue are checked continuously on a round-robin basis, these methods cannot support real-time flow control.

3.3 Workflow Activity

The activity is the basic execution unit of the workflow process. Different activities involve the execution of different codes. A reusable activity component can be reused to execute different execution codes. The design in Fig. 3A implements the execution code in the `Activity_1` class, and different tasks are implemented in different activity classes. In this case, neither the activity nor the execution code can be reused. In Fig. 3B, a task is implemented in an `ExecCode` subclass that is not directly related to the `Activity` class. Therefore, changes to the `ExecCode` subclass will not affect the `Activity` class, and the `ExecCode` subclass can be reused by other activities in different processes.

3.4 User Interface

In the execution of the WfMS, the activity usually needs to communicate with the user. The user interfaces are various and the user may connect to the system via different types of devices. The reusability is broken if the programmer needs to modify the codes in the activities when the WfMS is extended to accept a connection from a new client device. The design of the window components and the implementation of display protocols are the most important considerations for ensuring the reusability of user-interface components (UICs).

3.5 Failure Recovery

Some workflow systems (e.g., transactional WfMSs) that emphasize stability and reliability may implement a failure-recovery mechanism. The goal of failure recovery in the WfMS is to bring the failed workflow process back to some semantically acceptable state so that the cause of the failure can be identified. The problem can then be fixed and the execution resumed to ensure the successful completion of the workflow process. The basic failure-recovery process includes the following three steps:

1. The execution of the workflow process is terminated and the workflow engine then decides the *end compensation point* (ECP) and the *compensation set* of the failure.
2. Activities in the compensation set are compensated.
3. The workflow process is restarted from the ECP.

Previous failure-recovery models [18–22] have some drawbacks. First, they only allow the ECP and compensation set of a failure in an activity to be specified statically before the workflow process is compiled, which limits the flexibility of failure recovery. A more flexible way is to compute the ECP and compensation during the process run-time according to the execution results of activities. Second, the ECP and compensation set are specified by explicitly using the names (or some kind of identities) of activities. Therefore, inserting or deleting activities to or from a workflow process may require modification of the failure-recovery specification in another activity. Such dependency between activities reduces their reusability.

We propose a new failure-recovery model for WfMSs [15]. This model is supported with a new language, called the WfFH language, which allows the workflow designer to write programs so that s/he can use data-flow analysis technology to guide failure recovery during workflow execution. With the WfFH language, the computation of the ECP and the compensation set for failure recovery can proceed during the workflow process run-time according to the execution results and status of workflow activities. Also, the failure-recovery definitions programmed with the WfFH language can be independent, thereby dramatically increasing the reusability of the components in the WfMS.

3.6 Workflow Persistence

Workflow engines usually log the execution history of the processes they execute for evaluation and recovery purposes. Some of the systems store the history in the database. Reusable persistence components can provide database-independence operations for the developer of workflow applications.

4 The Functional Stack and Operational Architecture of the JOO-WfMS

In this section, we present the architecture of the JOO-WfMS. The operation of a WfMS is complicated. For example, the execution of an activity may involve the activation of the user interface, database manipulation, and failure recovery. Some of these are basic functions of the WfMS itself and others should be implemented by the

application programmer. Fig. 4A shows the functional stack of the components of the JOO-WfMS. From the viewpoint of software engineering, dividing the components into a stack provides at least two benefits: (1) promoting the reusability of all components in the system, and (2) increasing the efficiency in software development. The stack is divided into the following three layers:

1. **The fundamental layer:** This layer provides the basic components and mechanisms for the system, and is implemented and maintained by the implementer of the WfMS – the developer of the workflow application is not allowed to modify this layer. This layer includes the Java interfaces (i.e., the prototype) of all the components listed in Fig. 4 as well as the workflow engine that follows the defined interfaces. Also, mechanisms such as event handling, exception handling, and failure recovery, and the user-interface display protocols for different devices are implemented in this layer.
2. **The workflow basic component layer:** In this layer, the programmer implements all the needed classes for objects in Java according to the interface defined in the fundamental layer. It contains objects for transitions, activities, execution codes of activities, failure-handling objects, and process-definition interpreter objects.
3. **The workflow process composition layer:** In this layer, the programmer uses the objects implemented in the second layer to construct his/her workflow process.

Fig. 4B illustrates the operational architecture of the JOO-WfMS. Its core consists of the workflow engine, process, and activity. Several process instances are executed simultaneously whilst being monitored and controlled by the workflow engine. The process instances created by the workflow engine will notify the workflow engine of events that occur during execution. Several activity instances and transition instances are activated by a process instance. Note that the transaction objects control the flow of the process. Similarly, activity and transition instances will notify the process instance of the occurrences of events during execution. In the following subsections, we detail the components of the JOO-WfMS.

4.1 Workflow Engine

The workflow engine in the JOO-WfMS is responsible for the following tasks:

- Receiving messages from the workflow participant.
- Initiating the process and starting it according to the process definition.
- Receiving and dealing with events from processes.
- Recovering and resuming processes after a system crash or the occurrence of process failures.
- Supervising for administration and monitoring purposes.

In addition to handling messages from the workflow participant, the workflow engine deals with events from processes. By default, a process instance will send an event to the workflow engine when its execution state changes. Programmers can instruct the workflow engine to execute some actions when receiving events from processes, e.g., saving some information about the process. To promote the reusability, we adopt a compositional design for our workflow engine. We separate the implementation of the following functions from the `Workflow_Engine` class: message handling,

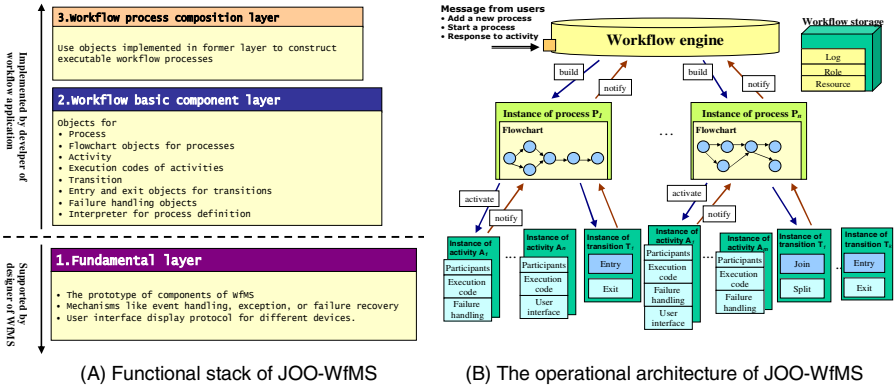


Fig. 4. Functional stack and operational architecture of the JOO-WfMS

system administration and monitoring, process control, process-definition interpretation, and event handling. These functions are implemented in other classes, and the workflow engine can invoke them as required. Therefore, we can change the invoked classes to meet various functional requirements rather than having to modify the `Workflow_Engine` class.

4.2 Workflow Process

In the JOO-WfMS, the workflow process component is responsible for the following tasks:

- Creating and starting the instances of activities and transitions.
- Receiving and handling events from activities and transitions.
- Updating the information of the Flowchart object.

Flow control is crucial to a workflow system, and comprises the following two tasks: (1) obtaining the activities that will be executed next, and (2) deciding the flow of process when meeting branches. However, object-oriented systems lack a procedural representation of control flow [24] due to the decomposition into classes that distributes flow control among different objects. Thus, the global control flow and behavior are less visible than in procedural programs [11]. Therefore, an additional challenge in building object-oriented workflow architectures lies in providing abstractions that maintain an explicit representation of the control flow without violating the principles of good object-oriented design. In the JOO-WfMS, the programmer defines the decision rules in Transition objects. A process instance can obtain information about successor activities and transitions from its Flowchart.

Fig. 5 shows the schematic of the Flowchart, in which circular and rectangle nodes represent the activity and transition, respectively. We present the details of the transition object in Section 1.6. The following six methods are provided to control the execution of process instance: `startProcess`, `suspendProcess`, `terminateProcess`, `abortProcess`, `resumeProcess`, and `restartProcess`. These methods may be invoked by the workflow engine or system administration tools (full details are available elsewhere [23]).

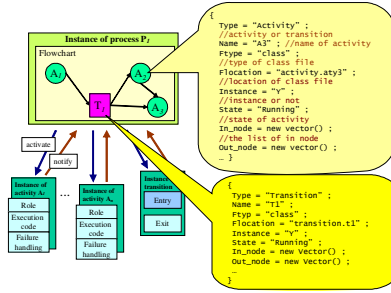


Fig. 5. A schematic of Flowchart

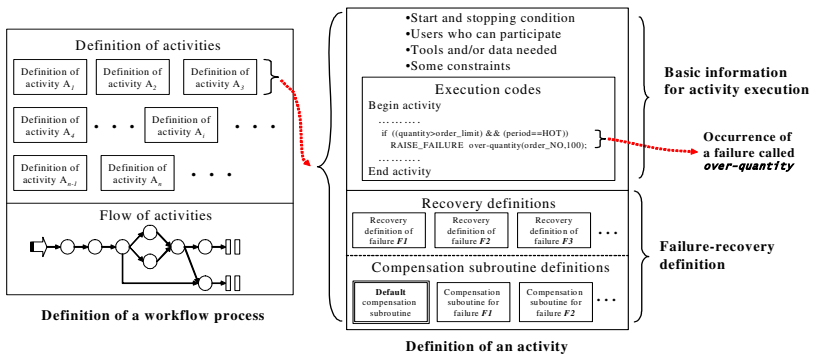


Fig. 6. Skeleton of an activity object

4.3 Workflow Activity

Fig. 6 illustrates the skeleton of an activity object in the JOO-WfMS. It contains basic information for the execution of the activity and additional objects for handling failure recovery. The basic information for activity execution includes at least the starting and stopping conditions, the users who can participate, the tools and/or data needed to complete the activity, the constraints on how the activity should be completed (such as the time limits), and the *execution codes* of the activity. The additional failure-recovery definition in the activity for failure recovery comprises recovery definitions and definitions of the compensation subroutines. The execution code is the program that executes the activity and which may trigger the failure-recovery process when the execution causes a failure. In the execution code, a user interface is invoked to interact with workflow users who participate in the execution of this activity.

4.4 User Interface

Our aim is to support a universal user interface in which the workflow participant can use different types of client devices to communicate with the workflow engine. The developer of the workflow application builds the user by embedding codes that invoke

an application program interface in the `ExecutionCode` object of the activity. This is very similar to window programming in Java – we call the window components provided in the JOO-WfMS UIC (full details of the functions and attributes of window components provided in the JOO-WfMS are available elsewhere [23]).

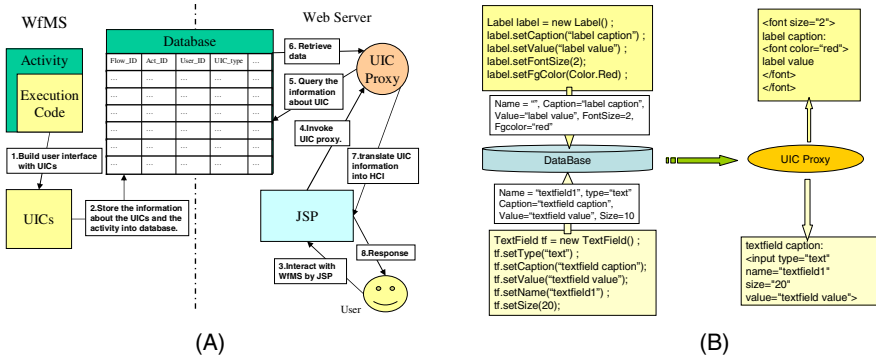


Fig. 7. Operation of the user interface in the JOO-WfMS

The operation of the user interface in the JOO-WfMS is shown in Fig. 7A. Although our architecture does not limit the types of client devices used by the user, we assume that the user connects to the JavaServer Pages (JSP) [25] server of the WfMS via a Web browser.

- Step 1: The programmer first invokes the UIC in the execution code of an activity (Fig. 7B).
- Step 2: The JOO-WfMS stores the UICs added in step 1 to a database.
- Step 3: An user sends a request to a specific JSP server via the Web browser.
- Step 4: After receiving the request, the JSP server consults the UIC proxy to retrieve data and translate data of UICs in this user interface.
- Step 5: The UIC proxy queries the data of UICs from the database.
- Step 6: The database returns the results of querying.
- Step 7: The UIC proxy translates the data of UICs into a specific format (e.g., HTML tags).
- Step 8: The JSP server sends the result to the user via HTTP.

4.5 Failure Recovery

In the JOO-WfMS, we adopt the WfFH language to support a workflow failure-recovery model [15]. The programmer defines the recovery definition of failure in the WfFH language. As shown in Fig. 6, the recovery definitions of failures are translated into failure-handling objects and then are embedded in the activity objects. In the JOO-WfMS, we use code-generation technology to translate the WfFH language into Java codes. The way to activate the failure-recovery process is specified in the execution code of the activity. In the JOO-WfMS, the execution code of an activity is also a Java program with the *failure-recovery extension*, which takes the following form:

```
RAISE_FAILURE Failure_Name(arg1, arg2, ..., argn);
```

The JOO-WfMS compiler translates the above statement into the following Java program (full details are available elsewhere [15,23]):

```
// To generate an array args to store  $arg_1$ ,  $arg_2$ , ..., and  $arg_n$ 
Object[] args= new Object[n];
args[0]= arg1;
args[1]= arg2;
...
args[n]= argn;
// To instantiate a failure object
Failure fail = activity.getFailure(Failure_Name);
// To set up the failure arguments
fail.SetArguments(args);
// To throw a Java exception to start the failure-recovery process
throw new RaiseFailureException(fail);
```

4.6 Workflow Transition

Most workflow languages support the basic constructs of sequence, iteration, splits (AND and OR), and joins (AND and OR) [26–28]. Fig. 8 shows the basic control flows of the WfMS. The JOO-WfMS allows the programmer to implement specific branch decisions according to the actual requirements. For example, in the “OR-split” shown in Fig. 8, the programmer may have to control the branch decision by examining the data stored in the database. The Transition component is the solution for flow control in the JOO-WfMS. Fig. 9 illustrates the structure of the Transition component, which has two objects: Entry and Exit. The programmer should implement the onEnter and onExit methods of the Entry and Exit objects, respectively. With different Entry and Exit objects, the transition can deal with any complicated join and split conditions (full details are available elsewhere [23]). One of the advantages of the control flow mechanism in the JOO-WfMS is that all the related objects can be easily reused, including the Entry, Exit, and Transition objects. The other advantage is that it can support real-time control flow: when the execution of an activity terminates, the onEnter method of the Entry object of its following Transition object(s) will be called immediately.

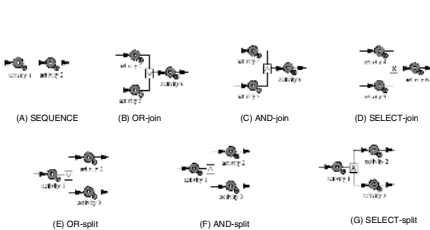


Fig. 8. Basic control flows in the WfMS

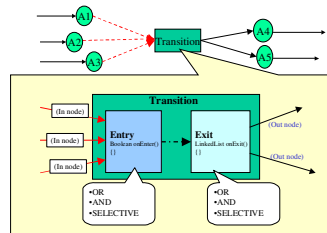


Fig. 9. Structure of the transition component in the JOO-WfMS

4.7 Workflow Persistence

Most commercial WfMS products use relational database management systems to store the histories. Some techniques (e.g., the data access object introduced in J2EE) and standards for database connectivity (e.g., ODBC and JDBC) help in the manipulation

of database systems. In addition to using databases, some WfMSs define their own data structure to store the execution. Therefore, the persistence component should allow the developer to use different data media to store the history. The event handlers of the workflow engine or process will invoke the implementations of the Persistence interface to store the logs. Two methods are defined in Persistence interface: `saveToStorage` and `retrieveFromStorage`. Developers may implement the policy for saving history (e.g., which data should be kept or which history should be stored into which table in the database) in the `saveToStorage` method and implement the way to retrieve the history from data storage in the `retrieveFromStorage` method.

5 Implementation and Experimental Results

To evaluate the reusability of components and performance of the JOO-WfMS, we first implemented layer one, followed by the second and third layers for several workflow applications including workflow systems for school administration and reciprocal E-learning. The user-interface protocol was implemented in JSP [25], which allows the user to connect to the workflow engine using a Web browser. Our experiences in implementing these applications demonstrated the ease of reusing the majority of the components in the second layers, including execution code objects and transaction objects.

Table 1. Times required to compute the compensation set and ECP (in seconds)

Failure name	Time required
Deputy_Not_Available	0.0004
Need_More_Attachment	0.0002
Improper_Goods	0.0008
Budget_Overspend	0.0006
Bad_Quality	0.0014
Invalid_Item	0.0006

Table 2. Average time required to perform persistence to database and text (in seconds)

The online system for ask for leave		Time required	
		Access	Text
Process		0.0156	0.0015
Activity ID	Activity name		
A ₁	login	0.0172	0.001
A ₂	Fill_form	0.0157	0.0021
A ₃	Deputy_choice	0.014	0.001
A ₄	Deputy_agreement	0.0157	0.001
A ₅	Division_agreement	0.0156	0.0011
A ₆	Department_agreement	0.0141	0.002
A ₇	Principal_agreement	0.0140	0.002
A ₈	Success	0.0171	0.001
A ₉	Notsuccess	0.0203	0.0015
A ₁₀	Personnel_agreement	0.0141	0.001
F ₁	Deputy_Not_Available	0.0141	0.001

Experiments were performed to evaluate the performance of the JOO-WfMS. All the experiments were run on a PC with a 2.4-GHz Pentium 4 processor, 512 MB of RAM, the MS Windows 2000 operating system, and Java Development Kit 1.4.2_02. Table 1 lists the times required to compute the compensation set and ECP for a purchase workflow, and Table 2 and Table 3 list the times for performing persistence in database transaction and Java serialization, respectively. The experimental results demonstrate that the performance is acceptable.

Table 3. Average time required to perform persistence by Java serialization (in seconds)

The online system for ask for leave		File size (kB)	Time required
Process		26	0.039
Activity ID	Activity Name		
A ₁	Login	9	0.0315
A ₂	Fill_form	13	0.0625
A ₃	Deputy_choice	19	0.0625
A ₄	Deputy_agreement	19	0.0625
A ₅	Division_agreement	19	0.0664
A ₆	Department_agreement	21	0.0705
A ₇	Principal_agreement	26	0.0708
A ₈	Success	29	0.0700
A ₉	Notsuccess	29	0.0725
A ₁₀	Personnel_agreement	25	0.0707
F ₁	Deputy_Not_Available	19	0.0235

6 Conclusions and Future Work

A WfMS with an architecture consisting of reusable components can be implemented by the application of techniques such as encapsulation, inheritance, polymorphism, and object composition of a traditional object-oriented programming language. However, we found that it is necessary to employ other schemes to break the dependencies among the components since these dependencies impede the reuse of the components. The JOO-WfMS adopts the WfFH language and user communication components to achieve this. The result is a system with fine-grained components, whose level of reusability was found to be high.

Future work should aim at building an application-development environment with a graphical user interface (GUI). Although the layered structure of the functional stack of the JOO-WfMS separates the code implemented by the WfMS implementer (layer 1) from the application designer of workflow (layers 2 and 3), a GUI development environment should offer the application designer of the workflow with appropriate methods to compose, reuse, and manage the fine-grained components.

References

1. D. Georgakopoulos, M. Hornick, and A. Shet. Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases, Vol. 3, No. 2, 1995, Pages 119–153.
2. Shi Meilin, Yang Guangxin, Xiang Yong, and Wu Shangguang. Workflow Management Systems: A Survey. International Conference on Communication Technology, 1998.

3. A. Elmagarmid, and W. Du. Workflow Management: State of the Art vs. State of the Market. Proceedings of NATO Advanced Study Institute on Workflow Management Systems, 1997.
4. Workflow Management Coalition. Workflow Reference Model. Workflow Management Coalition Standard, WfMC-TC-1003, 1995.
5. Workflow Management Coalition. Workflow: An Introduction. Workflow Handbook, 2002.
6. Cox, B.: Object-Oriented Programming: An Evolutionary Approach. Addison-Wesley, Reading, MA, 1987.
7. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1995..
8. M. Ader et al., "WooRKS, an Object Oriented Workflow System for Offices," Technical Report, Bull S.A. (Paris), Technics en Automatitzacio d'Officines S.A. (Barcelona), Dep. of Computer Science (University of Milan), Communication and Management Systems Unit (Athens), 1994.
9. Sunil K. Sarin: "Object-Oriented Workflow Technology in InConcert". COMPCON 1996: 446-450.
10. Stefan Rausch-Schott. TRIGSflow—Workflow Management Based on Active Object-Oriented Database Systems and Extended Transaction Mechanisms. PhD thesis, Institute of Applied Computer Science, Johannes Kepler University, Linz, Austria, February 1997. Published by Trauner Verlag, Linz, ISBN 3-85320-991-2.
11. Manolescu, D.A. Micro-Workflow: A Workflow Architecture Supporting Compositional Object-Oriented Software Development. Ph.D. Dissertation. Department of Computer Science. University of Illinois at Urbana-Champaign, 2001. Ralph E. Johnson, Advisor.
12. Ralph E. Johnson and Vincent F. Russo. Reusing object -oriented designs. Technical Report Technical Report UIUCDCS 91--1696, University of Illinois, May, 1991.
13. Workflow Management Coalition. Workflow Reference Model. Workflow Management Coalition Standard, WfMC-TC-1003, 1995.
14. Object Management Group. Workflow management facility specification, 2000. OMG Document Number formal/00-05-02. Available at <http://www.omg.org>.
15. G. H. Hwang, Y. C. Lee, and B. Y. Wu, "A Flexible Failure-recovery Model for Workflow Management Systems," International Journal of Cooperative Information Systems, Vol. 14, No. 1 (2005) 1-24.
16. N.S. Glance, D.S. Pagani, and R. Pareschi. Generalized process structure grammars (GPSG) for flexible representations of work. Proceedings of Conference on Computer Supported Cooperative Work, 1996.
17. WFMC. Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface -- XML Process Definition Language (XPDL) (WFMC-TC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002.
18. Weimin Du, Jim Davis, and Ming-Chien Shan. Flexible Specification of Workflow Compensation Scopes. ACM Group, Phoenix, Arizona, USA, 1997.
19. M. Kamath and K. Ramamrithan. Failure Handling and Coordinated Execution of Concurrent Workflows. IEEE International Council for Open and Distance Education, 1998.
20. J. Eder and W. Liebhart. Workflow recovery. IEEE International Conference on Cooperative Information Systems, 1996.
21. Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution language for Web Services, Version 1.1 dated may 5, 2003. <http://www-106.ibm.com/developerworks/sebservices/library/ws-bpel>.

22. David Chappell. Understanding BizTalk Server 2004. <http://www.microsoft.com/biztalk/evaluation/introduction.asp>.
23. Yung-Chuan Lee, "Towards the Reusability of Object-Oriented Workflow Management Systems." Master Thesis, Advisor: Gwan-Hwan Hwang, Dept. Information and Computer Education, National Taiwan Normal University, 2003.
24. Rex Hartson. User-interface management control and communication. IEEE Software, pages 62–70, January 1989.
25. Sun Microsystems (2002): Inc. JSR-000053 Java™ Servlet 2.3 and JavaServer Pages™ 1.2 Specifications.
26. B. Kiepuszewski, A.H.M. ter Hofstede, W.M.P. van der Aalst, Fundamentals of Control Flow in Workflows. QUT Technical report, FIT-TR-2002-03, Queensland University of Technology, Brisbane, 2002.
27. J. L. Peterson, Petri net theory and the modelling of systems, Prentice Hall, 1981.
28. van der Aalst, W., and van Hee, K., Workflow Management: Models, Methods, and Systems. The MIT Press. 368 pp., 2002. ISBN 0-262-01189-1.

Modeling the Behavior of Dispatching Rules in Workflow Systems: A Statistical Approach

Gregório Baggio Tramontina* and Jacques Wainer

Universidade Estadual de Campinas, Instituto de Computação,
Campinas SP 13084-971, Brazil

Abstract. Using scheduling techniques to reorder work in workflow systems can improve the performance of the business processes enacted by these systems. But this research area is still under-explored. This paper presents preliminary results on ongoing research on the modeling of the behavior of scheduling techniques in workflow systems using statistical analysis. It discusses the motivations for this approach and presents a first workflow scenario to be studied. Simulations of the use of dispatching rules to minimize the percentage of late cases are performed. The results are analyzed with statistical regression techniques to estimate functions that describe the observed data and compose a model to the behavior of the scheduling techniques. The model is evaluated against new data that was not present in the first observed data and the results show that it is feasible and could be used to assist systems administrators in applying scheduling techniques to workflow systems.

1 Introduction

One of the goals of a workflow management system (WFMS) is to improve the performance of the business processes it enacts. But one area in which they could help bring an important improvement is under-explored: the ordering of the execution of cases within the workflow itself. Most of today's workflow systems dispatch work on a first in first out (FIFO) basis. Some systems may also present the set of all possible tasks to their potential executor, who chooses the task he or she will perform next, which amounts to dispatching the work items at some random order. The tasks may be annotated with a priority or urgency indicator, but there is no guarantee that the executors will choose the high urgency tasks.

This paper describes preliminary results on ongoing research on applying scheduling techniques to workflow systems. It describes the first statistical models to the behavior of scheduling techniques in dynamic workflow systems with uncertainties in the processing times of the cases in its activities. These models were obtained by the statistical analysis of the simulation of dispatching rules in this workflow scenario. It is hoped that the models may be used as a guide to the application of scheduling techniques in future WFMSs.

* Partially supported by the Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP, Brazil, Proc. 03/12742-0.

This paper is organized as follows. Section 2 briefly discusses the application of scheduling to workflow and the related literature, as well as the motivation for this research. Section 3 describes the studied workflow scenario, the modeling of the uncertainties present in it, and the metrics used to evaluate the rules' performance. Section 4 defines and describes the chosen dispatching rules. Section 5 describes the analysis made to generate the statistical models. Section 6 evaluates the model and finally Sec. 7 concludes the paper and points to future work. Appendix A shows the numeric values that compose the model.

2 Applying Scheduling to Workflow

Applying scheduling techniques to workflow systems may seem intuitive, but it is not. The problems in mapping a workflow case ordering into scheduling are many. A list of a few of them is discussed in [1], which shows that there are great issues to be handled in next research steps. Many authors also recognize that scheduling and time issues are important in workflow systems and conducted research in related areas such as time management and time constraints in workflow systems, as well as business process management. Examples of research in these areas are [2,3,4,5,6,7,8]. The scheduling literature, specifically, does not cover workflow problems as a whole.

While studying the numerical results of the simulations, the authors noticed that they followed (or at least could be approximated to) a normal distribution. So, if there was a way to predict the means and standard deviations of these distributions, the results could be reconstructed and serve as a prediction tool to help in the application of scheduling techniques to workflow systems.

Progress has already been made in [1] with the study of the behavior of common scheduling techniques in a simple workflow scenario. But its main objective is to compare the performance of other techniques to the FIFO rule to show that performance improvements could be achieved by reordering work in another fashion. Significant advances, however, can be made by developing a model to the behavior of these techniques.

This paper analyzes a workflow scenario with uncertainties on the processing times of the cases in the activities and dynamic behavior using simulation. The uncertainties in these processing times means that one does not know their exact values, and there is at maximum a good approximation to them. These uncertainties are handled with the *guess and solve* approach described in [1].

3 The Example Scenario

The studied scenario consists of a sequence of three activities depicted in Fig. 1. Each activity is assigned a time interval from which the real execution times of the cases in each of these activities are taken (from an uniform distribution). These intervals are represented in Fig. 1 by the numbers above the activities. They are equal for all activities, which means they have the same means and standard deviations.

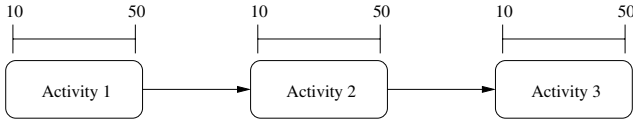


Fig. 1. The workflow scenario described in this paper

In order to apply scheduling techniques in this scenario it is necessary to map it to a scheduling problem. Due to its features, this workflow was mapped into a *flow shop scheduling problem*. The flow shop is defined as a set J of n jobs that have to be processed in one or more machines of a set M which contains m machines. All jobs have the same route to follow within the machine set. Each workflow activity is considered as a machine, so $M = \{1, 2, 3\}$, and each case is a job that needs processing in all of these machines, in this order.

Every job j has a *release date* r_j and a *due date* d_j . Release dates are the point in time when the jobs arrive at the system and are set so a certain machine utilization ratio is achieved (this will receive more attention later). Due dates are the point in time before which the jobs must be completed, and they are set multiplying the total processing time of a job by a constant (the *allowance factor*) and adding its release date to this value. In a mathematical formulation, the job due date is given by $d_j = r_j + A \sum_{i=1}^m p_{ij}$, where p_{ij} is the processing time of job j in machine i , A is the allowance factor, and m is the number of machines in the system. The amount of time within which a job can be completed without being late, calculated by $d_j - r_j$, is called the job's *allowance*.

3.1 Workload Generation

The *workload* is a crucial aspect of a system. It is defined as the number of tasks that are still being processed in the system [5,9]. The scenario's workload was generated using the model proposed by Mattfeld and Bierwirth [10]. This model controls the inter-arrival times of the jobs to determine the pace at which they arrive at the system. The mean inter-arrival time (λ) of the jobs is determined by the mean job processing time (\bar{p}) in the scenario, the number of machines in the system (m) and a desired machine utilization rate ($0 \leq U \leq 1$; $U = 0.75$ means a utilization rate of 75%). If \bar{p} is given, then λ can be calculated by $\lambda = \bar{p}/(mU)$. The inter-arrival times are then set using an exponential distribution with mean λ . This paper considers 9 different utilization rates, namely, from 0.15 to 0.95 with increments of 0.1, varying from lightly to heavily loaded systems.

3.2 Optimization Criteria

The performance of the scheduling techniques must be evaluated quantitatively. The optimization criteria, or objective function, analyzed in this paper, is the mean number of late jobs (cases) in the system that each rule “produces” when it is used to reorder the work in the system. This metric is defined straightforwardly as n_t/n , where n_t is the number of late jobs in the system and n is the total

number of jobs in the same system. It is a very important measure since late cases can incur penalties to the company and decrease customer satisfaction.

Guess and Solve Approach. The uncertainties in the jobs' processing times can be modeled by the guess and solve approach described in [1]. It is assumed that the system has a *guesser* capable of providing a prediction p'_{ij} of the processing times p_{ij} of the jobs with a maximum error f_p . It means that the property $f_p \geq \left| \frac{p'_{ij} - p_{ij}}{p_{ij}} \right|$ holds for every prediction. For our simulations, the chosen values of f_p were 0.10, 0.20, and 0.30, meaning a 10%, 20% and 30% error respectively.

The *guesser* component is an abstraction of a number of prediction techniques that can be used such as relying on someone's experience, using machine learning, or statistical techniques based on past cases. Once the execution times and routes are guessed, one can solve the resulting deterministic problem.

4 Dispatching Rules

Dispatching rules (or *priority rules*) guide the selection of the jobs locally in the machines. Whenever a machine becomes free, the job in its queue that best fits the rule's criteria is processed. The authors used the same set of rules from [1] with the addition of the SPT rule. These rules are explained next.

1. FIFO (First In First Out): selects the jobs in the order of their arrival times; very simple and robust;
2. SIRO (Service In Random Order): selects the jobs randomly, which represents the situation where the workflow cases are presented to their potential executors who choose which item they will work on based on personal criteria;
3. EDD (Earliest Due Date): selects the job with the nearest due date;
4. SPT (Shortest Processing Time): selects the job with the smallest processing time p_{ij} in the current machine;

The SPT was the only rule to be used with the guess and solve because it schedules the jobs based on their processing times, which are only available through the *guesser*. In this paper, "SPT-PURE" refers to the SPT rule used with a 0%-error *guesser*. SPT-10, SPT-20 and SPT-30 refer to SPT used with a 10%-, a 20%-, and a 30%-error-*guesser*, respectively.

5 Finding a Feasible Approximation

The first step to find a feasible approximation to the mean percentage of late jobs was to study the behavior of the selected rules according to the mean percentage of late jobs achieved by each of them in the proposed scenario. There are two main variables to be studied: the utilization rate (U) and the allowance factor (A). Greater utilization rates lead to a higher percentage of late jobs, while greater allowances give more time for the jobs to be completed and hence can decrease the late jobs.

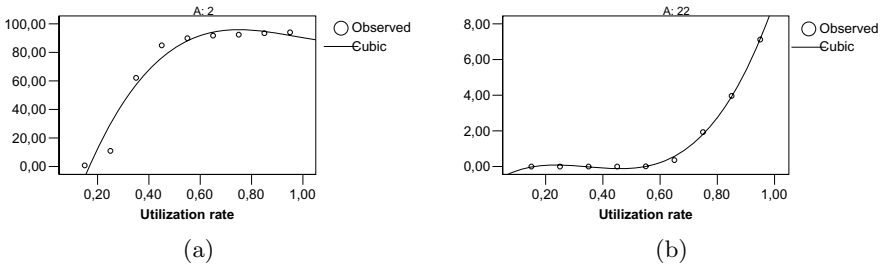


Fig. 2. Observed data and regression result for FIFO with $A = 2$ and $A = 22$

For each allowance factor considered in this paper, all utilization scenarios were simulated with 100 repetitions of 100 jobs for each pair allowance factor/utilization rate. Since there are many graphics and analysis to be shown, we will concentrate on the analysis of FIFO. The procedures adopted for this rule are repeated with the other rules.

The mean values achieved by FIFO for all utilization rates and allowance factors of 2 and 22 are shown in Fig. 2-(a) and Fig. 2-(b), respectively, as dots in the graphics. The Y-axis of the figures represent the mean percentage of late jobs achieved by FIFO for the utilization rates represented in X-axis. A statistical regression to estimate a function that represented the observed values was made. The regression showed that the cubic function was the best representative for the data. The cubic curve can be described mathematically by the expression $y(x) = c_0 + c_1x + c_2x^2 + c_3x^3$, where $c_0, c_1, c_2,$ and c_3 are called the “coefficients” of the equation. The job of the regression is to find such coefficients so that the function adjusts to the observed data. The regression procedure was then repeated for all curves of all allowance factors with the cubic function. As examples, the adjusted curves for the observed data in Fig. 2 are also plotted on the graphics. After the regression, a matrix of coefficients was available for each rule for all values of A . These coefficients can be found in appendix A.

The next step was to find a way to predict the coefficients of the functions for an arbitrary value of allowance factor. The function that predicted the mean percentage of late jobs would then be a two-variable function of the form $y(U, A) = c_0(A) + c_1(A)U + c_2(A)U^2 + c_3(A)U^3$, where the coefficients would be themselves functions of A . Many approaches were tested but the best one was to keep the coefficients in a matrix where one could retrieve them whenever needed, and when the coefficient $c_i(a)$ for an allowance a not present in the matrix was needed, one could infer it with a linear interpolation of the coefficients for the “nearest” allowance factors directly above and below a (a^+ and a^- , respectively). This operation can be described by $c_i(a) = c_i(a^-) + ((c_i(a^+) - c_i(a^-))(a - a^-))$.

6 Model Evaluation

The model was evaluated against a set of data that was not present in the set used to generate the model. For that, new simulations with new values of A

Table 1. Sum of errors for all rules with the new allowance factors

A	FIFO	SIRO	EDD	SPT-PURE	SPT-10	SPT-20	SPT-30
1.5	35.13	27.26	36.15	62.66	61.17	60.66	51.95
3.5	60.41	38.75	60.41	17.85	18.54	18.10	16.54
5.5	48.75	28.74	48.60	14.34	14.29	14.26	14.05
7.5	30.60	21.16	31.77	14.47	14.78	13.72	15.92
9.5	19.80	16.80	19.95	11.85	12.68	12.58	12.94
11.5	12.29	12.35	11.89	9.88	9.14	10.32	12.01
13.5	10.40	5.70	13.04	8.14	7.32	7.11	9.82
15.5	9.35	7.03	11.02	5.71	5.93	5.39	6.39
17.5	6.56	4.83	5.10	8.72	7.60	12.72	7.16
19.5	4.66	4.36	3.53	2.70	4.77	2.83	3.41
21.5	1.39	3.65	1.12	3.59	4.04	4.06	4.16

were performed and the prediction model was implemented in a C program. The results predicted by the program were then compared those obtained by simulation. This comparison is summarized in Tab. 1.

Table 1 shows the sum of the absolute values of the differences between the real and predicted values for the mean percentage of late jobs. Each cell of Tab. 1 represents the sum of all prediction errors of all utilization rates for each new value of A , that is, $\sum_{\text{for all } U} |\text{simul}(U, A) - \text{pred}(U, A)|$. All values are in percentual points. Notice that as the allowance gets larger, the errors decrease. Also, the errors for these higher allowances are relatively low compared to the amount of uncertainty embedded in the system. A deeper study showed that they concentrate on the region where $0.25 \leq U \leq 0.55$, being lower when $U > 0.65$. Since workflows are expected to face utilization rates higher than 0.65, the model shows itself feasible to predict the mean percentage of late jobs for the analyzed rules. In a broader comparison, the mean error for each allowance factor can be measured by dividing the numbers in Tab. 1 by the number of utilization rates analyzed (9). For example, for SPT-PURE with an allowance factor of 11.5, the mean error is 1.097 percentual points up or down, a quite feasible value.

7 Conclusion and Future Work

The modeling of the performance of the scheduling techniques in workflow systems is a challenging task. The uncertainty embedded in these systems makes it harder to predict their behavior. But it also represents a possibility of improvement to the business processes enacted by the WFMSs. The results in this work are the first step to this modeling. They present a way to predict the mean percentage of late jobs that a system will have when using one of the rules studied here. This is an important feature to help systems administrators on evaluating the different types of benefits (and drawbacks) each technique can bring.

Practical analysis shows that SPT is the best among the chosen rules to minimize the percentage of late cases in the studied scenario. The mean values

achieved by this rule are lower than the values achieved by the other rules in almost all cases. EDD becomes a better choice only when the allowance is greater than 16. This kind of information is also relevant for it helps the choice of the best scheduling technique.

This research, however, is ongoing and there is still progress to be made in the area. This includes the analysis of different configuration of sequences of activities (like different time intervals for the activities), the study of the other basic workflow patterns (the AND-split, OR-split, and loops, as defined by [11]), the inclusion of other scheduling techniques (like other dispatching rules and genetic algorithms, among others), and the conception of a complete model to workflow systems.

References

1. Tramontina, G.B., Wainer, J., Ellis, C.: Applying scheduling techniques to minimize the number of late jobs in workflow systems. In: Proceedings of the 2004 ACM Symposium on Applied Computing, New York, NY, USA, ACM Press (2004) 1396–1403
2. Bettini, C., Wang, X.S., Jajodia, S.: Temporal reasoning in workflow systems. *Distributed and Parallel Databases* **11** (2002) 269–306
3. Eder, J., Panagos, E., Pozewaunig, H., Rabinovich, M.: Time management in workflow systems. In: Proceedings of the BIS'99 3rd International Conference on Business Information Systems, Springer-Verlag (1999) 286–300
4. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Proceedings of the 11th International Conference on Advanced Information Systems Engineering, Springer-Verlag (1999)
5. Laguna, M., Marklund, J.: *Business Process Modeling, Simulation and Design*. Prentice Hall (2004)
6. Senkul, P., Kifer, M., Toroslu, I.H.: A logical framework for scheduling workflows under resource allocation constraints. In: Proceedings of the 28th International Conference on Very Large Data Bases. (2002) 694–705
7. Senkul, P., Toroslu, I.: An architecture for workflow scheduling under resource allocation constraints. *Information Systems* **30** (2005) 399–422
8. Zhao, J.L., Stohr, E.A.: Temporal workflow management in a claim handling system. In: Proceedings of the international joint conference on Work activities coordination and collaboration, New York, NY, USA, ACM Press (1999) 187–195
9. Pinedo, M.: *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, Englewood Cliffs, New Jersey (1995)
10. Bierwirth, C., Mattfeld, D.C.: Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation* **7** (1999) 1–17
11. van der Aalst, W., van Hee, K.: *Workflow Management: Models, Methods and Systems*. The MIT Press - Massachusetts Institute of Technology (2002)

A Table of Coefficients for the Rules

This appendix shows the tables of the coefficients found in the regression procedure for the rules. The decimal precision of these numbers was reduced so the tables could fit in the paper. But the real calculations are done with all possible precision.

Table 2. Coefficients for FIFO, SIRO, and EDD

A	FIFO				SIRO				EDD			
A	c ₀	c ₁	c ₂	c ₃	c ₀	c ₁	c ₂	c ₃	c ₀	c ₁	c ₂	c ₃
1	-70.92	750.84	-1077.07	499.84	-65.77	675.79	-926.86	415.49	-70.61	749.88	-1076.35	499.87
2	-85.15	612.10	-666.89	230.25	-61.16	432.13	-389.77	100.08	-84.79	603.47	-646.64	217.63
3	-54.81	329.41	-120.54	-74.78	-36.97	212.69	1.75	-106.16	-53.57	317.76	-99.24	-86.25
4	-32.18	136.04	223.94	-254.92	-22.72	92.13	198.56	-202.63	-30.70	124.12	244.12	-265.42
5	-13.51	-14.24	483.21	-389.24	-11.48	5.58	332.80	-266.79	-11.57	-29.35	507.78	-401.54
6	2.07	-129.87	659.06	-468.69	-4.35	-45.98	399.42	-292.95	4.55	-149.88	694.67	-488.46
7	13.87	-214.43	775.77	-516.53	2.76	-97.43	467.62	-319.36	15.72	-228.05	795.77	-526.16
8	17.95	-233.88	765.00	-492.95	4.40	-102.59	447.00	-297.31	19.88	-247.70	783.54	-500.75
9	24.93	-275.91	797.75	-493.44	8.01	-125.74	468.30	-301.82	26.81	-288.21	810.09	-496.36
10	25.99	-272.86	748.38	-450.66	9.28	-131.48	457.36	-288.64	27.09	-278.51	747.53	-446.38
12	24.59	-235.87	585.14	-327.04	12.70	-146.52	437.95	-261.97	25.59	-238.77	574.13	-315.24
13	22.77	-211.89	506.80	-273.00	11.71	-134.27	399.64	-237.01	22.79	-206.22	476.41	-249.27
14	19.96	-176.85	396.59	-197.15	14.36	-149.13	404.62	-232.21	19.14	-165.07	356.67	-169.50
15	16.14	-141.61	312.11	-148.99	11.85	-125.94	349.31	-200.17	14.24	-120.00	248.27	-105.62
16	10.91	-92.85	192.26	-75.80	12.22	-124.00	327.27	-183.05	7.89	-61.34	105.94	-18.93
17	8.15	-65.90	123.90	-35.26	11.45	-112.78	288.41	-156.49	3.02	-18.22	7.17	37.10
18	2.90	-19.18	15.33	28.76	9.34	-91.08	228.77	-117.72	-3.08	33.91	-107.80	101.61
19	1.09	-4.19	-14.70	40.69	7.59	-74.96	190.66	-97.19	-4.09	40.54	-113.69	94.34
20	-0.13	5.61	-32.12	43.37	7.67	-72.42	174.70	-87.61	-3.01	28.26	-73.87	56.60
21	-0.11	4.19	-23.77	31.85	6.84	-63.47	150.49	-73.86	-1.31	12.04	-30.70	22.87
22	-1.09	11.98	-38.38	36.73	5.78	-53.12	124.10	-59.43	-0.43	3.90	-9.84	7.25

Table 3. Coefficients for SPT-PURE, SPT-10, SPT-20, and SPT-30

A	SPT-PURE				SPT-10				SPT-20				SPT-30			
A	c ₀	c ₁	c ₂	c ₃	c ₀	c ₁	c ₂	c ₃	c ₀	c ₁	c ₂	c ₃	c ₀	c ₁	c ₂	c ₃
1	-53.3	571.1	-767.6	342.2	-51.5	553.9	-732.7	322.3	-52.2	559.0	-741.3	327.2	-47.8	537.9	-763.7	360.9
2	-27.0	174.5	-38.4	-40.8	-27.0	175.4	-38.5	-41.5	-26.5	173.6	-36.7	-41.3	-27.0	184.1	-72.8	-15.6
3	-12.7	42.7	195.1	-164.4	-11.7	36.6	207.8	-171.6	-12.0	40.3	197.5	-164.0	-13.8	57.7	163.1	-144.7
4	-7.8	1.9	254.5	-190.7	-6.8	-6.1	270.9	-199.0	-8.2	5.2	249.6	-187.2	-8.6	8.6	244.2	-184.4
5	-2.8	-37.8	317.0	-221.3	-2.8	-37.5	317.2	-221.5	-3.5	-32.5	310.5	-218.7	-5.1	-19.9	290.6	-208.8
6	0.2	-62.1	353.1	-238.3	0.6	-65.2	360.2	-242.2	0.1	-61.4	354.9	-240.0	-1.5	-47.7	332.3	-228.1
7	2.8	-80.1	371.6	-243.1	3.1	-82.4	376.1	-245.2	2.6	-78.4	369.9	-241.8	1.1	-68.7	360.9	-240.0
8	3.1	-78.9	357.0	-231.1	3.0	-78.5	357.4	-231.6	2.5	-74.1	348.9	-226.2	1.2	-66.7	345.7	-228.3
9	6.2	-100.9	386.8	-244.6	5.6	-97.2	384.0	-244.6	5.3	-93.7	373.7	-236.8	3.9	-84.9	368.5	-237.5
10	5.7	-95.3	366.6	-231.1	5.4	-93.6	363.9	-229.6	5.0	-90.6	359.8	-228.0	4.1	-86.5	362.6	-232.2
11	5.8	-93.6	349.7	-217.5	5.3	-90.0	344.0	-215.1	5.8	-94.8	357.9	-224.7	4.4	-86.2	351.8	-223.8
12	7.0	-98.5	342.1	-209.1	6.5	-95.3	338.6	-208.4	6.6	-96.4	342.0	-210.5	6.0	-96.3	359.9	-226.8
13	7.0	-102.6	341.6	-208.1	6.8	-95.4	330.2	-203.0	6.7	-94.4	329.9	-202.6	6.6	-99.0	356.9	-224.0
14	9.4	-110.8	341.0	-203.2	8.5	-104.7	332.7	-200.2	8.6	-106.7	337.6	-203.4	8.3	-108.4	361.6	-222.9
15	6.5	-85.7	283.3	-170.8	5.6	-79.0	275.1	-168.6	6.2	-83.9	286.4	-174.7	6.9	-95.8	331.8	-208.0
16	7.4	-88.7	273.7	-162.7	6.8	-83.6	265.4	-158.3	6.3	-80.7	262.6	-157.1	7.2	-93.3	305.7	-186.6
17	6.8	-80.2	243.2	-142.9	6.0	-74.4	235.6	-139.6	6.3	-80.7	262.6	-157.1	7.5	-92.1	289.1	-173.8
18	6.8	-80.2	243.2	-142.9	6.3	-56.1	183.9	-106.7	6.3	-78.5	247.5	-147.5	5.7	-73.6	238.1	-141.2
19	3.2	-44.2	147.7	-85.0	2.9	-43.3	152.8	-90.5	3.0	-44.3	156.2	-91.8	4.8	-63.9	211.4	-126.5
20	3.5	-43.0	133.7	-76.5	3.2	-41.7	136.0	-78.7	3.6	-45.8	147.0	-85.52	5.1	-61.6	190.1	-111.2
21	3.2	-37.1	111.5	-62.7	3.2	-39.0	121.9	-70.2	2.9	-36.5	117.0	-66.7	4.8	-56.6	171.8	-100.4
22	1.9	-25.3	80.6	-45.0	1.7	-25.4	87.8	-50.6	2.2	-29.8	97.8	-56.3	2.7	-35.7	117.0	-66.2

Collective Knowledge Recall: Benefits and Drawbacks

Naiana Carminatti, Marcos R.S. Borges, and José Orlando Gomes

Graduate Program in Informatics – NCE & IM
Federal University of Rio de Janeiro
naiana@posgrad.nce.ufrj.br,
{mborges, joseorlando}@nce.ufrj.br

Abstract. Organizations frequently need to recall past events that, for some reason, were not adequately documented when they occurred. The successful reconstitution of past events depends on several variables, such as how long ago the event occurred, and whether key people are still in the organization. It also depends on the supporting process. This paper examines three knowledge recall methods and compares them in a controlled experiment. A group storytelling approach is used in two of the methods, one of which is supported by technology. The results obtained favor the group approach, but the advantages of technology support are not conclusive. The paper also evaluates the benefits and the drawbacks of using a supporting technology.

1 Introduction

Knowledge is the most valuable asset in an organization. The appropriate management of this knowledge can make all the difference to some organizations. Organizations frequently need to recall past events that, for some reason, were not adequately documented when they occurred. To recall the relevant knowledge, organizations must rely on the people who witnessed the events or played a role in them. However, this is not an easy task. Incomplete information caused by lapses in memory and the lack of key facts are commonplace in the retrieval process.

The reconstitution of past events can be considered to be a transformation of tacit knowledge into formal knowledge. Its success depends on several variables, such as how long ago the event occurred, and whether key people are still in the organization. The more people there are to contribute, the higher the likelihood of completeness and accuracy. On the other hand, the more people there are, the greater the potential for controversy.

Telling stories is a natural way of transmitting tacit knowledge among individuals, groups, and organizations. When a story is told, the author's intention is to transmit knowledge to the listener. Stories are great vehicles for wrapping together many elements of knowledge such as: explicit and tacit knowledge, information and emotion, the core and the context [1]. Stories are a very powerful way to represent complex, multi-dimensional concepts. While a certain amount of knowledge can be reflected as information, stories hold the key to unlocking the vital knowledge, which remains beyond the reach of codified information [2].

Several approaches exist in the literature for the collective reconstruction of stories [3, 4, 5, 6, 7]. Some are based on interviews performed by a facilitator. Others use group dynamics in order to benefit from the group synergy, although there is some controversy as to whether groups perform better than individuals due to the “process losses” inherent in face-to-face settings [8]. Researchers have recently begun to examine the effects of computer-mediated communication (CMC) on group dynamics and have concluded that process losses can be overcome to some extent [9].

This paper extends prior research by analyzing, through experiments, the beneficial effects that group work can provide to knowledge recall. We compare the three main methods for knowledge recall: individual interviews, group dynamics in face-to-face settings, and group dynamics supported by a CMC tool. For the second and third method, we employ a technique that was developed to support the collective reconstruction of past events, which is known as group storytelling [10, 11, 12, 13]. This study builds from the hypothesis that groups that use the group storytelling approach perform better than groups that use individual interviews. We also examine the influence of a CMC tool on the group storytelling approach.

To perform the comparison we need a real situation experienced by a group of individuals who agree to serve as storytellers. In our experiment, the stories were picked from commercial motion pictures and were unknown by all the participants. Each film was divided into parts, which were selectively shown to the participants so that nobody had the entire view of the story. The task was to rebuild the story with their partial knowledge of the events, using any of the three methods for knowledge recall.

Although the experiment made use of fictitious stories, the results can be generalized to real stories because the development is very similar. People have a partial view of the events and they will only be able to reconstruct the entire story by grouping their pieces together. If our hypothesis holds, we believe the group storytelling approach can be used to recall past decisions and project stories in organizations [12].

The paper is divided as follows: Section 2 reviews the advantages and the drawbacks of collective knowledge recall. Section 3 describes the group storytelling approach, and Section 4 shows the supporting technology. Section 5 describes the planning and the implementation of the experiment, and Section 6 discusses the results. Section 7 concludes the paper.

2 Collective Knowledge Recall

Knowledge is the most valuable asset in an organization [14]. The appropriate management of this knowledge can make all the difference to some organizations [15]. The importance of the knowledge component has motivated companies to develop practices to facilitate its administration. As a result, Knowledge Management has been adopted in wide scale, supporting the definition of procedures, practices and technological tools that aim at capturing, storing and disseminating the knowledge in the organization.

Knowledge exists in both the mind of employees and in documents. Many organizations assign high priority to documentation but it is not correct to say that the most important type of knowledge is stored in documents [16]. The experience of the

organization members, their ideas and decisions are also part of the organization knowledge. Nonaka and Takeuchi define these elements as *tacit knowledge* [17]. It consists of technical abilities: mental models, faiths and ingrained perspectives not subjected to the easy manifestation. It is opposite to the *explicit knowledge*, which is simple to disseminate and share.

Both in tacit and in explicit knowledge, some sort of recall process is required when access to this knowledge is required. When the knowledge is tacit (i.e., it is in the mind of the employees), the knowledge recall process is complex and will depend on a collaborative attitude from the knowledge holders. One of the main challenges is to capture and to maintain the tacit knowledge because it is not logical and strictly documented. For tacit knowledge to be communicated it must be converted into elements that anyone can understand [16]. One possible approach is the transformation of tacit into explicit knowledge. This process is called *externalization*. In spite of the benefits, the process of converting tacit into explicit knowledge is expensive and complicated.

With today's level of specialization, knowledge rarely resides in a single mind. It is only when it is collected together from various sources that it can make any sense. In a different context, globalization has been forcing companies to maintain a presence in several parts of the world and to cooperate with each other. Knowledge in this case is not only distributed among several individuals, but these individuals might also be geographically distributed. More importantly, they may even be members of different organizations. As a result, most activities require some sort of teamwork, and knowledge should be captured and dealt with collectively. Although this imposes additional difficulty on individual work, it also provides some benefits.

When a group of people is involved in a knowledge recall task, they can work in at least two different modes. They can work individually and group their results as they progress, or they can work together with a single group result throughout the process. The knowledge generated by a collective process is usually richer than that generated by the same individuals working separately [18]. A collective process discloses different points of view, is stimulating and dynamic, and creates synergy among participants. To sum up, a collective process is expected to create more knowledge than the sum of individual processes. The probability of completeness and precision is greater if more people contribute. On the other hand, there is also more potential for controversy if there is a greater number of contributions. This is particularly true when group members have to recall facts from the past in which they had a partial role; for example, the documentation of a finished project.

The knowledge generated at the end of a collective recall process results from the combination of the skills acquired by each participant during the task execution. The knowledge can contain many more valuable details if more than one person participates in its creation since an activity normally involves more than one individual. However, just as any other group work, the collective capture of knowledge presents some difficulties that do not exist in individual work. In general, these difficulties have social or cultural causes, such as resistance to sharing knowledge, relationship difficulties, conflicts and quarrels, constraints, etc.

To assess the participant's understanding of the result generated by collective knowledge recall, we can make use of Bloom's taxonomy [19]. This taxonomy has been widely used in educational environments to help to evaluate the apprentice's

understanding of the concept being taught. We believe that with some adaptation, the taxonomy can also be used to represent the level of understanding of the knowledge recall process. The taxonomy is reproduced in the first column of Table 1. We added an extra row to represent the group synergy. In the second column we evaluated the collective knowledge construction when using a group storytelling approach.

Table 1. Bloom's Taxonomy applied to collective knowledge construction using a group storytelling approach

Bloom's Taxonomy of Cognitive Domains	Instantiation of the Taxonomy to Group Storytelling
KNOWLEDGE Recall of previous knowledge	Be able to recall the relevant facts from what they watched or witnessed
COMPREHENSION Understanding knowledge. Explaining and translating one form into another	Be able to represent their stories in the template provided. Translate from one form to another (images to words)
APPLICATION The use of knowledge to identify and solve new problems	Use stories to reach conclusions or to identify gaps between stories or misunderstandings of presented stories
ANALYSIS The identification of others' knowledge and the relationship between different pieces of knowledge	Filter the relevant part of other stories and establish a network of stories to form a single story or a group of stories
SYNTHESIS Rearranging previous knowledge into new patterns or structures	Rearrange or rewrite their own stories in response to others' questions or remarks
EVALUATION Determining how useful or valuable the knowledge is for a given purpose (suitability)	Evaluate stories and decide the value of a story to the main stories
GROUP SYNERGY Assessing and combining knowledge from others to create new knowledge	Recall new stories after being presented other members' stories. Filling gaps between stories

3 Group Storytelling

"A story can be defined as a narration of a chain of events that is told or written in prose or verse. The term narration comes from the Latin *narrere*, meaning to pass on knowledge" [11]. Storytelling can be considered to be as old as the human being, even considering that at the beginning of the human race, stories were of the most rudimentary form, such as the Rosetta Stone. The Egyptians registered their stories in illustrations. Indians have used oral storytelling as the main technique for knowledge propagation through the generations. The invention of the printing press made story dissemination widely available to many people at once, as copying material became much simpler [12].

Before a story can serve as knowledge transfer, it must be constructed or assembled. The assembly of a real story is the process of recalling knowledge from past events that have occurred. This can be an individual or a group task depending on whether the story fragments are remembered by one or more individuals. In the latter case, members of a group contribute to creating a story collectively, synchronously or asynchronously, in the same place or in a different place. This technique is called group storytelling.

Group storytelling is more appropriate than individual storytelling in contexts where there are several people involved in the execution of a task. A story that is told from several perspectives may bring out interpersonal and interaction complexities such as negotiation, communication, and coordination needs, which would not be manifested in a story told from a single perspective [18].

The idea of using a group storytelling mechanism is not a simple one. It depends on the existence of a Knowledge Management culture as well as that of a group culture. It also requires technological support. In Table 1, we present the instantiation of Bloom's taxonomy [19] to collective knowledge construction using group storytelling. This will help to understand the goals that can possibly be achieved with the group storytelling approach. We have added group synergy as one of the additional criteria because we wanted to compare the method with individual interviews.

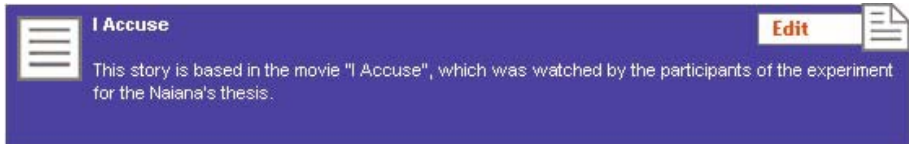
There are several ways of registering a story. Among them are texts, photos, audio, video, or a combination of them. "Video and audio help to bring ideas to life by adding more non-text clues, including body language, graphic illustrations, and sound effects or music. This helps activate many more parts of the brain than text alone, increasing most people's ability to pay attention and to recall what they have heard. It also draws in people who are not as comfortable in purely text-based communications" [2].

Instead of using simple text media, it is better to use various technologies (like multimedia). However, these technologies require richer production as well as skills that people generally do not have, like the definition of a good script, a good voice, a good characterization, and high quality pictures. [10].

The storytelling approach has been used in many works. Schäfer, Valle and Prinz [13] applied the group storytelling to create team awareness. The story in this case was a collection of annotated photos. They developed the *PhotoStory* application to generate and maintain a story through a sequel of pictures with subtitles. Participants should first set up a storyline. Then, they feed the story elements with annotated digital pictures. In their work, however, they aimed to create a story, which is different from the focus of our paper that aims at the recalling process.

4 The TellStory Groupware

The growth of cooperative work in organizations has stimulated the development of groupware, a type of support tool used by teams. It facilitates several activities that are traditionally performed in group work, such as coordination, communication, awareness and level of collaboration. In their research, Perret, Borges, and Santoro [12] argue that a consolidated system that gives support to the collaborative construction of stories does not yet exist in the literature.




I Accuse Edit

This story is based in the movie "I Accuse", which was watched by the participants of the experiment for the Naiana's thesis.

When you type title and description of a event and click on **modify event**, you are going to modify this event into database.

Although it is not required, filling one or more fields in Complementary Information box is essential to story's construction.

Edit the event:

title: 

description:

Complementary Information

Place:

Describe the place and scenario where this event occurred.

Fig. 1. Inserting an event. The story is made up of events that are inserted by each participant. Besides the event title and description, the participant can also add some descriptors such as place, characters involved, etc.

There are some Computer-Supported Collaborative Learning environments, which stimulate collective knowledge building, and there are also some Collaborative Authoring Tools. Based on some of the characteristics of these systems and the analysis of narratives and journalistic texts, Perret [20] has developed a groupware application called TellStory, which supports the group storytelling method. It is a web application, which offers text media and is implemented on the Zope platform [21].

The tool allows a group to tell a story through the contributions of each one of the members. Any registered member of TellStory can create a story and invite new participants. An individual can participate in the story by performing one of the following roles: (i) moderator: the creator of the story and the person responsible for the coordination of the actions inside of the story; (ii) user: contributor to the story; (iii) teller: the person that will write the final text; (iv) reviewer: the person who endorses the story built by the teller; and, (v) commentator: the person responsible for the identification of tacit knowledge externalization of the story. More than one person can assume the same role, and each role can be assumed by several people. The user role is common to all registered members.

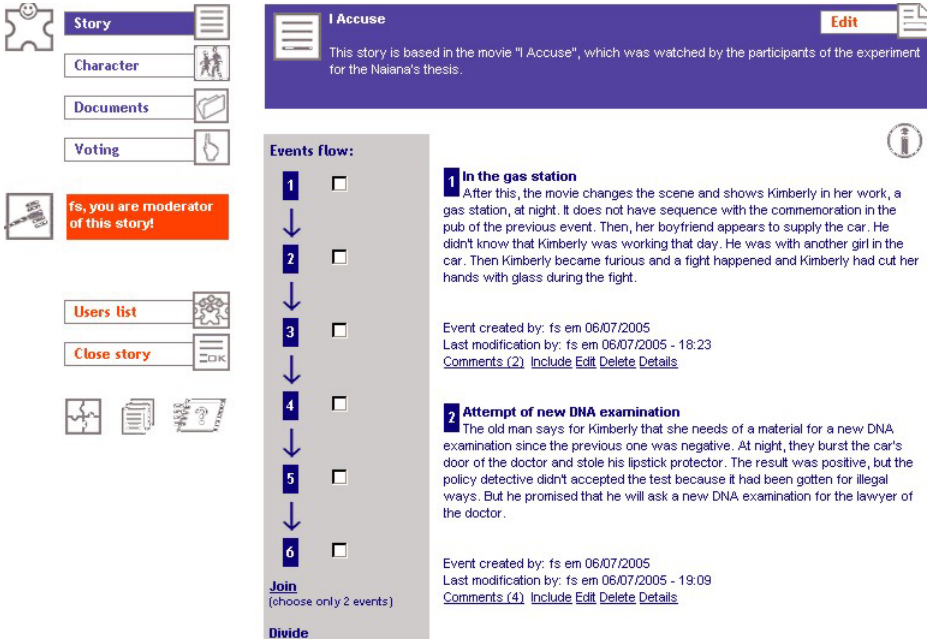


Fig. 2. Flow of a story. The participants decide on the chronological position of the event in relation to those already in the story when inserting the event. They can also change the position later on.

According to Holloway, a story is a sequence of events that are tied to each other by a full conductive thread of meaning, built by a causality relationship between a fact and its successor [22]. TellStory takes advantage of that definition to facilitate the construction of the story by a group. Each user can insert an event which he/she remembers (Figure 1), that is, a fact that happened.

The possible actions throughout the construction of the story are: inclusion, edition, exclusion, union, and fragmentation of events. The union of events occurs when two events can be considered as a single one; the fragmentation of an event occurs when an event is divided into two. These can be performed whenever necessary. The criteria that indicate whether a fact is an event, a sub-event, or a collection of events do not need to be explicitly defined by the participants. This makes the tool a flexible environment, where people can express themselves freely.

The events can be introduced in random order because they can be treated later, during coordination activities. At this point, they organize the events, discuss them, and vote to decide which order will be chosen. Figure 2 shows the flow of events: to the right, the events appear in a column in the corresponding order. If a modification of the events in the story occurs, the numbering of the events changes.

Once the users have input the events, they can discuss them with each other through comments in a forum format. They can eventually make decisions on certain subjects through voting that is organized by the moderator. One example of subject discussion is event truthfulness. If there is no consensus about the existence of a certain fact, the tool allows the story to have two versions, one that considers the

hypothesis of the event to have happened and the other that considers it not to have happened. However, the duplicity of versions should only be used if there is not a majority consensus on decisions related to a subject.

One of the most important benefits of TellStory is that it offers the possibility of using a template to address the elaboration of the story through the typical features of a narrative structure. For example, the template shows the users that an event should always have a cause and effect relationship with its successor or predecessor, according to the causality principle. The template also has a module in which the users can define and configure the characters, an activity which greatly aids externalization. The template works as a guide for the tellers, stimulating their memories and helping them to better structure their thoughts. When the group understands and the story already provides a sufficient flow of events, the moderator can conclude the task. At this point, the teller gathers the events and writes a final text based on the sequence. The reviewer corrects casual mistakes and has the authority to make any changes to arrange the logic of the final text. Finally, the commentator searches for tacit elements that can be identified in the story, which are registered in a module that is included in the final text.

5 Planning the Experiment

The goal of the experiment was to compare different alternatives for knowledge recall and to evaluate the benefits of a supporting technology for the group storytelling technique. The results should provide a preliminary evaluation of issues that we judged important to the design of a knowledge recall procedure. To achieve this, we compared the results obtained with each approach. The first insight was whether the group storytelling approach generates more commitment from the participants than the interview approach. We assigned some questions in the questionnaire related to the participants' satisfaction with the dynamics of the interaction. We compared the answers from the same group and also the answers from different groups using different approaches.

The second insight was the results obtained by each approach in terms of completeness and detailing level of the stories generated. We looked into the contents of each story and measured how far or close they were to the real story. We also looked into the stories generated from the same movie and checked for the differences. We were particularly interested in examining the knowledge produced by the combination of individual knowledge from different participants. In other words, we intend to assess the combination phase in Nonaka's knowledge transfer spiral [17].

Perret used the need for recalling a documentation of a complex organizational process that took place in an organization in Rio de Janeiro to test the TellStory tool [20]. This experiment, however, would not serve our purpose; we needed the same story to be recalled using different techniques and by different groups of people.

We opted to use two story recall techniques: one based on interviews and another based on the group storytelling approach. The first one is very common in organizations and consists of an interviewer asking questions to an interlocutor and compiling the answers to generate the story. In the group storytelling approach, members of a group contribute to the recall of a story collectively. The group

storytelling technique was carried out with and without a supporting technology. The supporting technology adopted was the web tool, TellStory [20], presented in the previous section.

Two groups of volunteers were prepared, each of which had four students from Computer Science and one professor acting as a moderator. The moderator is the person responsible for coordinating the techniques of stories recall. He did not watch the films and did not know their stories.

The task assigned to each group was the recall of the story told in the feature films (which had not been seen by the participants) using partial knowledge of their events. Each film was divided into parts ranging from five to twenty minutes, which were selectively shown to the participants so that nobody had the entire view of the story.

These parts had been previously selected by the coordinator of the experiment in order to create as much discussion as possible. Figure 3 shows the parts of the film watched by each one of the participants.

The experiment was divided into four parts. In Parts 1 and 2, groups A and B watched the same pieces of movie 1, but used different techniques to recall the story of the movie (group A used interviews; group B used the group storytelling technique without the tool). In Parts 3 and 4, the same groups watched the same pieces of another film. Both groups used the group storytelling technique, but group B used the TellStory tool, which they had been trained to use before the experiment. Table 2 presents a summary of the experiment.

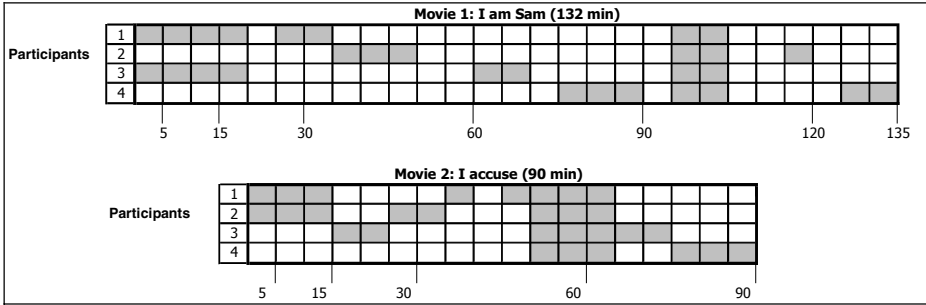


Fig. 3. Parts of the movies watched by participants. Some parts were watched by all, some by none; and some by a sub-group of the participants.

Table 2. Parts of the Experiment – Planning each evaluation

	View movie 1	View movie 2	Interviews	Group storytelling	Group storytelling + TellStory	
Part 1	X		X			Group A
Part 2	X			X		Group B
Part 3		X		X		Group A
Part 4		X			X	Group B

Each part of the experiment was divided into the following activities:

- a. Watch the corresponding pieces of the film without talking to other participants about it.
- b. Participate in the movie's story recall using one of the techniques.
- c. Elaborate the final writing of the movie's story.
- d. Answer the experiment evaluation questionnaire.

All the participants, including the facilitator, answered the questionnaire. Its objective was to generate a qualitative analysis of the differences between the techniques, to evaluate the benefits of a supporting technology, and to identify the difficulties that occurred during the experiment.

6 Findings

In this section, we present the results of the experiment based on the analysis of stories generated at each part, the observations made during the application of the techniques, and the answers of the evaluation questionnaires.

The stories generated by the groups were evaluated using the following criteria:

- **Completeness:** how complete the story generated by each group was; i.e., whether or not the group had covered all the important facts.
- **Level of detail:** if the story was presented as a summary of facts, or whether it had details.
- **The structure and the persistence of the knowledge generated:** if the knowledge about the story was kept at the tacit level or if it was formalized.
- **The interconnection between story fragments:** if the story was composed of loose fragments or if these fragments were well connected.

In addition, we took into account the geographic distribution of the participants and the possibility of asynchronous work. In the questionnaire, we asked the participants how difficult they found the use of each technique.

6.1 Interviews Versus Group Storytelling

The group storytelling technique generated better results than the interviews because it created a synergy among the participants, stimulating the contributions and discussions of the group. The contribution of each participant had a positive effect on the others, by making them remember relevant facts, recall forgotten information, argue conflicting points of view, and complement the story.

The story that was generated using the group storytelling technique presented greater completeness and a higher level of detail. Besides describing the main scenes, many details were reported. In the technique based on interviews, the story that was generated had several problems: lack of some of the main scenes, a low level of detail, several assumptions made by the facilitator, and several open questions that could not be answered by the facilitator. In the technique based on interviews, the moderator is responsible for putting together the story's fragments. This may distort the story because the moderator did not watch the film and s/he does not have enough knowledge to make her/his own deductions.

One of the disadvantages of both the techniques is that the participants need to get together in the same place at the same time. This is undesirable particularly in companies that are distributed geographically. In both cases, the use of verbal communication caused parts to be lost. Even when written communication was used, some participants filtered the information before reporting their views. This can lead to the omission of relevant facts in the final text. This reinforces the importance of formally registering all interactions.

According to the questionnaire's answers, the participants encountered difficulties when applying the techniques. At the interviews, some participants felt constrained; in the group storytelling technique, some participants felt uncomfortable in the presence of other people.

In both the group storytelling and interview techniques, incomplete events occurred, due to lapses in memory and to the lack of necessary information. However, they were more frequent in the interviews, when the participants did not interact in a group.

Another issue is the importance of the facilitator. Both techniques greatly depend on the facilitator's performance. However, in real situations, facilitators are well-trained people who usually follow a set of guidelines, including a strategy for extracting the best from participants. For example, although we had adopted the same technique in Parts 2 and 3 of the experiment, the strategy adopted was different, resulting in stories with very different characteristics and qualities.

6.2 Group Storytelling with the TellStory

Many problems were solved when the group storytelling technique was used with the support of TellStory. With the TellStory tool, all the contributions were persistent. The participants were able to organize their knowledge access the contributions made by other participants, access the tool at any point from anywhere. To sum up, they followed the story recall dynamics proposed by the tool [20], achieving some of the expected advantages.

Nevertheless, other problems appeared. The TellStory promoted less interaction among the participants and there was little intervention from the moderator due to the limitations of the tool. These limitations negatively affected the contribution of the group. Problems such as lack of awareness mechanisms, lack of coordination tools, and poor communication mechanisms were reported by the participants.

Although it produced important gains in time and energy, the asynchronous interaction did not motivate the participants' commitment. Because they logged on at different times and did not meet, the use of the tool requires a high level of compromise from group members. A synchronous interaction induces participants to reserve a fixed amount of time for the task, while the asynchronous interaction tends to assign low priority to the task. We believe that a mix of the two types of interactions would be more appropriate. In other words, to start with synchronous sessions followed by asynchronous interactions.

The story generated by the TellStory presented the same completeness and level of detail generated by the group storytelling approach without the tool. However, the scenes were not as ordered, which indicates some limitations of the tool.

The results confirmed that group storytelling is a very strong technique for recalling stories. However, when this technique is supported by a tool it needs:

- to be as transparent as possible so that people do not apply filters and feel confident in contributing;
- to provide mechanisms of awareness, coordination, and communication;
- to promote a reliable and motivating environment.

Table 3. Qualitative results obtained by observations from the stories

Aspect	Interviews	Group storytelling	Group storytelling with TellStory
Quality of stories	Organized, but poor and incomplete stories	Richer, more complete better organized stories	Rich, complete, but disordered stories
Interaction among participants	There is no interaction between the participants	Synergy of the participants	Synergy of the participants, but the qualities of the face-to-face interactions are lost
Documentation	Disorganized and not persistent knowledge	Disorganized and not persistent knowledge	Organized and persistent knowledge
Role played by the moderator/facilitator	The moderator is responsible for relating the film segments	The moderator questions or suggests the connection of the film segments	The moderator questions or suggests the connection of the film segments. He/She also monitors the participation in the task.
Group location	The group must be in the same place, at the same time	The group must be in the same place, at the same time	It allows access at any moment from anywhere
Expressiveness of participants	Constraints on interviews	Uncomfortable in the presence of other people	Inhibition about writing their remarks and beliefs

According to the questionnaire's answers, all the participants positively evaluated the dynamic proposed by the tool as being very useful for a story recall. They also reported that a face-to-face interface is richer and more stimulating. When this technique is not used, nonverbal information, such as facial expressions, gestures, voice intonation, and body movements are lost. Thus, the context, the individual perception of the contributions, and the channel of communications are also lost. The lack of direct interaction, a characteristic of the TellStory tool, made some participants less active, reducing their registered contributions. The future re-design of TellStory should provide face-to-face communication besides other media, such as video conferencing, audio, and graphic tools.

It is recommended that the story recall occur as early as possible to avoid the missing of details. On the other hand, the experiment indicated the need for better teamwork support. The story recall should also motivate participants by providing benefits of some kind at the cultural and social level.

In Table 3, we present a summary of the features of each technique based on the observations made during the execution of the techniques, the resulting stories, and the answers to the questionnaires

6.3 Limitations of the Experiments

The current findings have several limitations. Although these preliminary results provide useful insights to the collective knowledge recall, further experiments are necessary to confirm our hypotheses and the first round of findings. It is a consensus that group synergy has a positive effect on both the quality (completeness and accuracy) of the stories and the time spent to generate them. However, we have not confirmed the gains obtained with the groupware tool. This may have to do with the type of interaction, the functionality of the tool, or the set of experiments.

When groups are co-located in the same environment, the main advantage provided by the tool is the automatic documentation it generates. The face-to-face interaction, however, converge to a single story faster than the asynchronous interaction supported by the tool. The geography distribution of participants cannot be considered a constraint of the face-to-face interaction because it can be overcome by videoconferencing support.

The size of the groups can produce some effect on the results. On the experiments the groups were rather small (four people each). Large projects usually have many more participants. We believe this creates additional difficulties to face-to-face interactions, not only on the availability of all participants to get together at the same time, but also on the dynamics of the interaction itself. The asynchronous interaction can show some advantages in this situation.

The previous experience of participants working as a team can have some effects on the results. Teams that have worked together in task-oriented activities are expected to perform better than teams whose members have never worked together. In our experiments this was not an issue because the participants knew each other and had worked as a team in several tasks before the experiment. This situation can be assumed in most organizations. If, however, they work together for the first time, lack of trust can create additional barriers for groupware supported interaction [23].

7 Conclusions

This paper presents an experiment to examine the use of the group storytelling approach for knowledge recall. The group storytelling approach to knowledge recall was compared to the more traditional approach based on interviews performed by a facilitator who is also responsible for the final version of the story. We also compared the group storytelling approach with and without a computational support tool. The preliminary results show a clear advantage of the group storytelling method over the

interview approach. When we compare the group storytelling approach with and without the tool, there are advantages and disadvantages in both modes. Further experiment should be done to confirm these initial findings.

Knowledge recall is an important activity in organizations because many projects and jobs are carried out without any documentation of their procedures or results [24]. Knowledge recall would serve to support the design of future similar activities, trying to avoid mistakes and to repeat successes. In order to provide such support, it is very important to recover the tacit knowledge adopted during these projects. Formal documentation usually leaves out this important type of knowledge.

We have used feature films to simulate stories that are not completely known by any participants. We also have cases where parts of the story have been lost. This is usually the case when the organization cannot count on all participants to recall the knowledge.

The tool still needs some improvement. Some of expected benefits have not been achieved because the tool does not support an appropriate functionality. The experiment was important to generate insights into the requirements of future versions of the tool. Besides implementing the new functionality, we intend to make the tool customizable to be able to adapt to different knowledge recall situations.

One important target of our research is to use this approach for recalling the events that precede accidents that occur in organizations. To do this, the mechanisms of perception, communication, and coordination of the TellStory tool are being improved. Some new features are also being incorporated. The dynamic and the structure of stories proposed by Perret are also being adapted to the context of accidents and emergency situations [25].

Acknowledgements. Marcos R.S. Borges was partially supported by a grant from the “Secretaria de Estado de Educación y Universidades” of the Spanish Government. Naiana A. Carminatti is sponsored by NCE (Master Thesis Scholarship). The authors are grateful to all those who participated in the experiment: Célia Seabra, Débora Knight, Flávia Santoro, Igor Miranda, Maria Lopes, Rafael Gonçalves, Rosa Freitas, Simone Garcia and Viviane Diniz.

References

1. Sole, D.; Wilson, D.G. “Storytelling in Organizations: The power and traps of using stories to share knowledge in organizations”. Retrieved December 8, 2004, from LILA Harvard University Web Site: http://www.providersedge.com/docs/km_articles/Storytelling_in_Organizations.pdf
2. Ruggles, R., “The Role of Stories in Knowledge Management”. Storytelling Foundation. Retrieved December 8, 2004, from LILA Harvard University Web Site: Available at: http://www.providersedge.com/docs/km_articles/The_Role_of_Stories_in_KM.pdf
3. Fröhlich, P., Karandikar, H.: Driving organisational change: Using story to transform work processes at ABB. Knowledge Management, (2002). Retrieved July 7, 2005 from: http://www.nelh.nhs.uk/knowledge_management/km2/storytelling_toolkit.asp

4. Collins, T., Mulholland, P., Bradbury, D., Zdrahal, Z.: Methodology and Tools to Support Storytelling in Cultural Heritage Forums. Proceedings of the 14th International Workshop on Database and Expert Systems Applications, Prague, Czech Republic (2003) 105-109
5. Shen, C., Lesh, N.B., Vernier, F., Forlines, C., Frost, J.: Building and Sharing Digital Group Histories. Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW), New Orleans, Louisiana, USA (2002) 324-333
6. Lawrence, D., Thomas, J.: Social Dynamics of Storytelling: Implications for Story-base Design. Proceedings of the AAI Workshop on Narrative Intelligence, N. Falmouth, MA, USA (1999) 26-29
7. Davenport, G. Smarter Tools for storytelling: are they just around the corner? IEEE Multimedia Vol. 3, No. 1 (1996) 10-14
8. Steiner, I.D.: Group Process and Productivity. Academic Press, San Diego, USA (1972)
9. Kerr, D.S., Murthy, U.S.: Divergent and Convergent Idea Generation in Teams: A Comparison of Computer-Mediated and Face-to-Face Communication. Group Decision and Negotiation Vol. 13, No. 4 (2004) 381-399
10. Valle, C., Prinz, W., Borges, M.R.S.: Generation of Group Storytelling in Post-decision Implementation Process. Proceedings of the 7th International Conference on Computer Supported Cooperative Work in Design, Rio de Janeiro, Brazil (2002) 361-367
11. Valle, C., Raybourn, E.M., Prinz, W., Borges, M.R.S.: Group Storytelling to Support Tacit Knowledge Externalization. Proceedings of the 10th International Conference on Human-Computer Interaction Vol. 4, Crete, Greece (2003) 1218-1222
12. Perret, R., Borges, M.R.S., Santoro, F.M.: Applying Group Storytelling in Knowledge Management. Proceedings of the International Workshop on Groupware, San Carlos, Costa Rica, Lecture Notes in Computer Science, Vol. 3198. Springer-Verlag, Berlin Heidelberg New York (2004) 34-41
13. Schäfer, L., Valle, C., Prinz, W.: Group Storytelling for Team Awareness and Entertainment. Proceedings of the 3rd Nordic Conference on Human-computer Interaction, Tampere, Finland (2004) 441-444
14. Prusak, L.: Where did knowledge management come from?. IBM Systems Journal Vol. 40, No. 4 (2001) 1002-1007
15. Alavi, M., Leidner, D.E.: Knowledge Management Systems: Issues, Challenges, and Benefits. Communications of the AIS Vol. 1, Article 7 (1999) 1-38
16. Desouza, K.C.: Facilitating Tacit Knowledge Exchange. Communications of the ACM Vol. 46, No. 6 (2003) 85-88
17. Nonaka, I. Takeuchi, H.: The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, Oxford, England (1995) 21-45
18. Valacich, J.S., Dennis, A.R. and Connolly, T.: Idea generation in computer-based groups: A new ending to an old story. Organizational Behavior and Human Decision Processes Vol. 57, No. 3 (1994) 448-467
19. Bloom, B.S., Krathwohl, D.R., Masia, B.B. (Eds.): Taxonomy of Educational Objectives. The Classification of Educational Goals. Handbook 1: Cognitive Domain. David McKay Company, Inc, New York, USA (1956)
20. Perret, R.: The group storytelling approach applied to knowledge management. M.Sc. Dissertation, Graduate Program in Informatics, Federal University of Rio de Janeiro, IM &NCE (2004) (In Portuguese)
21. Zope: Open Source Web Application Server. Retrieved June 7, 2004, from: <http://www.zope.org/>
22. Holloway, J.: Narrative and structure: exploratory essays. Cambridge University Press, New York, USA (1979)

23. Rocco, E.: Trust breaks down in electronic contexts but can be repaired by some initial face-to-face contact. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI), Los Angeles, CA, USA (1998) 496-502
24. Gibson, C.B.: From knowledge accumulation to accommodation: cycles of collective cognition in work groups. *Journal of Organizational Behavior* Vol. 22, No. 2 (2001) 121-134
25. Carminatti, N.: Group Storytelling Applied to Collective Knowledge Recall. M.Sc. Dissertation, Graduate Program in Informatics, Federal University of Rio de Janeiro, IM &NCE (Forthcoming) (In Portuguese)

Developing Shared Context Within Group Stories

Flávia Maria Santoro^{1,2} and Patrick Brézillon³

¹ Post-Graduate Program in Informatics, NCE&IM, Universidade Federal do Rio de Janeiro,
PO Box 2324, CEP 20001-970, Rio de Janeiro, Brazil

² School of Applied Informatics, Universidade Federal do Estado do Rio de Janeiro,
Av. Pasteur, 458, Rio de Janeiro, Brazil
flavia.santoro@uniriotec.br

³ LIP 6, Université Pierre et Marie Curie - Paris VI,
8, rue de Capitain Scott, Paris, France
Patrick.Brezillon@lip6.fr

Abstract. Eliciting and re-using knowledge within an organization requires a very structured communication process among its employees in order to avoid misunderstanding and confusion. The transfer of knowledge among actors can only be successful if a common interpretative focus and its context are set up. So far, information about the real context that surrounded team's past activities can help their members to better understand situations at hand. In this paper, we argue that a combination of group storytelling technique and a groupware tool can help the elicitation and use of the context shared by a group. Moreover, our main goal is to discuss how groupware can help to structure and formalize the contextual information behind the scenes of a story told by a group, making it easier to understand, interpret and reuse the knowledge intrinsic to it.

1 Introduction

In business environments, knowledge is frequently defined as 'the capacity for effective action' [18][26]. Thus, knowledge *per se* is not directly of interest to organizations; primarily knowledge becomes valuable in its application. Procedures can not be dissociated from the way they have been accomplished in practice and learning a work process is not just being trained to, but also to observe different alternatives for doing it, and understand the reasons behind the choices according to specific circumstances. The ultimate purpose of knowledge sharing is to promote and disseminate 'effective action', either in the performance of specific tasks or in general behavior [29].

Eliciting and re-using knowledge within an organization requires an elaborated communication process among its employees in order to avoid misunderstanding and confusion. The transfer of knowledge among actors can only be successful if a common interpretative focus and its context are set up and shared [2]. So far, information about the real context that surrounded team's past activities can help their members to better understand situations at hand. This is generally called tacit knowledge, because for the most part relies on people's mind and is not registered in formal documents. It is necessary to capture and organize it in order to be useful.

However, extracting contextualized knowledge from teams and making it explicit is not an easy task. Besides people do not have time to spend in providing information, they are also not motivated since organizational protocols have a tendency to be somewhat dry and lacking in inspiration. Therefore, we suggest that a story is one possibility of registering full-bodied collective context. Stories can be a powerful approach to represent and convey complex, multi-dimensional ideas. “Well-designed, well-told stories can communicate both information and emotion, both the explicit and the tacit, both the core and the context” [28].

Storytelling engages people by means of amusing narrative structure with a more authentic language. Recently, the group storytelling technique has been proposed within the community of Computer-Supported Cooperative Work [1][20][27][33]. It is a collective activity of sense-building, with several individuals contributing with their recollections and interpretations about shared experiences.

In this paper, we argue that group storytelling technique allied with a groupware tool with specific functionality can help re-building a group shared context. Moreover, our main goal is to discuss how groupware can provide support to the externalization of the contextual information behind the scenes of a story told by a group, making it easier to understand, interpret and also to reuse the knowledge intrinsic to it. Therefore, we focus on the structuring context feature.

This paper is ordered as follows. Section 2 tells the implication of contextual information for sharing knowledge in group work. Section 3 presents how the research in group storytelling has been applied to elicit knowledge and exemplify with some business references. Section 4 reports the use of Tellstory, a group storytelling groupware and the way it makes context explicit and structured. Finally, Section 5 concludes the paper and discusses the next steps.

2 The Implication of Context in Group Work

Context is a complex description of the knowledge shared on physical, social, historical and other circumstances where actions or events happen. All this knowledge is not a part of the actions to execute or the events that occur, but will constrain the execution of an action or event interpretation [3]. For the global understanding of several actions and events, it is necessary to have access to important contextual information. Thus, in any domain where understanding, reasoning, problem-solving and learning are needed, the concept of context plays an important role.

In this work, we use the context model proposed by Brézillon and Pomerol [5]. Context is related to a focus, e.g., task, problem solving or decision making. At a given step of this focus, context is the sum of all the knowledge possessed by an actor on the whole process. The authors distinguish between the part of the context, which is relevant for the current focus of attention, and the part, which is not relevant. The latter part is called *external knowledge*. The former is called *contextual knowledge* because it has strong connections with the current focus although not directly considered in it.

Contextual Knowledge is personal to an actor and is evoked by situations and events determined by his focus. At a given focus, part of the contextual knowledge is proceduralized. *Proceduralized Context* is a part of the contextual knowledge, which

is invoked, assembled, organized, structured and situated according to the focus; it is used in the task performed at this focus (depicted in Figure 1).

The focus and its context are intertwined: the focus determines what must be in its context, and the context, on its side, constrains the focus. For example, when telling an event occurred during the development of a project in an organization, a professional might say “we used the method X to build the solution for the problem Y”. The focus was in building the solution for the problem Y applying the method X. Nevertheless, the context related to that event (not explained in the sentence) was: (a) one of team members was a specialist on method X; (b) methods W and Z were tried before but did not succeeded; and, (c) the supporting tool for method X had been recently bought by the company. The contextual knowledge, proceduralized at the time the focus arose can now explain it.

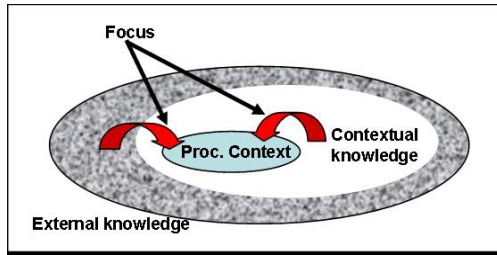


Fig. 1. Types of knowledge related to context and their dynamics

Knowledge intensive processes are intrinsically produced by collaborative efforts. Thus, context plays an important role in collaboration. It can be considered a shared space that is explored and exploited by participants in the interaction. The proceduralized context contains all the pieces of knowledge that have been discussed and accepted (or at least made compatible) by all the agents at a current step of the focus. These pieces of proceduralized context then become part of the shared contextual knowledge of each agent, even if they do not remain within the focus of the context as shown in Figure 2.

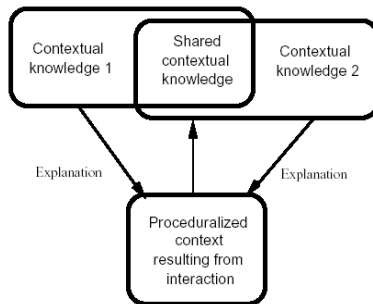


Fig. 2. Building shared proceduralized context and contextual knowledge

Participants must share contextual knowledge in order to reach an agreement for effective communication in a collaborative interaction. That is what Brézillon [4] calls *explanation* in the context of the interaction among a user and a system in a decision-making process; and the groupware research area calls *awareness* [9].

Groups share knowledge and build a collective context while working in a task or in a project. We observe that part of the context about the situations shared generally remains tacit, not registered, making collective experiences difficult to be explained, understood and communicated. Thus, we argue that contextual information should be recalled together with the knowledge produced and properly structured in order to be retrieved. We propose the group storytelling could be used for this purpose.

Next section presents a review of the group storytelling approach, and enlightens that the contextual information problem is generally mentioned, but not emphasized in most applications.

3 The Group Storytelling Approach

Storytelling is a long-established means of passing on wisdom and culture. In recent years, however, the attention to the role of narrative and sketchy information transmitted in the form of stories within organizations has been increasing. According to Sole and Wilson [29], it is due to the fact that the harder forms of knowledge that can be classified, categorized and analyzed are the most valued.

A story can be defined as "a narrative of an event chain told or written in prose or verse", while the word narrative means "to pass knowledge" [33]. A story "lives by itself", while the narrative of a story is just composed of the facts explicitly told. The narrative of a story helps to humanize the environment, involving emotions and provoking personal commitment [14]. Besides, telling a story is also an easier way to explain things informally, because of the needs for contextual cues to underline it, as for example, to explain how to ride a bike.

The popularity and the importance of the stories for the individuals have turned the storytelling a technique studied and applied in many fields and for various purposes such as education and learning [10][22][30], knowledge management in business [23][32], linguistics studies and artificial intelligence [16][25]. Methods and tools have been developed to support the stories capturing, registering and retrieving.

Among many examples of storytelling technique applied to promote the knowledge of management and organizational learning are Shell, ABB and NASA. Shell International Exploration and Production's program for managing its technical and business knowledge is focused on gathering and disseminating expertise within the company wherein the structure is geographically and multi-disciplinary distributed. Logan et al. [15] explain that storytelling helped them circulate employee *know-how* to the places it was most needed.

Post [21] discusses that for NASA's project managers be able to carry on demands aiming at faster, better and cheaper results, they used NASA's ASK Magazine, a storytelling tool for converting tacit to explicit knowledge. Fröhlich and Karandikar [8] describe the work undertaken in ABB, which deals with organizational process improvement by means of goal-oriented stories, told and derived from their domain experts' experiences and recorded in a story-base.

Most storytelling approaches applied in business are based on individual interviews made by a professional storyteller, who synthesizes the events collected and writes his own interpretation into a single text [12]. In this case, the story represents fractions perceived by each individual and joined in accordance to the viewpoint of the teller.

Nevertheless, real stories in organizations are generally experienced by teams. Following this perspective, some authors propose the group storytelling technique [1][20][24]. The group storytelling is a more appropriate method than the individual storytelling when there are several people involved in the setting that is being constructed. Groups build jointly a story about a work performed or a situation experienced by its members. Since each participant performed a role in the scenario, stories written by a team will probably contain more valuable details and everyone has the opportunity to present their view on what had happened.

A few groupware prototypes have been developed to support group storytelling and evaluate the results in terms of knowledge management. The diverse approaches are based on: texts [20]; graphics [1]; documents [27]; and images [24]. Although using different media, they all allow participants to add their contributions and discuss the facts told in a collaborative manner. Authors agree that some structure and aspects to sharpen memory should be offered. Most of them mentioned context as fundamental matter, but none were specifically concerned about structuring the shared context.

In a group storytelling, not only the story itself is interesting but the way in which the story is built, including opposition and negotiation between people, progressive construction of an episode from fragmented souvenirs of people. Generally all this context of the story building is beyond doubt lost and the story only presents a minimal dimension with no possibility to be adapted in another context.

Brooks [6] affirms that context means “seeing from a point-of-view”, often from somebody else’s point-of-view. When the readers distinguish the context of a story, they are able to adopt and accept the point-of-view of the storytellers, in other words, they become aware of “the world” made available to them.

A knowledge-sharing story offers a surrogate experience, as Sole and Wilson [29] explain. The narrative layout offers the reader an opportunity to experience in a replacement fashion the situation that was experienced by the storytellers. The listener can acquire understanding of the situation’s key concepts and their context, and even though the listener did not directly experience the story circumstances as is, he could experience a similar situation. Therefore, it is very important to make clear the context in which knowledge arises and consequently increase the chance of a significant knowledge transfer.

Meech [17] states that contextualization and narrative are active processes composed of several elements. Narrative is seen as the story representation (Story) and presentation (Discourse). The Discourse is the reproduction of the story onto some form of media. For this author, the Story is divided into events and entities; each one can be examined in terms of the contextualization that it is capable to provide.

Events illustrate parts of a story and many times are presented alone. However, a story is not just a collection of isolated events, instead it embodies many elements, called globally context, which links these facts transmitting to the reader a meaningful body of knowledge. Events are framed by context including politics, economy, sociology, literature, and also, personal interpretation, background and culture. The

connotation of a sentence (event) is not determinable in isolation; but requires relating the sentence to other sentences around it, prior experiences or some larger context. Just as knowledge, stories draw meanings from their contextual information [27].

Characters are also an important element of storytelling. The context can be provided using the story actors as the representation of social hints. In a similar way, “setting the scene” is the same of providing context. “Events may be compared with the concept of tasks, the sequencing, structure and composition of which provide vital contextual information” [17]. In this way, a narrative can be viewed as a conceptual framework for providing its actors with awareness about contextual constraints that were once shared by them. Furthermore the readers of the story should be capable to identify these contextual elements as well.

Based on these conclusions, we claim that the shared elements of context from a task performed by a group can be elicited and represented through group storytelling. Therefore, some formalization is necessary and we suggest that a groupware support tool can help to organize and structure this information making it able to be re-used.

In Section 4, we exemplify the issues discussed here, grounded by the theory presented on Section 2, and our implementation and experience in capturing and representing knowledge embedded with context through groupware.

4 Experiences with a Groupware to Structure Shared Context

Previously presented by Perret et al. [20], Tellstory is a groupware that supports collaborative stories’ building [31]. It is a web application where any registered member can start a story and invite new participants to join in, recollect and link important facts about a situation they have accomplished together.

According to Holloway, a story is a sequence of events that are tied to each other by a full conductive thread of meaning, built by a causality relationship between a fact and its successor [11]. Tellstory uses that definition to model the construction of the story in group. Each user can insert one or more events that are facts happened throughout the story, which he remembers. These events should be linked in a temporal flow. Tellstory main interface is shown in Figure 3. A map of events indicating the events’ sequence and a short description of each event are highlighted.

Individuals can participate on the narrative construction performing one of the following roles: (i) moderator: creator and responsible for the coordination of the actions about the story; (ii) teller: member that are able to contribute with events; (iii) editor: person that will write the final text; and, (iv) commentator: responsible for the identification of tacit knowledge externalization on the story. More than one person can assume the same role, as well as each role can be assumed by several people.

These roles are responsible for each step of the storytelling process. When the group (*tellers*) understands that the story already provides enough flow of events, the *moderator* can conclude it. By this time, the *editor* gathers the events and writes a final text based on the sequence. Finally, the *commentator* searches for tacit elements that can be identified in the story, which are registered enclosed to the final text.

A story-sharing system must offer flexible narrative, rather than rigid, pre-authored stories. It should provide enough structure to make possible that new members understand and re-tell the stories themselves, but not to make people locked into one

single way of describing the relevant events. The system should support differences in the way people share stories [27].

The actions along the construction of the story are: inclusion, edition, exclusion, union and fragmentation of events. The union happens when two events can be considered as a single one, yet the fragmentation of the event divides it in two, when necessary. The criteria that indicate if a fact is an event, a sub-event or collection of events do not need to be explicitly defined by the participants. This configures the tool as a flexible environment, where people can express themselves freely [20].

The focus of this research work is to discuss the ways to communicate correctly the contextual information that surrounds the events told in order to make them clear and understandable for all members of the group and mainly for the future consumers of the story. Tellstory helps users to externalize context in two ways: implicitly embed in the text and comments about the events and explicitly through a framework provided.

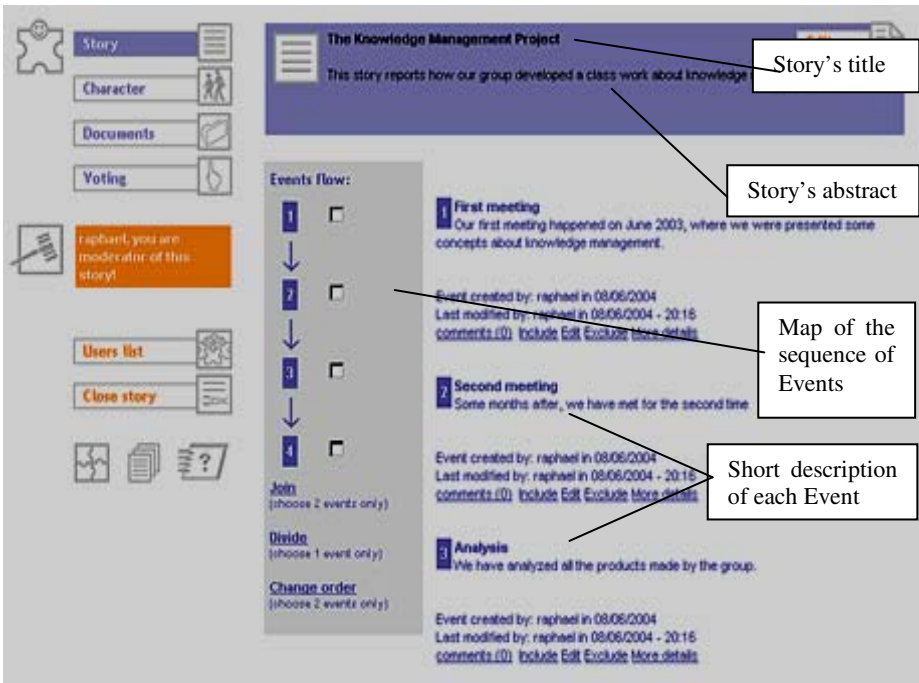


Fig. 3. The flow of a story in Tellstory [20]

4.1 Identifying Context in the Text and Comments

The first manner to express context is informally, through the users' contributions (events) and the notes (comments and discussions) about the others' contributions. Analyzing the text exposed in the description of each event, we can observe that many times the facts are naturally mixed with their context. On one hand it reveals the intention of explaining and detailing the whole story, besides the natural language stimulates free performance. On the other hand, whether a further interpretation of the

situation described is necessary in order to retrieve some specific facet desired, some structuring might be required.

Groups need support to express their thoughts and to solve the conflicts in order to produce real, interesting and useful stories. Once the tellers have included the events, they can discuss them within Tellstory, through adding comments in a forum format. The unstructured comments may complement the information presented, as well as generate conflicts. Individual contexts are proceduralized, allowing a shared and collective context to be built.

We reproduced here extracts from a case study made at a government organization in Brazil [19] to exemplify this situation. A group of five members (called M1 to M5) were invited to tell the story about the constitution of the central knowledge management team in that institution. They interacted through the Tellstory application during one month and pointed out the important events, reconstituting their shared context that had not been registered until that time. Twelve events were told and associated resulting in a final text; we examine two of them in this paper.

1st Event by M2: In the first meeting of the central team of knowledge management, the General Controller, the Secretary of Administration and the Executive Director of the institution had been invited to demonstrate the institutional support and to congratulate the group. Moreover, the coordinator of the group presented general concepts of Knowledge Management and the proposal, elaborated by the Knowledge Management Committee, describing the plans for the work to be performed.

2nd Event by M1: In December 2002 the second meeting of the central team of knowledge management was carried out. In this occasion, C.S., the Manager of the Corporative University of one of the institution's units, presented his project. In this meeting, the number of participants was reasonably superior to the previous one.

Comments made about this event in the forum:

M1: Do you have any suggestion for what the consequences of this event were?

M3: One important outcome was that the participants had been distributed in 3 thematic groups (organizational learning, organizational culture and information technology), to start the work of identifying already existing cases in the institution.

From this point of the story, we can observe that two events were told, related to two meetings of the group where some other people participated and some deals and decisions were made. In the first one, the goal was to formalize the group and establish its objective. The teller M1 explicated his following Contextual Knowledge (CK), the pieces of knowledge related to the event: (CK1) Some executives were present; (CK2) The executives gave institutional credibility to the event.

In the second one, the focus was on the speech of C.S. Nevertheless, M1, the member who told this event could not retrieve one piece of knowledge from his memory: the meeting led to the creation of thematic groups. Thus, the comments that M3 shared with the group helped to identify important contextual information related to this event: (CK3) Thematic groups were started; (CK4) Thematic groups should identify knowledge management initiatives within the institution. Other contextual information was pointed by M1: (CK5) The number of participants increased.

According to Brooks [6], the storyteller establishes story context in part through the relationship with the audience. The author suggests that “the words the storyteller uses in the story go a long way toward establishing context as well”. The relationship

is dependent on attributes of the situation or the composition of the audience. For example, a woman telling a story to a group of women from her own culture does not need to specify many details about femininity, for instance, because both teller and audience share a common cultural definition.

In our case, we would like the storytellers to provide as many details as they can, because the audience is unknown. A reader could be anyone in organization, even from other communities different from the tellers and the purpose is to transfer knowledge that ought to be found as much easily as possible. We can by far observe that some fine points were omitted in the narrative, as for instance the personal and professional information about the participants of the meetings.

When a participant in a group storytelling asserts some fact or makes a comment about some idea presented within the story, he may start a discussion process which can lead to a learning process by the group. Beyond the results, the process of learning is concerned in collaboration. The learning effect is personal to an individual because it supposes the integration of a new item in the existing mental schema of the individual. A support from another individual (in another mental schema) can only aim at making compatible the integration of the new item in their respective mental schemas.

We could notice that while the participants told their memories they also explained the situations by proceduralizing their contextual knowledge, re-building a shared context from the whole group.

4.2 Organizing Context Through the Complementary Information Framework

The second way used by Tellstory to elicit context is an attempt to extract information apart from the text of the events in a structured format. Therefore, it provides the users with a Complementary Information (Context) Framework to stimulate externalizing specific contextual information related to each event of the story.

For some authors [24][27][6], four questions are essential to storytelling: *who?*, *when?*, *where?*, and *what?*. These categories provide the type of contextual information expected to be captured together with the story. Based on studies about awareness in groupware applications, we introduce two more questions besides the **who? when? where? what?**: the **how?** and **why?**.

The contextual information that surrounds an event in a story should explain it. The answers for these six questions are supposed to provide that information. Therefore we suggest that they have to be represented and organized. A Framework based on these questions is presented in Tellstory interface while a participant is editing an event. The teller can use this space to inform the particulars about the event as well as create proper relationships among them.

The framework calls the users' attention to the typical characteristics of a narrative structure, in fact, working as a guide for the tellers, stimulating their memories and helping them to better structure their thoughts and expand their contribution by giving more details about the event told.

The subjects pointed in Table 1 compose the Context Framework asking the tellers to post some specific information related to the six questions mentioned.

In Figure 4, we show the creation of an event and the presentation of some parts of the Context Framework in Tellstory interface. The level of structuring the information proposed allows identifying relationships among the events declared and retrieve

them after. For example, participants are stimulated to describe the details about each *Character* of the story (general description, professional background, technical abilities, interpersonal relationship with the group, task involvement) and associate to a specific event the ones who really actuated in it (as highlighted in Figure 4).

Table 1. Subjects on the Context Framework

Subject	Asks the teller to:	Addresses:
Character	Detail the personages and their roles on the story (General Description, Professional background, Technical abilities, Interpersonal relationship with the group, Task involvement)	“Who?”
Period	Write date or period where this event occurred.	“When?”
Classification	Indicate to what part of the story this event belongs (Exposition, Complication, Climax or Outcome)	“When?”
Place	Describe the place and scenario where this event occurred.	“Where?”
Causes	Discuss what caused this event (events might be related to the previous events)	“Why?”
Effects	Type the consequences of this event (events might be related to the next events)	“What?”
Emotions	Describe perceived feelings while this event has occurred	“How?”

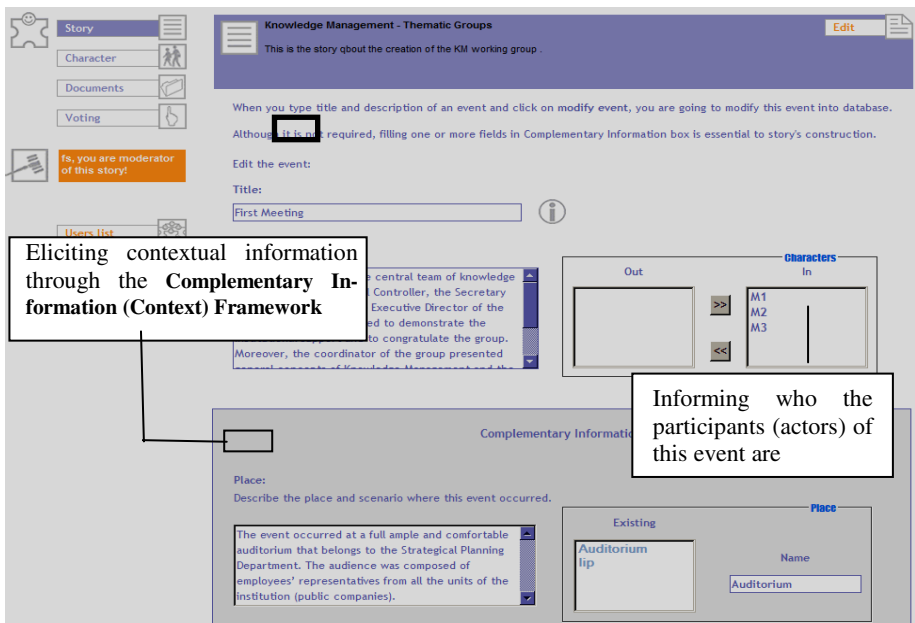


Fig. 4. Externalizing Context in Tellstory

The creation and description of characters are made apart. Figure 5 illustrates the description of a story character. After telling important information about each character of the story the whole list of characters (shown in Figure 4) are made available to be linked to a specific event.

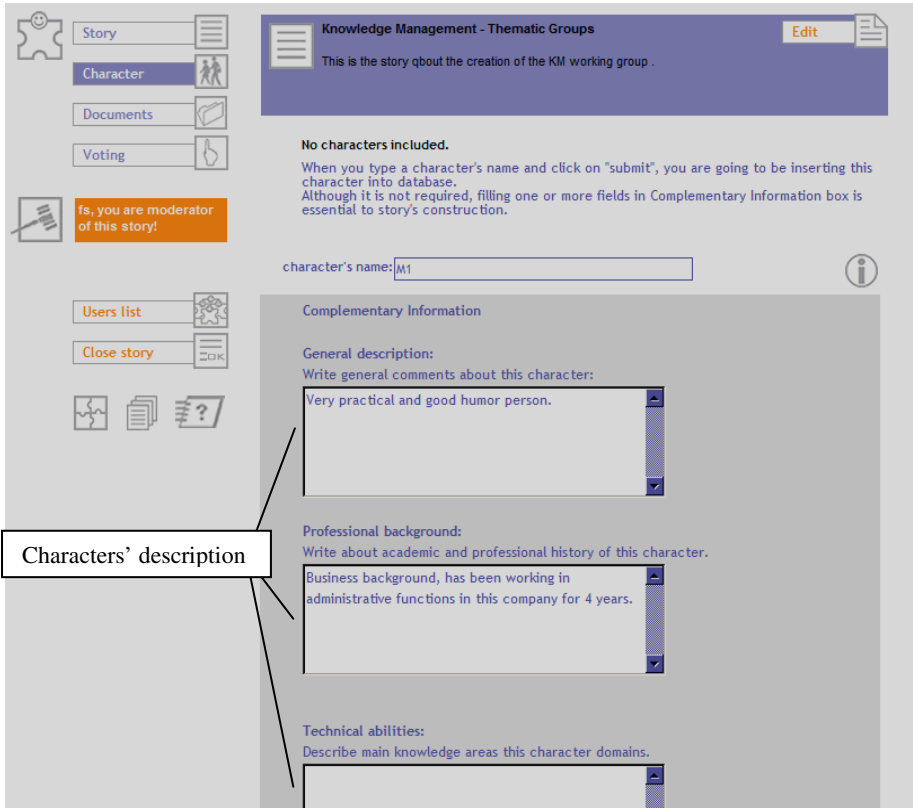


Fig. 5. Character Description in Tellstory

The same procedure is done concerning the *Place*, which the participants should depict and incorporate into a list. Besides, they indicate the exact *Period* when that event has taken place, along with its position in the progress of the story (Exposition, Complication, Climax or Outcome). This information may possibly determine the repercussion of a specific event for the whole results of the experience described.

The *Causes* and *Effects* clarify the non-temporal links among different events. It is important to realize that one fact had contributed to others even though they could have apparently no relationship at all. For instance, the exposition made by the coordinator at the first event of our case study might have motivated consequences in some decision later on the project. If someone wants to reuse this decision in a similar situation, it would be helpful to notice that some previous fact had contributed to it.

Finally, the personal feelings of people influence the way things go on in several settings. Trying to capture some of them, we seek to go deep in the explanation of how things happened. We let the tellers associate an iconic representation of *Emotions* to each actor involved in an event told.

The attempt to extract contextual information apart from the story makes possible to start a process of formalizing context. The Framework provides structured lists for all the contextual elements described, including the graphical icons to represent the

emotions felt by the actors in the course of an event. Establishing formal relationships among the various contextual elements makes it possible for the readers of stories to search for information that are related to the circumstances he deals with and so he can learn with the accomplishment of the story, reusing knowledge.

Back again to the example of our case study, while describing the events, the tellers also used the Context Framework proposed to detail and organize the information provided:

<p><i>1st Event by M2</i></p> <p><u>Characters:</u> M1, M2, M3</p> <p><u>Place:</u> The event occurred at a full ample and comfortable auditorium that belongs to the Strategical Planning Department. The audience was composed of employees' representatives from all the units of the institution (public companies). (M2)</p> <p><u>Period:</u> 14.11.2002, from 15:30 to 17:30. (M2)</p> <p><u>Causes:</u> As mentioned in Decree 21,683, of 04.07.02, the representatives of the municipal agencies would have to participate on specific or general meetings. All of them had been invited by means of an email posted by the work Coordinator. (M1)</p> <p><u>Consequences:</u> People heard the words of the authorities supporting the initiative and learned on the subject Knowledge Management and the proposal of the team. (M3)</p> <p><u>Emotions:</u> Most of the audience did not demonstrate in their faces credibility on the proposal. Many people were tense, confused, without knowing right what was happening. Some people had questioned the success possibilities of the work in face of the complexity and the cultural characteristics of the institution. However a few other people demonstrated excitement with the new perspectives of sharing among the institution's agencies. (M1)</p> <p><u>Classification:</u> Exposition (M1)</p>

Adding such information based on the framework, allowed the group to increase even more their collective knowledge about the event and the relationship among the others. M1, M2 and M3 revealed to the group new contextual knowledge that helped to explain how and why things took place at that time: (CK6) There is a decree that compels the employees of the institution to participate in such meetings; (CK7) The Coordinator invited people for the meeting. As a result or Proceduralized Context (PC) reached at that time, they agreed that: (PC1) People were not receptive to the proposal at first moment.

If we interpret the pieces of knowledge provided, it would be possible to write the following statement:

If (CK3) and (CK6) and (CK7) and (CK1) and (CK2)
 Then Focus ← Event I
 Result ← PC1

Such statement creates relationships among the isolated contextual information provided by teller, providing one more level of structuring.

We observe that after the interaction, the group has registered many of the knowledge about work they performed together. It was very natural for them to formalize the events and contextual information that surrounded them through storytelling. The next step is to move the application to an even more formal model such as ontology which will make possible not only to explore the relationships among diverse contextual information in deep, but also to infer non explicit ones.

5 Conclusions and Future Work

The conditions and constraints of knowledge use are as important as the knowledge itself. Research on context [5] recognizes that the capture, the management and the retrieval of explicit organizational knowledge must be considered jointly with the context in which it is captured, recorded and used. The lack of contextualization can lead to knowledge misuse or may cause wrong application since knowledge cannot be separated from its use.

In this paper, we highlight the importance of identifying contextual information allied to the pieces of knowledge shared within a group interaction while performing a task or a project in an organization. Due to the characteristics of this process we propose the group storytelling technique to support the structuring of context. We exemplify our discussion through a case study made with Tellstory [19], a groupware that supports collaborative story building, showing the viability of this proposal.

Narrative is a structure for conveying a series of related events. We observed that the story may omit details, but important agents, events, causes and results are pointed. A narrative describes a project history and its evolution over time. It may not be as complete as, for instance, videotapes of the entire work process, but it does communicate effectively how a project has taken form. By relating changes, problems faced and decisions made, a narrative can help make explicit some of the implicit knowledge the participants used to understand and implement the interventions, in other words, the whole context built. Thus, one might infer whether the results were applicable elsewhere.

Indeed we recognize there is not a clear distinction between the story and its shared context and this is why the study of context in the domain of storytelling presents a special interest. Contextual Knowledge could be placed at a meta-level if compared to the narrative, in the sense that it is a framework to classify the pieces of knowledge, turning the story able to be more easily adapted to different situations and also reuse alternatives initially abandoned.

We assume there are different granularity levels for Contextual Knowledge. For example, in our case study, taking the meeting as the focus, at one level, we can notice that the coordinator invited people (a very general context), but at another level, people went there obliged due to a decree (more specific context). The story unifies all the contextual elements providing a global sense to them.

Other kinds of contextual information can be in addition proposed. Asking for other relevant events, but not directly related to the story: What more was happening at the time the event took place? Is this any special date? (when). Asking for stakeholders: Who else knows about this event or could be interested in this topic? (who). Asking for extending knowledge about the event: What kind of professional information is there about this? What other applications are related to it? What more can be read about this? (what). Asking for relationships in space: Where else could it have occurred? (where). It would also be interesting to associate Emotions with Causes and Effects.

However it is important to draw attention not to overwhelm the participants with a big framework and many screens to pass through. Otherwise, telling a story would be distorted in a form fill practice. In our first experiences we have tried to let the tellers free to inform just whatever they want, not making any field obligatory.

The template provided by Tellstory is the first attempt to solve the context structure problem. Now we begin to study how to provide an even more formal structure for the pieces of knowledge captured by adding ontology format feature to the stories. Ontology will enhance the possibilities of making inferences on the information retrieved and provide the users of the stories the capability of associating them with their own contexts.

Because stories occur under a cultural and historical context, facilities to bring out background and contextual information could be provided, e.g. relevant news, to assist the user to interactively reflect on and share past experiences of the group. This could help participants to remember important facts, including personal ones, which might probably have affected the story. The current version of Tellstory allows users to upload documents associated to the story.

Group sense-building is a valuable function of storytelling and electronic story-bases can stimulate it by providing facilities such as comments and re-telling [7]. Nevertheless, discussions and disagreements will certainly arise. The forum format provided could be improved using a pre-defined model such as the ibis [13]. This way, information contained in the comments could also be linked and more easily used to advantage of the groups.

Also as a future work, one issue that should be discussed is the identification of the appropriate roles and what their contribution in terms of elements in the collective context linked to the focus could be. Proper interventions made by individuals with specific assigned roles may result in a story even more rich in details. We believe that the basic roles offered by Tellstory could be increased.

Besides other case studies wide-ranging in time should be carried on in other evaluate the next step in this process, which is the retrieval of knowledge contained in the stories from the community inside the organizations.

References

- [1] Acosta, C., Collazos, C., Guerrero, L.A., Pino, J.A., Neyem, H.A., Motele, O.: Story-Mapper: A Multimedia Tool to Externalize Knowledge. Proceedings of the XXIV Conference of the Chilean Computer Science Society, IEEE CS Press, Arica, Chile, November, 2004, pp. 133-140.
- [2] Araujo, R. M., Brézillon, P.: Modeling Software Organizational Knowledge through Context. Proceedings of Knowledge Sharing and Collaborative Engineering, St Thomas, USA, 2004.
- [3] Brézillon, P.: Context in problem solving: A survey. *The Knowledge Engineering Review*, vol. 14, n°1, 1999; pp. 1-34.
- [4] Brézillon, P.: Individual and team contexts in a design process. Proc. 36th Hawaii Int.Conf. on Systems Sciences. HICSS-36, Track "Emerging Technologies", R.H.Sprague (Ed.), Los Alamitos: IEEE, CD-Rom, January, 2003.
- [5] Brézillon, P., Pomerol, J.C.: Contextual knowledge sharing and cooperation in intelligent assistant systems, *Le Travail Humain* 62 (3), PUF, Paris, 1999, pp. 223-246.
- [6] Brooks, K.: The Context Quintet: Narrative Elements Applied to Context Awareness, In Proceedings of Human Computer Interaction, Greece, 2003.

- [7] Fraser, M., Stanton, D., Ng, M., Benford, S. D., O'Malley, C., Bowers, J., Taxn, G., Ferris, K., Hindmarsh, J.: Assembling History: Achieving Coherent Experiences with Diverse Technologies. Proceedings of ECSCW 2003 Helsinki, Finland, Kluwer, 2003.
- [8] Fröhlich, P., Karandikar, H.: Driving organisational change: Using story to transform work processes at ABB. *Knowledge Management*, 18 Mar 2002.
- [9] Gross, T. and Prinz, W.: Modeling Shared Contexts in Cooperative Environments: Concept, Implementation, and Evaluation. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, Volume 13, Issue 3, August 2004.
- [10] Guerrero, L. A., Mejías, B., Collazos, C., Pino, J., Ochoa, S.: Collaborative Learning and Creative Writing. Proceedings of the First Latin American World Wide Web Conference, IEEE CS Press, Santiago, Chile, 2003, 180-186.
- [11] Holloway, J.: *Narrative and structure: exploratory essays*. Cambridge University Press, New York, 1979.
- [12] Kleiner, A. and Roth, G.: How to Make Experience Your Company's Best Teacher. In: *Knowledge Management*. Harvard Business Review 75, no. 5, 1997.
- [13] Kunz, W. and Rittel, H.: Issues as Elements of Information Systems, Working Paper 131, Inst. of Urban and Regional Develop., U. California at Berkeley, 1970.
- [14] Lelic, S.: Fuel Your Imagination - KM and the Art of Storytelling. *Knowledge Management*, 2001.
- [15] Logan, E., Boyd, A., Vigers, B.: A pipeline for collaboration: Leveraging knowledge through storytelling at SIEP. *Knowledge Management*, 20 Nov 2001.
- [16] Mateas, M., Sengers, P.: Report from the 1999 Fall American Association for Artificial Intelligence Narrative Intelligence Symposium, USA, 1999.
- [17] Meech, J.F.: Narrative Theories as Contextual Constrains for Agent Interaction. In: 1999 Fall American Association for Artificial Intelligence Narrative Intelligence Symposium, USA, 1999.
- [18] Nonaka, I.: A dynamic theory of organizational knowledge creation. *Organization Science* 5(1): 14-37, 1994.
- [19] Perret, R.: The Group Storytelling Technique Applied to Knowledge Management. Master Dissertation, Federal University of Rio de Janeiro, Brazil, 2004 (in Portuguese).
- [20] Perret, R., Borges, M.R.S., Santoro, F.M. Applying Group Storytelling in Knowledge Management, Proceedings of International Workshop on Groupware, San Carlos, Costa Rica, *Lecture Notes in Computer Science*, Berlin, Germany, Springer-Verlag, 2004.
- [21] Post, T.: The ASK story: An insider's perspective on storytelling at NASA. *Knowledge Management*, 20 Nov 2001.
- [22] Roussou, M.: The Interplay between Form, Story and History: The Use of Narrative in Cultural and Educational VR. In O. Balet, G. Subsol, and P.Torguet (Eds.), International Conference on Virtual Storytelling 2001, LNCS 2197, Springer-Verlag Berlin Heidelberg, pp. 181-190, 2001.
- [23] Ruggles, R.: The Role of Stories in Knowledge Management. Storytelling Foundation. <http://www.storytellingfoundation.com/articles/business/stories-km.htm>. Accessed on August 2003.
- [24] Schäfer, L., Valle, C., Prinz, W.: Group Storytelling for Team Awareness and Entertainment. Proceeding of ACM NordCHI, Tampere, Finland, 2004.
- [25] Schank, R.: *Virtual Learning: A Revolutionary Approach to Building a Highly Skilled Workforce*. McGraw-Hill, 1997.
- [26] Senge, P.: *Sharing Knowledge*. Executive Excellence 14(11): 17-18, 1997.
- [27] Shen, C., Lesh, N.B., Vernier, F., Forlines, C., Frost, J.: Sharing and Building Digital Group Histories. Proceedings of ACM CSCW'2002, New Orleans, USA, 2002.

- [28] Snowden, D.: The Art and science of Story or 'Are you sitting uncomfortably?' *Business Information Review*, Dec 2000 17(4): 215-226.
- [29] Sole, D., Wilson, D.: Storytelling in organizations: The power and the traps using stories to share knowledge in organizations. LILA, Harvard, Graduate School of Education, 2002.
- [30] Stanton, D., Bayon, V., Neale, H., Ghali, H., Benford, S., Cobb, S., Ingram, R., O'Malley, C., Wilson J., Pridmore, T.: Classroom Collaboration in the Design of Tangible Interfaces for Storytelling. Proceedings of Computer-Human Interaction, Minneapolis, USA, 2001.
- [31] TELLSTORY: <http://chord.nce.ufrj.br:8080/tellstoryen>, 2005.
- [32] Thomas, J.C., Kellogg, W.A., Erickson, T.: The Knowledge Management Puzzle: Human and Social Factors in Knowledge Management. *IBM Systems Journal*, v.40, No.4 (2001).
- [33] Valle, C., Raybourn, E.M., Prinz, W., Borges, M.R.S.: Group Storytelling to Support Tacit Knowledge Externalization. Proc. of the 10th International Conference on Human - Computer Interaction. Crete, Greece, 2003.

Patterns of Collaboration and Non-collaboration Among Physicians

Claudia Barsotini¹ and Jacques Wainer^{1,2}

¹ Department of Health Informatics, Federal University of Sao Paulo (UNIFESP),
Sao Paulo, SP, Brazil

claudia@dis.epm.br

² Institute of Computing State University of Campinas (UNICAMP),
Campinas, SP, Brazil

wainer@ic.unicamp.br

Abstract. This work present an empirical evaluation of factors that discourage a stronger collaboration of physicians across time. By observing two different outpatient clinics in which a single patient is treated by a sequence of physicians for a long period, we were able to detect three aspects of a (paper) patient record that makes collaboration difficult: lack of diagnostic rationale, lack of treatment rationale and improper way of presenting the information contained in the patient record.

1 Introduction

The collaboration among physicians has always been a topic of interest among researchers in collaboration and CSCW. There are examples of very efficient groups, for example a surgery team, or a medical committee discussing case. In all these examples, the collaboration is synchronous and intense. We are interest in a asynchronous form of collaboration, specially a collaboration centers on a artifact - the patient records. We feel that an understanding of how physicians collaborate and more specifically how the patient record inhibits the collaboration among physicians could lead to important insights on how to develop computer based patient records that foster collaboration.

Long-term care of chronic or syndromic patients, seems to be an extremely interesting example of work to promote collaboration among physicians. In the case of syndromic diseases, in which patients have a set of signs and symptoms, one single physician may not be able to care for all aspects involved in the syndrome, requiring collaboration of other specialists. In cases of chronic diseases, in which definite cure is not possible, settling for stabilization of the disease manifestations, collaboration seems to be within the same medical specialty. In this case, collaboration takes place among physicians of the same specialty and in general they have different levels of knowledge about the specialty.

1.1 Relevant Research Studies on Medical Collaboration

There has been interest in medical collaboration for some time, especially in the areas of promotional tools, analysis and assessment of collaborative examples. In the

area of tool development, telemedicine may be seen as one of the main research lines involved in promoting medical collaboration. Almost all examples of telemedicine in teleconsultation described in the literature, either synchronous or asynchronous, involve collaboration between two healthcare professionals.

In the area of analysis and assessment of examples of medical collaboration, research studies point to medical records, either electronic or traditional, which require substantial improvement to enhance collaboration among professionals that use them.

Some studies about medical collaborative work show significant discrepancies in the assumptions of the role of electronic medical records in reaching integration of different medical services[2]. Other studies analyzed limitations of the electronic record systems when used by interdisciplinary groups [1,3].

Medical practice varies widely through different sectors, departments, hospitals and specialties[4,5,6]. This variation in practice is also translated into differences in use and type of source of information. Furthermore, there are significant discrepancies between collected data in the paper medical records and the necessary information to reach integration among physicians, considering the way physicians use and communicate information on the patients. What is written down is substantially different from what is said.

Other works have pointed out that paper medical records reproduce fragmented and ambiguous redundant information about patients [7,8].

2 This Study

In this study, we were interested in understanding how physicians collected information from medical charts (which were paper version in this case) to assess a new patient sent to them. We may understand it as a collaboration throughout time: one physician saw the patient for some time, recorded information about him/her in the medical file and a second physician, using the same medical record, has to learn about the condition and status of the patient. As we will see in the present paper, the information usually available in medical records is not enough for the collaboration to be effective.

Understanding the limitations of paper medical records may be relevant in designing electronic medical charts in order to maximize this type of collaboration.

The present study investigated two outpatient clinics in a large hospital in Brazil. It is a public university hospital in the city of Sao Paulo, in which patients have access to several procedures, from emergency care to periodical visits, exams or even hospital admission. Given it is a state-owned institution, services are paid by the government, reason why the hospital attracts mainly low-income people.

First of all, we studied the specialty of Clinical Neurology. Neurology is a specialty in which most of the cases are chronic or syndromic and patients are followed up by periodical visits to assess progression of the disease, to make adjustments in drug dosages, to control some symptoms, and make appointments with other specialties. The physicians that work in the neurological unit are resident physicians in the Medical Residence Program in Neurology, which lasts three years.

Secondly, we studied the clinical specialty of Nephrology, more specifically, Renal Lithiasis. The subspecialty of renal lithiasis is characterized by chronic patients that are controlled by periodical medical visits to assess disease progression and medication

adjustments. The physicians that work at the nephrology unit are resident physicians in the Medical Residence and Specialization Program in Nephrology, which lasts three years.

Thus, both the Neurology and the Nephrology outpatient clinics are integrated in the hospital infrastructure, especially regarding visit scheduling, patients' referral to other specialists and scheduling of exams and tests. Patients' records are controlled and managed by the hospital, known as the blue record file, and stored in the medical and statistical filing center (abbreviated in Portuguese as SAME). The medical chart is required from SAME by the outpatient unit management some days before the visit. SAME separates the records of scheduled patients and send them to the outpatient clinic on the day the patient has a visit. Almost all outpatient clinics also have their own specific information in a chart, creating a parallel network of information on the patient.

The present study was conducted from January to May 2004 in the Neurology General Outpatient Clinic of the Department of Neurology, Universidade Federal de Sao Paulo, Sao Paulo Hospital (HSP). Some characteristics of the outpatient clinic are as follows: chronic and syndromic patients, seen for long periods of time, and medical staff that is switched every two months owing to the schedule of the medical residence program.

We based our collection on three types of data: observation of visits, interviews with physicians, and analysis of documents. We analyzed the medical records of patients, both the specific and the general HSP charts.

We gathered 14 case reports and over 21 hours of observation. The observation of the medical visits was focused mainly on describing the situations in which physicians collaborated using the medical record of the patient. After observing the visits, some questions were asked to the physicians that had seen the patients in order to better understand the case. The description of two visits and the discussion with the physicians of the Neurology General Ambulatory are provided below.

2.1 Case Reports

We selected two vignettes to illustrate three points we find important.

Case 1. "... The physician read the three previous visits of patient MRS, male, 91 years old, who has been followed-up in other HSP units (Cardiology, Urology and Geriatrics) since 1990. The medical record has the diagnoses of coronary artery insufficiency, hypertension and Parkinson's disease (PD) since 1990. In the specific Neurology chart, the diagnosis of PD was made two years ago. The physician suggested that the presumptive diagnosis of PD was made two years and not 10 years before. The preceptor, who was in the room and knew the case from previous visits, suggested that the disease had in fact been in existence for 10 years and the family members of the patient agreed. The physician reported what was described in the first visit at the Neurology outpatient clinic and the preceptor refused some of the findings. The physician reported the drugs taken and the dosages and that the patient had interrupted drug use. The preceptor reminded them that the patient had discontinued medication because of therapeutic trial, and that this information was not in the medical record. The preceptor questioned the specific condition of the patient and its correlation with advanced age..."

Case 2. "... patient MHL, 37 years, currently being followed up at Neurology and other HSP units. The physician read aloud the clinical history to the patient and checked whether she still had the same symptoms and the main complaint. The patient reported worsening of pain in the leg; in the medical chart, it was described as normal physical examination. The physician did not find the topographic diagnosis of the condition in the chart. The patient complained that she had only undergone one test since she had started treatment. The physician asked the patient about other pains and aches, rashes, urine, and stools. The patient answered they were normal. The physician did not write these pieces of information down on the chart. The patient reported difficulty to come to the psychiatric visit and that she had been absent from some sessions; moreover, she had stopped taking the prescribed medication. The physician called the preceptor to discuss the case. The physician described the case since the beginning and the preceptor said he could not understand it. Then, the physician reported the case in a clearer fashion by chronologically describing clinical history, disease progression, diagnosis and exams..."

3 Discussion

The two vignettes above illustrate three issues that we find important regarding collaboration between physicians across time:

1. Lack of information about the diagnostic rationale.
2. Lack of information about therapy and medication used.
3. Inappropriate data collection and presentation.

Let us see them in details. Regarding the lack of information about the diagnosis rationale, in Case 1, the presumptive diagnosis was not included in the medical chart ("*In the specific neurology chart, the diagnosis of PD was made two years before.*"). That is, there was a suspicion of PD but until the Neurology staff made the diagnostic there was no record of it - and wrongly, once the diagnostic was made, it was placed 8 years before (because it explained 8 years of the patient's symptoms).

There are a set of characteristics of the patient record that explain this lack of rationale.

- The patient record is a legal document that can be used in legal processes as evidence both against and in favor of the doctor
- The records are also used by the health organizations for teaching, for quality control, and so on
- Finally in Brazil, the health records are legally property of the patient, and are only under guard of the health organization and health professional. Thus the patient can request his records at any time.

All these aspects of the patient records make them an artifact that it is not under control of the physicians who are the ones that deal and interact with it. This lack of control over future of the artifact constrain what the physicians are willing to record on it. From the physician's point of view, his legal and ethical obligations are to record the raw data (lab exams, physical exams) and his **decisions** but not his doubts. Not recording doubts and speculations, and the rationale for his actions may be something the physician learns

because other do the same, but a possible reason is that a physician does not want to record in a legal document what he does not know.

Case 1 also illustrate this lack of rationale information regarding the prescriptions. The reason why such drugs were selected, changes of dosage, and therapeutic trials were suppressed from the chart, and the preceptor had to clarify the reason why treatment was discontinued (“... *the physician reported the drugs taken and the dosages and that the patient had interrupted drug use. The preceptor reminded them that the patient had interrupted medication use because of a therapeutic trial*”). Again the fact that the records are a legal document may hinder recording the rationale - the legal important information is what was prescribed, and not why that drug was prescribed.

Case 2 shows a less clear problem. The physician included irrelevant information about the patient in the medical record, in addition to presenting difficulty to report the history in an organized and summarized fashion (“*The physician described the case since the beginning and the preceptor said he could not understand it. Then, the physician reported the case in a clearer fashion by chronologically describing clinical history, disease progression, diagnosis and exams...*”) It was necessary to chronologically reorganize the information from the chart to report the case to the preceptor. The charts record, at most, the details of a single consultation, but in a diagnosis process that is not an appropriate form to organize that information. Probably besides rationale, things like expectations, future plans, and information that can link one consultation to the previous and next one will be useful to reconstruct a epistemic history of the case and to more quickly understand what is the state of the patient’s treatment.

4 Conclusions

This paper has presented three shortcomings of paper medical records as tools that should facilitate the collaboration or at least communication among physicians through time. For long term patients, the medical records are the form by which a doctor should communicate his opinions and decisions about the patient to doctors that will take care of the patient in the future.

Two of the shortcomings derive from the fact that the records are not under control of the physicians - they are in some sense property of the patient, they are under control of the health organizations, and they may be seized by legal authorities. Given this lack of control over the artifact, the users are not willing to write down their doubts, expectations and so on, information that will be helpful for future doctors responsible for that patient.

The third shortcoming derives from different perspectives that are in place when the physician is filling the record and when he is reading it. The order and amount of information recorded is not conducive to a reconstruction of the case, which is what is needed when a new physician assumes the patient.

In broad terms, the needs of medical records to foster better collaboration are not dissimilar from the ones pointed out by researchers on collaboration in design, and so likely are the solutions and problems [10].

These problems must be addressed when specifying an electronic medical record system, if one expects improvement on the collaboration levels of doctors using the

system. The line of research presented in this paper, of understanding the practice of collaboration in health care, and other published research [11,12] on the same topic, hopefully will on the future finds its way into the requirements of future health care systems that can really improve the quality of care.

We feel that it is too soon to propose technical solutions to the problem of improving physicians collaboration. There are issues that still require further investigation; for example the issue of rationale. It is clear from our research that the recording of the decisions rationale is a useful tool for the collaboration, but it is yet unclear exactly what needs to be recorded and whether physicians are able and willing to record the rationale.

Nevertheless we are in the first stages of exploring a obvious technical solution to the problem - the separation of the patient record in a legal and “non-legal” or “private notes” part. The legal component would remain as it is, with the legally required information about the patient, diagnosis, treatment, and so on, and the other component would allow the physicians to express the rationale behind the decisions, their expectations and doubts regarding the patient treatment and so on. In fact we have evidence of double records, not in health organizations, but in privately own practices.

But this poses the question of the ownership of the non-legal component of the records - in the case of the private practice, the owner is the physician himself. But in the case of an outpatient public clinic the issue of ownership is less clear.

A different question, even if the ownership issue is resolved, is whether physicians are willing and have enough incentives to record the decision rationale and doubts. Physicians are trained to present themselves as very sure to the patients and they may feel very uncomfortable to disclose their doubts. There may be problems with a perceived loss of professional standing if a physician expresses his doubts to colleagues. Finally there is the ever present issue of the separation of those who do the extra work and those who profit from it [13] - why would a physician spend the extra time recording the rationale if he is not the one that should benefit from it?

In a related line of research we are in the first stages of investigating the use of the non-legal part of the record to automatically generate many kinds of different reports, including the legal component of the record, and a patient summary. This would not only encourage the physician to use the non-legal record but would also attempt to solve the third problem discussed in this paper, the quality and quantity of information needed to understand the state of a patient. A summary that links the different consultations regarding expectations, plans, and so on, may be extracted from the non-legal record if it has enough information. Again, more empirical research is needed to determine what information is really needed in these summaries.

References

1. Ellis CA; Gibbs, S.J. e Rein, G.L. Groupware: Some Issues and Experiences. *Communication of the ACM*, v.34, n.1, 1991, p.1-29.
2. Hartwood M, Procter R, Rouncefield M, Slack R. Making a Case in Medical Work: Implications for the Electronic Medical Record. *Journal of Computer Supported Cooperative Work*, 12, 2003. p 241-266.

3. Strauss A, et al. *Social Organization of Medical Work*. University of Chicago Press. 1985, Chicago.
4. Atkinson, P. *Medical Talk and Medical Work*. Sage, London: 1995
5. Berg, M. Patient Care Information Systems and Health Care Work: A Sociotechnical Approach. *International Journal of Medical Informatics*, vol. 55, pp. 87-101, 1999.
6. Grimson, J., W. Grimson and W. Hasselbring. The SI Challenge in Healthcare. *Communications of the ACM*, vol. 43(6), pp. 49-55, 2000. ACM Press.
7. Clarke K., Hartswood M., Procter R., Rouncefield M.. The Electronic Medical Record and Everyday Medical Work. *Health Informatics Journal*, 7(3/4), 2002. p. 168-170.
8. G. Ellingsen, E. Monteiro, A patchwork planet. Integration and cooperation in hospitals, *Journal of Computer Supported Cooperative Work*, 12: 71-95, 2003.
9. Berg, M. Search for synergy: interrelating medical work and patient care information systems, *Methods Inf. Med.*, July 2002.
10. T. Moran and J. Carroll, eds. *Design Rationale Concepts, Techniques, and Use* Lawrence Erlbaum Associates, 1995.
11. Ash, J., Berg, M. and Coiera, E. Some unintended consequences of information technology in health care: the nature of patient care information system related errors. *Journal of the American Medical Informatics Association* vol 11, n.2, 2004
12. Berg, M. Practices of reading and writing: the constitutive role of the patient record in medical work. *Sociology of Health and Illness*. Blackwell Publishers. vol 18 n. 4, 1996.
13. Grudin, J. Why CSCW Applications Fail. *Proceedings of the Conference on Computer-supported Cooperative Work.*, pp. 85-93, 1988

Shared Knowledge: The Result of Negotiation in Non-hierarchical Environments

Oriel Herrera¹ and David A. Fuller²

¹ Informatics School, Universidad Católica de Temuco,
Manuel Montt 56 Temuco, Chile

² Computer Science Department, Pontificia Universidad Católica de Chile
V. Mackenna 4860, Santiago, Chile

dfuller@ing.puc.cl, oherrera@uct.cl

Abstract. The knowledge building can be seen as a collaborative process of which negotiation is a fundamental aspect. The use of technology to support this process has been attempted in various groupware systems. However, there is no adequate support system for the process of negotiation, which generally relies on voting as the element for reaching agreement in decision-making. On the other hand, the best approaches to this problem have been formalised in learning environments where there is a clear hierarchical structure. When the environment is not hierarchical, new problems arise which require special attention. This article presents a model of knowledge building that has negotiation as its basis in a group that is non-hierarchical in its structure. The model is implemented on a prototype tool named ShaKnoMa, which is tested on common tasks in an environment such as that proposed here. For the knowledge representation, concept maps are used which act a scaffolding to classify, index and search the information.

Keywords: Knowledge-Building, Concept Maps, Group, Visual Language.

1 Introduction

The collaborative knowledge building covers various stages and actions that the members of a group must carry out. Within this collaborative process of knowledge building, a fundamental aspect is negotiation. This aspect becomes a social phenomenon in which the members of the group agree on that knowledge which they consider valid for the group. The valid knowledge of the group will correspond to the knowledge previously negotiated and which the group will now accept as shared knowledge. In this context, we find a scenario in which technological support can assume a leading role. Computer-Supported Cooperative Work (CSCW) is an application field that has dealt with this subject matter, initially orientated towards group decision support systems (GDSS). Hence, negotiation can be seen through various approaches [1]: voting, decision approval [2], access permission [3], intertwining of perspectives [2]. Voting is the central mechanism in each of these approaches. Those who make decisions do so on the basis of previous knowledge, which is considered valid by each

person individually. However, negotiation has much deeper and more complex implications than simply deciding based on a vote. In learning environments, negotiation in knowledge building acquires other connotations. Voting loses its prominence, and now the process of knowledge building on the basis of negotiation among the work group's participants becomes the most important. The new knowledge begins to be negotiated, so each member shares in a reflective process, incorporating new concepts into his preconceptions. The article proposes a model of knowledge building for groups with a non-hierarchical structure, based on the negotiation of knowledge between participants. Such negotiation and representation are carried out using a visual language based on concept maps, reaping all the benefits of Ausubel and Novak's theory of learning [4]. In section 2 the representation diagram of knowledge and its life cycle are described. In section 3 the aspect of knowledge negotiation is dealt with in-depth. In section 4, general aspects of a prototype, which implements the model, are described. Finally, some results and conclusions are presented.

2 Representation of Knowledge

2.1 Visual Language

In order for knowledge to be defined and subsequently used, it must be represented explicitly. For this purpose, the proposed model defines a visual language based on concept maps. A concept map is a graph that consists of a combination of labeled nodes, linked by arcs, which may also be labeled. [5]. These were initially defined by Novak and Gowling [6], who established that concept maps are a vehicle for representing on "paper" the neuron connections in our brain, which facilitates the understanding of the internal processes of the mind. The success of their use lies in their simplicity and user-friendliness, due to the fact that human beings by nature have a remarkable capacity for employing and interpreting symbols. Software like Belvedere [13] uses this approach. This language has two main components: Concepts (nodes) and Relationships (arcs). The group defines a knowledge domain in which will work. For this domain, a project P is initialized, where P is defined as $P = \{CM\}^+$, that is to say, each project will be conformed by a finite group of one or more concept maps CM . On the other hand, a conceptual map CM is defined as $CM = \langle \{C\}^+, \{R\} \rangle$, that is to say, a tuple composed by a finite group of one or more concepts C and a finite group of relationships R . Now, each concept C will have the form $C = \langle N, M, E, \{Q\}, \{A\}, \{L\} \rangle$, where C is a tuple composed by a name N , a meaning M , one or more examples E , a set of questions Q , a set of attached files A , and a set of links L to other maps CM . Each one of these elements is specified as follows:

N = string, comprises a group of one or more descriptive words that explain the concept C ; M = string, refers to a description that defines the concept C within the context in which the group is working; E = string, corresponds to one or more examples that reinforce the meaning M of the concept C ; $Q = \{p \text{ string} / C \text{ gives answer to } p\}$, comprises a question, which can be responded by the information related to the concept C which allows more flexibility and speediness in search process; A = file, comprises one or more files attached to the concept C that gives

support to C ; $L = \{e \text{ link} / e \text{ links to another } CM\}$, is a link that allows to connect the concept with another concept map CM of the project P . In the same way, each relationship R can be defined as $R = \langle N, J, E, W, \{Q\}, \{A\}, \{L\} \rangle$, that is to say, a tuple composed by a name N , a justification J , one or more examples E , a key word W , a set of questions Q , a set of enclosed files A , and a set of links L to other maps CM . The elements N , E , Q , A and L have the same meaning than the concept's elements. $J = \text{string}$, is the description that explains the reasoning behind the relationship between the concepts in the knowledge field under study. $W = \text{string}$, corresponds to a key word belonging to a lexicon [8] which models the types of links.

In figure 1, a concept map belonging to a project is showed. The concept map is composed by three concepts and two relationships. Therefore, $CM = \langle \{\text{Courseware, Distance Learning, Tools}\}, \{\text{Oriented, Include}\} \rangle$. In this case, the concept *Courseware* is selected. On the right-hand side the elements associated with the concept are shown: $N = \text{"Software package to supplement or replace traditional course activities"}$, $E = \text{"WebCT and Learning Space are the most popular coursewares"}$, $A = \{\text{URI, URI, Document, Image, Document}\}$, L , there are not links to other maps for this concept.

Relationships are associated with certain semantic elements. On the one hand, the author is free to impose any expression for labeling the relationship, and this facilitates the creation and interpretation of knowledge [7]. On the other hand, the author must select from a dictionary the type of relationship which he is creating, and this facilitates the search for and inference of knowledge [8]. These semantics are based on what Turoff propounds on the basis of a complete taxonomy of nodes and links based on Guilford's *intellect theory* [9]. Therefore, it become very easy to introduce network analysis techniques and structural modeling methods that can be applied to aid the users in getting a computer supported analysis of the web [10].

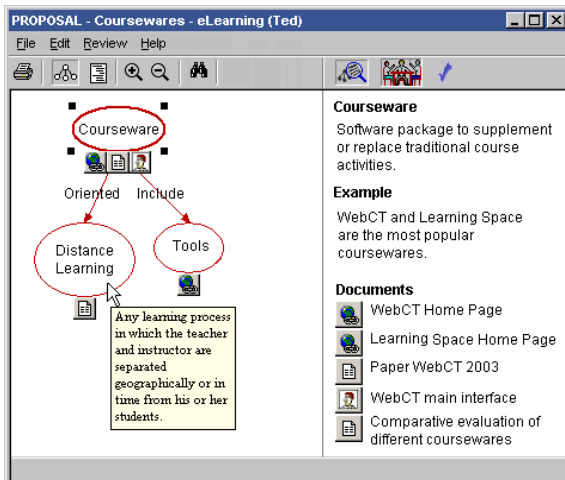


Fig. 1. Interface showing a concept and its associated elements

2.2 Interaction Model

We understand by *knowledge* in this context every concept, or relationship between concepts, or structure of both, with their components. Figure 2 shows the life cycle of knowledge, which defines the interaction model of the group.

Each of the arrows represents the taking of a decision, which causes the knowledge to change its state. The life cycle of knowledge involves seven main states, represented in figure 2 by the rounded rectangles. The initial state of knowledge is the *Proposal*, which reflects the participant’s intention to incorporate new knowledge into the shared knowledge repository. Whoever makes the proposal has control over it, and is who decides whether to present it to the group or not. In this state, the participant works from an individual perspective, incorporating those elements determined by his own experience. In the *Under Approval* state, the knowledge is subject to discussion. Here the components of the knowledge (*name, meaning/justification, examples, questions, attached files, links*) can undergo modifications derived from negotiation of the group. In the *Under Revision* state, the participant who made the proposal submits the knowledge for review, incorporating the changes derived from the negotiation. The states *Approved*, *Outdating Process*, *Outdated* and *Rejected* are the outcome of the negotiation (for more detail, see [11]).

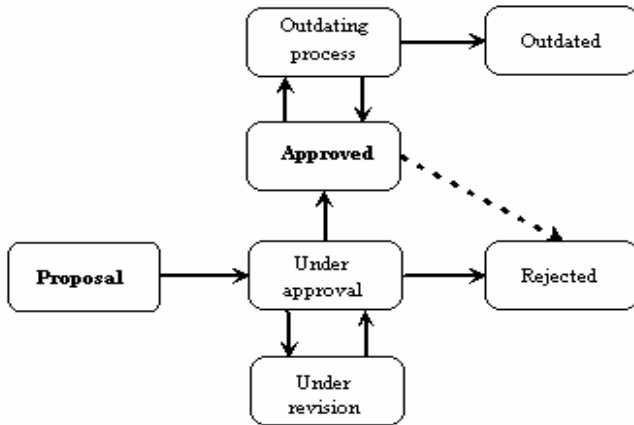


Fig. 2. Life cycle of knowledge defined by group interaction

3 Negotiation of Knowledge

Let us imagine a scenario in which a participant makes a proposal to the group, which is negotiated and ultimately approved. Figure 3 shows a trace diagram which describes the interaction that is produced from the time that a participant makes the proposal until it is approved and incorporated as shared knowledge. The group revises the proposal, suggesting changes to the participant who made it, or making comments on the suggestions of other members of the group.

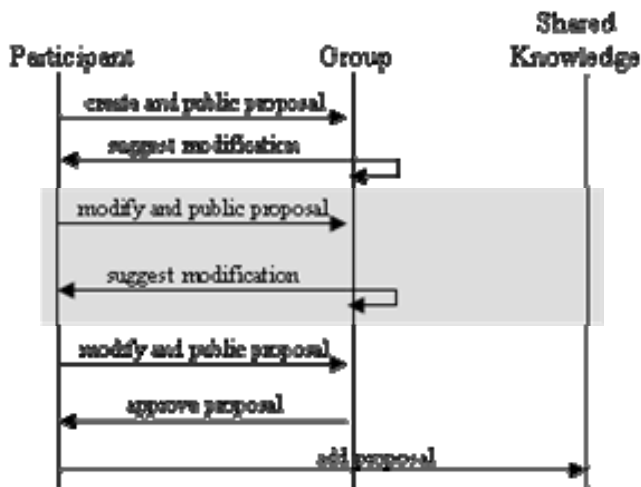


Fig. 3. Trace diagram of group negotiation to incorporate a knowledge proposal into the repository of shared knowledge

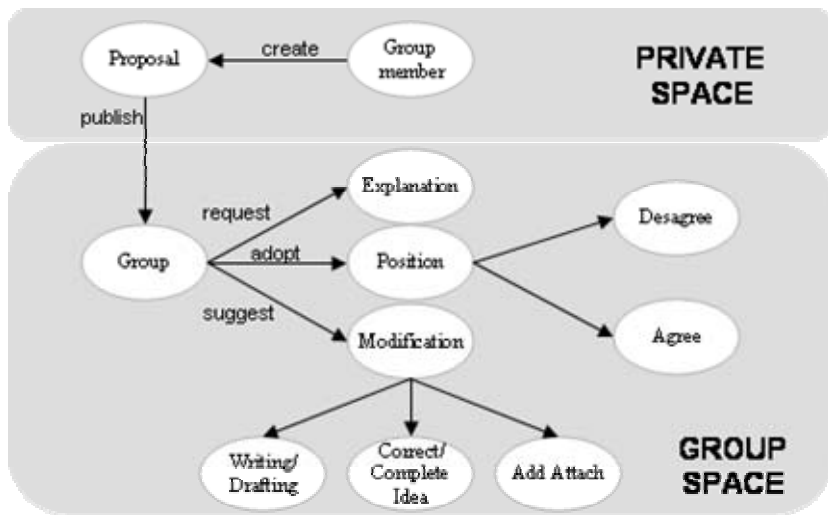


Fig. 4. The possible actions with respect to participants’ proposals in the negotiation process

The participant who made the proposal assimilates changes as often as necessary (cycle with a grey background in figure 3), until agreement is reached, allowing for the incorporation of the proposal into the shared knowledge.

In the negotiation process, group participants can take various actions. Each participant individually makes relevant contributions within the group space (see figure 4).

The model suggests a scaffolding to guide the discussion process through categories of revisions: *Request an Explanation* (require the author to explain some

element included in the proposal), *Suggest a Modification* (it could corresponds to writing/drafting, completion or correction of an idea, and incorporation of attached files), *Adopt a Position* (voting). Once the user has voted, he cannot revise the proposal, unless he withdraws his vote.

4 Prototype

All the elements corresponding to the group's work are associated with a project. A participant creates a new project and invites the rest of the group to join it. A

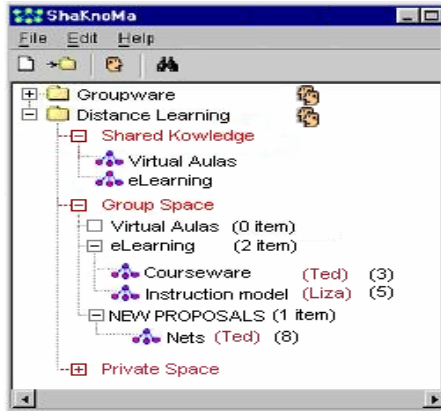


Fig. 5. Organization of the workgroups based on projects. The user is participating in two projects (Groupware and Distance Learning), each with a different group.

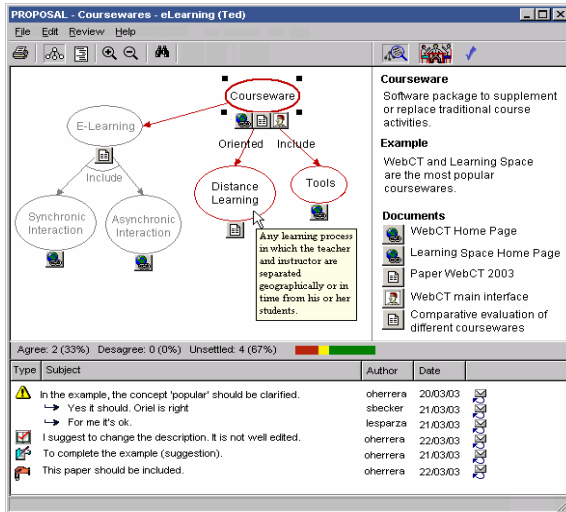


Fig. 6. Interface showing a new proposal and the negotiation of the group

participant can have access to various projects: those he himself has created as well those to which he has been invited. In the figure 5 two projects are shown in which the user is participating (Groupware and Distance Learning).

Each project consists of three workspaces: Shared Knowledge, Group Space and Private Space. The Shared Knowledge corresponds to the knowledge which the group has already negotiated and which is considered valid knowledge. The Group Space corresponds to the proposals that the group participants have submitted for discussion and revision; and it is from here that proposals are admitted into the Shared Knowledge. Each participant has a Private Space where he/she creates proposals before submitting them for revision.

Each proposal is visible to the entire group. In figure 6 the revision interface of user Ted's proposal, called *Courseware*, can be seen, containing three new concepts that are proposed for incorporation into the *eLearning* map. The participants can see all the information associated with each concept and relation-ship. The proposal continues to evolve according to the negotiation of the group. The author is responsible for making the changes to the proposal until a consensus is arrived at.

5 Results and Conclusions

Some experiments, according to the traditional methods used in social sciences, were done; meaning experiments with internal and external validity, so the results can be generalized, trying to control possible variables that can be presented (history, experience, selection, etc.). The experiment "pre-test, post-test with control group" was used, which is one of the mostly used in these situations [12].

A summary of the results is presented. In order to demonstrate three different hypothesis, we worked in parallel with a group of senior students majoring in education and a group of students in computer science. A control group for each one of these groups was considered. The first hypothesis refers to the knowledge building, regarding quantity, quality and knowledge distribution. Preliminary results are:

- The groups that used the proposed model generated more documents, including some written by themselves and some obtained from Internet and digital libraries.
- The quality of the knowledge (evaluated by the professor) was similar.
- The group that used the model had a better mastery of the knowledge. In addition, each student mastered the documents, while in the control group certain documents were not known by some students in the group.

The second hypothesis aims to the availability of knowledge, regarding search activi-ties, summary and material preparation in a collaborative form. Some results were:

- All groups required face to face sessions to distribute the work.
- The groups that used the model had the information centralized. They manifested more satisfaction with the access to the information than the control groups.
- All groups delivered the total of the requested information.

The third hypothesis refers to the interaction of the group. The results were:

- The tool was classified by the students as friendly. It facilitates the interaction, but puts bureaucratic obstacles when facing obvious decisions.
- Inside the university the performance of the tool is acceptable, but when connected off campus the connection is slower.
- The model did not generate conflicts among the students.

For a work group, being able to define what can be considered shared knowledge is an essential task within the collaborative effort. Furthermore, if the interaction of the participants is made in an asynchronous distributed scenario, technology plays a very important role. A model of knowledge negotiation for asynchronous non-hierarchical environments has been presented, based on concept maps, which is given expression in a computer tool. This focus allows members of a group working separately to construct knowledge in a particular field, reaping the very benefits presented by Ausubel and Novak's theory of learning. The first trials are being carried out with a group of students in their final year of a program in the education field, who must develop their degree-qualifying seminar. It is also being applied to a research group, which defines fields of knowledge that form the basis of research projects and written articles.

References

1. Stahl, G. (2003). Negotiating Shared Knowledge in Asynchronous Learning Networks. Proceedings of Hawaii International Conference on System Sciences (HICSS-36), USA.
2. Stahl, G. & Herrmann, T. (1999) Intertwining perspectives and negotiation, In: Proceedings of International Conference on Supporting Group Work (Group '99), Phoenix, AZ.
3. Wulf, V., Pipek, V., & Pfeifer, A. (2001) Resolving function-based conflicts in groupware systems, *AI & Society*, 15, pp. 233-262
4. Ausubel, D. P., Novak, J.D. and Hanesian, H. (1978). Educational psychology: A cognitive view. 2nd edition. New York: Holt, Rinehart, and Winston. Reprint, 1986.
5. Lambiotte, J., Dansereau, D. Cross, D. y Reynolds S. (1984). Multirelational Semantic Maps. *Educational Psychology Review* 1(4): 331-367.
6. Novak, J., and Gowling, D., (1984) *Learning How To Learn*, New York, Cambridge University Press.
7. Brown, E., & Chignell, M. (1995). End user as developer: Free-form multimedia. In Edward Barrett and Marie Redmond (Eds.), *Contextual media: Multimedia and interpretation*. MIT Press, Cambridge, MA: pp. 189-211.
8. Turoff, M., Rao, U., Hiltz, S. (1991) Collaborative Hypertext in Computer Mediated Communications. *Proceeding of the XXIV Hawaii International Conference on System Sciences*. Vol. IV, pp.357-366.
9. Guilford, J. (1967) *The Nature of Human Intelligence*. McGraw-Hill Publishers.
10. Lendaris, G. (1980), *Structural Modeling: A Tutorial Guide*, *IEEE Transactions on Systems, Man & Cybernetics*.
11. Herrera, O., Fuller, D. (2000). Soporte al Proceso Colaborativo de Creación y Uso de Conocimiento en Grupos con Estructura no Jerárquica. *Proceedings de la XXVI Conferencia Latinoamericana de Informática, CLEI 2000, Septiembre, México DF*.
12. Campbell, D. T., Stanley, J.C. (1977) *Experimental and quasi-experimental designs for research*. Rand McNally College Publishing Co., Chicago, IL.
13. Paolucci, M., Suthers, D., Weiner, A. (1995) *Belvedere: stimulating students' critical discussion*. *Proceeding of Conference on Human Factors in Computing Systems*, pp 123 – 124, Denver, Colorado, United States.

A Mediation Model for Large Group Collaborative Teaching

María Ester Lagos, Miguel Nussbaum, and Francisca Capponi

Department of Computer Science,
P. Universidad Católica de Chile, Santiago, Chile
mn@ing.puc.cl

Abstract. The incorporation of computer resources into the classroom has given rise to the need for understanding how such technology can be used to achieve effective teaching practices. Collaborative learning aided by wirelessly connected mobile computers is a work mode that has generated benefits for student learning in small-group activities. It is therefore of interest to investigate whether these benefits are also present when working in large groups such as an entire class. This paper proposes a Mediation Model for Large Group Collaborative Teaching based on the interactions that occur between the components of a classroom technologically mediated using wirelessly connected mobile devices. A particular case of the model is presented, followed by an example of it in the form of a collaborative activity.

1 Introduction

The mobility and size afforded by handhelds (PDAs) facilitates their utilization in any classroom or other location on school premises, meaning that they can be employed more extensively than PCs installed in a laboratory [7]. Thus, from occasional use as a computer supplement they can transition to a frequent and integrated role [4].

Roschelle et al. [6] have suggested a taxonomy for wirelessly connected handheld devices that reveals their impact on face-to-face collaboration and how they influence learning, motivation, commitment and the development of mutual understanding. Complementing the foregoing, Zurita and Nussbaum [9] have demonstrated that wirelessly connected mobile devices support collaborative activity by supplying a space for negotiation and coordination between the different states of an activity through the mediation of participants' synchronization and interactivity.

However, the benefits of wirelessly connected mobile devices have only been demonstrated for learning by students in small groups, where building discussion and consensus is much simpler to achieve [8]. It remains an open question whether these benefits are also obtainable by large groups such as a whole classroom. If so, it would permit the development of more robust and more varied ideas [5] and enable a diversity of visions on the part of the students and a richer pedagogical labor by the teacher [2].

To understand how this technology can be used in collaborative teaching with large groups, we must investigate the interactions that emerge between the components of a classroom (teacher, students, groups, etc.) and the incidence of group size on the

students’ work and learning, so that mediation between the components can then be utilized to create a classroom in which the students collaborate effectively.

The present study proposes a Mediation Model for Large Group Teaching based on the interactions that arise among the components of a classroom that is technologically mediated by wirelessly connected mobile devices. A particular case of the model is also presented and illustrated by an example application.

2 Mediation Model

Our Mediation Model assumes interaction to be the basic unit that occurs among various actors in a classroom with technological support. The roles of the different actors in the established dynamic will have to be elucidated and the requirements for implementing a given mediation model in the classroom will then be determined.

To define how and between whom information flows in a classroom technologically mediated by mobile devices connected to a wireless network, we define an Interaction Model for the main classroom components (see Fig.1).

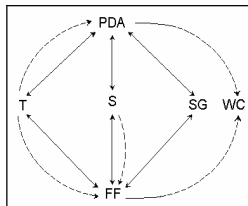


Fig. 1. Interaction Model

The components of the model as shown in Fig.1 are the following:

- **Actors:** Persons or groups of persons among whom information flows. They are information emitters and receptors, and include:
 - Student (S)
 - Whole Class (WC), comprising the N students in the class
 - Small Group (SG), a subset of S students in the class, which consists of N/S groups
- **Mediators:** The entities through which the information flows. Since the information is transformed in the process, they are more than just communication channels:
 - Personal Digital Assistants (PDA). PDAs act as instruments that support and regulate relations between actors, and provide:
 - organization of the information
 - a negotiation space
 - coordination between activity states
 - Face-to-face relationships (FF). The human medium in which information is exchanged, impacting on the students’ commitment to their responses and their group as well as on the development of mutual understanding between the different actors.

- *Actor and mediator*: Teacher (T). In addition to sending and receiving information, the teacher is responsible for electing the curriculum activities to be implemented and for guiding the students toward the achievement of the desired goals. The teacher is also in charge of delivering feedback to the students and filtering the information flowing between them and among the groups in such a manner that the discussions take the desired course.

The interactions in the model, as shown in Fig.1, are centered on the student and use PDAs and a face-to-face relationship (FF) as mediator components for the interactions with the other actors: the teacher (T), the small group he or she belongs to (SG) and his or her class (WC). The teacher also acts as mediator of the interaction and communication between the students (S), the small groups (SG) and the class (WC). The arrows connecting the components indicate the direction of information flow.

The set of valid interactions in the model is enumerated below:

- T→PDA→S : Teacher sends information to student in form of question or exercise.
 T→PDA→SG : Teacher sends information to small group in form of question or exercise.
 T→PDA→WC : Teacher sends information to class in form of question or exercise.
 T→FF→S : Teacher speaks with student to explain or mediate a specific point.
 T→FF→SG : Teacher speaks with a specific small group, mediating its concerns.
 T→FF→WG : Teacher speaks with whole class to give a general explanation.
 SG→PDA→S : Small group sends information to student to exchange coordination information.
 SG→PDA→T : Small group sends information to teacher in response to question sent.
 SG→FF→S : Small group discusses with student to exchange ideas and arrive at agreed response.
 SG→FF→T : Small group speaks with teacher to request information, an explanation or his/her mediation.
 S→PDA→T : Student sends information to teacher in response to question or exercise sent.
 S→FF→T : Student discusses or requests information or help from teacher.
 S→PDA→SG : Student sends information to small group to exchange coordination information.
 S→FF→SG : Student discusses with small group to which he/she belongs to arrive at agreed response and/or for asking help.
 S→FF→WG : Student discusses with class to defend his/her response or that of his/her group and/or to support a given student.

Note that the relation of student to student is included within the relation between student and small group or within the relation between student and whole group.

These interactions are the key element in understanding the individually and group-generated learning process. They also show that in a technologically mediated classroom with wirelessly connected mobile devices, three work modes can be implemented: individual mode (I), small group mode (SG) and whole class mode (WC).

The *individual work mode (I)* teaches the student individual responsibility by requiring him or her to justify his or her work to the teacher and the next work unit, the small group and/or the class group. In this mode interactions occur between the student (S) and the teacher (T).

The *small group work mode (SG)* is useful for various reasons. First, by emphasizing learning in peer groups the teacher will have more time to help students who are having difficulties and to assign enrichment activities to students who have already

mastered the prescribed material. Second, it gives teachers greater flexibility to adapt the learning and teaching objectives to individual learning needs. Third, students who learn together in small groups may be motivated toward cooperation instead of competitiveness. Fourth, the obligation and the opportunity to work with others in order to learn allows students to develop social and communication skills [1].

The *whole class work mode (WC)* teaches both individual and group responsibility. Each student assumes responsibility for his or her response to the class, and the class response is then built from those given by the individual students as mediated by the teacher. However, because the teacher must attend to the needs of a whole class, the emphasis is on uniformity rather than diversity and individual explanations [1].

In short, we may distinguish three work modes (see Fig.2): individual work (I), collaborative work in small groups (SG) and collaborative work involving the whole class (WC).

Taken together, the work modes (Fig.2) and the set of possible interactions (Fig.1) generate a number of possible classroom work sequences as shown in Fig.3:

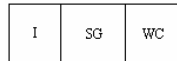


Fig. 2. Classroom work modes

I	SG	WC
$T \rightarrow PDA \rightarrow S$ Teacher sends a task to the student $S \rightarrow FF \rightarrow T$ } $T \rightarrow FF \rightarrow S$ } Teacher mediation $S \rightarrow PDA \rightarrow T$ Student sends answer to the teacher	$T \rightarrow PDA \rightarrow SG$ Teacher sends a task to the group $S \rightarrow FF \rightarrow SG$ } $SG \rightarrow FF \rightarrow S$ } Discussion $S \rightarrow PDA \rightarrow SG$ } $SG \rightarrow PDA \rightarrow S$ } $SG \rightarrow FF \rightarrow T$ } Teacher mediation $T \rightarrow FF \rightarrow SG$ } $SG \rightarrow PDA \rightarrow T$ Group sends answer to the teacher	$T \rightarrow PDA \rightarrow WC$ Teacher sends a task to the class $S \rightarrow PDA \rightarrow T$ Voting $T \rightarrow PDA \rightarrow S$ } $S \rightarrow FF \rightarrow T$ } Justification $S \rightarrow FF \rightarrow WC$ } Argumentation $T \rightarrow FF \rightarrow WC$ } Teacher mediation

Fig. 3. Classroom work sequences

In the individual work mode (I), information is sent and received by the teacher through the technological network and his or her mediation in person with the student. In the small group work mode (SG), sending and receiving information between the teacher and the group also takes place through the technological network. Collaborative work in this case takes the form of face-to-face discussion generated by exchanging information for purposes of coordination, synchronization and negotiation, technologically mediated by PDAs. Finally, in the whole class work

mode (WC), student voting and the justification of responses to the class as mediated by the teacher are the basis for group discussion [3]. These characteristics of the three modes can be generalized in the form of the model displayed in Fig.4.

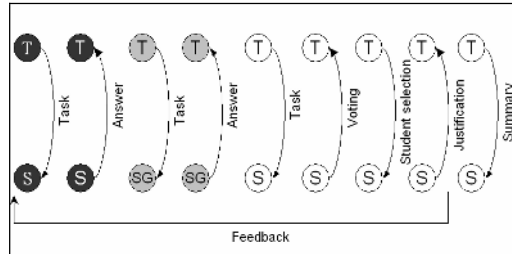


Fig. 4. Classroom Mediation Model

The interaction flows observable in Fig.4 are as follows:

- 1) **T→S:** The teacher chooses a subject and sends certain information to the students, who may ask a question..
- 2) **S→T:** The students send the teacher responses that express their individual positions.
- 3) **T→SG:** Small groups are created and the teacher sends each group information resulting from the individual work. (Discussion takes place within the groups.)
- 4) **SG→T:** Each group sends its response to the teacher.
- 5) **T→S:** The teacher chooses certain responses from those received and sends them to the students.
- 6) **S→T:** The students vote on the responses.
- 7) **T→S:** The teacher sends a justification notice to a randomly chosen student regarding his or her response.
- 8) **S→T:** The student justifies his or her response to the class, with the teacher mediating.
- Feedback** from the cycle (return to first step until teacher decides to end the activity).
- 9) **T→S:** The teacher sends the students a summary containing the main ideas of the subject the students worked on.

In the mediation model in Fig.4, the three work modes can be observed as one moves from left to right: individual (steps 1 and 2), small group (steps 3 and 4), and whole class (steps 5 through 9). The order of the steps is not strict, however. The sequence in Fig.4 is presented in such a manner that the activity displays a logical flow, and is intended as a proposed ordering of the interactions that preserves the natural flow of the classroom activity.

3 An Example of the Mediation Model

To illustrate the application of the Mediation Model in Fig.4, we analyze a subset of it in which the teacher works with students individually and then with the whole class, without using small groups (see Fig.5).

The example we use for this illustration is an activity designed to teach students about sound in the context of a physics course at the secondary school level. The purpose behind the activity is to get the students to discover the relationships between the various characteristics of phenomena such as vibration, reflection, light, frequency,

and wave by using sound as a unifying concept. Thus, the work model involves relating content through concepts and characteristics. The students must match a concept to characteristics sent to them by the teacher, and then hold a group discussion using a voting mechanism mediated by the teacher to eliminate responses that do not match.

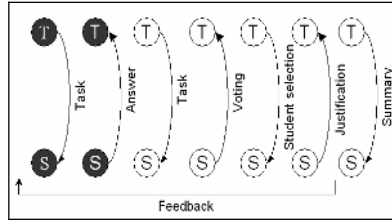


Fig. 5. Example of Mediation Model

We now analyze this application by following the flow of interactions.

- 1) **T→S**: The teacher chooses a subject and the concept to be found, and then *sends* each student a characteristic (Fig.6a).
- 2) **S→T**: Each student *sends* the teacher his or her *response*, that is, the concept that matches the characteristic received. Fig.6b shows the introductory screen seen by the students upon receiving the activity, while Fig.6c, d and e display the screens of three different students in which they must write the concept they believe matches the indicated characteristic.

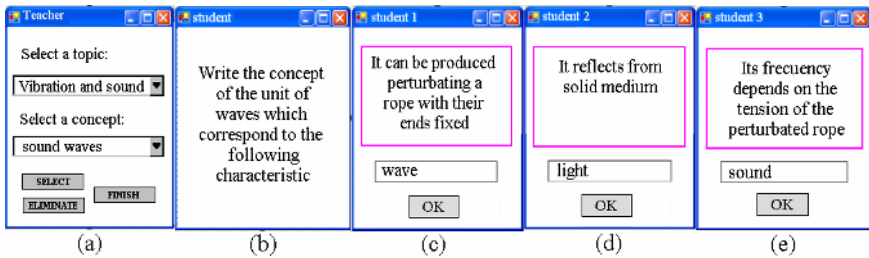


Fig. 6. Individual Work

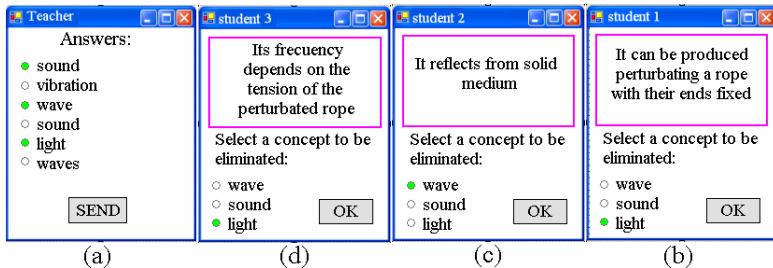


Fig. 7. Class Work – Voting

- 3) **T→S**: The teacher chooses certain of the responses received and *sends* them to all students in the class (Fig.7a). Then on each student’s screen there appears the characteristic he or she had originally received and the list of responses sent by the teacher for reconsideration (Fig.7b, c, d).
 Each student must then eliminate the concepts that do not match the characteristic on the screen. But now they see the responses of their classmates and thus have a broader view of the problem. They then decide whether to reaffirm or change their original response. Student No. 1, for example (see Fig.7b), will see that “light” is the incorrect concept and vote to reject it by pressing the “OK” button.
- 4) **S→T**: The students *vote* either for a response they want to eliminate or for the correct response, depending on the specific case. Both the teacher (Fig.8a) and the students (Fig.8b, c, d) see a graphic on their screen showing the votes obtained for each concept.
- 5) **T→S**: The teacher (Fig.8a) then chooses at random one of the students who voted for a given concept and *sends* him or her a *justification notice* requiring that he or she justify that vote, thereby stimulating a dialogue with the class (“JUSTIFY” button).
- 6) **S→T**: The randomly chosen student is notified by the appearance of a red screen on his or her PDA (Fig.8c). The student then *justifies* his or her response to the whole class, thus initiating a class *discussion*.
- 7) **T→S**: The teacher *sends* the students a *summary* containing the *main ideas* of the subject the students worked on. Once feedback has taken place, the teacher sends the students new characteristics relating to the concept they must find, and the students have the option of adding the correct concept if it is not on the list of responses given by his or her classmates.

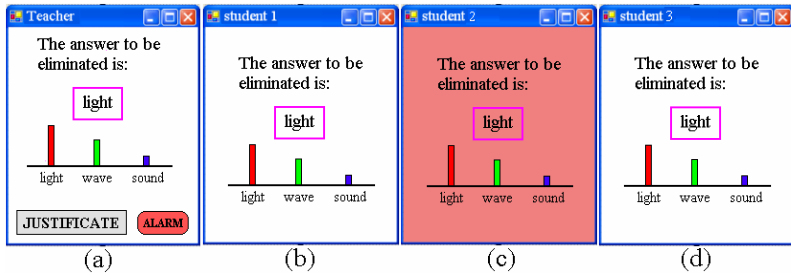


Fig. 8. Class Work – Discussion

4 Conclusions

We have presented a general model of technological mediation for the classroom that was designed on the basis of the possible interactions. The model enables us to design collaborative activities both for small groups and a whole class, incorporating individual work as one more element. This characteristic is not generally found in CSCL studies.

The Mediation Model given in Fig.4 constitutes a contribution for teachers because it demonstrates how technology can be used to mediate interactions between actors in

the classroom. It enables activities to be designed to fit the curriculum while developing social and communication abilities among students.

Progress on our research is currently in the implementation stage of the educational application described in this paper. This and other examples will be validated so as to obtain a set of such applications that will allow us to draw both quantitative and qualitative conclusions regarding the support provided by wirelessly connected PDAs to the set of valid interactions we have defined.

Acknowledgments

This work was partially funded by FONDECYT grant 1040605.

References

1. Abrami, P., Lou, Y., Chambers, B., Poulsen C. & Spence, J. (2000). Why Should we Group Students Within-Class for Learning? *Educational Research and Evaluation*, Vol. 6, No.2, pp. 158-179.
2. Hegedus, S. and Kaput, J. (2002). Exciting new opportunities to make mathematics an expressive classroom activity using newly emerging connectivity technology.
3. Mazur, E. (1997). *Peer Instruction: A user's manual*. NJ: Prentice Hall.
4. Roschelle, J., & Pea, R. (2002). A walk on the WILD side: How wireless handhelds may change computer-supported collaborative learning. *International Journal of Cognition and Technology*, 1(1), 145-168.
5. Roschelle, J., Abrahamson, L. & Penuel, W. (2004). Integrating Classroom Network Technology and Learning Theory to Improve Classroom Science Learning : A Literature Synthesis. Paper presented at the Annual Meeting of the American Educational Research Association, San Diego, CA, April 16, 2004.
6. Roschelle, J., Rosas, R. & Nussbaum, M. (2005). Towards a Design Framework for Mobile Computer-Supported Collaborative Learning. *Computer Supported Collaborative Learning Conference*, Taiwan, July 2005.
7. Vahey, P., & Crawford, V. (2002). *Palm Education Pioneers Program: Final Evaluation Report*. Menlo Park, CA: SRI International.
8. Zurita, G. (2003). *CSCL Activities and Mobile Technology*. Thesis submitted to the Office of Research and Graduate Studies in fulfillment of the requirements for the Degree of Doctor in Engineering Sciences, Pontificia Universidad Católica de Chile.
9. Zurita, G., & Nussbaum, M. (2004). mCSCL: Mobile computer supported collaborative learning. *Computers & education*, 42(3), 289-314.

Analyzing the Organization of Collaborative Math Problem-Solving in Online Chats Using Statistics and Conversation Analysis

Alan Zemel¹, Fatos Xhafa², and Gerry Stahl³

¹The Math Forum, Drexel University, 3210 Cherry Street,
Philadelphia, PA, USA
arz26@drexel.edu

²The Open University of Catalonia, Barcelona, Spain
fxhafa@uoc.edu

³College of Information Science and Technology,
Philadelphia, PA, USA
gerry.stahl@cis.drexel.edu

Abstract. In this paper we describe how a statistical test on a hypothesis regarding collaborative math problem solving using online chats showed an unexpected result, whose understanding required the use of qualitative methods. The phenomenon behind the result is identified using Conversation Analysis. This paper demonstrates the importance of using qualitative methods to describe the perspective of participants as a way of interpreting statistical results, revising hypotheses and developing alternative coding schemes and procedures. The combined approach of quantitative and qualitative methods is applied on real data coming from Virtual Math Teams research project (Drexel University) and is identifying issues not addressed so far in the analysis of online collaborative group activity.

1 Introduction

The analysis of the use of groupware is particularly problematic. Most methods of human-computer interaction were developed for single-user systems and are not applicable to computer mediation of group interaction. A common approach to analyzing the use of groupware is to compare statistical measures of usage across conditions or cases. However, this can be criticized for not investigating and taking into account qualitative differences that may be crucial to understanding the quantitative differences [1]. While there is a widespread feeling that fields like CSCL and CSCW need to take a multidisciplinary approach incorporating a variety of analytic methods, it is difficult to see how quantitative and qualitative approaches built on fundamentally incompatible theoretical foundations can work together. This paper reports a case in which a quantitative discovery led to qualitative analysis that explained the significance of the quantitative results and suggested modifications of the quantitative approach.

In the Virtual Math Teams (VMT, [2]) project at Drexel University, we investigate online problem-solving chat interactions from a variety of analytical and methodological perspectives. On the one hand, a coding scheme has been developed and applied to logs of online chats among actors participating in math problem solving. This provides a basis for a quantitative analysis of the chat logs. On the other hand, conversation analytic methods have been applied to these chat logs as a way of describing the procedures participants use to make sense of their ongoing activity.

Conversation analysis (CA) and statistical analysis (SA) are uneasy partners in the analytic enterprise. These two orientations to analysis derive from very different perspectives on the role of the analyst and the kinds of assumptions that can be made with respect to the data and its interpretation. In statistical analysis, hypotheses are put forward and tested. Coding schemes are devised which are designed to facilitate the testing of these hypotheses and statistical methods are applied to coded data. In this approach, it is the analyst's perspective that is privileged. The analyst:

- proposes the hypotheses,
- produces the coding scheme to capture the relevant data from an experiment designed specifically to allow for testing of the hypothesis, and
- assesses and interprets the statistical results [3].

Statistical analysis of data gathered from online collaborative learning experiments plays a central role in many CSCL studies [4], [5], [6], [7]. A whole range of statistical methods, from descriptive statistics to multilevel and other sophisticated methods have been used to analyze the underlying features (variables) of the collaborative activity that takes place in a small group.

Conversation analysis, on the other hand, is an analytical methodology that attempts to describe the actions of participants in terms of the relevances demonstrated by participants in and as their interaction [8], [9]. This methodology privileges the perspective of the participants over the analyst's perspective [10]. Actions are seen as situated within a stream of ongoing action and are sequentially organized. Furthermore, conversation analysts presume that actors design and 'customize' their action for the particular circumstances in which they are accomplished.

The differences between SA and CA are consequential. For statistical analysts, validity and reliability are significant concerns. These are not concerns for conversation analysts. Conversation analysts are concerned with providing adequate descriptions of the sense-making procedures used by participants as they interact. Where statistical analysts would discover what might be 'present' as frequently observed regularities in interactions, conversation analysts are concerned with how specific actions are made relevant by prior actions and how a current action make relevant subsequent actions over the course of a particular sequence of actions. For conversation analysts, it is sufficient that the participants in a particular interaction treat their ongoing actions as sensible. The conversation analyst's task is to describe these sequences of actions as sense-making procedures.

While these two types of analysis, statistical and conversational, may seem incompatible, it turns out there are circumstances in which they can be mutually informative [11]. In this paper, we describe a situation in which a puzzling statistical

result was made intelligible by conversation analytic investigation. This is a novel approach to analyze the organization of the interaction in collaborative math problem-solving activities in online chats. Indeed, existing approaches in the literature treat quantitative and qualitative methods separately. Our results show the strength of using a combined approach. Specifically, by using a quantitative approach, we detected an unexpected result in a hypothesis test. This made further investigation necessary. The qualitative method enabled us to identify the phenomenon that produced the unexpected result in the hypothesis test.

2 Data Collection

The Virtual Math Teams (VMT, [2]) project at Drexel University investigates small group collaborative learning in mathematics. In this project an experiment is being conducted, called *powwow*, which extends The Math Forum's (mathforum.org) "*Problem of the Week (PoW)*" service. Groups of 3 to 5 students in grades 6 to 11 collaborate online synchronously to solve math problems that require reflection and discussion. AOL's Instant Messenger software is used to conduct the experiment in which each group is assigned to a chat room. Each session lasts about one to one and a half hour. The *powwow* sessions are recorded as chat logs (transcripts) with the handle name (the participant who made the posting), timestamp of the posting, and the content posted.

2.1 Coding Scheme

Both quantitative and qualitative approaches are employed in the VMT project to analyze the transcripts in order to understand the interaction that takes place during collaboration within this particular setting. A coding scheme has been developed in the VMT project to quantitatively analyze the sequential organization of interactions recorded in a chat log. The unit of analysis is defined as one posting that is produced by a participant at a certain point of time and displayed as a single posting in the transcript.

The coding scheme includes nine distinct dimensions, each of which is designed to capture a certain type of information from a different perspective. They can be grouped into two main categories: one is to capture the content of the session whereas another is to keep track of the threading of the discussion, that is, how the postings are linked together. Among the content-based dimensions, conversation and problem solving are two of the most important ones which code the conversational and problem solving content of the postings. Related to these two dimensions are the Conversation Thread and the Problem Solving Thread, which provide the linking between postings, and thus introduce the relational structure of the data. The conversation thread also links fragmented sentences that span multiple postings. The problem solving thread aims to capture the relationship between postings that relate to each other by means of their mathematical content or problem solving moves (see Figure 1).

Each dimension has a number of subcategories. The coding is done manually by 3 trained coders independently after strict training assuring a satisfactory reliability.

Regarding the statistical approach, this paper considers 4 dimensions only; namely the conversation, problem solving, social reference, math move and system support dimensions.

Line #	Handle	Statement	Time	Conversation Thread	Conversation	Problem Solving Thread	Problem Solving
45	AVR	Okay, I think we should start with the formula for the area of a triangle	8:21:46		Offer		Strategy
46	SUP	ok	8:22:17	45	Follow	45	
47	AVR	$A = 1/2bh$	8:22:28		Offer	45	Perform
48	AVR	I believe	8:22:31	47	Extension	47	
49	PIN	yes	8:22:35	47	Setup	47	
50	PIN	i concue	8:22:37	49	Agree	49	Check
51	PIN	concur*	8:22:39	50	Repair Type		
52	AVR	then find the area of each triangle	8:22:42		Offer	45	Strategy
53	AVR	oh, wait	8:22:54		Regulation		
54	SUP	the base and heigth are 9 and 12 right?	8:23:03		Request		Offer
55	AVR	no	8:23:11	54	Setup	54	
56	SUP	o	8:23:16		No Code		
57	AVR	that's two separate triangles	8:23:16	55	Critique	55	Reflect
58	SUP	ooo	8:23:19	55	Setup	55	
59	SUP	ok	8:23:20	58	Response	58	

Fig. 1. A coded excerpt from Pow2a

2.2 Data Collection

The sample used in this study consisted of six powwows that were chosen from a larger set of powwows with the aim at conducting a first data analysis. The criteria for choosing the sample is based on one of the characteristics of the powwow experiment, namely, for some powwows the math problem was announced in advance while for some others the math problem was announced just at the time of starting the chat session¹. Thus, the sample of six powwows is made up of three powwows in which the math problem was announced at the beginning of the session, whereas in the rest the problem was posted on the Math Forum’s web site in advance² (see Table 1). It should be noted that for the math problem being announced in advance doesn’t necessarily mean that the participants of the chat already solved the problem in advance.

Table 1. Description of the coded chat logs

PoW-wow Session #	Facilitator	Members	Number of Utterances	PoW Name	Announced Before?
1	MUR	PIN, GOR, REA, MCP	334	Finding CE	No
2a	GER	FIR, PIN, SUP, OFF	724	Equilateral Triangle Areas	No
2b	MUR	MCP, AH3, REA	204	Equilateral Triangle Areas	No
9	POW	EEF, AME, AZN, LIF, FIR	715	Making triangles	Yes
10	MFP	AME, FIR, MCP	582	The perimeter of an octagon	Yes
18	MFP	AME, KOH, KIL, ROB	488	A tangent square and circle	Yes

¹ We will refer to this as “known – not known” criterion.

² We will refer to the first group as “NO group” and to the second as “YES group.”

3 Statistical Analysis

3.1 First Level: Statistical Analysis Based on Main Dimensions

Our first objective was to test whether there is any significant effect of the “known – not known” criterion on the sample of the powwows. To this end, we started by computing³, through descriptive statistics, the distribution of frequencies in different dimensions (*Conversation, Social Reference, Problem Solving, Math Move* and *System Support*) for the six Powwows and used Means and ANOVA⁴ to test the existence of significance difference due to the “known – not known” criterion. The study showed that there was no such effect, at a usual confidence level of 95% (in fact, significance in differences, that is significant pairs, were not noticed even at 90% confidence level). The fact that there is no clear effect of the criteria “known – not known” allows us to conclude that the classification of the sample of Powwows into groups according to “known – not known” criterion is not relevant. We could also observe this by computing the Boxplot representation of the variables under study (see Figure 2).

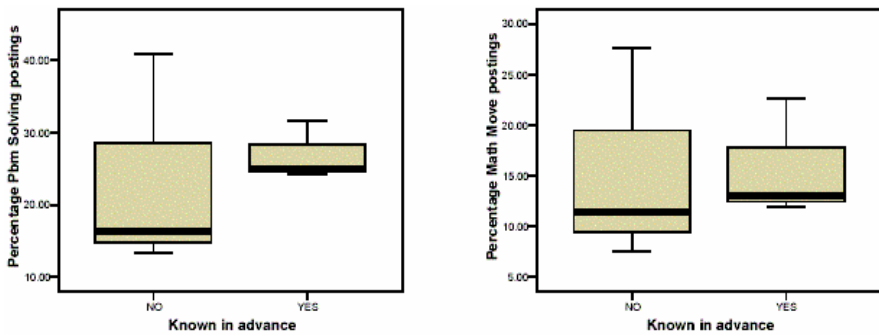


Fig. 2. Boxplot representation of Problem Solving and Math Move dimensions

Given the above finding, we refined the statistical analysis by looking at the correlation between vectors of values of the six powwows (grouping “known – not known” was now maintained just for visual effect). By computing similarities between the powwows we could see which powwows are similar to each other and which are different from each other. We computed thus the correlations (Pearson correlations) through proximity matrix shown in Table 2.

From Table 2 we observe the following:

- a) Pow2b (3:NO in the table) is *negatively* correlated to the powwows of the NO group (pow1 and pow2a) and *positively* correlated to the powwows of the other group (pow9, pow10, pow18). Moreover, significant correlation of pow2b with pow10 (5:YES) and pow18 (6:YES) is observed and not significant correlation with the pow9 (4:YES).

³ The statistical computations are done in SPSS 12.0.

⁴ Note that the different dimensions are independent of each other.

Table 2. Pearson Correlation of Vector Values of Six Powwows

Proximity Matrix

	Correlation between Vectors of Values					
	1:NO	2:NO	3:NO	4:YES	5:YES	6:YES
1:NO	1.000	0.756	-0.452	0.567	0.108	-0.197
2:NO	0.756	1.000	-0.219	0.912	0.603	0.067
3:NO	-0.452	-0.219	1.000	0.202	0.620	0.956
4:YES	0.567	0.912	0.202	1.000	0.867	0.470
5:YES	0.108	0.603	0.620	0.867	1.000	0.791
6:YES	-0.197	0.067	0.956	0.470	0.791	1.000

This is a similarity matrix

- b) There is a significant positive correlation of the pow9 with pow1 and pow2a of the NO group. In pair wise terms, pow9 is more correlated to the powwows of the NO group than to the powwows of the YES group (its group).
- c) There are some pairs of powwows positively and strongly correlated, namely (powwow2a, pow9) and (pow2b, pow18) which suggest taking a closer study of the possible common features of these powwows.

The previous observations on the correlations between powwows from different groups not only supports the claim that there is no significant effect of the “known – not known” criterion but also shed light on the reason why these two groups are not really separated. Indeed, the negative correlation of the pow2b with the powwows of the NO group shows that its place is not in the NO group. Even more, its positive correlation with the powwows of the YES group indicates that this powwow is better grouped with the powwows of the YES group.

In our next step, we decided to exclude the *System Support* dimension from the analysis; indeed, this dimension is less relevant in the context of the interaction analysis and could have thus introduced some noise in the analysis. We run again the statistical computations by re-computing the correlations through proximity matrix as shown in Table 3.

Table 3. Pearson Correlation of Vector Values of Six Powwows (system support excluded)

Proximity Matrix

	Correlation between Vectors of Values					
	1:NO	2:NO	3:NO	4:YES	5:YES	6:YES
1:NO	1.000	0.999	-0.427	0.868	0.376	-0.145
2:NO	0.999	1.000	-0.396	0.884	0.407	-0.112
3:NO	-0.427	-0.396	1.000	0.080	0.678	0.957
4:YES	0.868	0.884	0.080	1.000	0.787	0.366
5:YES	0.376	0.407	0.678	0.787	1.000	0.862
6:YES	-0.145	-0.112	0.957	0.366	0.862	1.000

This is a similarity matrix

By excluding the System Support dimension, we observe a clear effect on the correlations, namely:

- a) On the one hand, an increased negative correlation of the pow2b (3:NO) with the powwows of its group (pow1 and pow2a, 1:NO and 2:NO, respectively) is now observed. Notice also that the correlation between pow1 and pow2a is almost perfect correlation. On the other hand an increased positive correlation of the pow2b (3:NO) with the powwows of the other group (pow9, pow10, pow18) is observed. Interestingly, pow2b is now less correlated to pow9 (4:YES in the table).
- b) An increased positive correlation of Pow9 with the powwows of the NO group (pow1 and pow2a) is now observed. Moreover, we observe a decrease in its correlation with pow10 and pow18.
- c) Finally, pow18 is now negatively correlated to both pow1 and pow2a.

We repeated the above computations by standardizing the variable values by z-score.

Table 4. Proximity Matrix

	Correlation between Vectors of Values					
	1:NO	2:YES	3:YES	4:NO	5:YES	6:YES
1:NO	1.000	.987	-.999	.869	-.921	-.993
2:NO	.987	1.000	-.977	.778	-.845	-.999
3:YES	-.999	-.977	1.000	-.894	.939	.986
4:NO	.869	.778	-.894	1.000	-.993	-.808
5:YES	-.921	-.845	.939	-.993	1.000	.870
6:YES	-.993	-.999	.986	-.808	.870	1.000

This is a similarity matrix

According to the statistical computations indicated above, the powwows show the following two clusters:

- Cluster 1: (pow1, pow2a, pow9)
- Cluster 2: (pow2b, pow10, pow18)

By re-computing⁵ the Boxplot representation of this new clustering we could observe the significant separation between variables under study for the two groups (see Figure 3).

In other words, we expected the chat logs to be clustered based on the idea that in some chats, participants had access to the problem prior to their participation in the chat, while in other chats, participants had no access to the problem. However, the statistical analysis demonstrated that the clustering of chats was organized according to some other basis. At this point, we determined to conduct a qualitative approach to identify the reasons for this alternative organization of the online chats.

⁵ Compare to Figure 2.

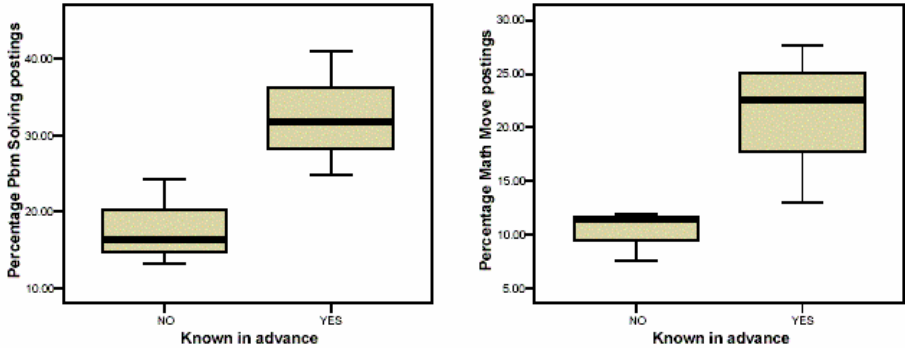


Fig. 3. Boxplot representation of Problem Solving and Math Move dimensions

4 Participation Frameworks and the Organization of Online Interaction

To discover possible reasons for the failure of our initial hypothesis, we reexamined the chats using Conversation Analysis (CA). With this approach, we examined logs of the online chats to identify participants' perspectives on their own actions with an eye to describing their actions as sense-making procedures. The work of conversation analysis involves close inspection of interactional data. In conventional face-to-face interaction, this involves inspecting video and audio recordings of interaction. When it comes to online chats, logs of the chats, which display the text postings of participants and the time stamp associated with each posting, are the data that are inspected.

“Conversation analysis studies the order/organization/orderliness of social action, particularly those social actions that are located in everyday interaction, in discursive practices, in the sayings/tellings/doings of members of society” ([9], p.2). The object of inquiry in conversation analysis is not exclusively conversation per se, but rather talk and social interaction. Thus, as Ten Have describes, “CA’s interest is with the local production of [social] order and with ‘members’ methods’ for doing so” ([8], p.19).

Using the methods of CA, we began to notice that the organization of social order in these chats could be differentiated according to the way that participants oriented to the production of problem solutions. In particular, we noticed that, in some circumstances, participants reported on *work they had already completed*, whether it was work done prior to the chat or work done offline and without the participation of others in the production of that work during a chat. This organization of participation we have termed **expository participation**. On the other hand, we noticed that there were circumstances in which participants engaged each other in the *conjoint discovery and production* of both the problem and possible solutions. This organization of participation we termed **exploratory participation**.

Expository participation in the chats we examined involved one actor producing a report as an extended narrative of an activity performed by that actor. Such reporting is designed to project recipient participation in terms of the production of assessments

of the report or the reported work. Recipients of that report have not participated in the work being reported. The report is designed and presented either as an already achieved understanding of the problem in terms of a candidate solution or as steps anyone with appropriate understanding of the problem might take to produce a solution. One version of expository participation is where one actor first announces that a solution has been achieved and then, upon prompting from recipients, proceeds to tell recipients what the solution is and how he or she produced the candidate solution. For example, an actor might report something like: "I've got the answer" which calls upon recipients to solicit the result. Announcing a result makes it relevant for recipients to ask for an explanation. Explanations might be offered in ways that describe the production of the solution as having been already achieved by the actor reporting the result, as in, "First I did ... and then I computed ... which equals ..." Another way to produce an explanation involves the circumstance where an actor describes how a competent person would go about solving the problem, as in "First you do ... then you compute ... which equals ..." In this regard, these approaches to the exposition of a problem's solution is much like the telling of a story (see e.g. [12]). This is illustrated in the chat excerpt below:

24 AH3 I think I have the solution!
 25 REA what
 26 MCP I guess 15
 27 REA k
 28 MCP I think it's like the Pythagorean idea, applying to triangles.
 29 AH3 $\sqrt{5^2 + 7^2} = \sqrt{74}$
 30 MCP Yes, 30-60-90 is needed fact
 31 AH3 The solution is $\sqrt{74}$
 32 REA how
 33 MCP 7?
 34 AH3 Go to...
 35 AH3 <http://mathforum.org/dr.math/faq/formulas/faq.triangle.html>
 36 AH3 Under scalene triangle, the formula for the area of any triangle is...
 37 AH3 $K = a^2 * \sin(B) * \sin(C) / [2 \sin(A)]$
 38 AH3 Why is that smiley their
 39 AH3 $K = a^2 * \sin(B) * \sin(C) / [2 \sin(A)]$
 40 AH3 Where a = an edgelength of an isosceles triangle

Fig. 4. Example of Expository Chat (Powwow 2b)

An expository report is a way that an actor constitutes a problem as solvable. This is, in fact, a position we support because there is evidence in the transcripts that actors themselves orient to these reports in just this way. For example, the actor producing the report treats the problem as having already been solved and thereby constitutes a participation framework in which that he or she acts in the manner of an instructor, explaining what is already known by the instructor to an audience that presumably does not yet know. Constituting such a participation framework is a delicate business

in the conduct of these chats. To do so, actors often draw upon the resources of news reporting by indicating they have something newsworthy to report, i.e., the solution to the problem. The actor reporting the solution designs his or her report in a way that allows the recipients of the report to “discover” in the report how the problem can be seen as solvable and solved.⁶

Exploratory participation, on the other hand, involves participation in which actors interact so as to constitute, in and as their chat, an understanding of a problem in terms of the conjoint production of possible organizations of mathematical activity from which a solution could be achieved. In such circumstances, actors use the resources afforded them by their interaction to constitute the math problem and their understanding of that problem as an emergent sequence of possible and/or achieved math activities designed to produce what may come to be subsequently recognizable and treated as a solution to the problem. If expository participation is a form of “news” reporting, then the distinguishing feature of exploratory participation is that the actors themselves are constituting the “news” as their ongoing interaction rather than reporting it and receiving the report. This is shown below:

- 119 MCP What's this extra saying? Like, if both of the smaller triangles are sitting on their bases, the base of one is 5 and the base of the other is 7?
Is that the interpretation?
- 120 REA I guess it is 10
- 121 MCP If they're oriented with corresp angles in corresp locations?
- 122 REA or should I say 9.8
- 123 REA what do you think
- 124 REA i used the proportions
- 125 MCP Oh, I guess this is where that 7 from AH3's answer came from,
way back there. I didn't know where that came from.
- 126 MCP I still need to make sure I know what the wording is saying. Am I interpreting
the q right?

Fig. 5. Example of Exploratory Chat (Powwow 2b)

Actors engaged in exploratory work do not have a solution in hand as they do when they are engaged in expository work. Instead, they work to discover ways to produce such a solution by 1) allocating participation among actors in the chat and 2) by constituting and drawing on resources for producing a solution that are distributed among participants and which are made available by actors' participation in the chat. Like expository participation, the work of exploratory participation also constitutes the problem in terms of its solution, but with exploratory participation the solution is not yet known to participants. Exploratory interactions involve putting forward proposals for consideration and assessment, negotiating ways of formulating the problem in terms of different solution strategies, quick exchanges among multiple participants rather than extended postings, etc. Thus the work of exploration involves something developing alternative understandings of the problem in terms of the development and assessment of alternative possible solutions.

⁶ This is similar to the way Livingston finds mathematicians doing proofs [13].

It is important to note that expository and exploratory work may be done during the same chat. Furthermore, expository participation requires that the expositor did the work of producing a solution “offline,” i.e. without the participation of other actors in the chat. One of the affordances of chat is that “offline” activities are possible even as a chat is occurring because participants only have access to the messages they post. An actor’s work with a pencil and a pad of paper beside his or her computer is not available to others unless and until it is posted in the chat system.

By examining the Powwow chats, we were able to see that there were considerable differences in the way participation was organized. Despite the fact that actors in Powwow 2b had not seen the problem in advance of their chat, they did their work “offline” during the chat and displayed an expository organization of participation in common with Powwows 10 and 18. Despite the fact that the actors in Powwow 9 had access to the problem in advance of the chat, they displayed an exploratory organization of participation in common with Powwows 1 and 2a. Thus, using CA, we were able to identify the same relation among the powwows showed by the statistical analysis and, moreover, explain the phenomenon in terms of the organization of participation in the chats.

One important question we considered was whether or not the coding scheme that had been used to identify these puzzling clusters initially could have been used to identify these different organizations of participation. We decided that it would not have been possible. The reasons for this decision are as follows. The existing coding scheme treated each post as the primary unit of analysis. Codes applied to individual chat postings but could not be used to characterize larger sequences of postings. This made it impossible to analytically identify the organization of participation which is understood as the relation among groupings of posted chat messages. While an alternative approach to coding might have made such an analysis possible, the work involved in developing such a coding scheme was formidable. Furthermore, it pointed to the logical problem of consistency that coding schemes are often designed in ways that lend themselves to find things for which there are codes. If we want to understand how participants organize their participation, if we want to understand a sequence of actions from participants’ perspectives, then coding schemes need to capture these perspectives rather than the perspectives and interests of the researcher.

5 Hypothesis Testing and Discovery

As this work has shown, analytically understanding social interaction can be a tricky business. The conduct of inquiry into social interaction has traditionally utilized theories and analytical methodologies that allowed the analyst to test hypotheses against a collection of coded data [14]. By proposing hypotheses and testing them against coded data derived from “real world” phenomena, analysts are presumed to be able to check the validity of their theories about social interaction. On occasion, anomalies appear. Unexpected results are either dismissed as “outliers” or other methods of analysis are deployed to provide some explanation.

Many of the problems associated with statistical analysis of social phenomena derive from the coding schemes that are used. The procedures for producing codes and for applying them to interactional data are sometimes problematic. One problem

is that the sense-making procedures analysts use to produce and apply codes are not independent of the sense-making procedures participants in the observed social interaction use to make sense of their ongoing activity. While this can be seen as a failure of coding schemes, it can also be viewed as a resource for doing initial investigations of social interactional phenomena which are then supplemented by close inspection of the sense-making procedures actors use. This is the perspective and approach we have taken in the VMT project.

Using the existing coding scheme as applied to six chat logs, we explored the hypothesis that we would expect to see a difference in the way that problem-solving chats were organized if participants did or did not have the opportunity to inspect the problem in advance of the chat. We hypothesized a difference could be detected and used statistical techniques to describe ways that the chats were grouped together. What we found was counter to the hypothesis. Rather than dismiss the results or question the value of the coding scheme, we opted to treat the unexpected result as an indicator of phenomena that required further investigation and closer analysis of the way that participants organized their activities in these chats.

6 Conclusions and Further Work

In this research, we were able to exploit the mutually informing features of quantitative and qualitative analysis. This has allowed us to discover a far more nuanced explanation for the observed grouping of chats. However, in order to determine whether our qualitative results provide an adequate explanation across multiple cases, we need to re-specify a coding scheme that derives from the perspective of the participants (for further discussion, see [14], [11]). According to [11] practitioners of CA have often made informal distributional claims with respect to observed interactional phenomena. However, certain questions about the ‘typicality’ or distribution of certain features of interactions of a particular type can only be assessed quantitatively. In such cases, questions arise as to the appropriate way to code data such that the requirements of valid statistical and quantitative analysis can be met without violating the requirements of preserving the sequential organization of, participants’ perspectives on and relevances with respect to emergent, unfolding action sequences.

Based on this research, we have begun to explore a ‘top-down’ approach to coding, based on the ways that interactants organize themselves and their interaction into recognizable activities. This approach uses CA methods to identify closings and openings of action sequences by which participants organize their activities into “long sequences” [12] of identifiable action types. For example, we have begun to identify sequences in which math problem solving activities are being conducted, as distinct from various other kinds of non-math social interaction. In this way, we are developing a coding scheme that preserves actors’ orientations, concerns, relevances and the sequential organization, of the ongoing interaction. This proposed approach to coding makes possible the comparison of different instances of social interaction in ways that preserve the local organization of interaction and exploit that local organization as a source of insight into the ways we come to treat action sequences as sequences of particular sorts.

This paper points the way to achieving an understanding of computer-mediated interaction among multiple participants. As this research has shown, questions concerning the ways that groups are formed and sustained through online interaction can be explored using multiple analytical methodologies as long as care and consideration are given to the differences in the assumptions that inform these methodologies. By using qualitative methods to explain an unexpected analytical result, we have shown how it is possible to interpret the organization of group participation in online interaction.

References

1. Stahl, G. (2002). Rediscovering CSCL. In T. Koschmann, R. Hall & N. Miyake (Eds.), *CSCL 2: Carrying forward the conversation* (pp. 169-181). Hillsdale, NJ: Lawrence Erlbaum Associates. Retrieved from <http://www.cis.drexel.edu/faculty/gerry/cscl/papers/ch01.pdf>.
2. VMT Project: <http://mathforum.org/wiki/VMT/>
3. Mason, R. L., Gunst, R. F., Hess, J. L. (2003). *Statistical Design and Analysis of Experiments: With Applications to Engineering and Science, 2nd Edition*. Wiley.
4. Dillenbourg, P., Baker, M., Blaye, A. & O'Malley, C. (1996) The evolution of research on collaborative learning. In E. Spada & P. Reiman (Eds) *Learning in Humans and Machine: Towards an interdisciplinary learning science*. 189-211. Oxford: Elsevier.
5. Avouris, K., Margaritis, F. A Tool to Support Interaction and Collaboration Analysis of Learning Activity, *CSCL 2002*
6. Strijbos, J. W. (2004). *The effect of roles on computer-supported collaborative learning*. Doctoral dissertation, Open University of the Netherlands, The Netherlands.
7. Daradoumis, Th., Martínez, A. and Xhafa, F.: An Integrated Approach for Analysing and Assessing the Performance of Virtual Learning Groups. *CRIWG 2004*: 289-304, Springer
8. Ten Have, P. (1999). *Doing Conversation Analysis: A Practical Guide*. London: Sage Publications.
9. Psathas, G. (1995). *Conversation Analysis: The Study of Talk-in-Interaction*. Thousand Oaks: Sage Publications.
10. Pomerantz, A., Fehr, B. J. (1997). Conversation Analysis: An Approach to the Study of Social Action as Sense Making Practices. In Teun A. Van Dijk (Ed.), *Discourse as Social Interaction, Discourse Studies: A Multidisciplinary Introduction Volume 2*, (pp. 64-91). London: Sage Publications.
11. Heritage, J., Roth A. (1995) Grammar and Institution: Questions and Questioning in the Broadcast News Interview. *Research on Language and Social Interaction*, 28, 1, 1-60.
12. Sacks, H. (1992). *Lectures on Conversation*. Oxford: Blackwell.
13. Livingston, E. (1986). *The ethnomethodological foundations of mathematics*. London, UK: Routledge & Kegan Paul.
14. Kaplan, A. (1964). *The Conduct of Inquiry: Methodology for Behavioral Science*. San Francisco: Chandler Publishing.

Collaboration for Learning Language Skills

Luis A. Guerrero¹, Milko Madariaga¹, Cesar Collazos²,
José A. Pino¹, and Sergio Ochoa¹

¹ Department of Computer Science, Universidad de Chile

² FIET, Universidad del Cauca, Colombia
{luguerre, mmadaria, ccollazo,
jpino, sochoa}@dcc.uchile.cl

Abstract. A Collaborative activity is designed and a software tool is developed to support teaching grammar to primary education students. The activity is intended to create interdependencies among students. The software tool helps to implement the activity. Activity and tool were designed for teaching Spanish grammar, but they can be adapted for teaching other languages.

1 Introduction

Group work has long been used as a pedagogical tool in a variety of learning situations and, indeed, according to Slavin “many studies have shown that two or more individuals can solve problems of different kinds better when they work in groups than when they work independently” [16]. A specific type of group learning is that supported by collaborative techniques: Computer Supported Collaborative Work (CSCL). Collaborative learning technologies must go beyond generic groupware applications, and even the basic technology is not yet well developed [17].

CSCL can of course be applied to teach language skills. Language knowledge is considered one of the most important assets for a person’s life. Thus, language acquisition courses constitute a substantial portion of the primary and secondary curriculum in many countries. Our own University students still have deficiencies in writing abilities; despite the fact the acceptance selection tests include this subject in their evaluation. This situation is further aggravated by recent technology uses - such as textual chat through cellular phones - which do not motivate youngsters to apply good grammar for communication.

Our purpose was to develop a tool to support teaching grammar to primary school students. It is assumed the tool will be used with Collaborative Learning techniques. Specifically, it can be applied to 6th, 7th or 8th grade Spanish grammar students, but it may be adapted to teach other languages grammar as well.

The paper is organized as follows. Section 2 presents related work. The design of the collaborative activity used as a basis for the tool is presented in Section 3. Section 4 includes the main features of the tool. Section 5 presents some practice with the system and finally, section 6 presents the conclusions and future work.

2 Related Work

Sánchez [14] has developed an Internet site for non-specialist students working on Spanish grammar, designed to encourage their autonomy. The site is intended as a complement to regular classes and to the conventional tools. By offering a choice of learning paths, the designers seek to help students to acquire or to reinforce aspects of the declarative knowledge they need to perform language tasks during regular classes.

E-Cid [19] is an online Spanish course replacing traditional lessons. This course is based on contrastive grammar, and has been designed in modular form.

ELMA (Electronic Language Material Archive) [20] is a Web-searchable tool that can be used to customize syllabi according to content-based learning practices. This Web-based content will be accompanied by a battery of activities aimed at activating a student's previous knowledge, facilitating the student's ability to organize information and develop interpretive skills, and at generating class discussion.

Rodríguez et al. have designed collaborative learning games using palmtops for students in the area of Spanish language, obtaining high levels of pupil motivation, attention and concentration [11].

Klein has developed a Spanish class in a collaborative manner without computer support. In his course, Klein improves speaking and writing skills via extensive and intensive practice in both areas [22]. Ying-Hong et al. have developed an English distance learning system (English multimedia corpus). It includes English articles, dialogues and videos [18].

Hardy has developed a Web site to introduce the basic syntactic structure of Modern English and the most common prescriptive rules in formal writing, containing thousands of exercises; students may immediately know the correctness of their answers [21].

The University of Sydney has built web sites, which may be visited to learn English grammar, introducing some basic concepts in English grammar: parts of speech, groups and phrases and subject and predicate [23].

There are many other computer-supported experiences to teach Spanish and English grammar. These cases focus on student group work, not collaborative groups: collaborative activities do not just happen when people are put together and required to do a task in unison [6]. A supportive social milieu and a task infrastructure are required. In this paper, we focus on collaboration as a group phenomenon in which complex tasks are managed through close, step-by-step, apparently casual monitoring by participants of each other's actions, often cued through language.

3 The Collaborative Activity

The decision to use collaborative learning (CL) arose primarily from a desire to innovate and to increase student participation. A cooperative group does not automatically improve the construction of high order cognitive skills and complex knowledge structures. In order to increase the possibilities for mutual understanding and task-related social interaction, interaction tools are needed that are adequately related both to the new concepts to be learned and to the previous experience and knowledge of the students [9]. There should be flexible methods available for the

students, to help them externalize their preliminary ideas and make their thinking process transparent to others. From a constructivist perspective, CL can be viewed as one of the pedagogical methods that can stimulate students to negotiate information and to discuss complex problems from various perspectives. This can support learners to elaborate, explain and evaluate information in order to re- and co-construct (new) knowledge or to solve problems [3, 15]. That is our rationale to design a teaching-learning activity based on collaborative learning techniques.

The designed activity was based on modifying the Language and Communication curricula for 6-8th grades from our Ministry of Education [10]. The Ministry suggests a series of individual activities for grammar contents.

The designed activity includes elements of CL [7]. In particular, the activity was designed to generate interdependencies among group members, such as the need for information interchange during task performing, work splitting into several roles, and the need for explicit knowledge sharing [13]. These interdependencies are the key to collaboration, and it is not easy to achieve them. We based our work on Collazos et al., who have developed a mechanism to structure positive interdependence through software tools intended to make students think in terms of “we” instead of “me” [1]. When positive interdependence is clearly structured and understood, group members perceive that they – and their work- are linked for mutual benefit, that the efforts of each group member will be unique, and that the unique efforts of all members will contribute to success.

The activity has two roles: teacher and student. The teacher prepares the activity and acts as a facilitator. The students work in small groups. They must perform the tasks assigned by the teacher and solve any stated problems. The teacher must select a set of students to do the activity. The number of students should not exceed six, since several studies suggest small groups are best to generate maximum participation and idea interchange [2]. As an example, the activity development will be explained below for a group of four students.

The teacher must choose content for the activity. The content for the example will be a morphological analysis: classify each word from a text to belong to just one category, according to context within its sentence. At the beginning, the teacher must select the work categories (for instance: nouns, adjectives, verbs, adverbs).

The teacher must then find suitable work texts. The number of texts must agree with the number of students who will participate in the activity. The teacher has also to find relevant reference material for the students and make it available to them.

Planning of the activity is done as follows. In our example we have four students numbered S1-S4, and four texts labeled T1-T4. There are four work cycles; each of them has individual work and then group work. Both work types are instances of Problem Based Learning (PBL) [5]. In PBL, the teacher assigns tasks to students, and they must do research and other actions to solve the problems by themselves.

The individual activity consists of studying one morphological element for one of the texts during each cycle. The student must identify which words correspond to this morphological element in the text (Fig. 1). This activity involves a kind of positive interdependence: resource interdependence, allowing students to share materials, information and other resources. This encourages additional conversation and planning and gives elements to the teacher to monitor the activity.

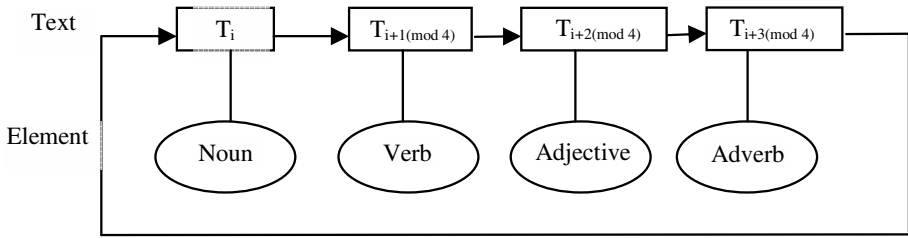


Fig. 1. Assignments for Student S_i

The student must do work in two aspects:

1. Learn about the theory concerning the grammatical element under study. The student may or may not have previous knowledge. S/he may use the reference material, ask his colleagues or consult other information sources.
2. Apply the theory to identify words being the grammatical element in the text, according to their context.

The student tries to find all words belonging to his category in this individual work phase. Three cases may occur: i) s/he rightly chose words belonging to the category; ii) s/he made a wrong choice when selecting words which do not belong to the category; and iii) s/he omitted to choose words belonging to the category. The teacher uses the whole of these cases to determine the student's strengths and weaknesses and to evaluate his/her performance. Note this problem solving involves an understanding of the grammatical elements; it is not an automatic task.

After the individual work, the students must do group work. It consists of co-located correction and discussion of the previous activity. The students must have access to the performance as a group they have obtained thus far. This group activity is very important. When an individual member of the group expresses his/her opinion in relation to the shared public understanding of the group, this will be an attempt to synchronize his/her own understanding with the group-accepted version and make clear the disagreements if there are any. Depending on the outcome of this process there may be further interaction and negotiation until a new meaning or understanding is fully accepted by the group. The key aspects of co-construction of knowledge, meaning and understanding lie on this process interaction among individuals, as well as on their shared and individual cognition.

The group activity ends when the group passes a threshold of performance, e.g., suppose students S_1 and S_2 chose a certain word as noun (correct) and adjective (incorrect) at the same time. If individual performances were to be considered, there will be a right classification (favourable points) and an incorrect one (no points counted). On the other hand, the group numerical performance would be null, because it is incorrect to classify the word both as noun and adjective for that sentence.

The students will have to justify their choices during this group activity, generating discussion. According to Doise and Mugny, the benefits of collaborative learning are explained by the fact that two individuals will disagree at some point, that they will feel a social pressure to solve that conflict, and that the resolution of this conflict may lead one or both of them to change their viewpoint [4]. The social pressure in this case is done by group members wishing to improve the group performance.

The teacher makes the evaluation to determine whether or not the group has passed the performance threshold. In case the group does not approve, the students must continue discussing changes to word classifications. If they pass, each student has probably mastered his grammatical element and learned something on the other ones.

A new cycle is then started with each student in charge of a different element from the one s/he worked in the previous cycle (Fig. 1). This strategy lets each student deal with all concepts of the activity contents. The strategy is consistent with recommendations from standard CL literature: Johnson et al., e.g., recommend rotating roles while the activity be in development [8]. The number of cycles and the number of different texts, then, must agree with the number of students. The teacher can control the difficulty of the text for each cycle; s/he will probably increase it depending on the previous rate of improvement and to keep students' interest. It is also expected the students will increasingly move from consulting reference material to asking colleagues who have already mastered concepts.

4 The Computer Tool

There are three types of tool users: teachers, students and a system administrator. A teacher can create and monitor activities, input texts, input grammatical categories, input reference material and register students. A student can read the activity description and is allowed to do individual and group tasks. The administrator maintains activities and users for the system.

The base work unit is the activity. It has name, description, students assigned to it, a specific grammatical category and a text. Automatic correction of students' classifications is provided if the teacher has done the classification beforehand. Please note this does not mean the activity to be done by the students is going to be mechanical or without reasoning.

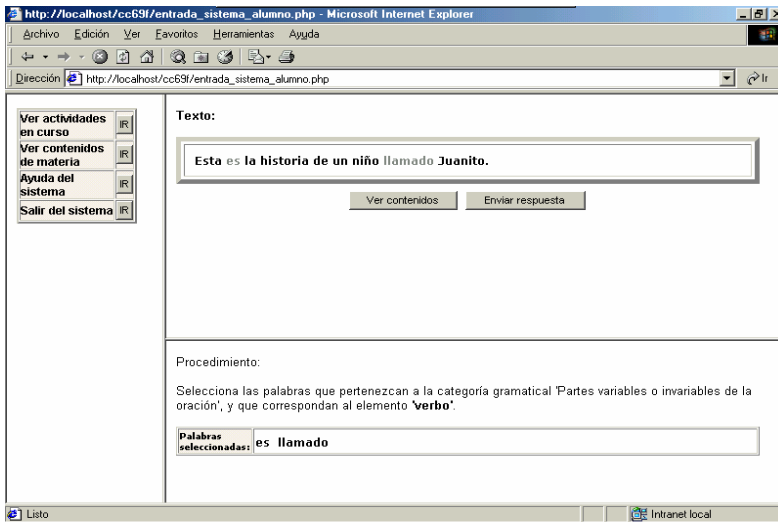


Fig. 2. Individual work UI

Monitoring is provided through statistical reports showing individual and group performance in terms of number of rightly, incorrectly classified and omitted words. The screen also identifies the problematic words. This information, after each cycle, lets the teacher support students by providing hints or suggesting changes.

Each student gets a personalized screen, describing his/her activities. Fig. 2 shows the screen for individual work, which may be asynchronous and distributed. Group work, by contrast, requires face-to-face, synchronous work in just one computer.

Color cues are provided in the screens for easy visualization. Thus, the screen for group work presents all words classified by a specific student with the same color. A distinctive color is used for conflicting words, i.e., those ones chosen by two or more students. The current group performance is also presented in graphic form. Finally, the positive or negative difference with respect to the threshold is also shown.

Both individual and group work user interfaces have a quick access button to the reference material. Therefore, they can easily review relevant theory.

5 Experimenting with the System

A preliminary experimentation was done with 32 seventh grade students (12-13 years old) at a public school in our country, divided in eight groups of four students. We would have liked to assess whether a collaborative activity such as the proposed one actually makes students learn the subject. Furthermore, we should compare this activity to traditional ones to determine the value of the collaborative approach. However, our research is still on-going, and thus, a first pre-experiment was intended just to have a first input about the usability of the tool.

The experiment consisted of two sessions with a questionnaire at the end of each of them. Some improvements to the usability of the software were done between the first and second session, according to the comments of the students. Some of the improvements were: use of standards in colours and icons, a simplified way to enter the application (avoiding the use of login and passwords), use of nicknames, more graphical information (instead text-only interfaces), use of a more simplified language (in the directions and messages), the possibility to change the student data (the nickname, the colours), a simplified way to manage the software security (login and passwords are very complicated concepts).

Table 1. Second questionnaire results

Question	Likert average	Disagree (1 & 2)	Neutral (3)	Agree (4 & 5)
The activity improved my Spanish language knowledge	3.8	13.3%	13.3%	73.3%
The group work improved my personal knowledge	3.6	20.0%	16.7%	63.3%
We finished the activity in a successful way	4.0	6.7%	16.7%	76.6%
I like the group work activity	3.8	6.7%	33.3%	60.0%
Four people were a good group size	3.5	13.3%	33.3%	53.3%
I contribute to my group knowledge	4.3	3.3%	16.7%	80.0%
I liked the activity	3.8	6.7%	30.0%	63.3%
I liked the software tool	4.1	6.7%	23.3%	70.0%

Table 1 presents some of the results of the second anonymous questionnaire. Answers to the questions were in a Likert 5-value scale (5-totally agree; 4-agree; 3-neutral; 2-disagree; 1-totally disagree). Most interesting results were the following ones: over 70% of the students think the activity improved their Spanish language knowledge; 60% of the students also liked group work, and most of the rest were neutral about group work. Only two students (6.6%) did not like the activity.

6 Conclusions and Future Work

Collaboration is not simply a treatment with positive effects on participants. Collaboration is a social structure in which two or more people interact with each other and, in some circumstances, some types of interaction occur having a positive effect [12]. Activities should then be designed accordingly to get a shared understanding of the problematic situation.

Our basic assumption is that CSCL tools must be associated to CL techniques to be truly considered “collaborative”; otherwise it may be just “group” or “collective” learning. The chosen technique in our case was PBL: the activity begins as a task the students must achieve. It is while trying to do the assignment when students need background theory and concepts. Of course, most of the required information is easily available from the reference material, but it is while trying to assimilate it when that information is transformed into useful knowledge. Note that some PBL characteristics such as freedom to decide the methods or plan development, do not apply here.

The developed activity attempts to generate a CL environment, where individual experimentation and group collaboration play a key role in the teaching/learning of grammatical concepts. The software, on the other hand, is intended to simplify the teacher’s task in terms of activity creation and monitoring: the tool automatically corrects students’ assignments and it also provides statistical reports on students’ performance both currently and in its evolution in time. Despite the fact the developed activity was designed to support teaching of Spanish grammar, we think it can be easily adapted to the grammatical elements of other languages.

Finally, it is possible to consider the use of some alternative development and implementation platforms, which could provide additional flexibility to the tool. Specifically, we could include wireless mobile devices as PDAs (Personal Digital Assistants). Naturally, the impact of this technology on the design of the application must be evaluated. The evaluation must include both the technical feasibility and the pedagogical and psychological aspects modelled in the collaborative tool. Our first impression is that both individual and group tasks can be supported with these devices. Individual tasks can be made in an asynchronous distributed way and thus, it should be easy to support them. The synchronous face-to-face group activity could also be supported in its discussion with PDAs.

Acknowledgements

This work was partially supported by Fondecyt (Chile) grants No. 1030959 and 1040952.

References

1. Collazos, C., Guerrero, L., Pino, J., Ochoa, S.: Collaborative Scenarios to Promote Positive Interdependence among Group Members. LNCS 2806, 2003, 356-370
2. Cooper, J.: Small-group Instruction in Science, Mathematics, Engineering and Technology (SMET) Disciplines: A Status Report and an Agenda for the Future. Cooperative Learning and College Teaching Newsletter 6 (2), 1996
3. Dillenbourg, P., Baker, M., Blake, A., O'Malley, C.: The Evolution of Research on Collaborative Learning. In Spada, H. and Reimann, P. (eds), Learning in Humans and Machines, 1995
4. Doise, W., Mugny, G.: The Social Development of the Intellect. Oxford: Pergamon Press, 1984
5. Duch, B., Gron, S., Allen, D.: The Power of Problem-Based Learning, A Practical "How To" for Teaching Undergraduate Courses in Any Discipline, Stylus Pub. LLC (2001)
6. Galleger, J., Kraut, R., Egido, C.: Intellectual Teamwork: Social Foundations of Cooperative Work. Hillsdale: Lawrence Erlbaum, 1990
7. Johnson D., Johnson, R., Holubec, E.: Circles of learning (4th ed.). Edina, MN: Interaction Book Company, 1993
8. Johnson D., Johnson R, Holubec, E.: Cooperation in the Classroom. Interaction Book Company, Edina, MN, 1998
9. Katz, S., Lesgold, A.: Collaborative Problem-Solving and Reflection in Sherlock II. Workshop on Collaborative Problem Solving, Edinburgh, 1993
10. Ministry of Education [our country]. Curriculum on Language and Communication for 6-8th grades, 2003 (in Spanish)
11. Rodríguez P, Nussbaum M, Zurita G, Rosas R Lagos F.: Personal Digital Assistants in the Classroom: An Experience. Ed-Media, Tampere, Finland, 2001
12. Roschelle, J., Teasley, S.: The Construction of Shared Knowledge in Collaborative Problem-solving. In C.E. O'Malley (Ed), Computer Supported Collaborative Learning, Berlin: Springer-Verlag, 1995, 69-197
13. Salomon, G.: What does the design of effective CSCL require and how do we study its effects? SIGCUE Outlook, Special Issue on CSCL 21(3), 1992, 62-68
14. Sánchez, P.: Intégration d'un outil informatique dans l'enseignement du niveau intermédiaire d'espagnol à l'Université de Technologie de Compiègne, Apprentissage des Langues et Systèmes d'Information et de Communication, 5(2), 2002, 209-229
15. Scardamalia, M., Bereiter, C.: Computer Support for Knowledge-building Communities. Journal of the Learning Sciences, 3(3), 1994, 265-283
16. Slavin, R.: Using Student Team Learning. Baltimore, MD: Center for Social Organization of Schools, Johns Hopkins University, 1980
17. Stahl, G.: Groupware Goes to School. Lecture Notes in Computer Science 2440, 2002, 1-24
18. Ying-Hong, W., Chih-Hao, L.: A Multimedia Database Supports English Distance Learning, Information Sciences, Informatics and Computer Science: An International Journal, 158(1), 2004, 189-208
19. <http://virtualcampus.ch/E-Cid>
20. <http://www.ucltlc.org/funding/2000.01/elma.htm>
21. <http://textant.colostate.edu/grammarbook/title.html>
22. <http://www.uiowa.edu/~spanport/personal/Klein/w116/116-Hm.htm>
23. <http://www.arts.usyd.edu.au/departs/english/grammar/default.html>

Collaborative IS Decision-Making: Analyzing Decision Process Characteristics and Technology Support

Bjørn Erik Munkvold, Kristin Eim, and Øyvind Husby

Department of Information Systems, Agder University College,
Servicebox 422, 4604 Kristiansand, Norway
Bjorn.E.Munkvold@hia.no, k_eim@hotmail.com,
ohusby@yahoo.com

Abstract. The paper presents an analysis of a collaborative decision-making process related to the selection and implementation of a new corporate solution for collaboration and information management in a Norwegian oil company. Several challenges were identified in the decision-making process, related to ensuring continuity between different phases, enabling efficient communication among the different stakeholder groups, and gaining involvement and commitment from the business areas. The analysis also focuses on the utilization of various collaboration technologies in the different phases of the decision-making process. The study contributes to increase our understanding of collaborative IS decision-making processes, and the role of collaboration technologies for supporting these.

1 Introduction

The role of information technology (IT) and information systems (IS) in supporting decision-making is well acknowledged, enabling more effective decision-making in the complex and information intensive operations of today's organizations [13]. While Decision Support Systems (DSS) traditionally have been confined to supporting individual decision-makers, the scope of these systems is now broadening to also include different forms of collaborative support [12]. While much research has been conducted on group decision-making using group support systems (GSS), there is still limited empirical research on how a more extended collaborative IT infrastructure may support organizational decision-making processes that may involve both dispersed and asynchronous collaboration [6].

This paper presents an analysis of the collaborative decision-making process related to the selection and implementation of a new corporate solution for collaboration and information management in Statoil, a Norwegian oil company. Spanning a two-year period, this decision process involved several phases with many participants, using different collaboration technologies for communication and information sharing. We identify several challenges experienced in the decision process, and discuss the potential role of collaboration technologies in meeting these challenges.

The acquisition and/or development of IS applications in organizations represents a particular type of decision-making process that has not been extensively focused in research [2]. According to Boonstra [2, p. 207], “*Many current decision-making models and approaches in the MIS-field use assumptions that are mainly based on the rational model of decision-making and ignore fundamental differences in IS decisions, organizational features and external factors.*” Our case analysis focuses explicitly on the attributes characterizing the IS decision-making process studied.

The aim of this study is thus twofold: 1) to contribute to increase our understanding on the potential role of collaboration technologies in supporting organizational decision-making processes, and 2) to add to the scarce knowledge on IS decision-making by providing an in-depth analysis of the collaborative decision-making process related to the acquisition and implementation of a corporate IS solution.

The next section presents a brief overview of relevant research related to IS decision-making and IT support for collaborative decision-making. Section 3 presents the methodological approach, and Section 4 describes the case organization and the decision-process analyzed. The results from the data analysis are presented in Section 5 and discussed in Section 6. Section 7 presents conclusions and implications.

2 Relevant Research

2.1 IS Decision-Making

An IS decision-making process is defined as “a set of actions that begins with the identification of a stimulus (the IS-related problem) and ends with the IS decision”, where an IS decision is defined as “a decision to invest (or not to invest) in new IS” [2, p. 196]. While IS decision-making can be regarded as a case of organizational decision-making, it has also been pointed to how high complexity, significant resource demands, challenges related to evaluation, and lack of prior experience on which to base the decision may represent distinguishing characteristics that make it important to develop a better understanding of strategic IS decision-making processes [2, 11]. Boonstra [2] combined a phase-based [9] and an attribute-based [11] approach in analyzing 20 IS decision-making processes. This analysis showed how IS decisions can be extremely diverse and have many perspectives, thus calling for a more differentiated perspective on IS decision-making processes than the dominating rational model. The following five factors were identified as resulting in major differences in IS decision-making processes:

- **Stimuli** - the degree of *urgency* and *necessity* from the perspective of the decision-makers (crisis, problem, opportunity)
- **Design mode** - whether there is scope to *design* a solution (ready-made, modified or customized)
- **Style** - whether the IS decision can be *subdivided* in order to follow a more gradual process path (planned vs. incremental)
- **Search process** - whether a *search* must be made to find distinct IS alternatives
- **Participants** - the number and influence of *stakeholders* involved in the process, and the extent that their interests vary and contrast.

In this study we apply these factors for characterizing the collaborative IS decision-making process studied.

2.2 IT-Supported Collaborative Decision-Making

There is a rich body of research on the use of IT support for group decision making, often termed group support systems (GSS) [5, 6]. While much of this research has been conducted as experiments in academic setting, there is also a growing number of studies on organizational decision-making supported by GSS [6]. This research shows that the use of a GSS can have positive effects on both efficiency and decision quality, while also providing enhanced participation and increased buy-in to group decisions. However, most of these studies have focused on use of GSS in meeting room settings, thus being limited to synchronous use of electronic meeting systems (EMS) in relatively small groups. Few studies have yet reported on the use of a broader range of collaboration technologies for supporting organizational decision-making processes of a larger scope and time span. Brézillon et al. [4] report positive effects on editorial decision-making in a newspaper from the implementation of a collaborative IT system. However, the lengthy decision process related to the implementation of this system was not reported as being supported by any collaboration technology. A study by Borges et al. [3] addresses how lack of accompanying information and a common context may result in a gap between decision-makers and implementers, thus producing wrongly implemented or lost decisions. They present a possible solution to this problem combining workflow technology and discussion support tools. In our case study analysis we also document how combining different collaboration technologies may be effective for meeting potential challenges in collaborative decision-making processes.

3 Methodological Approach

The focus of this study was a complex decision-making process, involving multiple stakeholder groups. We applied a qualitative research approach, using a single case study design. The basis for this research project was an initiative from the case organisation, suggesting a study of their collaborative decision-making processes and related use of technology support. Thus, this could be described as an opportunity-based design [8]. The unit of analysis was the decision-making process related to the selection and implementation of the new corporate solution for collaboration and information management in Statoil.

Data was collected through interviews and document analysis during Spring 2004. Two of the authors interviewed 13 Statoil employees who all had been involved in the decision-making process. These participants covered all major roles in the case process, such as project sponsor, project manager, and members of the project group, steering group, and reference group. The participants had from 5 to 25 years experience in Statoil. The interviews were semi-structured, using an interview guide that focused on the respondent's background and role in the project, the structure applied in the decision process, the communication and involvement in the different phases of the project, the IT tools used throughout the project, and the respondent's overall view on the project. All interviews were taped and transcribed. In addition, we

analyzed an extensive number of Statoil documents related to the different phases in the decision process, as well as more general documents on the project methodology, e-collaboration strategy, and Statoil business processes.

The data was analyzed through several iterations, involving interpretation, categorization and discussion among the researchers. All interview statements were validated by the respondents, and a final results report was approved by Statoil, thus ensuring internal validity.

4 Case Overview

Statoil is the world's third largest exporter of crude oil, with approximately 24,000 employees in 29 countries and a total revenue exceeding US \$ 45 billions (2004). Statoil's IT unit (about 630 employees) is a division of the corporate services business area, responsible for adapting and delivering IT services to internal customers in the company.

In December 2001, a project for implementation of a new corporate solution for collaboration and information management in Statoil was initiated. The motivation for the project was a fundamental need to establish a corporate-wide foundation for IT-supported collaborative work practices, including coordination and document management in all phases of the value chain. From 2002 to 2004, several actions were conducted through different project phases which eventually led to a specific IT solution for enhancing collaboration and information management in Statoil.

During this project, Statoil's existing portfolio of collaboration technologies was used extensively for supporting communication, coordination and information sharing among the project participants. Lotus Notes constituted the core technology in this existing solution, providing support for information management through the Sarepta Arena system developed in Statoil. In addition, Statoil also has an established infrastructure for audio, video, and desktop conferencing [1].

4.1 Case Decision Process

All activities related to development and implementation of new IT solutions in Statoil are governed by their standard project methodology for IT projects – ProMIT. According to the ProMIT model, a product development process involves several phases, the transition of which is represented by a decision gate (DG), i.e. a milestone or decision to be made before entering the next project phase. ProMIT specifies the following phases: pre-study, concept specification, execution, and conclusion. Table 1 provides an overview of the case decision process according to these phases, based on documentation from Statoil.

The project is divided into five separate decision phases spanning a two-year period, followed by a two year implementation period. The project was initiated in 2001 with a decision on developing a strategy for effective e-collaboration in Statoil. Phase 1 was completed in August 2002, and resulted in a strategy comprising several goals for enabling efficient collaboration and information sharing. This strategy formed the basis for the initialization of the feasibility study in the next phase. The feasibility study included product demonstrations and architectural workshops in cooperation with vendors, literature studies, and strategy workshops attended by

participants from different business areas in Statoil. As the desired capabilities of a new solution seemed to be insufficiently supported and understood by the vendors, Statoil decided to extend the feasibility study.

Table 1. Overview of project phases

Period		Project phases and decision points	ProMIT phases & decision gates
2001	Dec	Decision on e-collaboration strategy	"DG0"
2002	Jan-Aug	Phase 1: Development of strategy	Prestudy
	Jul-Aug	Decision on mandate for feasibility study and concept development	DG1
	Aug-Dec	Phase 2: Feasibility study	Concept spec.
2003	Jan-Feb	Decision on mandate for further feasibility study and concept dev.	Concept spec.
	Feb-May	Phase 3: Feasibility study, concept dev.	Concept spec.
	Jun-Aug	Choice and approval of concept, approval of implementation project	DG2
	Jul-Sept	Phase 4: Request for information from relevant vendors (RFI)	Execution
	Oct	Choice of vendors to be asked to propose a solution	Execution
	Oct-Dec	Phase 5: Preparation of requirement specification, request for proposal from selected vendors (RFP)	Execution
	Dec	Choice of vendor	DG3
2004-2005		Implementation and deployment	Execution, DG4, DG5, Conclusion

Phase 3 included technical and organizational feasibility studies, development of alternative solution scenarios or concepts, and different analyses of consequences and costs. Based on the different solution scenarios, a proposed solution concept and architecture was presented. The request for information (RFI) process was conducted in phase 4, which involved gathering information and evaluation of the solution concepts from selected vendors. Based on this evaluation, a choice was made regarding vendors to be asked to propose a solution. Phase 5 involved preparation of requirement specification and a request for proposal (RFP) to these vendors. This phase resulted in the choice of Microsoft as the key vendor for the new solution, which is now being implemented.

The first three phases were primarily aimed at competence-building and maturation, acquiring knowledge of new collaboration and information management technologies. This justifies the durability of these phases, of one and a half year all together. The extensive scope of the project also resulted in much time spent on deciding whether to continue with the project, especially in the transition between phases 3 and 4. The project would affect all employees in Statoil, and several rounds

back and forth with the business areas were needed to discuss whether the employees were mature for this level of change in their work practices. Phases 4 and 5 had a more specific focus, thus lasting only three-four months each.

4.2 Case Project Organization

The project comprised several groups with participants from various parts of the organization. The *steering group* was composed of process owners and IT leaders from different business areas having a “customer” role, which involved evaluation and decision making associated with the deliveries. The chairman of the steering group was the *project sponsor*, who also represented corporate management. The project sponsor was assisted by the steering group and was formally responsible for making the final decision. The *user responsible* worked for the sponsor, and was responsible for ensuring that the end-users got the information they needed about the project. The *project manager* was the leader of the project group. The project manager role was assigned to several persons during the project. The *project group* was composed of staff members of the corporate services/IT department, working on the project on a daily basis. Members of the project group varied somewhat during the project phases.

The *reference groups* were composed of representatives from relevant discipline networks and various business areas representing end-users. The reference groups were in this project referred to as quality assurance/control groups. The involvement of representatives from different business areas was considered important in order to maintain the viewpoint of the end-users. However, as will be described in the following section, the involvement of these groups varied to a certain extent between the business areas. Finally, *working groups* were put together with representatives from both the reference groups and the project group, for working on specific task assignments.

5 Case Study Findings

This section presents major findings from the case study, categorized according to the following themes emerging from our data analysis: project continuity, project communication, commitment and user involvement, and collaboration technology support. Interview quotes are used to illustrate and exemplify the findings, with identification of group affiliation for each informant.

5.1 Project Continuity

The described project was relatively comprehensive with its separate phases or sub-projects spanning a two year period. One challenge was to ensure continuity throughout the project, i.e. maintaining knowledge and other resources during all project phases:

“A lot of time has been spent on learning, and it is important to ensure that the learning is passed on to the next phases, from either the same persons or in such a way that the competence and knowledge are reused. That is a challenge when the duration has been that long.” (Steering group)

Some of the respondents argued that the project suffered from a continuous mobilizing and demobilizing of project participants:

“(...) we [the project group] were demobilized from the project in December 2002, and didn’t know when the project was starting up again in 2003. Then, when people were assigned to other tasks, a new decision was made on continuing with the project which required a new mobilization.” (Project group)

The loss of continuity was also referred to as a result of people switching roles during the project phases. This discontinuity in the process was especially evident in the transition from phase 3 (the feasibility study) to phase 4 (the RFI process). Some of the respondents described the waiting time between these phases as quite frustrating, due to insufficient overview of the project and the constant need for keeping themselves posted with the information in Sarepta Arena folders to update their knowledge about the project. However, some of the respondents claimed that the continuity between the phases was satisfactory and could not have been done otherwise.

The results indicate that the perception of continuity among the respondents varies according to their role and group affiliation within the project. The frequency of meetings varied among the different groups in the project. Whereas the project group working with project-related tasks on a daily basis had weekly meetings, the other groups had meetings in a less frequent manner (monthly or every third week). In general, respondents working on the project on a daily basis described the transition between phases as rather frustrating, while respondents involved in groups working on the project in a more infrequent manner did not regard the loss of continuity as problematic. However, all respondents agreed that the project continuity from project start until phase 3 was satisfactory.

The demobilization of project members was also reported to affect their access to project-related information. Whereas project members who were actively engaged in the project throughout the project lifecycle had relatively good access to information about the project, members of other groups such as the reference groups and participants who gradually were removed from the project expressed a certain lack of available information:

“I’m organized in the same unit as many of the project members. In addition I was involved in the first three phases in the project, but not anymore (...) However, I don’t know much about the project now, I don’t get much official information. And I am a Nosy Parker, so if there was anything, I would have known...” (Project group)

“The accessibility of information was very good in the period I was involved in the project. But when that period was over, we didn’t hear much.” (Reference group)

Most of the respondents agreed that the accessibility of information was sufficient in periods when they were directly involved in the project. This was obtained through reports containing needed information prior to and after each meeting:

“I’ve read information about the project through Sarepta Arena. And that has been very good. They have been very good at documenting the processes. (...) They have followed up quickly with information prior to and after the meetings. Actually, that has been exemplary.” (Reference group)

The perceived lack of information was mostly evident during phases 4 and 5. Two possible reasons for this were identified. First, these phases were confidential and hence all project-related documentation was kept in closed folders. Second, by the start of phase 4, many project members were either appointed to new projects or assigned to new roles in the project. Generally, fewer participants were involved in this phase, as the project sponsor and his advisors to a certain extent dominated these processes. As a consequence, several of the former project members were now on the outside and no longer automatically updated with the progress of the project. The technologies used for information sharing among the participants are described in Section 5.4.

5.2 Project Communication

The case process involved several forms of communication between the various groups in the project (ref. Section 4.2). Among the project members it was a common agreement that the internal communication worked well. These worked tightly together and were at times co-located in a dedicated office environment. However, during hectic periods with several parallel activities there was less time for internal communication, and collaboration with surrounding environments became a low priority task. The participants in the reference groups attended more or less formal hearing meetings chaired by the project group. Some of these hearings were characterized as very successful regarding communication, attributed to being well prepared in advance and facilitated by the group leader.

Terminology was mentioned by several informants from the different stakeholder groups as one factor preventing effective communication. The project group spent a lot of time exploring and learning about the subject area, developing a certain jargon. This resulted in a barrier to effective communication between the project group and the other groups in the project, i.e. the steering group and the reference groups:

“I think that what has been the main weakness is communication, a dialogue, which is the lack of a common conception, a common perception of what we are talking about. And that the framework is mutual.” (Project group)

Although involvement of users and business representatives was described as strong, communication was seen as a main challenge by the project group. This involved translating the rather complex project terminology into a message that could make the business areas understand “what’s in it for me”. The project group here stressed the importance of spending adequate resources on gaining necessary involvement from the user representatives. Yet, some respondents were still critical regarding the project group’s use of terminology:

“That is - to put it mildly - totally incomprehensible! That is a common problem - whether you work within IT or other parts of the enterprise, everybody has their own language. Once talking to others that do not share this language, there is a cultural conflict.” (Steering group)

A reference group member also pointed to how “IT language” tends to be a general concern among all employees in the company, contrary to other disciplinary jargons that are only used within the related professional groups.

Another communication challenge was referred to as a gap in ambition level between the project group and the steering group regarding the expected deliveries. This was particularly related to the feasibility studies in phases 2 and 3, where the deliveries from the project group exceeded the expectations of the steering group regarding level of detail. While the result was perceived to be of good quality, this was also seen as evidence of how the communication between these groups could have been better. Some members of the steering group also regarded the level of detail in the documentation to be too high, arguing that it could have been possible to reach the same decision without this extensive decision basis, and within shorter time. Other members of the same group considered the extensive and thorough process necessary for obtaining a consensus decision. This illustrates how there tended to be differing perspectives both within and between the different groups on these challenges.

5.3 Commitment and User Involvement

The use of reference groups served to obtain input and requirements from the business areas and users, and to anchor the decisions to be taken by developing maturity in the business areas through learning together with the project group. The overall perception seemed to be that the business areas were well involved:

“What has been particularly good is that they [the project group] have involved a large share of the corporation, the business areas and staff functions. A representative sample.” (Reference group)

“And, that we feel more ownership causes that we have a better attitude, and understand the benefit. It is even so that we see that we are able to take a broader view and realize what consequences it has to Statoil, and not solely think of ourselves.” (Reference group)

The early participation by the business in the initial phase of developing the e-collaboration strategy, and the feasibility study and lab demonstrations, were stated as factors contributing to commitment from the business representatives.

However, several respondents pointed to how the involvement from different business areas had varied to some extent:

“We had so-called reference groups from the business that were to contribute. I noticed that there was poor participation from some business areas. However, that improved. But the participation from the business was too variable, I guess. That is a classical problem - those that ought to take part often do not have the time because they are key personnel. The steering group stressed the importance of participation, and then it improved. We did not have optimal participation from day one, but it improved gradually.” (Steering group)

As stated by several respondents, varying commitment will often be an issue when participation is voluntary and time is limited. Some also addressed the challenge of selecting the right representatives from the business areas, arguing that some business areas missed their chance of influencing the process by sending delegates to workshops “that did not know why they were there”. It was also pointed to how it was important to involve different groups of users, but that involving too many could lead to “endless discussions”.

Overall, the interviews indicated that the project group, the steering group and the business representatives all thought that the level of involvement and commitment was somewhat satisfactory, but that all parties also suggested that it could have been better.

5.4 Collaboration Technology Support

Statoil has a well established infrastructure of collaboration technologies [10]. Table 2 provides an overview of the different collaboration technologies used in the various phases of the project, and the experiences with these.

The technologies include both asynchronous and synchronous tools, supporting information sharing, distributed meetings, electronic brainstorming and evaluation. Typically, the technologies have been used on a demand basis, utilizing the technologies considered most suitable for the different tasks. This can be seen to reflect the rather internalized use of the different technologies among the employees, at least in the project group.

Table 2. Technologies, usage, and experiences

Technologies (Product)	Areas of use	User groups*	Experiences	Phases
E-mail (Lotus Notes)	Coordination	P, S, R	Effective for coordination and communication outside meetings.	All phases
Shared document folders (Sarepta Arena, Lotus Notes)	Information sharing and document management	P, S, R	Important for common access to project documentation.	All phases
Net meetings (MS NetMeeting/ Lotus Sametime with telephone conf.)	Distributed meetings	P, S, R	Effective for meetings with participants at different Statoil offices.	2 (mainly)
Videoconferencing (Tandberg)	Distributed meetings	P, S	Used mostly for meetings with externals.	2 and 3 (mainly)
Electronic meeting support (GroupSystems)	Workshop	P, R	Effective for brainstorming and gaining user input.	3
Surveying (Confirmit)	Evaluation	R	Useful for priority-setting of solution requirements.	3

*P = Project group, S = Steering group, R = Reference groups

A lot of documentation was produced in the process and stored in Sarepta Arena, the Lotus Notes based general solution for document management in Statoil. This involved different types of documents, i.e. working documents, temporary reports, final project reports and executive summaries. Access to a common document repository was reported to ease collaboration within the project, and several respondents complimented

the project group for providing timely and well structured information in this repository. Yet, the project group saw a need for presenting their reports to the steering group in meetings, as the reports were considered to warrant more explanation rather than being evaluated “on their own”.

Net meetings and videoconferencing were mainly used in the concept specification phase. Clear meeting agendas and material made available in advance, combined with skilled meeting facilitation, were emphasized as important factors for effective accomplishment of such meetings. Sarepta Arena was here used for feedback and questions on shared documents prior to meetings. One respondent here pointed to how net meetings seemed to be preferred over videoconferencing, at least in connections when the participants knew each other, as arranging a net meeting on the desktop was considered easier than “running to a videoconferencing room”. Videoconferencing thus tended to be used more with external parties.

During creation of different solution scenarios in the concept specification phase there was a need for richness of ideas developed through brainstorming. A session with this purpose was organized as an all-day workshop using GroupSystems for electronic meeting support, with a skilled GroupSystems facilitator from outside the project. This workshop was perceived as a good utilization of such a service that lowered the threshold for communication and contributed to effective dialogue and lots of ideas in a short period of time:

“Killer app! It’s an excellent method when you are going to generate many ideas within a short time-frame because it ensures that everyone gets to speak one’s mind, without everybody talking at the same time. However, that demands having a good facilitator - which we had.” (Reference group)

The abilities to systematize, organize and prioritize were also appreciated. Anonymity and equal possibilities for participation were mentioned as factors that eased participation of parties that would otherwise not have been so eager to contribute.

In addition, a surveying tool was used for evaluation purposes. This was stated to be particularly suitable during priority-setting of solution requirements in the concept specification phase, with feedback gained from a larger number of employees providing a good starting point for further analysis.

6 Discussion

6.1 Characterizing the IS Decision-Making Process

Similar to most of the IS decision-making processes studied in [2], this process had a relatively long time-span of two years (not including implementation). The process can be characterized as rational, i.e. a decision “*based on objectives, perceived or not, and undisputed facts, [...] taken only after a thorough analysis as a well-orchestrated and coordinated series of actions*” [2, p. 199]. The project was run “by the book”, i.e. following the detailed specifications of the ProMIT methodology. Table 3 characterizes the decision-making process according to the distinguishing factors presented in Section 2.1, based on [2]. This characterization provides the background for the further discussion on the role of collaboration technologies in the decision process, and the experienced challenges in the process.

Table 3. Characteristics of the IS Decision-Making Process

Attributes	Decision Process Characteristics	
Stimuli	Problem Decision (Opportunity)	The decision process was initiated based on the need for a more effective solution for collaboration and information management. Also some element of opportunity decision, as the new solution is intended to enable new and innovative collaborative work practices.
Design mode	Modified	No ready-made solution existed covering Statoil's needs – thus the solution combines given and customized features, which can be modified to make them fit specific business conditions.
Style	Incremental	The decision process comprised a series of decisions, also involving decisions on continuing the project.
Search process	Extensive	The concept specification phase involved an extensive search process of solution alternatives.
Participants	Many involved	The process involved several stakeholders, represented by the project, steering and reference groups, with different needs and concerns.

6.2 The Role of Collaboration Technologies in the Decision Process

We have previously reported how several collaboration technologies were used to support the decision-making process in this case. In this section we discuss the extent to which the full potential of these technologies has been exploited, and the implications from our findings regarding how collaboration technology may be used for alleviating the experienced challenges related to collaborative IS decision-making. In sum, the main challenges were identified as: 1) ensuring continuity in the project; 2) ensuring effective communication among the different stakeholder groups; and 3) gaining involvement and commitment from the business areas.

Project Discontinuity. The incremental nature of this decision process created problems with continuity, due to project members being demobilized and assigned to new projects between the different phases. The Lotus Notes based information repository, Sarepta Arena, provided common access to project documentation throughout the project, and thus served as an important means for maintaining continuity through documentation. Thus, the problems with discontinuity were more related to non-optimal use of competence resources, than to decisions being “lost” in transition between phases [3]. It should also be noted here that some respondents actually pointed to a need for some membership fluctuation throughout the process, for bringing different qualities into the project.

Yet, with the final phases of the project being closed, this implied restricted access to the contents in the repository. Thus, project members from the earlier phases who were no longer part of the project group now experienced difficulties with keeping up to date with the project. This seems to imply a need for a more differentiated access

policy to the information repository, allowing more peripheral members to still gain access to high-level status information on the project. Also, use of electronic discussion lists could perhaps have been useful as a forum for stakeholders to maintain informal communication and discussion about the project [3].

Project Communication. Statoil has a well developed infrastructure for electronic communication, including desktop conferencing (net meetings), and audio and video conferencing. This was reported to be effective in supporting communication among project members in different locations, thus also providing more flexible access to key personnel. The challenges related to communication identified in this decision process were thus related to contents, and not infrastructure.

The subject of this IS decision-making process was complex and comprehensive, implying an emphasis on learning by the project group and the need for an extensive search for solution alternatives. With few comprehensive IS solutions like this having yet been implemented in industry, there was also limited possibility for learning from other companies. The communication problems reported were mainly related to perceived gaps in technical jargon and ambition level among the different stakeholder groups. This illustrates the importance of developing *cross understanding*, i.e. the understanding that individual group members have about the mental models of other individual group members [7]. The combination of a high level of knowledge diversity and a high level of initial or achieved cross understanding is considered optimal for enabling decision groups to produce high quality decisions [7]. Conscious selection of group members with knowledge diversity, and frequent dialogue and exchange of perspectives are presented as strategies for achieving this.

Commitment and Involvement. As evidenced in the case analysis, the project group was complimented for having facilitated involvement from a large share of the organization. This took the form of workshop sessions where the different business areas were invited to send their representatives. The use of skilled facilitators for these workshops was emphasized as a positive factor for the outcome of these. Yet, the time pressure in some of the decision stages also was reported to result in hearing meetings where the agendas proved too ambitious, leaving time mostly for presentation from the project group rather than real discussion and input from the business representatives.

One full-day workshop was also conducted with GroupSystems, supporting brain storming and idea organization. This was characterized as very effective, exposing many of the benefits commonly attributed to this type of technology, such as more effective communication through parallel input of ideas, and increased participation through anonymity [5, 6]. The use of a trained facilitator was highlighted as an important requirement for successful use of this technology, in line with previous findings [1, 5, 6].

The findings also show how involving the right people was a challenge, since the appointment of these was handled by the different business areas. Deciding on the level of involvement was also characterized as a balancing act, where “too much democracy” could risk falling into “endless discussions”:

“Presumably we could talk with all 17000 employees, and they would all have different opinions, and still not be satisfied with the level of involvement. Thus, one has to find the proper balance.” (Steering group)

The survey tool was reported to be effective for gaining feedback from a larger number of people. Use of Sarepta Arena for feedback on documents prior to meetings represented another possibility for involving a broader base of people, regardless of time and place. In addition, online discussion forums could possibly have been used for enabling involvement from interested people in the business areas. Providing more frequent and easily available information updates to people outside the core project group was also identified as a potential way of maintaining a more continuing commitment to the project.

Some Final Comments on Technology Selection and Use. Several of the challenges reported in this study can be ascribed to time pressure, as the deadlines in the various phases of the project were tight. This clearly restricted the possibilities for providing extensive information services to “outsiders”, and for moderating and maintaining electronic discussions. In that respect, our recommendations may be somewhat discarded as wisdom of hindsight. Yet, we argue that the findings also contain implications for how collaboration technology support may contribute to decision process improvements, even within the reality of hectic project schedules. For example, increasing use of multi-mode meetings, i.e. meetings comprising interaction in different time/place combinations may enable a broader base of employees to contribute at their time and convenience, thus providing more flexibility to the decision process [1]. Examples of this form of multi-mode interaction identified in the case decision-making process include discussion of documents in Sarepta Arena prior to meetings, and the use of the survey tool for gaining feedback from a larger number of people as a starting point for further analysis.

Finally, the positive experiences from the GroupSystems session indicate how this constitute an effective tool for quickly collecting and structuring input from a larger number of people than would be possible in traditional meetings without technology support. One may thus question why this tool was not utilized more throughout this decision-making process. However, the answer to this goes beyond the assessment of functionality alone, as a decision to discontinue the use of electronic meeting support in Statoil was made independently of the new e-collaboration solution concept.

7 Conclusions and Implications

The article has presented an analysis of the collaborative decision-making process related to the specification, selection and acquisition of a new IT solution for collaboration and information management in Statoil. Overall, the process was characterized as successful, reaching a consensus decision among the stakeholders involved. Yet, despite being conducted according to a detailed process methodology, and supported by an extensive portfolio of collaboration technologies, the project experienced several challenges related to project continuity, communication between the stakeholder groups, and commitment and involvement. This illustrates the complexity in this type of IS decision process, characterized by many stakeholders, extensive search, and modified design. Overall, our findings support earlier calls for a more differentiated view on IS decision-making processes, with explicit attention to the specific attributes characterizing each process.

The study also contributes insight into the potential role of collaboration technologies for supporting IS decision-making processes. The use of a common

information repository and a well established communications infrastructure were important factors for successful completion of the project. A more differentiated and role-based access to the information repositories, and more extensive use of online discussion forums were identified as potential means for increasing involvement in the process, and closing communication gaps between stakeholder groups through increasing cross understanding. Further, combination of different technologies for supporting *multi-mode interaction* stands out as a potential way of enabling more effective interaction in time pressed activities. Finally, the positive effects from the use of electronic meeting support indicate that this technology could favorably have been utilized more in this project.

Further research should investigate more the potential role of different forms of collaboration technology support for organizational decision-making processes. By analyzing relationships between attributes of these processes and technology use, this may further increase our understanding of suitable combinations of technology support for different phases and activities in collaborative decision-making processes.

Acknowledgments

We are grateful to the Statoil interviewees for sharing their experiences with us, and to Bjørn Tvedte for facilitating the data collection for this study and for providing useful comments.

References

1. Anson, R., Munkvold, B.E.: Beyond face-to-face: a field study of electronic meetings in different time and place modes. *Journal of Organizational Computing and Electronic Commerce* 14(2) (2004) 127-152
2. Boonstra, A.: Structure and analysis of IS decision-making processes. *European Journal of Information Systems* 12 (2003) 195-209
3. Borges, M.R.S., Pino, J.A., Araujo, R.M.: Bridging the Gap Between Decisions and Their Implementations. In: de Vreede, G.-J. et al. (eds.): *CRIWG 2004. Lecture Notes in Computer Science*, Vol. 3198. Springer-Verlag, Berlin Heidelberg (2004) 153-165
4. Brézillon, P., Adam, F., Pomerol, J.-C.: Supporting Complex Decision Making Processes with Collaborative Applications – A Case Study. In: Favela, J., Decouchant, D. (eds.): *CRIWG 2003, Lecture Notes in Computer Science*, Vol. 2806. Springer-Verlag, Berlin Heidelberg (2003) 261-276.
5. Fjermestad, J., Hiltz, S.R.: An Assessment of Group Support Systems Experimental Research: Methodology and Results. *Journal of Management Information Systems* 15(3) (1998-1999) 7-150
6. Fjermestad, J., Hiltz, S.R.: Group Support Systems: A Descriptive Evaluation of Case and Field Studies. *Journal of Management Information Systems* 17(3) (2000-2001) 115-160
7. Huber, G.P., Lewis, K.: Cross Understandig in Decision Groups: Analysis and Support. In: Meredith, R. et al. (eds.): *DSS2004 Conference Proceedings*, Prato, Italy (2004) 381-391
8. Miles, M.B., Huberman, A.M.: *Qualitative Data Analysis. An Expanded Sourcebook*. 2nd edn. Sage, Beverly Hills, California (1994)
9. Mintzberg, H., Raisinghani, D., Theoret, A.: The structure of unstructured decision processes. *Administrative Science Quarterly* 21(2) (1976) 246-275

10. Munkvold, B.E., Tvedte, B.: Implementing a Portfolio of Collaboration Technologies in Statoil. In Munkvold, B.: *Implementing Collaboration Technologies in Industry: Case Examples and Lessons Learned*. Springer-Verlag, London (2003) 81-107.
11. Sabherwal, R., King, W.: An empirical taxonomy of decision-making processes concerning strategic applications of information systems. *Journal of Management Information Systems* 11(4) (1995) 177-214
12. Shim, J.P., Warkentin, M., Courtney, J.F., Power, D.J., Sharda, R., Carlsson, C.: Past, present, and future of decision support technology. *Decision Support Systems* 33 (2002) 111-126
13. Todd, P, Benbasat, I.: The Impact of Information Technology on Decision-Making: A Cognitive Perspective. In: Zmud, R.W. (eds.): *Framing the Domains of IT Management*. Pinnaflex, Cincinnati, Ohio (2000) 1-14

Software Requirements Negotiation Using the Software Quality Function Deployment

João Ramires^{1,4}, Pedro Antunes^{1,3}, and Ana Respício^{2,3}

¹ LaSIGE – Large Scale Information Systems Laboratory

² Centro de Investigação Operacional

³ Department of Informatics of the Faculty of Sciences of the University of Lisboa,
Bloco C6, Campo Grande, 1700 Lisboa, Portugal

{paa, respicio}@di.fc.ul.pt

⁴ Centro Nacional de Pensões, Av. República 102, 3º Lisboa, Portugal
joao.j.ramires@seg-social.pt

Abstract. We propose a groupware tool supporting the Software Quality Function Deployment approach to software requirements validation. The design challenge is to involve several stakeholders, having conflicting views and attitudes which may be difficult to reconcile, in the requirements validation. The adopted approach integrates collaboration and negotiation support. Negotiation models inspired the development of a set of mechanisms promoting integrative attitudes and avoiding distributive ones. Experiments with the tool revealed some usability problems, but also showed that it is convenient to use and beneficial promoting consensus.

1 Introduction

Best engineering practices recommend that product quality should be addressed before and constantly evaluated during the product development. Furthermore, this vague notion of “product quality” should refer to concrete system attributes, addressing both the stakeholders’ needs and technical activities necessary to deploy the product. This perspective is central in the Total Quality Management (TQM) trend adopted by many organizations pursuing excellence in software development [2].

Quality Function Deployment (QFD) [3] is often used to implement TQM. QFD aims to define relationships and ultimately match the users’ and technical requirements [4]. QFD is used by manufacturing firms and has been lately applied to software production [2, 5]. In this later context, the fundamental value provided by QFD is focussing the software development process on the users’ perspective: the Voice of the Customer (VoC [6]). Although the traditional software development processes recognize the importance of the users, they do not offer simple mechanisms to verify the compliance with users’ requirements through all development stages (lifecycle tracking [4]). The Software QFD (SQFD [5]) fills this gap in software engineering. The SQFD approach is considered part of the Capability Maturity Model (CMM) level 4 implementation [6].

In a very simplified view, SQFD is a matrix of correlation values between requirements and specifications. This matrix is used in the following way [5]: (1) users' requirements are solicited to relevant stakeholders and placed in the left-hand side; (2) with the help from the stakeholders, the requirements are converted to technical specifications and placed at the upper side; (3) the stakeholders are then invited to complete the matrix with their perceived correlations; (4) a list of requirements priorities is defined; and (5) a list of technical specifications priorities is defined.

The correlations may be expressed in several ways, although a four-point scale ("none," "weak," "medium" and "strong," or 0, 1, 3 and 9 numeric values) is most often used [5]. The selection of a correlation is a qualitative task, where the objective is to identify the most appropriate link between "what" will be implemented and "how" the implementation corresponds to the stakeholders' expectations. Since there are many stakeholders involved, it is natural that different values may be proposed, according to different perspectives about the system, interpretations of what is involved in system development, hidden agendas, etc.

Three alternatives for obtaining SQFD correlations have been documented in the literature: (1) requesting individual responses and averaging the results, possibly using a moderation factor such as the relative importance attributed to each stakeholder [4]; (2) using multi-criteria preference analysis to combine individual preferences into some utility function [7, 8]; and (3) in a meeting, where the stakeholders must negotiate their different opinions until a consensus is achieved [9].

Although there are differences between the first two approaches, their focus is on the individuals, while the later approach stresses the commitment of the whole group to the SQFD process. This later approach is considered beneficial for team building, increasing the involvement in product development, obtaining overall consensus about "what to do," and preserving momentum when the group changes [9].

One problem with the later approach is that, being based on meetings and a definite need to negotiate, the evaluation process may become time-consuming. According to [10], two consequences of increased participation in meetings are a decrease in response time and a decrease of total time available for decision making in organizations. This problem naturally increases with the size of the SQFD matrix.

We aim to develop a groupware tool supporting parallel work and facilitating consensus on the SQFD matrix, thus reducing the required amount of time to accomplish the task. The proposed groupware tool, which is named MEG, integrates SQFD support with collaboration, negotiation and argumentation support. MEG is at the same time a Group Support System (GSS) and a Negotiation Support System (NSS), thus falling in a category of tools commonly designated Group Decision and Negotiation Support System (GDNSS [11]).

More generally, we aim to research the integration of GSS and NSS functionality, addressing their differences in the conflict dimension. Regarding the low-conflict facet of the GSS perspective, we propose an approach to promote integrative attitudes and avoid distributive ones. Recognizing the high-conflict facet of the NSS perspective, the proposed approach also supports parallel negotiation processes, argumentation, bargaining and firm attitudes from the users.

The paper is structured in the following way. We start describing the requirements for integrating SQFD with collaboration and negotiation models in a coherent groupware tool. We then provide a detailed description of MEG. Finally, we describe two experiments with the tool, discuss the related work and present the obtained results.

2 SQFD and Negotiation – Requirements Definition

The SQFD matrix has cells correlating users’ requirements with technical specifications. A correlation value measures the preference for the technical specification to fulfil the satisfaction of the requirement corresponding to a cell. We adopt the four-point scale with values 0, 1, 3 and 9. For reading convenience, correlation values will be referred as C . For example, in Figure 4, the requirement “Reply to mails easily” relates with the specification “Integrate external editor” with $C=9$, suggesting that this specification is strongly satisfying.

Assuming that several stakeholders work in parallel and select different C leads to a conflict situation addressed by our initial requirements:

- R 1.** *MEG will support the specification of different preferences for C , so that stakeholders may express their different views while working in parallel.*
- R 2.** *When alternative C have been proposed, MEG will support the negotiation of C until a final value is accepted by all stakeholders.*

Any negotiation requires information exchange. The SQFD model does not explicitly represent such information, but only the proposed C . One clear advantage of adopting groupware to implement SQFD is the possibility of extending the SQFD matrix with a shared memory component preserving the positions and arguments produced by the stakeholders. We adopted IBIS for that purpose. In Figure 1 we illustrate how SQFD and IBIS are combined to organize information pertaining to the negotiation of one SQFD cell [12]. The conflicting C generate an issue, for which there is an initial bid and a subsequent negotiation to reach an agreement. In that process, the stakeholders may add arguments to their preferences. This approach is articulated by the following requirement:

- R 3.** *MEG will handle multiple preferences for C as an issue, identifying an initial bid and positions against or in favour, and will allow the stakeholders to express their positions and arguments in favour of different C values.*

One further aspect related with SQFD and negotiation concerns the degree of information sharing supported by groupware. Very often, groupware assumes that the group has a shared goal and conflicts may be resolved with the support to shared representations of problems, issues and alternatives [13]. On the contrary, negotiation assumes conflict between the parts involved, turning more difficult the creation of shared representations while increasing the process weight. The challenge then is that

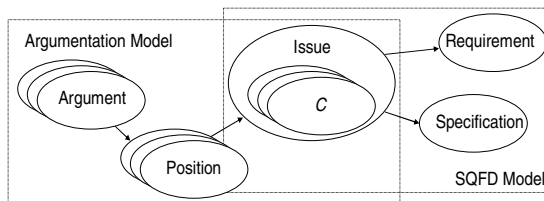


Fig. 1. Integration of SQFD and argumentation models for one SQFD cell

the integration of both perspectives creates some tensions between the support to the individual and group goals, and the support to shared representations and negotiation processes. These observations lead us to define a clear frontier between the stakeholders' hidden and shared knowledge [14]:

R 4. *MEG will not disclose the individual preferences of C, only their positions and arguments.*

We will now address the negotiation of *C* for one SQFD cell, considering that all cells are handled in the same way. Negotiation behaviour can be analysed according to two different strategies paradigmatic in negotiation research [15]:

- *integrative*, where an agreement is found in an inventive and collaborative way, exchanging information about preferences and priorities, and seeking common gains – both parts win (Win-Win);
- *distributive*, persuading the other part to accept an offer while disregarding the counteroffers – this is a game of winning and losing (Win-Lose).

The negotiation process often follows a differentiation-before-integration pattern [16], where the negotiation starts with distributed behaviours until an impasse is reached, and then the participants switch to integrative behaviours to avoid failure.

Most academic and non-academic literature shows a bias towards the integrative strategy [17], because of two main reasons: (1) it represents a zero-sum solution, since the gains obtained by one party represent losses from the other; and (2) the fundamental values behind the integrative strategy – interpersonal trust, cooperation and search for mutually acceptable outcomes – are favoured by most scientists' value systems. We will also follow the policy of favouring the integrative strategy.

According to [16], the switch to the integrative behaviour requires the combination of two conditions: an impasse and the willingness to engage in integrative behaviours. We investigate another alternative: using groupware to foster users engaging in integrative behaviours. To accomplish this endeavour, we have to further explain the differences between the integrative and distributive strategies, based on the following set of negotiation attitudes defined by [18]:

Competition - when one party tries to convince the other to accept a stake that is only favourable to self interests. This clearly corresponds to a Win-Lose attitude.

Collaboration - when both parties collaborate to maximize common gains (Win-Win).

Compromise - when both parties split the benefits. It is a satisfactory, although not necessarily an optimal result, since each party may not achieve all intended goals. This attitude leads to moderate Win-Win situations.

Obliging - when one party accepts a stake that is only favourable to the other party. This attitude occurs for several reasons, e.g. to close rapidly the process or simply because the issue is perceived as not important. This is usually considered a Lose-Win attitude. However, the literature reports that the obliging effects are unclear on the long run[19]: producing positive effects by eliminating conflicts, but on the other hand losing the opportunity to maximize common gains. In our context, we regard this attitude as neutral in terms of integrative/distributive behaviours. This view assumes that in SQFD parties engage in multiple negotiations, and thus the importance of a single Lose-Win is reduced.

Avoidance - when one or both parties decide to retreat. If there is a dependency on the negotiation process, this attitude frustrates the other's intentions (Lose-Lose strategy). It is also used, for instance, when one party seeks to use time pressure to own benefits (pursuing a Win-Lose strategy).

Based on the above attitudes, we adopted three fundamental requirements:

- R 5.** *MEG will favour Win-Win behaviours, which includes support to collaboration and compromise.*
- R 6.** *MEG will provide some resistance to Win-Lose and Lose-Lose behaviours, i.e. competition and avoidance.*
- R 7.** *MEG will be neutral about the Lose-Win behaviour, i.e. obliging.*

3 Design of the SQFD Negotiation Tool

In this section we describe our solution model. We start focusing on accomplish requirements R1, R2, R3 and R4.

Given a pair (requirement, specification), the SQFD matrix stipulates the correspondent correlation value, $SQFD: R \times SP \rightarrow \{0,1,3,9\}$, where R is the set of requirements, SP is the set of specifications and $\{0,1,3,9\}$ is the set of feasible correlation values (a zero value corresponds to an empty cell).

When MEG starts, $SQFD_{rs} = 0, \forall r \in R, \forall s \in SP$, meaning that the default C is zero. Without loss of generality, in the following, we will consider a generic SQFD cell and the negotiation of the correspondent C .

The initial bidder is the first stakeholder specifying a non-zero C , while the value specified is the initial bid. MEG associates them to the cell through the concept of *ISSUE*, stated as

$ISSUE = (initial-bid, initial-bidder)$, where $initial-bid \in \{1,3,9\}$, $initial-bidder \in ST$, and ST is the set of stakeholders.

The initial bid is public and its instantiation opens up the opportunity for other stakeholders to express their preferences for C (requirements R1 and R2). All subsequent stakeholders attributing a value to the same cell will be treated as supporters or opponents to the initial bidder. A stakeholder may instantiate more than one preference for C . $PREF(S_i)$ specifies the preferences' tuple of stakeholder S_i :

$PREF(S_i) = \langle c_1, \dots, c_k \rangle$, where $S_i \in ST$, k is the tuple size ($0 \leq k \leq 3$), and $c_j \in \{1,3,9\}$ is the j -th preference value S_i , stated ($j \leq k$).

Only stakeholders participating in the definition of C have a non-empty *PREF* tuple ($k > 0$). These preferences are part of the hidden knowledge maintained by the system (requirement R4), since individual preferences are kept undisclosed to the other stakeholders. Based on issues and preferences, MEG identifies the supporters and/or opponents to the initial bided (requirement R3). Whenever a stakeholder has a set of preferences compatible with the *ISSUE* (i.e. considering stakeholder S_i , at least one of the values in $PREF(S_i)$ is equal to the initial-bid), MEG registers a position in

favour of the initial bid. When there is no such compatibility, MEG registers a position against. This is done by computing $POSITION(S_i)$ for all $S_i \in ST$, where

$$POSITION(S_i) = \begin{cases} In-Favour, & \text{if } \exists j: PREF(S_i)_j = initial-bid \\ Against, & \text{otherwise} \end{cases}$$

Stakeholders are offered the possibility of attaching arguments to their positions (requirement R3), which confers them additional negotiation abilities. This means that for a stakeholder S_i it may be defined a tuple of arguments, defined by

$$ARGUMENTS(S_i) = \langle Arg_1, \dots, Arg_k \rangle, S_i \in ST, Arg_j \in Ontology, j \leq k, 0 \leq k$$

An argument is a very short piece of text, such as “human factors” or “failure”. MEG assumes the ontology necessary to implement this functionality has been previously supplied. The idea behind this approach is that the stakeholders do not have to write their own arguments; they can select relevant and meaningful ones from the ontology. We have not addressed this aspect in great detail, since the ontology varies from organization to organization. In our experiments we relied upon generic roadmaps for quality assurance provided by software engineering literature.

We move now our attention to the negotiation support, which description is based on the states machine displayed in Figure 2. MEG assumes that, if there is at least one position against the initial bid, then there is a conflicting situation requiring a negotiation process. To handle that process, MEG deals with the concept of cell state. We consider an *Equilibrium* state, referred by E , that is reached whenever there is no ongoing negotiation for that cell, either because: a) has no preference assigned; b) has one single C ; or c) there is no position against the current C . A negotiation is successful whenever its end leads to state E , i.e., E is both the starting state and the only accepting state. All the other states (denoted by S, F, WW, WL and LL) are *negotiation* states. In Figure 2, plain arcs correspond to transitions associated with user actions, while dashed lines represent transitions related with system events (such as evaluating positions or attitudes).

The machine moves from E to S when at least one stakeholder has a position against the initial bid, thus starting a negotiation. F is reached when all positions against the initial bid have disappeared and the negotiation process is close to a final.

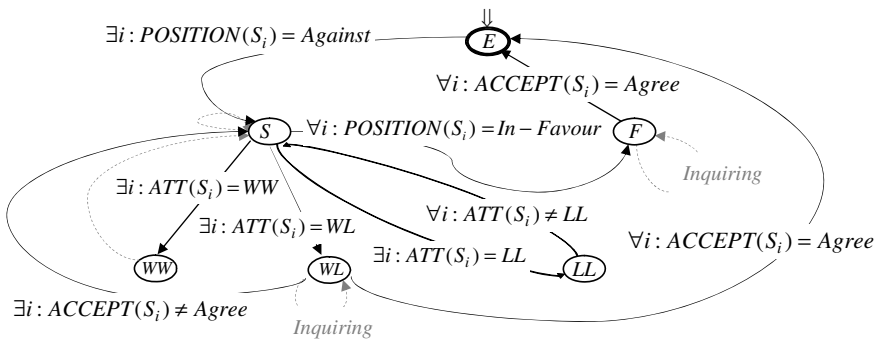


Fig. 2. Negotiation support (states machine)

MEG then requires all stakeholders involved in the negotiation to explicitly agree to finish the process (thus moving to E). $ACCEPT(S_i)$ denotes this explicit acceptance of stakeholder S_i when inquired by MEG. If S_i agrees to finish the negotiation $ACCEPT(S_i)=Agree$ and, otherwise, $ACCEPT(S_i)=Not-Agree$.

The other states intimately relate with the attitudes we have identified in requirements R5, R6 and R7. Stakeholder S_i may take an attitude $ATT(S_i)$ of the following types: Win-Win (WW), Win-Lose (WL), Lose-Win (LW) or Lose-Lose (LL).

The state WW is reached whenever a stakeholder takes a WW attitude. In this case, MEG re-calculates the set of positions (returning to S) and, if the conflict has disappeared (thus moving to F), attempts to finish the negotiation. The WL state is reached whenever a stakeholder changes preferences in a WL attitude. Movements out of WL depend on the result of users' inquire. Finally, the LL state is reached whenever a stakeholder adopts a LL attitude, a situation that requires MEG to suspend the cell negotiation until that attitude is revoked.

To understand the MEG functionality we also have to specify how these different attitudes are detected by the system. The specification is provided in Table 1.

Table 1. Behaviour detection

Attitude	Detection
Win-Win	$PREF$ became "closer" to <i>initial-bid</i>
Win-Lose	"Firm" option has been selected (see section 4.2 for explanation)
Lose-Win	$PREF$ has been removed
Lose-Lose	"Block" option has been selected (see section 4.2 for explanation)

3.1 Supporting the Integrative Approach

With requirements R5, R6, and R7 we declared the objectives to favour integrative strategies and resist to distributive strategies. We now describe how we addressed these issues. Expressing the problem in more concrete terms, our objective is to facilitate Win-Win, be neutral about Lose-Win, and create difficulties to Win-Lose and Lose-Lose attitudes.

According to [20] an integrative strategy is founded on "principled negotiation": (1) separate people from problems; (2) focus on interests, not on positions; (3) create options for mutual gains; and (4) use objective criteria. Our solution addresses these principles in the following ways:

- The stakeholders' identities are undisclosed. When a stakeholder originates an issue, position or argument, the information about who took that action is not displayed. This approach allows separating people from the problem.
- MEG does not show the stakeholders' preferred C , but only their positions relatively to the initial bid. This approach gives some latitude to changing positions and allows focussing more on interests than positions. MEG also allows the stakeholders to freely change their positions at any time during the negotiation.
- MEG creates opportunities for mutual gains by proposing a consensus value. The calculus of the consensus value is explained below.
- The ontology provides a standard mechanism for objectively arguing in favour or against an issue.

Whenever possible, under a conflicting situation, MEG proposes a consensus value for C that is obtained in the following way. Considering stakeholder S_i , the stakeholder weight in the negotiation is given by $SW(S_i) = 1 - (n_i \cdot 10^{-3})$, that decreases with n_i , the number of Win-Lose or Lose-Lose attitudes S_i has taken in the past. $UP(S_i, x)$ is the un-weighted preference for the correlation value $x \in \{1,3,9\}$ stated by that stakeholder, while $WP(S_i, x)$ is the corresponding weighted preference, now considering the stakeholder weight in the negotiation. These values are given respectively by

$$UP(S_i, x) = \begin{cases} 1, & \text{if } \exists j: PREF(S_i)_j = x \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad WP(S_i, x) = UP(S_i, x) \cdot SW(S_i).$$

$P(x) = \sum_{S_i \in ST} WP(S_i, x)$, $x \in \{1,3,9\}$, computes the total preference for x expressed by the stakeholders. And finally, $CONSENSUS = \max_{x \in \{1,3,9\}} (P(x))$ is the correlation value that obtained the highest number of occurrences in all the preferences' tuples, or Null if there is no such value.

In summary, we used majority voting, where votes are weighted according to the number of distributive attitudes taken during the system use. This approach is aiming at benefiting the stakeholders that take integrative attitudes. When a *CONSENSUS* value is obtained, MEG proposes it as a fair solution to the negotiation process, on par with the initial bid. MEG does not enforce the stakeholders to accept that value.

Now, we turn our attention to the mechanisms built in MEG to create difficulties to Win-Lose and Lose-Lose attitudes. MEG allows Lose-Lose attitudes using a "blocking" mechanism (mentioned in Table 1). Basically, the blocking mechanism allows one stakeholder to lead the negotiation to a suspended state (*LL*), so that the process stops until the stakeholder removes that condition or the SQFD task is concluded without consensus. To create some resistance to this attitude, MEG makes the user interaction with this mechanism difficult: the action is not easily accessible and several confirmations are required before activation.

MEG allows Win-Lose attitudes using a "firm" mechanism: one stakeholder may express to the others that he/she has a firm position about C . When this mechanism is activated, MEG informs all the other stakeholders and asks them if they accept that position or not (moving to state *WL*). In case all stakeholders accept, the negotiation process is finished, otherwise the negotiation continues. To create resistance to the usage of this mechanism, when MEG informs the stakeholders that someone has a firm position, it also informs about the total number of similar attitudes taken by that stakeholder. This information may influence the stakeholders not to accept firm positions from persons that have wield too many distributive attitudes in the past.

4 Implementation Details

MEG is a client-server tool implemented with MS Excel 2002, Access and Visual Basic 6.0. The system architecture is shown in Figure 3. The SQFD matrix was implemented with an Excel spreadsheet using RTD technology. Users may interact with

the spreadsheet but cannot directly modify the cells. Those modifications are requested to MEGCLIENT, which communicates with MEGSERVER, which in turn maintains the shared information stored on an Access database. The database is accessed through XML. The RTDSERVER updates the distributed spreadsheets using DCOM. The interaction between Excel and RTDSERVER is explained in [21].

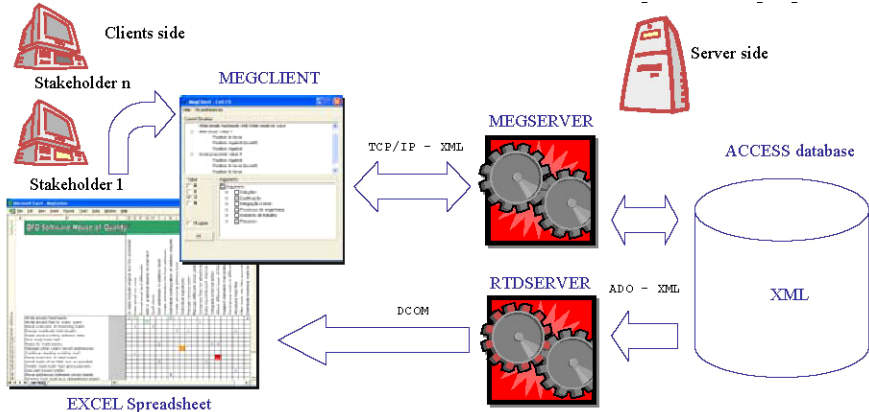


Fig. 3. System architecture

4.1 Illustration of MEG Functionality

Consider a set of three stakeholders: S1, S2 and S3. Figure 4 shows one SQFD matrix with several conflicting situations and ongoing negotiations. Observe that some cells present correlation values (1, 3 and 9, since 0 corresponds to a blank cell), while some others show the symbols “?”, “F” and “L”. These symbols are shown when the cell is under negotiation. The “?” indicates that the process is ongoing; while “F” and “L” indicate that a user expressed a firm position and locked the cell, respectively. The users do not directly manipulate the SQFD matrix. Instead, MEGCLIENT is invoked whenever one user double clicks on a cell.

We use the sequence of actions shown in Table 2 to illustrate how users interact with MEGCLIENT and the correspondent system reaction. Using MEGCLIENT to modify the SQFD cell E5, S1 selects C=1. Since the cell was previously empty, MEG creates an issue with 1 as initial bid and propagates it through the system. 1 will appear in E5 for all stakeholders. (Figures 5-8 illustrate interaction of S2 with MEG).

Afterwards, S2 decides to analyse E5, double clicking E5 to open MEGCLIENT. The issue is displayed, showing the proposed correlation but without identifying S1 as initial bidder (Figure 5). S2 does not agree with the correlation and selects C=3. MEG recognizes two conflicting proposals for E5 and initiates a negotiation process. The preferences list is constructed with one supporter (S1) and one opponent (S2) to the initial bidder (Figure 6). Note that the identity of the supporters and opponents is undisclosed. Furthermore, a “?” appears in E5.

S3 decides to enter the negotiation and proposes C=3. MEG recalculates the preferences, to come with one supporter and two opponents to the issue. MEG also analyses if there is a consensus value. Since no previous “firm” or “block” positions have

been used, the obtained consensus value is 3 ($P(1)=1$, $P(3)=2$ and $P(9)=0$). Therefore, MEG proposes 3 to the stakeholders (Figure 7).

S1, analysing the consensus value, decides to adopt a compromising attitude and adds 3 to the range of accepted correlations. MEG realizes there is one possible agreement on 3 and requests confirmation from all stakeholders (Figure 8). All users agree and the negotiation process finishes. The value 3 finally appears in cell E5.

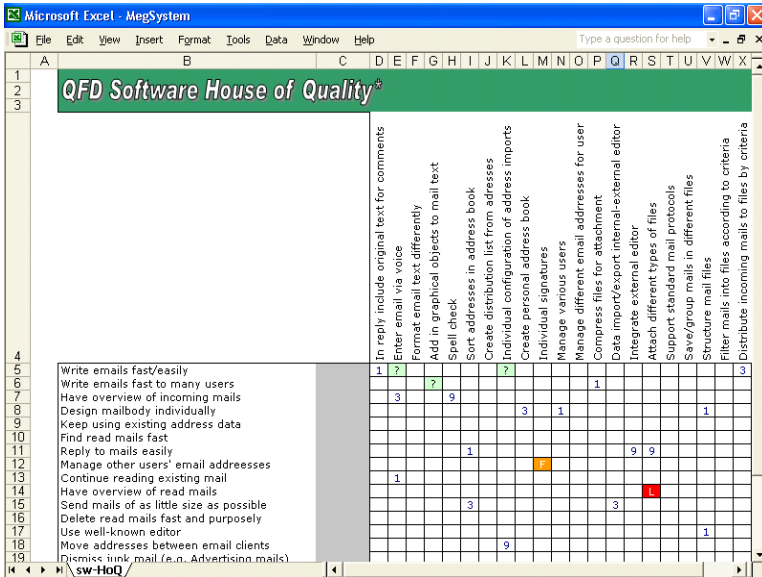


Fig. 4. SQFD matrix (example from [1])

Table 2. Sequence of actions accomplished by S1, S2 e S3 and system events

Time	S1	S2	S3	Action
t1	1			S1 selects $C = 1$
t2				S2 analyses cell
t3		3		S2 selects $C = 3$
t4				S3 analyses cell
t5			3	S3 selects $C = 3$
t6				MEG proposes $C = 3$
t7	1, 3			S1 adds $C = 3$ to selection
t8				MEG requests agreement

5 Evaluation

MEG was evaluated in two pilot experiments involving two stakeholders each. The participants had the following background: (A) more that 30 years experience in software development and requirements negotiation with outsourcing organizations;

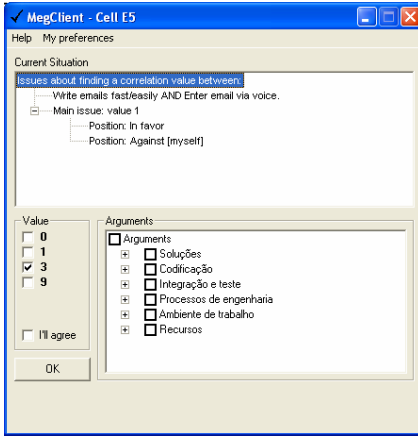


Fig. 5. MEGCLIENT in t2 for S2

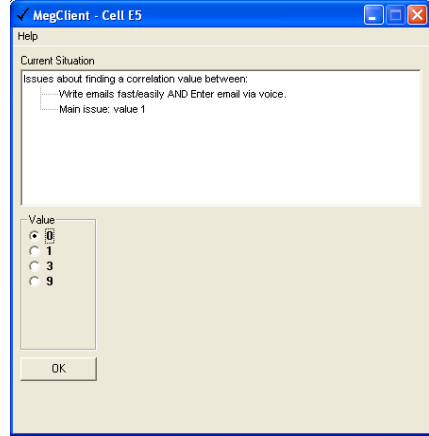


Fig. 6. MEGCLIENT in t3 for S2

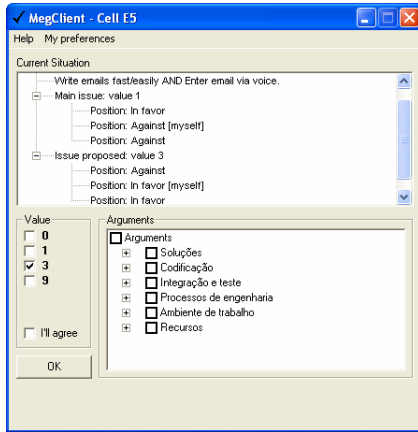


Fig. 7. MEGCLIENT in t6 for S2

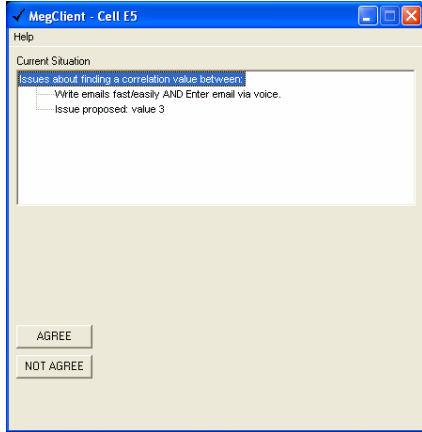


Fig. 8. MEGCLIENT in t8 for S2

(B) 6 years experience in systems analysis; (C) project coordinator in a large company; and (D) analyst/programmer in statistics and operational research.

The experiments were accomplished in the context of a governmental agency responsible for the national pensions system. The project concerned the introduction of a new formula for computing pensions. The goal set for the pilot experiments was to construct and evaluate the SQFD matrix designated as “House of Quality” (HoQ). The HoQ correlates preliminary lists of user and technical requirements, so that priorities can be set early in the project.

The HoQ was specified in the following way. We interviewed stakeholder A, who is deeply knowledgeable about the problem context. His recommendations allowed us to specify the user and technical requirements. We followed ISO/IEC 9126 to finally structure the quality requirements:

Functionality	Apply new formula Integrate with current formulas in the pension application
Reliability	Detailed contingency plan
Usability	Provide adequate training Document new functionality Clearly define modifications to existing processes
Maintenance	Add new functionality with minimum operational modifications
Portability	Detailed migration plans

The following list of technical requirements was specified by the authors based on the recommendations of stakeholder A:

Functionality	Calculus of pensions for person P Calculus and demonstration of pensions for entities E1 and E2 Store data according to user profile Client can operate in different OS Display specific legislation used in calculus User authentication
Reliability	Complete a transaction cycle without execution errors Continue operation if data is not available Continue operation if write error Use secondary server if main server fails Recover operations completed before a power failure
Usability	Organize items logically in screen Provide online explanations of calculus Provide online help Alert that new functionality is available
Maintenance	Show how to realize calculus Support configuring codes and parameters Use several modules Reuse some modules Facilitate data access in every screen Data must look the same in every printer
Portability	Export data to Excel Use install rules from internal doc PPP/2004 Use portability rules from internal doc 002/2004

The resulting SQFD is therefore an 8x24 matrix with 192 correlation values. Each pilot experiment started with a brief tutorial about MEG, which took approximately about 15 minutes. Then, the SQFD was negotiated by a pair of stakeholders until a consensus was obtained. During the experiment, whenever necessary, additional help about the MEG functionality was provided by the authors, which participated in the process as observers.

Beyond obtaining the SQFD matrixes with correlations, we requested the participants to fill up a questionnaire with questions about the MEG functionality and usability, as well as open questions about the most positive and negative aspects. The following quantitative results were obtained:

Functionality	1 (<)	2	3	4	5 (>)
Convenience (the available functions are appropriate for the task at hand)				4	
Precision (the obtained results reflect your opinions)			2	2	
Agreement (you agree with the consensus and majority voting approach)				3	1
Usability					
Comprehension (your effort to understand the application logic)			2	1	1
Learning (your effort to learn how to use the application)			2	1	1
Operability (your effort to control the negotiation process)			2	1	1

The list of positive and negative aspects was as follows:

Positive	Participant
Easy finding point of agreement	A
Knowing arguments from others to evaluate and eventually revise my position	A
Better understanding of the overall ideas from stakeholders	B
The "current situation" closes every time a change is made, which is positive because it obliges to read modifications	B
The system does not show how many others have confirmed their positions	B
The negotiation model is efficient, although for top management it should be more graphical	C
The integration of negotiation attitudes with the QFD affords obtaining reliable results	D
Negative	
Very slow	A
Unusable by common users	A
It is more intuitive to qualify correlations by names than numbers	A
The situation where all stakeholders are in favour but one does not press the option "I agree" is confusing, because the consensus was rejected but all were in favour	A
If 2 stakeholders obtain an agreement, that value goes to a cell. If another pair negotiates a different value, the initial pair is not informed	B
System is slow	B
The information shown in "current situation" should be presented graphically	C
The graphics should be more intuitive. For instance, it is more intuitive for a manager to see that there are N stakeholders in favour or against a value	C
The value obtained by consensus by a group of stakeholders may be substituted by another group of stakeholders without notifying the first one	D

Combining the quantitative and qualitative results obtained from the questionnaire, we arrived at the following conclusions about MEG:

- **The system is considered difficult to use.** The three criteria related with usability had the lowest score in the questionnaire. This situation is reinforced by negative comments from stakeholders A and C. Two stakeholders (A and B) also considered that the system had bad performance (this situation is caused by DCOM).
- **All stakeholders agreed that the system is convenient to use.** This position is reinforced by several positive comments about understanding the overall positions from others, revising own positions and ease finding agreements.
- **All stakeholders agreed that the consensus approach, complemented by majority voting, is beneficial.** The agreement criterion was the one that received the highest score, reinforced by positive comments about the ease to reach consensus.

Several minor functional and user interface details were also raised by the stakeholders, for instance, about the use of the "I agree" button, renegotiation of cells overriding previous consensus, and difficulties in obtaining summary view of the negotiation processes. These comments should be used in future versions of MEG but do not reflect any significant issues about the core design decisions made.

6 Related Work

In Table 3 we show a comparison of MEG with related systems using four criteria: (1) support to GSS features; (2) support to NSS features; (3) argumentation support; and (4) dependence on the facilitator.

Table 3. Comparison between MEG and related systems

	GSS	NSS	Argumentation	Facilitator
MEG	Supports parallel activities Shared memory component Proposes consensus value	Bid support Supports private preferences	Based on IBIS Uses pre-defined ontology	No
Easy-WinWin [22]	Uses GroupSystems Follows Win-Win methodology	No	No	GroupSystems requires facilitator
ME-DIATOR [23]	No	Defines goals and utility spaces Identifies equilibrium Suggests compromises	No	Facilitator aids the construction of shared representation
Hermes [24]	Discussion forum	Updates process status Recommends solutions Finds inconsistencies	Based on IBIS	No
Virtual QFD [1, 25]	Web-based tools: discussion panels, VoC tables, evaluation panels and QFD matrixes	No	No	Facilitator manages data during meetings
Co-Decide [26]	Multi-user spreadsheet extension Offers OLAP features	No	No	No

Comparing EasyWinWin and MEG, we observe the former neglects negotiation support. EasyWinWin uses generic GSS tools (GroupSystems' brainstorming, categorizing and voting tools) with two major consequences: dependence on the facilitator to manage the technology; and limited support to parallel activities. EasyWinWin follows the Win-Win principle [27] that all stakeholders should win, and guides the users through a process where winning conditions are identified and negotiated until mutual agreements are obtained.

MEDIATOR is strictly a NSS where problem representation evolves by sharing individual points of view and searching for a point of equilibrium. The system uses a set of dimensions to define goals and utility spaces. The evolution of utility spaces is displayed in matrix or graphical form. Like MEG, compromising solutions can be suggested. The system supports a facilitator who aids in the construction of a shared problem representation.

Hermes is the system more closely related with MEG: it organizes arguments using IBIS, assists the negotiation process with updated information about the process status, recommends possible solutions, and also searches for inconsistencies among users' preferences. However, it offers limited GSS support, and the adoption of a discussion forum makes it inadequate for handling a large number of requirements.

Virtual QFD basically supports data sharing before and during meetings using the Web. Available tools include discussion panels, VoC tables, evaluation panels and QFD matrixes. Unlike MEG, a facilitator is required to manage data during meetings.

Finally, Co-Decide is a multi-user extension to a single-user spreadsheet. The basic idea behind Co-Decide is to extend typical OLAP features to multiple users. Unlike MEG, Co-Decide does not support the negotiation process.

With this necessarily brief overview we show that the fundamental characteristic of MEG is bringing together several characteristics of GSS and NSS, in particular support to shared and private data, parallel work, bidding and argumentation.

7 Discussion

MEG improves the effectiveness of SQFD combining functionality attributed to GSS and NSS (Table 4). A unique characteristic of MEG is that it attempts to stimulate users to assume integrative attitudes based on a set of subtle interventions at the user-interface level, underpinned by models of negotiation processes. Experimented solutions included: 1) reducing the accessibility to Win-Lose and Lose-Lose attitudes, making difficult the access to associated buttons and requesting unnecessary confirmations; 2) associating a cost to Win-Lose and Lose-Lose attitudes and showing that cost to the group; 3) supporting Win-Win and Lose-Win attitudes, avoiding focus on definite values, facilitating position changes and multiple choices; 4) promoting Win-Win attitudes, recommending a consensus value based on majority voting. MEG also supports ontology based argumentation. The objective is to reduce the levels of conflict by exchanging standard messages, meaningful in the domain, instead of free text.

Table 4. Combining GSS and NSS perspectives

NSS (high-conflict)	GSS (low-conflict)
- Bargaining	- Multiple parallel negotiations
- Firm positions	- Stimulate integrative attitudes
- Negotiation blocking	- Avoid distributive attitudes
	- Ontology based argumentation

One interesting outcome from this combination of GSS and NSS functionality is that the resulting tool offers more latitude and flexibility handling group strategies: the system supports low-conflict collaborative situations, but is also capable to cope with increased levels of conflict in a flexible way. The results from the pilot experiments indicate that the approach is considered beneficial for reaching consensus.

Acknowledgements

This paper was partially supported by the Portuguese Foundation for Science and technology, Project POSI/EIA/57038/2004.

References

- [1] G. Herzwurm, S. Schockert, U. Dowie, and M. Breidung, "Requirements Engineering for Mobile Business Applications," Proceedings of the First International Conference on Mobile Business, Athens, Greece, 2002.
- [2] R. Zultner, "Tqm for Technical Teams," *Communications of the ACM*, vol. 38, pp. 79-91, 1993.
- [3] Y. Akao, *Quality Function Deployment: Integrating Customer Requirements into Product Design*. Cambridge, Massachusetts: Productivity Press, 1990.
- [4] A. Stylianou, R. Kumar, and M. Khouja, "A Total Quality Management-Based Systems Development Process," *The DATA BASE for Advances in Information Systems*, vol. 28, pp. 59-71, 1997.
- [5] S. Haag, M. Raja, and L. Schkade, "Quality Function Deployment Usage in Software Development," *Communications of the ACM*, vol. 39, pp. 41-49, 1996.
- [6] G. Antonioli, S. Gradara, and G. Venturi, "Methodological Issues in a Cmm Level 4 Implementation," *Software Process Improvement and Practice*, vol. 9, pp. 33-50, 2004.
- [7] H. In, D. Olson, and T. Rodgers, "Multi-Criteria Preference Analysis for Systematic Requirements Negotiation," Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02), Oxford, England, 2002.
- [8] P. Chuang, "Combining the Analytic Hierarchy Process and Quality Function Deployment for a Location Decision from a Requirement Perspective," *Advanced Manufacturing Technology*, vol. 18, pp. 842-849, 2001.
- [9] V. Bouchereau and H. Rowlands, "Quality Function Deployment: The Unused Tool," *Engineering Management Journal*, pp. 45-52, 2000.
- [10] V. Vroom and A. Jago, *The New Leadership: Managing Participation in Organizations*. Englewood Cliffs: Prentice-Hall, 1988.
- [11] E. Bellucci and J. Zeleznikow, "A Comparative Study of Negotiation Decision Support Systems," Thirty-First Annual Hawaii International Conference on System Sciences, Kohala Coast, Hawaii, 1998.
- [12] Y. Reich, "Ai-Supported Quality Function Deployment," *Artificial Intelligence in Economics and Management*, pp. 91-106, 1996.
- [13] M. Bichler, G. Kersten, and S. Strecker, "Towards a Structured Design of Electronic Negotiations," *Group Decision and Negotiation*, vol. 12, pp. 311-335, 2003.
- [14] J. Johannessen, J. Olaisen, and B. Olsen, "Information Management in Negotiations: The Conditions under Which It Could Be Expected That the Negotiation Partners Substitute a Competitive Definition of the Situation for a Cooperative One," *International Journal of Information Management*, vol. 17, pp. 153-168, 1997.
- [15] R. Lewicki and J. Litterer, *Negotiation*. Homewood, Illinois: R. D. Irwin, 1985.
- [16] F. Harinck and C. Dreu, "Negotiating Interests or Values and Reaching Integrative Agreements: The Importance of Time Pressure and Temporary Impasses," *European Journal of Social Psychology*, vol. 34, pp. 595-611, 2004.
- [17] R. Lewicki, S. Weiss, and D. Lewin, "Models of Conflict, Negotiation and Third Party Interventions: A Review and Synthesis," *Journal of Organizational Behavior*, pp. 209-252, 1992.
- [18] K. Thomas and R. Kilmann, *Thomas-Kilmann Con.Ict Mode Instrument*. Escondido, California: Blanchard Training and Development, 1974.
- [19] R. Friedman, S. Tidd, S. Currall, and J. Tsai, "What Goes around Comes Around: The Impact of Personal Conflict Style on Work Conflict and Stress," *The International Journal of Conflict Management*, vol. 11, pp. 32-55, 2000.

- [20] R. Fisher and W. Ury, *Getting to Yes: Negotiating Agreement without Giving In*. New York: Penguin Books, 1983.
- [21] P. Cornell, "Building a Real-Time Data Server in Excel 2002," Microsoft Corporation 2001.
- [22] B. Boehm, P. Grünbacher, and R. Briggs, "Developing Groupware for Requirements Negotiation: Lessons Learned," *IEEE Software*, vol. 18, pp. 46-55, 2001.
- [23] M. Jarke, T. Jelassi, and M. Shakun, "Mediator: Toward a Negotiation Support System," *European Journal of Operational Research*, vol. 31, pp. 314-334, 1987.
- [24] N. Karacapilidis and D. Papadias, "Computer Supported Argumentation and Collaborative Decision Making: The Hermes System," *Information Systems*, vol. 26, pp. 259-277, 2001.
- [25] G. Herzwurm and S. Schockert, "Virtual Product Development," Proceedings of the Fifth International Symposium on Quality Function Deployment and the First Brazilian Conference on Management of Product Development, Belo Horizonte, Brazil, 1999.
- [26] M. Gebhardt, M. Jarke, and S. Jacobs, "A Toolkit for Negotiation Support Interfaces to Multi-Dimensional Data," *SIGMOD Conference*, pp. 348-356, 1997.
- [27] B. Boehm and R. Ross, "Theory-W Software Project Management Principles and Examples," *IEEE Transactions on Software Engineering*, vol. 15, pp. 902-916, 1989.

The Design and Field Evaluation of a Repeatable Collaborative Software Code Inspection Process

Pushpa G. Koneri¹, Gert-Jan de Vreede^{1,2},
Douglas L. Dean³, Ann L. Fruhling¹, and Peter Wolcott¹

¹ College of Information Science & Technology,
University of Nebraska at Omaha

{pkoneri, gdevreede, afruhling,
pwolcott}@mail.unomaha.edu

² Faculty of Technology, Policy and Management,
Delft University of Technology, The Netherlands

³ Marriott School of Management, Brigham Young University
doug_dean@byu.edu

Abstract. The use of software products in today's world has increased dramatically making quality an important aspect of software development. There is a continuous need to develop processes to control and increase software quality. Software code inspection is one way to pursue this goal. This paper presents a collaborative code inspection process that was designed during an action research study using Collaboration Engineering principles and techniques. Our inspection process was implemented as a sequence of thinkLets, chunks of facilitation skill, that were subsequently field tested in a traditional paper-based and Group Support System (GSS)-based environment. It was found to be successful in uncovering many major, minor as well as false-positive defects in inspected pieces of code. Results illustrate the process' efficiency in identifying duplicate defects thereby reducing follow-up time to correct each defect. The inspection process' flexibility was observed as it was successfully applied to inspect both pieces of code or an entire module. Overall the collaborative inspection process was considered to be productive for code inspection and was satisfactory for the inspectors involved.

1 Introduction

A software defect is a “material breach of the contract for sale or license of the software, if it is so serious that a customer can justifiably demand a fix or can cancel the contract, return the software, and demand a refund” [21]. Software inspection is aimed at detecting software defects thereby improving the quality of software products. Inspections can focus on the design documents or the source code itself. Fagan [8] defines inspections as a “formal, efficient and economical method of finding errors in design and code”. Well-executed software inspections can add substantial value to the software development process. According to Klimas [15], around 80% of anomalies are removed through inspection, while one hour of

inspection will uncover as many anomalies as four hours of regression testing. Furthermore, every hour invested in software code inspection can save up to five hours in later stages [15].

Software can be inspected at different stages of its development [24]. In the *Software requirement specification* stage the specifications requirements document is inspected for its completeness. An inspection plan is developed that includes requirement specification, description of inspection procedures, schedules and milestones. In the *Design* stage, the adequacy of the software design is determined by checking the consistency of design with the requirements specified. The various design products can be inspected through analysis, simulation, and walkthroughs. In the *Software coding* stage, the code is checked for its correct syntax, logic and execution. Different aspect of codes such as logic error, data error, input/output error, processing time, programming standards that are followed, and conflicts with the specification requirements are also inspected. In the *Software testing* stage, the test cases are inspected for their correct functional objectives. Test procedures, test schedules and test milestones are also inspected for their accuracy. Finally, in the *Software Maintenance* stage, inspectors check to keep the software product in line with the current industry trends.

At each of these stages, software inspection requires the proper coordination of the activities involved because inspecting a software product is not a one-person task. Usually the software inspectors are not the same people who develop the product. Also, inspectors have to work together to perform a comprehensive inspection as modern software products are too large to be inspected by a single person. Finally, the same elements of a software product are often inspected by more than one inspector to ensure higher inspection quality. In other words, software inspection is a collaborative process that has to follow a systematic approach to maximize both inspection efficiency and effectiveness [6]. A well-defined software inspection process lays out the ground rules that allow objective rather than subjective decision-making regarding potential defects [4].

The research reported in this paper focuses specifically on software *code* inspection. The objective of software code inspection is to identify and remove software bugs before testing the code [9]. Each defect identified during code inspection saves around 9 hours of time in later stages [9]. A survey [9] showed some striking examples of the value of code inspections:

- IBM managed to remove 82% of all defects before testing even starts.
- AT&T found inspections led to 14% increase in productivity and tenfold increase in quality.
- HP found 80% of the errors detected during inspections were unlikely to be caught by testing.
- HP, Shell Research, Bell Northern, and AT&T all found inspections 20 to 30 times more efficient than testing in detecting errors.

This paper reports on a study in which a collaborative software code inspection process was designed and evaluated in the field. The design of the process followed Collaboration Engineering principles to create a repeatable inspection process that inspection teams could execute by themselves [16,23]. Earlier research suggests that code inspection is an example of a collaborative process that can potentially be

facilitated by the participants themselves [10]. The evaluation of the collaborative code inspection process was performed by applying it to two cases in two organizations and determining the process' perceived efficiency and effectiveness, as well as the inspectors' satisfaction with it. Furthermore, the process was evaluated in two configurations: a paper-based implementation and a Group Support System (GSS) based implementation.

The remainder of this paper is structured as follows. Section 2 discusses code inspection and collaborative support for code inspections in more detail. Section 3 describes our research approach. The design of the collaborative code inspection process is presented in Section 4, along with the criteria to evaluate the process. Section 5 presents the results of the evaluation of process in the field. Finally, Section 6 discusses the findings and implications of our research and summarizes limitations and future research directions.

2 Background

Code inspection methods can be broadly classified into two categories: *Formal* and *Informal Inspection* methods. Formal Inspection methods are conducted in a well defined systematic and structured manner. These methods mostly execute a detailed, formal process and keep a record of the meeting results for later analysis to improve both the quality of the code and also of the inspection process itself. Informal Inspection methods do not follow any standard procedure. They can take place at regular project status meetings or even over a coffee. They do not follow any specific agenda and there is no documented output. Below we describe formal methods in more detail and discuss collaborative support for such methods.

2.1 Formal Software Code Inspection

A typical formal software code inspection consists of six steps (table 1, [8,11]). Formal code inspection methods have a number of advantages and disadvantages. According to Wiegers [24], the key advantages include a more thorough coverage of the software code, the identification of the most number of defects, and it being a good test of understandability and maintainability of the code. Among the disadvantages are issues like the need for thorough preparation prior to the inspection meeting, and the high costs per defect because of the use of multiple inspectors and the slow coverage [24].

Fagan [7] argues, the main reason for inferior inspection results is that the formal inspection process is not well understood and is often poorly executed. The satisfactory outcome of an inspection largely depends on how well the purpose of inspection is defined for the inspectors. The purpose can vary from understanding the code, maintaining the code, to identifying corrective actions on the defects uncovered. A clearly understood purpose affects how the inspectors examine the code. Further, the manual process of error documentation is tedious and prone to errors in case of miscommunication among programmers and code inspectors. Thus the entire inspection process can be time consuming and costly.

Table 1. Steps in a formal code inspection process

Step	Description
Planning	The materials to be inspected are arranged, the designated inspectors are checked for their availability, and the meeting place and the time is decided.
Overview	The inspection team is briefed about the scope and purpose of the code to be inspected to help them understand what is expected from them. This may include the “distribution of documents, role assignments and training in code inspection procedure, rules, and checklists” [11].
Preparation	The inspectors go through the software code individually and make an effort to understand it completely.
Inspection Meeting	The inspectors meet to identify issues (defects) in the code that is inspected. They are encouraged to find as many major issues as possible.
Rework	The identified issues are reviewed by the code author and are classified as defects that need to be corrected.
Follow up	The moderator makes sure that all necessary actions are undertaken on all identified defects. The moderator also has to make sure that no defects are introduced during rework. Finally, he decides if there is any possibility or need for re-inspection.

The Preparation step requires inspectors to understand the document and identify defects individually. During the inspection meeting the moderator then collates these defects. A lot of time is wasted here as inspectors often find and record duplicate defects due to the lack of a mechanism to share the defects as soon as they are identified by individual inspectors. Therefore a major amount of inspectors’ time is spent on “*redundant search*” [6]. During a typical inspection meeting each inspector waits for his turn to report defects to the moderator. This time could be used more effectively for detecting new defects thereby increasing inspection productivity.

Another limitation of traditional inspection process is that it relies heavily on face to face meetings [14]. If the inspectors are scattered in different geographical areas or simultaneously working on different projects then it is difficult to find time and place for inspection meetings. Mashayekhi et al. [18] propose a distributed collaborative software inspection environment that removes spaces constraints and provides sufficient structure to partially relax time constraints. In such an environment, developers and inspectors need not be present in the same location for the code inspection session. This considerably reduces the time and cost, and increases the flexibility of inspection [18].

2.2 Collaboration in Software Inspection

Software code inspection is a collaborative task. Software code is typically inspected by more than one inspector, each of which follows a set inspection guidelines and works together with others to accomplish the common goal of identifying the defects. Software code inspection is people-intensive. The inspection team typically consists

of the author of the code, a moderator, individual inspectors and at times even a software manager and a scribe. The moderator is responsible for getting the source code from the author. (S)he coordinates with the inspectors and distributes the code. The manager gives a short description about the code to the inspectors and explains the purpose of the inspection and what is expected from the team. The individual inspectors assemble to discuss the software code and detect as many defects as possible. During the meeting the defects are reported and discussed. The scribe records the identified defects in a formal defect log. The defect log is finally submitted to the author of the code for further review. Thus, the software code inspection process can be rightly termed as a collaborative process.

Inspection teams may use different types of tools to support their activities. Such tools can support the inspection activities itself, the collaboration between the people involved in the inspection process, or both. Computer support allows the inspection teams to handle inspection artifacts online, to record inspector comments and create project management reports which reduces much of the bulky printed materials and the forms that are normally generated by inspections [20]. For example, Koo et al. [17] designed a PC based application that supports verification and validation activities based on Fagan [8] inspections. They feel that a desirable attribute of inspection is “rigor” and using computers to support the process helps provide this.

Inspection teams also use groupware tools to “communicate, cooperate, coordinate, solve problems” [13]. One particular family of groupware tools that is increasingly being used in the context of code inspections are Group Support Systems (GSS). A GSS provides a set of configurable modules which can be used to support different collaborative activities for the collection, categorization, and evaluation of inspection artifacts [13]. According to recent studies, a GSS supported inspection process can significantly improve the number of defects found during software code inspection and appears to be more effective and efficient in finding defects [6,10]). The advantages of using GSS in code inspection include:

- GSS provide automatic report generation. All information collected during the inspection session can be formatted into a formal report that summarizes the entire session.
- GSS allow inspectors to work in parallel and also to look into other inspectors’ contributions. Thus inspectors can contribute and access information easily and efficiently. As inspectors working in a group can view defects identified by other inspectors, the number of duplicate defects is reduced.
- GSS allow inspectors to capture their comments themselves as they occur. This reduces the risk of misinterpretations by the scribe or forgetting them at a later stage in the inspection process.
- GSS support both distributed and face to face collaborative work. Inspectors need not be in the same room. They can contribute from their own office desk and also see what others have contributed.

However, GSS are typically complex systems to apply effectively in a sustained fashion [3]. They require the assistance of a skilled facilitator to be configured effectively [23]. For example, the GSS feature of anonymous communication appears to impact group performance in inspection tasks: Vitharana and Ramamurthy [22] found that non-anonymous groups were more effective and had a more positive attitude towards theory inspection task than anonymous groups.

However, most organizations will not have the resources available to employ skilled facilitators to operate GSS for inspection meetings [3]. It will be of more value to such organizations to have access to a repeatable collaborative method that clearly instructs an inspection team how to perform a collaborative inspection by themselves. The design of such a GSS-based process is the objective of the research described in this paper. The next section will elaborate on the research method that was used to achieve this objective.

3 Method

Our research objective was to design and evaluate a repeatable collaborative code inspection method in practice. Action research was considered the most suitable research methodology for this purpose. Action research has the dual intention of improving practice and contributing to theory and knowledge [1,5]. We followed a model [25] that states that an action research study may consist of four activities that can be carried out over several iterations: ‘Plan’ concerns exploration of the research site and the preparation of the intervention. ‘Act’ refers to the actual intervention. ‘Observe’ concerns the collection of data during and after the actual intervention to enable evaluation. Finally, the ‘Reflect’ activity analyses the collected data and infers conclusions that may feed into the ‘Plan’ activity of a new iteration.

Action research was selected as our research approach for several reasons. First, it is especially appropriate to address ‘how to’ research questions. Our research aimed to develop a process to perform collaborative code inspections. Second, we felt that there was too limited understanding of the collaborative aspects in code inspections to study it in a constructed, and tightly controlled, experimental setting. Finally, action research is very well suited for continuous learning. It allows researchers to continuously evaluate and improve their products during a series of interventions. In our research, we initially produced a few prototypes of the collaborative inspection process. The way in which each of the four steps was executed is as follows:

- Plan: We conducted a literature review and approached other researchers working on software code inspection. This fostered an understanding of limitations in the existing inspection processes used, which was later considered while designing the process for this research. This understanding informed the first rough process design for collaborative code inspection which was modified and improved on in a few iterations. Details of the process are presented in Section 4.
- Act: In this step we applied the collaborative code inspection process in practice to observe and study it in action. We carried out two case studies:
 - Case I: In an IT department of a Midwestern US Logistics firm, a piece of C++ code was inspected. The inspector team consisted of 4 experienced professional software developers with sound knowledge of software code inspection. This case study helped to observe the performance of the designed inspection process in an industrial setting.
 - Case II: In a Computer Science department of a Midwestern University in the US, an entire software module in PERL and HTML was inspected. The inspector team consisted of 3 experienced student programmers. The module inspected is a part of a larger application developed by the department and

commercially applied in the field. Although most inspectors lacked industry experience, they were experienced software developers with sufficient knowledge of software code inspection. This case allowed to evaluate the inspection process in an environment with less experienced inspectors.

- **Observe:** In this step we collected and analyzed data to evaluate the collaborative inspection process. We used different instruments, including informal interviews with a selection of the inspectors involved and questionnaires for all participants in the inspection sessions. With the questionnaire we collected both, qualitative and quantitative data regarding the perceived *effectiveness* and *efficiency* of the inspection process and also regarding the participants' *satisfaction* with the inspection process and its outcomes. We also analyzed the actual inspection results on various aspects such as numbers of Major/Minor defects identified and the time taken for the inspection. Given the developmental character of the research and the limited sample size, the analyses were exploratory in nature.
- **Reflect:** In the final step, we analyzed our observations. This pointed out some limitations of the design process and set the stage for future research topics.

4 The Collaborative Inspection Process

This section presents the design of the collaborative inspection process. We first elaborate on the foundation for the process design in terms of the design criteria that were considered and a standard collaborative inspection process that was used as a starting point for our design. Second, we describe the Collaboration Engineering principles and techniques that we used to create our design. Finally, we present our design and discuss its paper-based and GSS-based implementation.

4.1 Foundations

The collaborative inspection process was designed in a few iterations. Each of these iterations was done with an eye on several design criteria. The iterations were necessary to optimize the different criteria as much as possible. The following criteria were considered:

- **Efficiency:** The collaborative inspection process should not require a lot of time to arrive at satisfactory results.
- **Ease of use by inspectors:** The collaborative inspection process should not be complex and should be easily understood by the code inspectors.
- **Comprehensiveness of results:** The process should be able to identify different types of defects as defined before the inspection execution. Such defects may include syntax and semantics errors, logic errors and inefficient ways of coding.
- **Non-redundancy of results:** The process should minimize the redundancy of identified defects as much as possible. Having a group of inspectors inspecting the code, the probability of identifying duplicate defects is considerable.
- **Applicability to varying quality of software code:** Software code can vary in terms of quality from being well-documented superior code to inefficient code with a large number of errors. The collaborative inspection process should be effective for software code of different quality.

We did not design the collaborative inspection process from scratch as researchers had already proposed different processes in the literature. We used the process proposed by Dean et al. [6] from a case study with the Baan Development Applications Department in the Netherlands as our basis. This process is based on existing industry inspection standards and has two phases, each consisting of a number of activities. In the *pre-meeting phase*, the code to be inspected is printed with each page numbered and handed over to the inspectors. The inspectors understand the code and the type of errors they have to identify. Then they identify and document the defects that they find. In the *discussion meeting phase*, the defects identified by the inspectors are collated into a common defect log. Overlapping defects are removed to arrive at a list of unique defects which are classified as major defects, minor defects, or false positives. False positives are defects which the inspectors initially consider as defects but, after discussion, conclude that they are false alarms. During the meeting, the inspectors also discuss any further doubts or comments they have with respect to the code. After the meeting is adjourned, the results should consist of a list of clear, unique, and classified list of defects which is submitted to the author of the code for further review and resolution.

4.2 Collaboration Engineering

We designed the collaborative inspection process following Collaboration Engineering (CE) principles and techniques. CE is an approach to designing, modeling and deploying repeatable collaboration processes for recurring high-value collaborative tasks that are executed by practitioners without the ongoing intervention of professional facilitators [23]. Within CE several design steps are taken. The ones relevant for our study are the decomposition of the process in terms of collaborative activities, their classification in terms of patterns of collaboration, and the selection of appropriate group facilitation techniques, called thinkLets, to guide the execution of each activity [23]. These steps are normally executed iteratively.

The decomposition of the collaborative inspection process in terms of collaborative activities was based on the standard process presented above. Each of the collaborative activities in that process could be classified as aiming to achieve a particular pattern of collaboration. A pattern of collaboration represents observable regularities in the behavior and outcome that emerge over time in teamwork. Within CE, six patterns of collaboration are used [2]:

- *Generate*: To move from having fewer concepts to having more concepts. The goal of generation is for a group to gather or create concepts that have not yet been considered by the group. Brainstorming is an example of a generation process.
- *Reduce*: To move from having many concepts to having a focus on fewer concepts deemed worthy of further attention. The goal of reduction is to decrease a group's cognitive load by limiting the number of concepts they must address.
- *Clarify*: Moving from less to more shared meaning for the concepts under consideration. This is important because people frequently use the same label for different concepts, use different labels for the same concepts, and use labels and concepts that are unfamiliar to others.

- *Organize*: To move from less to more understanding of the relationships among concepts. The goal of organization is to reduce the effort of a follow-on activity. A group might, for example, organize a list of ideas into a number of categories.
- *Evaluate*: To move from less to more understanding of the benefit of concepts toward attaining a goal. The goal of evaluation is to focus a discussion or inform a group's choice based on a judgment of the worth of a set of concepts with respect to a set of task-relevant criteria. For example, an evaluation process may involve a team using a five-point scale to rate a set of alternatives.
- *Build Consensus*: To move from having more disagreement to having less disagreement among stakeholders on proposed courses of action. The goal of consensus building is to let a group of success-critical stakeholders arrive at mutually acceptable commitments.

Table 2. Examples of thinkLets

ThinkLet Name	Pattern of Collaboration	Purpose
Directed-Brainstorm	Generate	To generate, in parallel, a broad, diverse set of highly creative ideas in response to prompts from a moderator and the ideas contributed by team mates.
LeafHopper	Generate	To generate, in parallel, ideas in depth and detail on a focused set of topics.
BroomWagon	Reduce	To eliminate the least important ideas from a large set.
FastHarvest	Reduce & Clarify	To have pairs of team members extract a list of key ideas on assigned topics from a raw set of brainstorming comments.
ChauffeurSort	Organize	To organize ideas into categories through a short group discussion of each idea.
Concentration	Organize	To remove overlap among ideas to create a unique set.
StrawPoll	Evaluate	To evaluate a number of concepts on one or more criteria.
MoodRing	Build Consensus	To continuously track the level of consensus within the group with regard to the issue currently under discussion.

To achieve these patterns of collaboration in a group, a facilitator or moderator may apply facilitation interventions called *thinkLets*. ThinkLets are repeatable, predictable, transferable facilitation techniques to assist a group in reaching its agreed goal [3]. A ThinkLet encapsulates an expert facilitator's best practice for producing a known pattern in the behaviors of a group of people who collaborate. Some thinkLet examples are given in table 2. ThinkLets can be used and re-used as building blocks for team process designs in any domain where collaboration is required [23]. A sequence of thinkLets and the transitions from thinkLet to thinkLet comprise a design for a collaboration process [16]. We used thinkLets for our collaborative inspection process design to make the design more predictable and repeatable in practice.

4.3 The Design

The final design of the collaborative inspection process was the result of three iterations. Early iterations were deemed less desirable because of perceived inefficiencies in the discussion of found defects, the disproportionate amount of time required to complete the process for pieces of code with a large number of defects, and the incomplete recording of all defects. The final design is presented in figure 1.

The process starts with the inspectors individually familiarizing themselves with the code and understanding it. They then are invited to individually identify and record defects. Through the DirectedBrainstorm thinkLet they are stimulated to think about different types of defects in parallel. Compared to traditional, sequential brainstorming where participants speak in turn, the DirectedBrainstorm thinkLet has shown to be more productive and creative [19]. During the brainstorming activity, the participants use the page and line number on the printed version of the code. With each page/line number combination they record a short description of the defect.

The discussion meeting phase starts with a Concentration thinkLet. The inspectors first discuss their findings for each page of code. Only unique defects are reported as the inspectors keep track of duplicate defects while listening to other inspectors' findings. Identified duplicate defects are removed. Thus this activity generates a non-overlapping and a clean list of defects. During the discussion, it may transpire that the inspectors could identify more defects, triggered by each other's contributions. If this is the case, the inspectors perform the DirectedBrainstorm thinkLet again followed by a Concentration to remove the duplicates from the additional defects.

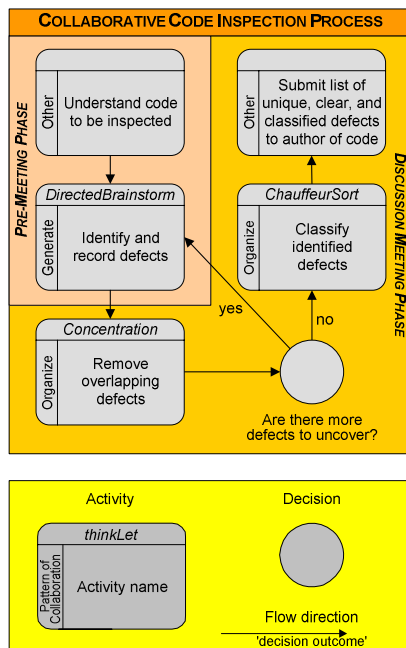


Fig. 1. The design of the collaborative inspection process

The resulting unique list of defects is then classified into the pre-defined defect types (major, minor, false positive) using the *ChauffeurSort* thinkLet. In this activity, the inspectors determine for each defect what type it is. Different opinions are aligned through a brief discussion. The result is a defect list consisting of unique, non overlapping defects which are clearly described and classified into the pre-defined categories. This list is handed to the author of the code for correction.

The collaborative inspection process design was evaluated in two implementations. Each thinkLet specifies which capabilities have to be afforded to the group so that they can follow the thinkLet's rules during its execution [16]. We implemented the process on traditional "paper-based" tools (prepared defect sheets, voting stickers etc.) and on a GSS, GroupSystems Workgroup Edition 3.4. In GroupSystems, each code page was assigned to a separate entry in an outline. Inspectors could enter defect descriptions with associated source code line numbers into their respective page entry. The discussion of the *ChauffeurSort* was focused by having the inspectors vote on their preferred classification for each defect in GroupSystems' Vote module.

Both the paper-based and GSS-based implementations were executed in each case study. The results of the case studies are presented in the following section.

5 Results

This section presents the results from the two case studies. Data was collected and analyzed regarding the process' effectiveness, efficiency, and inspector satisfaction.

5.1 Inspection Effectiveness

The process was evaluated for its effectiveness in uncovering most of the defects. In Case I, the project manager knew the expected number of defects in the code before the inspection was conducted. After the session the total number of defects found was *more* than the expected number. Also table 3 shows that a considerable number of major defects were uncovered in all the sessions. We cannot compare the number of major defects found across sessions as every session inspected a different set of code.

Table 3. Measured inspection effectiveness

	Major	Minor	Duplicate	False-positives	Lines of Code
<i>Paper</i>					
Case I	16	9	15	4	267
Case II	6	49	14	5	592
<i>GSS</i>					
Case I	21	3	2	3	319
Case II	5	23	0	2	137

The data show that the collaborative code inspection process does not favor one type of defect over others. Finding minor defects is essential for a number of reasons. First, the distinction between major and minor helps the code author to prioritize the corrections to make. Second, such a classification is conform international standards

on coding and documentation. This serves programmers other than the code author. The collaborative inspection process also appeared to be effective in identifying false positive and minor defects and in classifying them. This helped in reducing the amount of time spent by the developers on false alarms. Thus the process not only helps to improve the code, but also saves time on correcting false errors.

To measure the inspectors’ perception on the process’ effectiveness, a questionnaire was used on the scale of 1-7, where 7 was the most positive. The reported values in the table 4 are the averages for the questions asked on the effectiveness of the process. Feedback from the questionnaire supported the above findings and indicated that the inspection process was perceived to be effective in uncovering most of the errors.

Table 4. Perceived Effectiveness of the process, mean (standard deviation)

Perceived Effectiveness of the process	Case I	Case II
Paper-based Inspection	4.89 (1.13)	5.83 (1.04)
GSS-based Inspection	4.78 (1.17)	5.39 (1.42)

5.2 Efficiency

The process’ efficiency was evaluated in terms of ease of understanding of the process, flexibility of the process i.e. ability to incorporate last minute changes, and the repeatability of the process. From the sessions executed, it was observed that the inspectors found the inspection process very simple and extremely easy to understand. The moderator’s instructions were considered to be thorough and clear. As all inspectors had a strong software background, it was not surprising they were very comfortable using the GSS tool. In Case I, a single piece of C++ code was inspected whereas in Case II a complete module consisting of different software programs was inspected. The efficiency results for both these sessions are almost similar as seen from the table 5. This illustrates the process’ flexibility.

To measure the inspectors’ perception on the process’ efficiency, a questionnaire was used on the scale of 1-7, where 7 was the most positive (table 6). As can be seen, the inspectors had a positive perception on the process’ efficiency. They also found the process to be repeatable and versatile. Though in this research the inspection process was used only to inspect software codes, the inspectors suggested that it could easily be used to inspect software requirement documents, functional design documents, and other artifacts by defining new defect categories relative to the inspected document.

Table 5. Measured inspection efficiency

	Formal defect log generation	Removing overlaps and refining the defect list	Defect classification	Total
Paper				
Case I	30 min	10 min	5 min	45 min
Case II	47 min	5 min	10 min	62 min
GSS				
Case I	0 min	7 min	5 min	12 min
Case II	0 min	5 min	5 min	10 min

Table 6. Perceived Efficiency of the process, mean (standard deviation)

Perceived Efficiency of the process	Case I	Case II
Paper-based Inspection	5.33 (1.76)	5.47 (1.77)
GSS-based Inspection	5.40 (1.30)	6.07 (1.42)

5.3 Satisfaction

The inspectors' feedback indicated that they were satisfied with the collaborative inspection process and considered it to be very useful. Table 7 (scale of 1-7, 7 most positive) shows that the inspectors gave positive feedback on the inspection process and were satisfied with the process and the outcome of the meeting. They indicated that with this process, it was likely to attain stated inspection goals.

Table 7. Perceived Satisfaction, mean (standard deviation)

Perceived satisfaction	Case I	Case II
Satisfaction with the process		
Paper-based Inspection	5.33 (0.65)	4.75 (1.06)
GSS-based Inspection	5.83 (0.83)	6.25 (0.87)
Satisfaction with the outcome		
Paper-based Inspection	5.25 (0.75)	5.00 (0.85)
GSS-based Inspection	5.50 (0.80)	6.08 (0.67)

5.4 Conclusion

The results show that in both cases the perceived effectiveness was somewhat higher for paper-based inspection than GSS-based inspection. For perceived efficiency and user satisfaction the results are reversed. Overall, the results for the three performance metrics are well above average and hence satisfactory. Thus, the collaborative inspection process appears to perform well in both paper-based and GSS-based formats.

6 Discussion and Conclusions

The collaborative inspection process was successful in uncovering many major, minor as well as false-positive defects in the inspected code. The process was found to be efficient in identifying duplicate defects thereby reducing the code author's valuable time working on the submitted defect list. The inspection process' flexibility was observed as it was successfully applied to inspect both pieces of code or an entire module. Overall the collaborative inspection process was considered to be productive for code inspection and gave a sense of satisfaction with the inspectors involved. In this final section, we first discuss and reflect on the results of our research. We conclude by identifying the limitations and future research directions.

6.1 Discussion

While most research on software code inspections focuses on formal inspection methods and tool support for defect recording and tracing, our study concentrated on the design of collaborative procedures to execute an inspection. We built on the research by Dean et al. [6] and found similar encouraging results. Yet there are a number of noteworthy differences. First, in addition to minor and major defects, we also included a focus on false positives and duplicate defects. In line with Grünbacher et al. [12], we felt this would benefit the quality of the inspection results. Second, our process was executed by a dedicated moderator that was not part of the inspection team itself. In Dean et al.'s [6] work, the moderator was one of the software professionals. This was found to impact the fluency of the inspection process as the moderator had to divide his attention between finding defects and guiding the process.

We focused on the design of the *process* and not directly on the collaborative tool support. The process design was evaluated through a dual implementation in a paper-based and GSS-based environment. In fact, this research's contribution is that it showed that a sound process design can be implemented and executed successfully in different technological environments. Both implementations were found to produce, to a large extent, similar results.

Yet, the GSS-implementation proved to be more useful for a number of reasons. First, consistent with [12], it is apparent that using the GSS-environment saved significant time on process execution. While generating a formal defect log, all identified defects by the inspectors were discussed and collated into a common file. In a paper-based method this was executed as a separate activity that took around 30-50 minutes of the execution time. Moreover, the time spent on this activity is directly proportional to the number of defects identified by the inspectors. Second, the GSS allowed inspectors to work in parallel and view fellow inspectors' lists of defects. This transparency avoided to some extent recording duplicate defects as well as reduced the time required to remove the overlapping defects from the defect log. Third, the voting feature of the GSS supported the categorizing of identified defects. Finally, the GSS offered automatic report generation documenting the entire session data into a formal report. However, even though the paper-based implementation may be more time-consuming, it still yielded satisfactory results. The inspectors pointed out that they were relaxed and at ease while working truly alone in the pre-meeting part of the inspection. This could be attributed to the absence of peer pressure as no inspector was aware of the productivity of their colleagues.

6.2 Limitations

A number of limitations have to be taken into account when interpreting the results of our research. First, the mindset of the participating inspectors may play quite a significant role in relation to the quality of the inspection. Unenthusiastic inspectors may not be willing to give their 100% to the inspection session which will affect the measured and perceived quality of the inspection process. Although we felt we were working with enthusiastic inspectors, we were not able to take this into account. Second, the GSS application used in this research is a general purpose tool; it is not specifically designed for code inspection. Therefore, inspectors may have experienced

sub-optimal functionality concerning the recording of defects. Finally, the sample size of the study is limited. However, we feel that the limited sample size is offset by the high level of realism that we achieved by executing the process in two real situations.

6.3 Future Research Directions

We foresee two directions for future research. First, our collaborative inspection process was implemented in a co-located collaborative environment. It is useful to explore the value of the collaborative inspection process in a distributed environment. Such an exploration could focus on comparing the efficiency, effectiveness, and satisfaction between the co-located and dispersed executions. Second, our research design did not allow for an in-depth comparative analysis of the paper-based vs. GSS-based implementation. Our results only let us conclude that our process design worked well in both implementations. An in-depth comparison between the paper and GSS version of our process would require, for example, an experimental design in which a number of inspection groups all inspect the same piece of code, half of the groups in the paper environment and the other half in the GSS environment.

References

1. Argyris, C., Putnam, R., MacLain Smith, D.: *Action science – Concepts, methods and skills for research and intervention*, San Francisco: Jossey-Bass (1982)
2. Briggs, R.O., Vreede, G.J. de, Dean, D.L.: The Process and Pattern Layer in Collaboration Engineering, *working paper*, University of Nebraska at Omaha (2005)
3. Briggs, R.O., Vreede, G.J. de, Nunamaker, J.F. Jr.: Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems, *Journal of Management Information Systems*, 19(4), (2003) 31-63
4. Budd, A.: The Importance of Process in Web Design. http://www.andybudd.com/archives/2004/01/the_importance_of_process_in_web_design/index.php. Retrieved on 10 November, 2004 (2004)
5. Checkland, P.B.: *Systems thinking, systems practice*, Chichester: Wiley & Sons (1981)
6. Dean, D.L., Rodgers T., Nunamaker, J.: Increasing Inspection Efficiency through Group Support Systems, *Proceedings of the 37th HICSS*, IEEE Computer Society Press (2004)
7. Fagan, M.: Hitting the bull's eye – Improving your quality, schedule, cost, and performance, *Proceedings, 11th International Software Quality Conference*, (2001)
8. Fagan, M.E.: Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems journal*, 15(3), (1976) 182-211
9. Ganssle, J.G.: A guide to Code Inspection, <http://www.ganssle.com/Inspections.pdf>, Retrieved on 02 June 2004 (2001)
10. Genuchten, M.V., Dijk, C. van, Scholten H., Vogel, D.: Using Group Support Systems for Software Inspections, *IEEE Software*, 18(3), (2001) 60-65
11. Gilb T., Graham D.: *Software Inspection*, Addison – Wesley, Wokingham, England (1993)
12. Grünbacher, P., Halling, M., Biffel, S.: An Empirical Study on Groupware Support for Software Inspection Meetings, *Proceedings of the 18th IEEE International Conference on Automated Software Engineering* (2003)
13. Halling, M., Grünbacher, P., Biffel, S.: Groupware Support for Software Requirements Inspection, *Workshop on Inspection in Software Engineering*, (2001) 20-29

14. Harjumaa L., Hedberg, H., Tervonen, I.: A path to virtual software inspection, *Proceedings of Asian-Pacific Conference on Quality Software*, IEEE Computer Society Press, Los Alamitos, CA, (2001) 283-287
15. Klimas, E.: SmallTalk Code Inspection Process, <http://www.lineaengineering.com/Resources/Inspection/inspection.html>, Retrieved on 27 April 2004, (2001)
16. Kolfschoten, G.L., Briggs, R.O., Appelman, J.H., Vreede, G.J. de: ThinkLets as Building Blocks for Collaboration Processes: A Further Conceptualization, *Lecture Notes in Computer Science*, Berlin, Springer Verlag, (2004)
17. Koo, S.R., Son, H.S., Seong, P.H., Yoo, J., Cha, S.D., Son D.S., Choi, S.S.: Toward Easy Inspection and Effective Use of Formal Methods in NPP Software Fields, *Transactions of the American Nuclear Society*, 86(1), (2002) 73 – 74
18. Mashayekhi, V., Drake, J.M., Tsai, W.T., Riedl, J.: Distributed, Collaborative Software Inspection, *IEEE Software*, 10(5), (1993) 66-75
19. Santanen, E., Briggs, R.O., Vreede, G.J. de: Causal Relationships in Creative Problem Solving: The role of active facilitation in EBS, *Journal of Management Information Systems*, 20(4) (2004) 169-200
20. Sify, H.P.: Identifying the Mechanisms to Improve Code Inspection Costs and Benefits, *PhD. Dissertation*, University of Maryland (1996)
21. Software Defects: www.kaner.com/pdfs/defects4.pdf, Retrieved 29 May 2004 (1996)
22. Vitharana, P., Ramamurthy, K.: Computer-Mediated Group Support, Anonymity, and the Software Inspection Process: An Empirical Investigation, *IEEE Transactions on Software Engineering*, 29(2), (2003) 167-180
23. Vreede, G.J. de, Briggs, R.O.: Collaboration Engineering: Designing Repeatable Processes for High-Value Collaborative Tasks, *Proceedings of the 38th Hawaiian International Conference on System Sciences*, Los Alamitos: IEEE Computer Society Press (2005)
24. Wiegers, K.E.: Improving Quality through Software Inspections, *Software Development*, 3(4), (1995) 55-64
25. Zuber-Skerritt, O.: *Action research for change and development*, Gower Publishing (1991)

Handheld-Based Electronic Meeting Support

Gustavo Zurita¹ and Nelson Baloian²

¹ Universidad de Chile, Departamento de Sistemas de Información y Auditoría,
Diagonal Paraguay 257, Santiago de Chile, Chile
gnzurita@facea.uchile.cl

² Universidad de Chile, Departamento de Ciencia de la Computación,
Avenida Blanco Encalada 2120, Santiago de Chile, Chile
nbaloian@dcc.uchile.cl

Abstract. Many studies have reported on the problems that arise when trying to carry out successful meetings. Various authors have developed computerized tools for supporting the different stages of a meeting, but most of these have been conceived for large PCs or Notebooks, which tend to distract the participants from face-to-face interaction. Also, many meetings are organized in a spontaneous manner, sometimes with no access to PCs. In this paper, we propose a meeting support tool for handhelds that overcomes many of the problems inherent in the use of devices with large screens. However, the small size of handheld displays leads to other problems, especially in human-handheld and human-human interactions. The system proposed here is designed using gesture and concept-map principles that enable these problems to be resolved.

1 Introduction

Face-to-face meetings are a frequent activity in any organization [1], and as such their effectiveness and productivity is an important requirement [1], [2]. Various surveys indicate that meetings take up 40% to 50% of management's time. One-half of meeting participants found them to be lacking in productivity, with 25% of the time devoted to irrelevant matters and the total time they take up now twice what it was 20 years ago [3]. Thus, meetings have come to be seen as time-consuming and unproductive [4].

Despite the existence of procedures, rules and mechanisms designed to ensure that meetings are both effective and productive [1], [5], [6], they continue to suffer from various problems (see Section 2) such as no agenda or agenda-setting process, lack of a common workspace for participants, difficulties in the drawing up of minutes, lack of follow-up on commitments, and the absence of voting mechanisms [2], [7], [9].

To solve these problems, technological scaffolding has been developed and tested based on personal computers (PCs). Known as EMS (Electronic Meeting Support), these solutions provide procedures and mechanisms aimed at achieving effective and productive face-to-face meetings [6], [8]. Nevertheless, it has been demonstrated in [10] and [11] that the PC and notebook interfaces and screens used for meeting support capture the attention and cognitive concentration of participants to such an extent that social interaction is reduced. Furthermore, if PCs are employed, meetings must be

held in specific physical spaces [9], [12], making coordination and cohesion more difficult in project scenarios that involve people from various organizations or work teams who need to meet face-to-face in a variety of locations [12]. As pointed out in [11], the ability to bring technological support to the meeting place requires the mobility offered by notebooks and handhelds. According to [10], handhelds are easier to use as a support tool for face-to-face meetings.

In [13], [14] and [15] it is posited that handheld portable computer devices are non-obstructive and create a feeling of belonging to the user, given that they may be employed in various organizational tasks and can be carried permanently on one's person to any place and used at any time. Handhelds are considered to be a good platform for reading brief, concrete content because their interface is simple and insensitive to content formats, thus allowing information to be read quickly, and are also felt to be suitable for providing support to diverse collaborative work groups [16]. However, their reduced screen size and use of virtual keyboards or widgets for entering and handling information introduces new complexities into the person-handheld interaction [17].

In this paper we propose a prototype for a face-to-face meeting support system based exclusively on the use of handhelds wirelessly connected through a peer-to-peer ad-hoc network. This system allows people to meet in any place where the handheld connection is able support the various tasks and processes, both individual and collaborative, that arise over the life-cycle of a meeting. Its design incorporates the following principles: a) Interaction is based exclusively on gestures for managing, organizing and reviewing the notes made by meeting participants. Users are limited to employing a handheld pen and freehand text or graphics, thus minimizing the number of widgets and virtual keyboards; b) Content entered during the meeting is structured, whether it be individual or collaborative notes through three-dimensional concept apps, thereby giving "depth" to the handheld screen. In addition, the system provides the necessary support for group memory, minutes, agenda organization and various commitment and voting processes.

2 Problems of Face-to-Face Meetings

The most common problems of face-to-face meetings as found in [2], [7], [9] and [18] may be characterized in terms of the different meeting stages or life-cycle [18], which consists of an implicit sequence of activities that occur before (pre-meeting), during and after (post-meeting) any actual meeting.

- Pre-meeting: Non-existence of a work agenda or deficiencies in its construction, absence of times assigned for each agenda item. Lack of work methodologies for organizing meeting attendees' contributions, presenting an idea to the other participants, contributing and discussing ideas and recording notes.
- During the meeting: Absence of organization and coordination of attendees' participation due to the lack of an individual or collaborative work area where notes, points of view, ideas and opinions can be shown. Lack of follow-up closely based on the agenda. Discussion of irrelevant matters and information due to absence of agreement mechanisms and the consequent loss of time. Non-existence of records of commitments made by attendees.

- Post-meeting: Inability to carry out a follow-up due to the lack of group memory in the form of a record of notes, activities, tasks, progress and conclusions, resulting in the loss or forgetting of participants' contributions. Deficient or non-existent follow-up of commitments, hindering future follow-up action and between-meeting activities.

In order to ensure that meetings are effective and productive [2], the system should support the following elements: a) construction and follow-up of work agenda, b) organization and coordination of individual and collaborative work, c) negotiation for arriving at agreements, and d) follow-up and management of commitments.

3 Related Work

Various analyses have been carried out of both proposed and already-developed EMS systems that use freehand input, concept maps and especially handhelds, as well as the functionalities offered by handhelds for supporting face-to-face meetings: agenda creation, distribution and discussion support; task and processes development support; distributed on-screen viewing; individual note-making; and generation of minutes (see Table 1).

The Dolphin project [19] uses PCs connected to a LiveBoard to provide support for face-to-face meetings and persons distributed among different physical locations. Dolphin uses concept maps to link up the different issues dealt with at a meeting, so that a given issue can give rise to other sub-issues. Each issue and sub-issue is handled through a shared work area, with the option for attendees to make personal and private notes in the same system.

The We-Met project [20] supports face-to-face meetings using tablet PCs for each of the participants all of whom are interconnected through a PC. Attendees can work in the same virtual work area on their tablet screens, which is shared through the connection with the PC and is freehand input-based. The project's objectives are (a) to facilitate communication between meeting participants, and (b) to facilitate documentation of knowledge and information generated by the meeting for easy review. Users of this system found that it was necessary to have private work areas where they can develop ideas that are not yet ready to be presented to the other attendees.

The Pebbles project [21], though not conceived to be used exclusively for meetings, can be used to provide support to collaborative groups in various contexts. It consists of applications that interconnect handhelds through a PC. The devices are used as though they were PC mice or keyboards. The project's objective is to mediate social interaction techniques between persons through a shared screen.

RoamWare [10] is a handheld architecture that supports informal face-to-face reunions, including those held in such places as corridors. Each handheld can detect and interconnect to others located within a limited space, while the participants make notes on their devices. These notes are sent to a central computer where they are stored for later distribution.

Costa et al. [22] have developed the idea of combining handhelds and a PC to explore the relationships that may exist between a meeting and these technologies. They show that the use of handhelds is neither annoying nor obstructive to the flow of the meeting, and suggest the devices be utilized as tools to generate reports, a traditional

technique for linking meeting processes to organizational ones. The authors of the study also attempt to improve meeting report generation by making use of the capacities handhelds can contribute to the EMS for managing individual and group information.

Table 1. Comparison of face-to-face meeting support systems using handhelds

Characteristics of implemented/proposed EMS	Dolphin	We-Met	Pebbles	Roam-Ware	Costa et al.	Antunes and Costa
Freehand input based	✓	✓	✓	✓		✓
Use of concept maps	✓					
Use of handhelds		Tablet PC	✓	✓	✓	✓
Use of PCs	✓	✓	✓	✓	✓	✓
Wireless network interconnection			✓	✓	✓	✓
Support for creation, distribution and discussion of agenda						✓
Support for development of tasks/processes	✓	✓	✓	✓		✓
Distributed viewing of tasks and processes on screen	✓	✓	✓	✓	✓	
Ability to take individual notes	✓			✓	✓	
Creation of minutes		✓		✓	✓	✓

Antunes & Costa [23] have studied the impact of including handhelds as a support to meetings, pointing out the important role they can play in managing individual information. The authors note the following requirements: a) creation and distribution of an agenda; b) support for the development of the issues on the agenda; c) recording of decisions taken; d) inclusion of the foregoing in the minutes for later distribution; e) support for typical meeting structures; and f) support for various agenda, issue, decision, report and logistics templates.

Table 1 shows the findings of a comparative analysis of the above-described meeting support systems. Particularly noteworthy is that only one system uses concept maps to support collaborative work (Dolphin), while Antunes and Costa are the only ones to propose the creation, distribution and discussion of the agenda. None of the systems provides any support for negotiations aimed at reaching agreements or for commitment follow-up, and most importantly, none use gestures as a solution to the restrictions imposed by the small size of the handheld screen.

4 Design Principles

The system design principles proposed in this paper that constitute a novel contribution compared to other solutions are described below:

- **Handheld screens acquire greater depth through three-dimensional concept maps.** The provision of shared visual spaces may be seen as a facilitator for various processes between persons working in groups because of the support it gives to externalization. This plays an important role in the organization and creation of knowledge in the sense that these spaces support the transition from tacit and indi-

vidual knowledge to explicit knowledge. Shared visual spaces such as concept maps have been applied in discussion groups [24], design groups and collaborative activities. We propose the use of concept mapping techniques for providing support to group design of meeting agendas and meeting development as well as group memory handling. Furthermore, handheld screens can be given greater depth by virtue of the fact that the explosion of each node implies the generation of a new screen on which an aspect specified by the parent node can be worked on, thus resulting in the creation of three-dimensional concept maps. The third dimension affords the option of overcoming the disadvantage of handhelds' reduced screen size by displaying a new screen for the development of additional aspects.

- **Interface simplicity: Gestures.** The design of interfaces for applications that can be built for handhelds pose a challenge due to the small size of the screen. Touch screens are an existing freehand input-based technique for facilitating communication between the user and a handheld, allowing the user to create widgets (buttons for actions such as review, insertion, deletion and change of location). Note, however, that these decrement the amount of useful screen space (see <http://www.palmsource.com/developers>), a single button using up to 10% of the device's screen. Gestures are entered with a pen through predetermined designs, with a result that is efficient, powerful and practical [25], albeit some gestures are not easily remembered and may be difficult to recognize. Generally speaking, the design of a gesture-based interface should incorporate the following three considerations: (a) gestures should be easy to learn and remember, (b) they should be reliably recognizable by the system, and (c) users should be continually informed on the available options. In addition, a zoom feature allows the user to see the structure of the concept maps on the handheld and sounds can be associated as a support to the use of gestures.

5 Prototype Design of HEMS

The proposed prototype, which we will call HEMS (Handheld-Based Electronic Meeting Support), is oriented toward providing support for dealing with the problems identified at the end of Section 2. Figure 1 shows the functionalities that HEMS can support (double-line rectangles) within the meeting life-cycle, the various *gesture* and *concept map* principles (ovals) that support the complete system, and the support components it provides for the *individual work space*, the *group work space*, *voting*, and the assignment and monitoring of *time periods*.

Since handhelds' reduced screen size restricts the amount of information that can be displayed, the design of the interface must be given particular attention [17]. According to [26], one solution for obtaining effective interaction between the user and screen content is to use pen-based *gestures*. A pen-based system facilitates the use of freehand input and is a natural method of making notes during a meeting [20]. Gestures can also support creative processes such as brainstorming, shared visual representations, collaborative publishing of graphic designs, and visual sketch displays [24]. Gestures on the screen will be automatically detected as such and interpreted semantically by HEMS. For easy retrieval and follow-up work, reusable

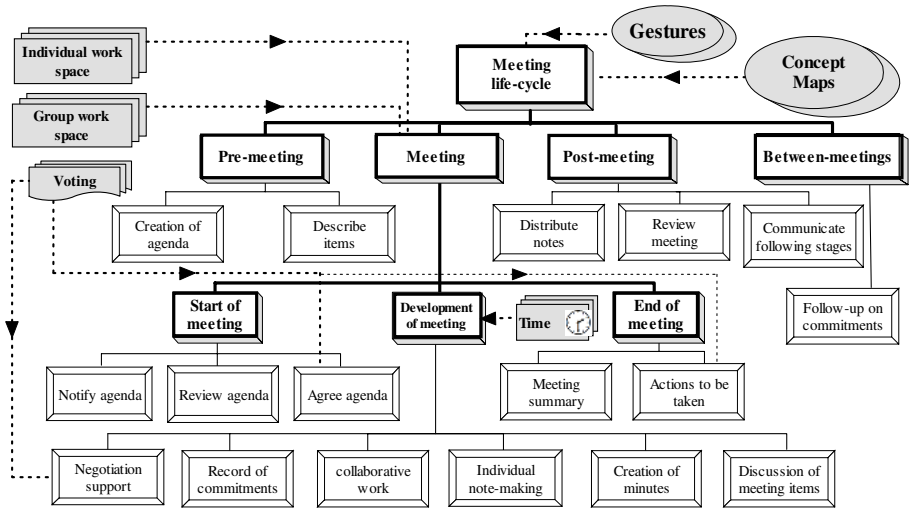


Fig. 1. Support provided by HEMS prototype at the various stages of the system

materials are stored with the semantic structure. Seen from the users' perspective, the final goal would be to reduce all necessary interaction with the Handheld to the moderating gestures and documentary writing on the screen

Features such as agenda creation, note-making, commitment assignment, and support for voting are implemented using *concept maps* that allow a hierarchical nesting of any individual or group issue to be dealt with. Kristoffersen and Ljungberg [27] show that viewing graphic elements and using concept maps on which users arrive at an agreement as to their meaning through explanations help people establish effective social interaction for dealing with any given issue.

The nesting incorporated in concept maps ensures organization, ease of follow-up, and flexibility of creation, modification and management while at the same time avoiding changes in context due to the three-dimensional semantic graphs provided by the maps. As an example, consider Figure 2, in which a person named Ann puts (Figure 2a) forward three issues to be dealt with at a meeting: a new employee, a future project and the budget. Once all the participants (John, Ann, Eva, Tom and Max, as shown at the bottom of the handheld screens) are agreed on the "new employee" issue, one participant (Ann again) selects it using the "select item" gesture and a new blank node appears that is dependant on the issue. If the participants are not in agreement on the "budget", the "delete item" gesture is used. In Figure 2.b, John introduces two sub-issues ("how old" and "knowledge"), both of which are part of the "new employee" concept.

To navigate the concept maps, a chosen issue is double-clicked (for example "new employee", Figure 2.a) and its sub-issues are displayed (in this case, "how old" and "knowledge", Figure 2.b). A double click outside of the selected issues will display the screen shown in Figure 2.a. The gestures "previous" and "next" are used to navigate through a screen or node related to a given concept. The structure of the issues to be dealt with (the concept map) can be seen in Figure 2.c. For the voting process, the

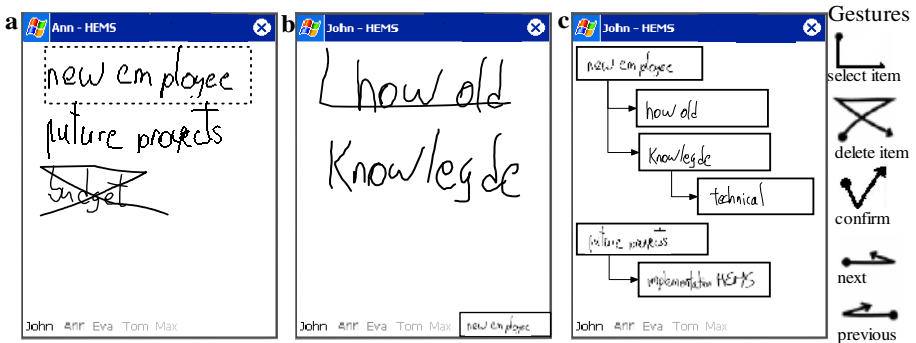


Fig. 2. Screenshots of HEMS and the basic gestures

gestures “confirm” (agree) and “delete item” (disagree) may be used by each participant on a given issue.

The functionalities of the HEMS system (Figure 1) that facilitate the provision of support and mediation for dealing with the phenomena discussed at the end of Section 2 are described in what follows.

- **Agenda construction and follow-up.** These functionalities include the ability to a) facilitate the creation and description of agenda items individually (pre-meeting); b) notify agenda and review it as a group (start of meeting); c) review agenda items (start of meeting); d) agree upon the agenda based on the issues proposed by each participant in shared and collaborative fashion, propose alternative issues to be dealt with, and have a voting component for arriving at agreements (start of meeting); e) provide support for meeting follow-up through the assignment and management of estimated time periods for each issue [6], an important factor for promoting effectiveness and productivity ([2]) by supplying elapsed time alerts and progress indicators on matters being discussed at the meeting.
- **Organization and coordination.** The use of concept maps generates a natural mental structure that ensures the participants remain focused on the issues to be dealt with [6]. HEMS supplies a work area for each issue that is to be developed individually or collaboratively [20]. It can be used by attendees to make handwritten notes.

In the final stage of the meeting, the deep organization of the concept maps a) enables the drafting of a meeting summary through the follow-up of the structure, and b) facilitates the determination of actions to be taken. Additionally, in the post-meeting stage this feature makes it possible to a) distribute each participant’s notes as well as those made by the group as a whole, b) review notes and commitments at a later time, and c) inform those involved regarding the stages to follow.

Given that the attendees’ notes were made during the processes of agreeing upon an agenda, developing the issues discussed, taking votes, etc., the **minutes** of the meeting will be saved by individual and group contribution for each attendee as well as by issue dealt with, including a record of the times associated with each issue. In this format, the minutes constitute a memory of the meeting so that the user

may consult them for information at any moment and maintain the links with the corresponding issues discussed at other meetings.

- **Negotiation.** HEMS includes a voting tool to support negotiations and discussions, allowing attendees to agree upon the issues to be placed on the agenda, those that are to be dealt with at the actual meeting and/or the actions to be taken (see voting component in Figure 1). The voting system may use any mode of agreement (unanimity, simple majority, two-thirds majority, etc.). Because it works through pen-based gestures, the system provides the necessary flexibility for adapting to various “mental scenarios” that may arise.
- **Commitments.** If an attendee must carry out a particular activity at some later time, a note is made in the handhelds stating that the activity must be executed by a certain deadline and by the person associated with the note. The system also ensures the necessary functionality for sharing this information, thus allowing the commitment to be tracked by all participants.

In shared mode, the individual annotations of a given participant can be viewed by all in a single work area, and each of them may specify whether or not their notes are to be private. Commitments and minutes can be accessed by various criteria such as concept maps and issues contained in the agenda, time elapsed before dealing with a given issue, or an issue’s position relative to a given participant’s note [20].

HEMS is entirely based on a peer-to-peer ad-hoc wireless network. Note that meetings typically last 2 to 3 hours, which is less than the useful charge life of currently used handheld batteries. Finally, the amount of information needed to be stored is relatively small, so that the limits imposed by handhelds’ reduced memory size do not constitute a problem.

6 Conclusions

The use of handhelds would appear to be an interesting option for coordinating meetings that can be held at any time and in any place due to the devices’ ability to make notes and share small items of information, their ease of deployment in any collaboration scenario and their ad-hoc communication support. Handhelds are also a good choice in that they allow brief, spontaneous notes to be expanded later into fuller contributions. In cases where large amounts of data must be inputted, solutions involving keyboards or other high volume input devices are required and handhelds would be less applicable.

However, even in situations where handhelds are appropriate, their reduced screen size constitutes a challenge when designing human-handheld and handheld-mediated human-human interactions. The system proposed in this paper, founded on two principles aimed at improving these two classes of interactions, implements functionalities that help overcome what are recognized in the literature as the most frequent problems with meetings. The first principle is the use of an interface that is based wherever possible on an interaction with gestures so that widgets occupying scarce screen space are not needed. The second principle is the application of a simple structure to the notes made by meeting participants. This simplifies the communication of ideas, and thanks to the tridimensionality of the structure when expanded, each node iteratively increases the depth of the screen. The structure must be kept simple to

ensure it can be easily retained by the mind, a condition that is fulfilled by a hierarchical structure. In view of the foregoing, we believe that the tool presented here can be an effective support for spontaneous face-to-face meetings, a hypothesis we hope to confirm in experiments planned for the near future.

Acknowledgments. This paper was partially funded by Fondecyt 1050601 and DI - Universidad de Chile Nro. I2 04/01-2. Special thanks to Lorena Quezada and Javier Martinez.

References

1. Allen, T.: Organizational Structure for Product Development. Sloan School of Management, MIT: Cambridge (2000) 1-24
2. Tropman, J.E.: Effective meetings: Improving Group Decision Making. Sage Publications (1996)
3. Matson, E.: The Seven Signs of Deadly Meetings. Fast Company. 2. April/May (1996) 122
4. Mosvick, R., Nelson, R.: We've Got to Start Meeting Like This! A Guide to Successful Business Meeting Management. Glenview, IL: Scott, Foresman, (1992)
5. Cranes, W. T.: Effective Meetings for Busy People: Let's Decide It and Go Home. New York: McGraw-Hill Inc. (1980)
6. Hayne, S.: The facilitators' perspective on meetings and implications for group support System. Database, 30(4) (1999) 72-91.
7. Drew, J.: The 3M Meeting management team. Mac. Graw Hill. (1994)
8. Jessup, L., Valacich, J.: Group Support Systems: A New Frontier. New York: MacMillan (1993)
9. Nunamaker, J.A., Dennis, A., Valacich J., Vogel, D., George, J.: Electronic Meeting Systems to support group work. CACM 34(7) (1991) 40-61
10. Wiberg, M.: RoamWare: an integrated architecture for seamless interaction in between mobile meetings. Conference on Supporting Group Work archive. Proceedings of the 2001 International ACM SIGGROUP (2001) 288-297
11. Bergqvist, J., Dahlberg, P., Kristoffersen, S., Ljungberg, F.: Moving out of the meeting room: exploring support for mobile meetings. Proceedings of the Sixth European conference on Computer supported cooperative work. Copenhagen, Denmark (1999) 81-98
12. Hutchins, E.: The Technology of Team Navigation. In Galegher, J., Kraut, R.E., Egidio, C. (eds.): Intellectual Teamwork: Social and Technological Foundations of Cooperative Work. Lawrence Erlbaum Associates, Hillsdale, NJ. (1990) 191-221
13. Marshall, C., Ruotolo, C.: Reading-in-the-Small: A Study of Reading on Small Form Factor Devices. JCDL'02, Portland, Oregon, USA. (2002) 13-17
14. Luff, P., Heath, C.: Mobility in Collaboration. In Proceedings of Computer Supported Collaborative Work, CSCW'98. ACM Press (1998) 305-314
15. Perry, M. O'hara, K., Sellen, A., Brown, B., Harper, R.: Dealing with mobility: understanding access anytime, anywhere. ACM Transactions on Computer-Human Interaction (TOCHI), 4(8) (2001) 323-347
16. Schmidt, A. Lauff, M., Beigl, M.: Handheld CSCW. Workshop on Handheld CSCW at CSCW '98, 14 November, Seattle (1998)
17. Guerrero, L., Pino, J., Collazos, C., Inostroza, A., Ochoa, S. Mobile Support for Collaborative Work. Proceedings of 10th International Workshop on Groupware, CRIWG 2004, LNCS 3198, Springer Verlag, San Carlos, Costa Rica, September (2004) 363-375

18. Bostrom, R., Anson, R., Clawson, V.: Group Facilitation and Group Support Systems. In Group Support Systems: A New Frontier. Jessup, L. Valacich, J., (Eds.) (1993) 146-168
19. Streitz, N., Geißler, J., Haake, J., Hol, J.: DOLPHIN: Integrated Meeting Support across Local and Remote Desktop Environments and LiveBoards in Proceedings of the 1994 ACM conference on Computer supported cooperative work, CSCW'94, ACM (1994) 345-358
20. Wolf, C., Rhyne, J.: Communication and Information Retrieval with a Pen-Based Meeting Support tool. CSCW Proceedings. ACM (1992) 322-329
21. Myers, B.A., Stiel, H., Gargiulo, R.: Collaboration using multiple PDAs connected to a PC. Proceedings of the ACM, Conference on Computer Supported Cooperative Work. (Seattle, WA) (1998) 285-294
22. Costa, C., Antunes, P., Dias, J.: The Meeting Report Process: Bridging EMS with PDA. Third International Conference on Enterprise Information Systems, ICEIS 2001. Setubal, Portugal. (2001) 821-826
23. Antunes, P., Acosta, C.: Handheld CSCW in the Meeting Environment., CRIWG 2002, LNCS 2440 (2002) 47-60
24. Hoppe, U., Gaßner K.: Integrating Collaborative Concept Mapping Tools with Group Memory and Retrieval Functions. Proceedings of the Computer Support for Collaborative Learning (CSCL) 2002 Conference (2002) 716-725
25. Long, A., Landay J., Rowe L., Michiels, J.: Visual similarity of pen Gestures. In: Proc. of the Human Factors in Computing Systems SIGCHI 2000 2(1) (2000) 360-367
26. Nicholson, M., Vickers, P.: Pen-Based Gestures: An Approach to Reducing Screen Clutter in Mobile Computing. In Proceedings of Mobile HCI 2004 (2004) 320-324
27. Kristoffersen, S., Ljungberg, F.: An Empirical Study of How People Establish Interaction: Implications for CSCW Sessions Management Models. Proceedings CHI 99 Conference on Human Factors in Computing Systems Pittsburg (1999) 1-8

Sharing Information Resources in Mobile Ad-hoc Networks

Andrés Neyem, Sergio F. Ochoa, José A. Pino,
and Luis A. Guerrero

Department of Computer Science, Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
{aneyem, sochoa, jpino, luguerre}@dcc.uchile.cl

Abstract. Many people are sharing digital resources through networks in order to facilitate, enhance or improve collaborative work. Information sharing is not only important to support collaborative work but it also represents the basis for design and implementation of solutions for typical design aspects of groupware applications, such as: floor control, group memory, shared objects replication and sessions and users management. Advances in mobile technology have extended the sharing information scenarios to Mobile Ad-hoc Networks (MANETs), which has brought new challenges. This paper presents a simple service platform to share information resources among members of a MANET-supported groupware session. People interact using notebooks and PDAs. In addition, a shared presentation tool which has been developed using the services of the platform is described. This presentation tool can be used to assist other collaborative activities, such as: technical presentations, casual interactions, meetings for decision making and software technical reviews.

1 Introduction

Gartner's report estimated that PDA revenues in 2004 reached a record \$4.3 billion for a 16.7 percent increase, compared to 2003 figures [2]. IDC estimate that 13 million handheld devices are sold each year and the estimation for 2005 is 71 million units to be sold [8]. Advances in mobile technology and the price reduction of computing mobile devices have prompted the spread of this technology to many scenarios, such as: schools, hospitals, police, government and business. However, handheld machines are not massively used to assist group work yet; their main use still is to support personal activities.

These devices incorporate communication capabilities - usually based on Wi-Fi - which allow them to interact with each other using wireless (one hop) and mobile (multihop) networks [17]. Therefore, any physical scenario providing these communication services to people on the move becomes a potential collaboration arena. Examples of these scenarios are: shopping malls, offices, universities, hotels and airports. Software reviews, brainstorming sessions, shared presentations and synchronous learning activities are some of the collaboration activities that could be supported using these devices. However, supporting these collaboration activities in

mobile networks, also named MANETs (Mobile Ad-hoc NETWORKS) [17], involves finding MANET-based data sharing solutions.

This paper presents a service platform that allows collaborators be grouped in ad-hoc sessions and share information resources on MANETs, by using notebooks and PDAs. That platform can also be considered as a basis to develop solutions to support groupware design aspects, such as floor control, group memory, shared objects replication and sessions and users management. Solutions found to support these design aspects will depend on the MANETs information sharing strategy.

Next section describes the challenges to share information on MANETs. Section 3 presents related research work. Section 4 describes the service Platform for Ad-hoc Sharing Information Resources (PASIR), a presentation tool which has been developed using the services of the platform, and a discussion on PASIR strengths and weaknesses. Finally, Section 5 presents the conclusions and future work.

2 Sharing Information in MANETs

For many years the CSCW and CSCL communities have used shared information as a way to support or enhance collaboration among people [14, 15]. Shared information has also being used to develop software solutions supporting design aspects of groupware applications [4, 13]. The most common strategy to share information among collaborators involves centralizing data and services. Many groupware platforms were designed following this strategy [12], and they show good results in distributed systems supported by stable wired and wireless networks. However, this strategy is useless when collaborators are communicated through an unstable network like a MANET [1, 5]. The network structure becomes highly dynamic since collaborators move continuously, and each centralized resource represents a failure point for collaborative solutions in term of ensuring the communication availability. Hence, low availability of the shared data space jeopardizes the collaboration process. Although sharing information on mobile systems is not a new challenge, most of the proposals do not consider the use of handheld devices, such as PDAs, and unstable communication services, which is a particular feature of MANETs. These particularities bring new challenges for sharing information for collaboration.

PDAs typically constrain groupware applications mainly in terms of screen size, processing power, memory capacity and networking services provided by the operating systems. These services allow notebooks and PDAs be integrated in the same workgroup scenario [5]. On the other hand, the signal instability and the tight bandwidth in MANETs represent the main restrictions for the groupware communication services design. Collaborative systems using such communication services should exhibit high shared data availability to avoid jeopardizing the collaborative process. These restraints also show the need for new solutions able to keep high availability of shared information even in that unstable scenario.

3 Related Work

There are several research initiatives that are trying to provide good solutions to support sharing information in peer-to-peer networks. Some of these related works are

tuple-based distributed systems derived from LINDA [3], such as: FT-LINDA, JINI, PLinda, T-spaces, Lime and JavaSpaces [6, 11]. Although these implementations allow sharing information in peer-to-peer networks, they use centralized components to provide binding among components of the distributed system. Such centralized components become critical failure points in unstable networks.

Another related project is the iClouds framework which offers spontaneous mobile user interaction and file exchange in mobile ad-hoc networks [7]. This framework does not require centralized components because it does a full replication of any shared file, which is appropriate in MANET scenarios. However, it does not provide support to exchange shared objects, just files. In addition, iClouds does not distinguish among copies of a same shared file (e.g. master and slave copies) and does not support distributed operations on those files either. Similarly, the Proem platform provides support for shared files, but on Personal Area Networks [9].

Another interesting platform is XMIDDLE. It allows mobile hosts to share XML documents across heterogeneous mobile hosts, permitting on-line and off-line access to data [10]. However, these capabilities do not allow manipulating compound documents (like MS-Office documents or Adobe Acrobat documents), which are used by many people to support the collaboration activities.

Next section presents a software platform named PASIR (Platform for Ad-hoc Sharing Information Resources), which was designed to share files and compound documents using the functionality provided by the .NET framework. It allows users to share information resources through a distributed data space.

4 PASIR

The proposed platform is implemented using C# programming language and reuses the services provided by the .Net framework for object and file manipulation, and also for networking. There are no centralized services or data in the platform. Every component of PASIR is fully replicated in order to keep high availability of resources even when a session member gets isolated. Although a groupware system supported by PASIR is composed of three layers (Fig. 1), the proposed platform involves the two lower layers.

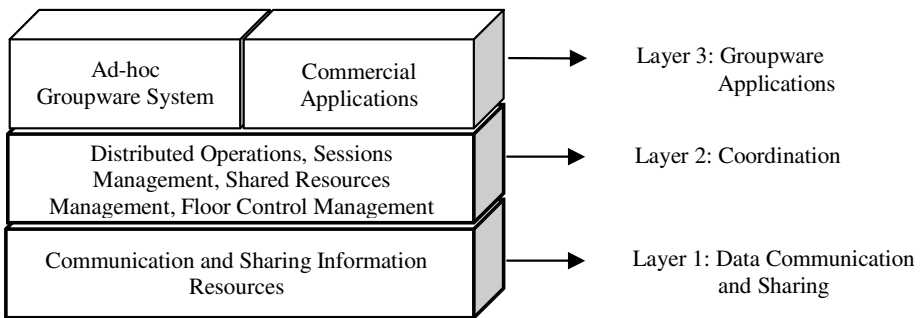


Fig. 1. Architecture of a PASIR-supported Groupware System

Each layer carries out a specific function and it communicates with the adjacent layers through a well-defined interface. The lower layer is in charge of providing all the networking and data sharing services among the groupware applications. The coordination layer uses the services of the lower layer to implement the typical groupware services for collaborative applications. It coordinates distributed operations and it generates a consistent vision of the group activities. The services currently implemented in this layer correspond to the services required by the prototype application used to test the platform. They include session and user management, floor control and shared objects synchronization. Nevertheless, many other services for groupware applications can be included in this layer.

Finally, services for data sharing provided by PASIR can be embedded in ad-hoc collaborative applications and also in some commercial software products, such as Microsoft Office, Adobe Acrobat and Photoshop. This particular functionality allows some monolithic applications support collaborative activities. Thus, it is possible to reuse all data-manipulation applications functionality if the data is based on COM (Component Object Model) objects [16]. These applications represent the upper level of the architecture. The current implementation of PASIR supports just on-demand shared objects synchronization through services provided by the coordination layer.

4.1 Data Communication and Sharing Layer

The PASIR communication services are asynchronous and based on UDP over IP. UDP does not guarantee packet delivery as TCP; however, it is suitable for mobile environments because it is connectionless. In addition, it allows using IP multicast to detect the presence of reachable hosts. Communication services provided by PASIR can be used on any network platform able to use UDP over IP, such as Bluetooth, IrDA and IEEE 802.11x. Consequently, PASIR can be used in stable and unstable communication environments. These communication services are the same provided by the .Net framework for both notebooks and PDAs. Yet, the semantics of the messages was specifically designed to allow information sharing on MANETs.

People are able to share two types of resources using the communication services: flat files and COM objects. Every shared resource has an XML descriptor specifying its features and indicating whether the resource is a master or slave copy. These shared resources are located in a folder that every user has for each opened session. Some shared resources could be a collection of shared interrelated objects (e.g, a MSWord document or a PowerPoint presentation can be considered a collection of shared linked COM objects). Besides, the comments a user can include in that collection are also shared objects that are part of the collection. These shared objects can travel together or they can be filtered to decouple them for synchronization purposes. The synchronization services are provided by the coordination layer.

4.2 Coordination Layer

The coordination layer is based on a fully replicated session manager. It locally records information about users, sessions and shared resources (Fig. 2), and allows users to interact with the shared objects through a visual interface (Fig. 3a and 3b). Shared objects are grouped in sessions. Every session also groups users sharing

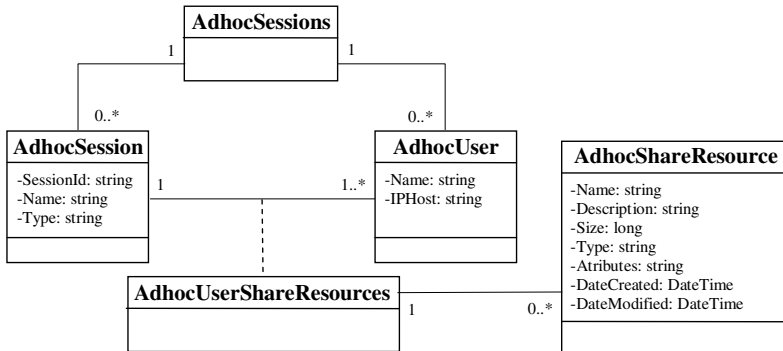


Fig. 2. Structure of the PASIR Session Manager

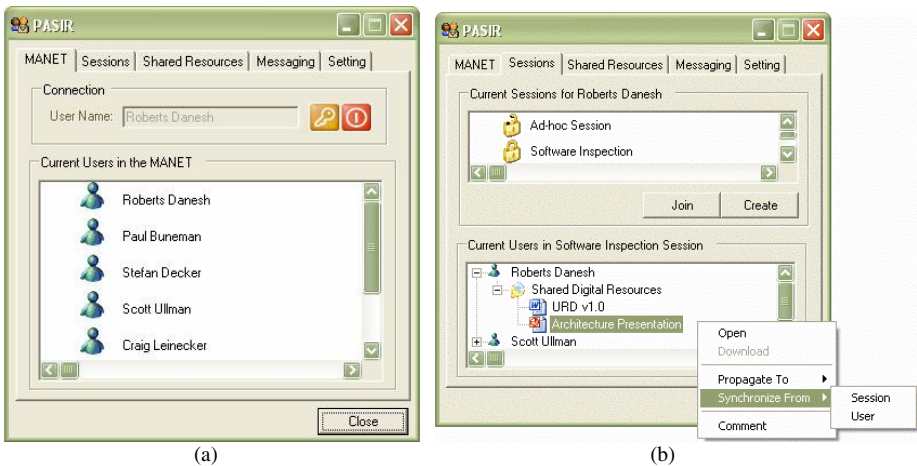


Fig. 3. User Interface of the PASIR Session Manager

information resources with the remaining session members. Every user in the MANET can be part of more than one session. Users showing interest to connect to the MANET are registered by the *AdhocSessions* class (Fig. 2) and they become available in the shared environment (Fig. 3a).

Once inside the environment, a user (*AdhocUser* class) can access a session in two ways (Fig. 3b): (a) creating a session, in that case she is automatically inside; or (b) requesting access to an existing session (*AdhocSession* class). When a user creates a session the platform gives her a *SessionId* which is not visible to the rest of the MANET members. The *SessionId* should be sent to the users that are invited to the session. The invitation and the *SessionId* will be delivered using multicast.

Once a user has access to a session, a local shared folder of the session (*AdhocShareResource* class) becomes visible to the rest of the session members (Fig. 3b). Thus, that user can synchronize her shared resources with the resources of a specific partner or with the rest of the session members. The attributes of each shared object are analyzed and compared in order to carry out the synchronization process.

However, the synchronization of files that are not composed of COM objects (e.g., plain text documents) is done using an XML descriptor of the files as support.

Remote shared resources can be downloaded or remotely accessed using the local session manager. When a user leaves a session, the local shared resources are kept available for the local user who can work asynchronously. The COM objects included by the user in the local copy (e.g., comments on a MSWord document) can be transferred to the master copy next time the user with the master copy and the user with the comments are reconnected to the working session. Every working session is potentially alive even if no users are currently connected and it gets available when the first user gets in. A user can leave a working session for good indicating that decision to the local session manager. A session is potentially alive while a registered user exists (even if he is not connected).

4.3 Groupware Applications Layer

Several groupware applications can be developed using PASIR services. Even some commercial products are able to interact with the platform using the services. As an example, this section briefly describes the design of SPT (Shared Presentation Tool). This tool makes shared PowerPoint documents accessible to several users in a session. Such users are also able to link comments to this shared document. This process can occur both when the document is being edited and when the presentation is being delivered. Author and presenter use the regular MS PowerPoint product, whereas reviewers connected to the same session use the commenter module to make comments. Every comment is linked to a slide. Comments made by reviewers are linked on-demand into the master copy of the PowerPoint document, that is the one used by the presenter. The synchronization process can be done following four strategies: one sender and one receiver, one sender and many receivers, many senders and one receiver, and many senders and many receivers (whole synchronization). Every comment can be a COM object or an attribute specified in the XML descriptor depending on the .Net framework version being used (full or compact version). These comments can be anonymous or not. Moreover, the same strategy can be used to make comments or corrections to a paragraph of a shared MS Word document. This tool can be useful to support collaborative activities such as software technical reviews, paper presentations or meetings for decision making. This implementation has the same user interface for notebooks and PDAs.

4.4 Discussion

The PASIR platform is easy to deploy in notebooks and PDAs communicated through a MANET. It allows sharing the information resources and keeps a high availability of both shared information resources and services. That functionality can be considered a basis to develop solutions to support the groupware design aspects and also collaboration among people. The PASIR current implementation supports distributed asynchronous work because of the services currently available in the coordination layer. Nevertheless, prototypes of synchronous coordination services are under construction.

The relationship between PASIR and commercial frameworks and applications allow developers reuse functionalities from these commercial products to support collaborative activities or to create new groupware applications. Although this reuse constrains groupware systems, usually it also represents a reduction in development effort and an improvement on the product quality.

The main limitation of this proposal refers to the fact these advantages can be obtained only by using the MS Windows family of operating systems. Furthermore, the integration of PDAs and notebooks and the synchronization mechanisms also depend on it, because most of them are provided by the .Net framework.

The proposed platform can also be used in stable (wireless or wired) communication settings, using several computing devices and desktop PCs. The platform functionality in that case is the same. However, the stability of the communication services allows including the server in the network to ease and to improve the efficiency of the information sharing process.

5 Conclusions and Future Work

A platform easing information resource sharing in MANETs using notebooks and PDAs has been presented. The platform functionality can be considered a basis to develop solutions to support the groupware design aspects and also collaboration among people. Unlike other initiatives, the proposed platform takes advantage of the relationship between .Net and COM frameworks [16] and well-known commercial products in order to provide a scenario to assist collaborative activities.

The main advantage of PASIR is the relationship it has with commercial frameworks and applications. It allows developers to reuse functionality available in these frameworks and also in the applications. However, these advantages are only available for the MS Windows family. It restricts the portability of the groupware systems and programming languages that can be used to develop, extend or integrate these systems. Although PASIR was designed for MANETs, it is also possible to use it in stable communication settings including desktop PCs.

In order to complete the PASIR support for synchronous and asynchronous information sharing, the authors are currently working on the implementation of new coordination services that will allow synchronous sharing of the COM objects and files. Future work includes embedding awareness components in monolithic commercial applications, using add-in capabilities to improve the support for the collaborative work. Moreover, the groupware services provided by the coordination layer will be increased and improved. Finally, experimentation in real scenarios should be carried out to evaluate the proposal and get feedback to improve it.

Acknowledgments

This work was partially supported by Fondecyt (Chile), grants N°: 1030959 and 1040952 and by MECESUP (Chile) Project N°: UCH0109.

References

1. Aldunate, R., Ochoa, S., Peña-Mora, F., Nussbaum, M. Robust Mobile Ad-hoc Space for Collaboration to Support Disaster Relief Efforts Involving Critical Physical Infrastructure. *ASCE Journal of Computing in Civil Engineering*. In press.
2. Gartner, Inc. Gartner Says Worldwide PDA Shipments Grew 7 Percent While Revenue Increased 17 Percent in 2004. (2005). URL: www.gartner.com/press_releases/asset_120374_11.html
3. Gelernter, D.: Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems* 7(1), (1985), 80-112.
4. Guerrero, L., Fuller, D. A Pattern System for the Development of Collaborative Applications. *Information and Software Technology* 43(7), (2001), 457-467.
5. Guerrero, L., Ochoa, S., Pino, J., Collazos, C. Favorable Cases for the Use of PDAs in Collaborative Work. Accepted for special issue of Group Decision and Negotiation (2005).
6. Handorean, R., Payton, J., Julien, C., Roman, G. Coordination Middleware Supporting Rapid Deployment of Ad Hoc Mobile Systems. *Proc. MCM'03, USA*, (2003), 363-368.
7. Heinemann, A., Kangasharju, J., Lyardet, F., Mühlhäuser, M. iClouds - Peer-to-Peer Information Sharing in Mobile Environments. *Lecture Notes in Computer Science* 2790 (2003), 1038-1045.
8. IDC. IDC Remains Optimistic About Handheld Devices, Forecasts 71 Million Shipments by 2005. (20 June 2001). URL: <http://www.idc.com>
9. Kortuem, G. Schneider, J., Preuitt, D., Thompson, T., Fickas, S., Segall, Z. When Peer-to-Peer Comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad-hoc Networks. *Proc. P2P'01, Sweden*, (2001), 75-93.
10. Mascolo, C., Capra, L., Zachariadis, S., Emmerich, W. XMIDDLE: A Data-Sharing Middleware for Mobile Computing. *Journal on Personal and Wireless Communications* 21(1), (2002), 77-103.
11. Nemlekar, M.: Scalable Distributed Tuplespaces. MSc. Thesis. Department of Electrical and Computer Engineering, North Carolina State University, Chapter 5. 2001.
12. Ochoa, S. Guerrero, L. Pino, J., Collazos, C. Reusing Groupware Components. *Lecture Notes in Computer Science* 3198 (2004), 262-270.
13. Schuckmann, C., Schiimmer, J., Seitz, P. Modeling Collaboration using Shared Objects. *Proc. GROUP'99, ACM Press, USA*, (1999), 189-198.
14. Siirtola, H., Heimonen, T. Scalable Support for Work Groups and Groupwork. *Proc. MobileHCI'01, Dunlop and Brewster (Eds.)*, France, (2001), 129-134.
15. Talja, S., Hansen, P. Information Sharing. In: *New Directions in Human Information Behavior*. Ed. A. Spink & C. Cole. Dordrecht: Kluwer. (2005).
16. Templeman, J., Mueller, J. *COM Programming with Microsoft .Net*. Microsoft Press, Redmond, Washington. (2003).
17. Tschudin, C., Lundgren, H., Nordström, E. Embedding MANETs in the Real World. *Proc. PWC'03, Italy*, (2003), 578-589.

Towards a Model of Cooperation

Adriana S. Vivacqua^{1,3}, Jean-Paul Barthès³, and Jano Moreira de Souza^{1,2}

¹ COPPE/UFRJ, Graduate School of Computer Science

² DCC/IM, Institute of Mathematics, Federal University of Rio de Janeiro,
Rio de Janeiro, Brazil

³ UTC, Université Technologie de Compiègne, Compiègne, France
avivacqua@cos.ufrj.br, barthes@utc.fr, jano@cos.ufrj.br

Abstract. Researchers from several different domains have conducted studies about cooperation, and a wealth of different models and theories have been generated as a result. In this paper we describe an initial version of an integrative model for the initiation of cooperation, with the theoretical background from which it was created. Our main goals were to gain a better understanding, organize and structure the most important aspects and recurrent themes that show up in cooperative behavior research, adapting them when necessary. We are especially interested in the initiation of cooperation, and in the determination of factors that lead to the establishment of cooperative endeavors with the final goal of understanding what affects and how to encourage cooperation. A model such as this could be applied to groupware tools to increase the levels of cooperation between users.

1 Introduction

Rapidly evolving environments and frequent political, economical and technological changes have led to changes in working environments [5]. Work now often involves experts from different domains and is often remote, as evidence by an increase in adoption of virtual work teams [10]. The constitution of working groups allows individuals to pool their expertise, enabling the group to jointly tackle problems.

Having others with whom to discuss alternatives and provide feedback or insight into the problem often proves valuable, as new ideas and possibilities spring from these interactions. Complex problems greatly benefit from diverse opinions and points of view [18] [19]. Groupware systems seek to support these groups of individuals in their joint activities, facilitating knowledge and information exchange and archival, competence management and awareness of collaborators [5] [22].

Cooperation, however, has its costs: group work usually entails secondary activities to ensure appropriate communication and information exchange, which involve coordinating, controlling and mediating relationships and articulating interdependent activities [22]. Team distribution can make this articulation process harder and eventually decrease levels of cooperation [14].

During our studies of computer support for spontaneous interactions, we felt the need to better understand the process of establishing cooperation and what factors

contributed to or detracted from it. Researchers from such different domains as management, social psychology, economics and computer science have studied influences on cooperative behavior, and we have attempted to integrate the most important concepts and theories found in literature into a single model [4]. We are concerned with the engagement phase, that is, the moment when actors see the opportunity for interaction and decide to act on it.

With this model, we attempt to uncover the decision-making strategy employed by individuals when deciding whether to initiate cooperation. The point is to tease out from existing theories the elements that might influence the decision to (a) initiate an interaction with a cooperative intent or to (b) respond to this interaction, engaging in cooperation. We integrate these elements into a model that maps the influence of the elements on the final decision (whether positive or negative). To that end, we have reviewed a number of theories and studies, and tried to integrate them, taking the most frequently mentioned elements and linking them together.

The purpose of such a model is twofold: (1) to provide a better understanding of individual reasoning and how it affects group work and, (2) when used in conjunction with groupware or recommendation systems, to enable better matchmaking and foster cooperation. With this model, we expect to be able to construct systems to not only support but also foster cooperation, to help distributed users establish productive cooperative relationships. Our final goal is to instrument applications in such a way as to induce and enable cooperation, facilitating group formation.

The remainder of this paper is organized as follows: the next section provides a brief overview of the background literature reviewed. In Section 3, we present the model and its main elements. Section 4 finalizes with a discussion of how this model can be applied and directions for future work.

2 Theoretical Sources

We started our search for a feasible model by reviewing a number of sources that deal with influences on cooperation or participation in groups. Much literature is based on game theoretic analysis, which plays a central role in the work of economists, sociologists and psychologists.

Game theory is the formal study of conflict and cooperation and its concepts apply whenever the actions of a group of agents are interdependent [24]. Game theorists study cooperative behavior through social dilemmas, or situations where individually reasonable behavior leads to a less than optimal situation for the group [12]. Researchers use social dilemmas to study cooperation between members of a group, strategy adoption and the effects of changes in certain elements of the games (such as repetition or identifiability) on strategy adoption [1]. One of the shortcomings of game theory is that it seeks to rationally explain actions that are sometimes performed irrationally. Thus, game players are poor approximations of real people, who have limited rationality and perception, and are influenced by habit, instinct and custom [23].

It is unlikely that a “game person” will be a perfect match to a real person. To try to address this problem, we also reviewed studies from psychology and sociology, which could provide further insight into aspects that hold for real people. Experimentation-based social psychological studies provide models such as VIST (Valence,

Instrumentality, Self Efficacy, Trust), that maps factors that influence participation in virtual work teams [10] and the Collective Effort Model, that maps reasons for social loafing (non-cooperation) [3] [13].

Research has also acknowledged that different types of tasks lead to different participatory behavior in groups [13] [17] (for instance, in additive tasks, such as tug of war, people tend to exert less effort when they are working in a group), but task and resource related elements (e.g. time, availability) were left out of for the time being. At this stage, we are focusing on factors relating to the individuals, as these play an important part when engaging in cooperation. These are also frequently mentioned as influencing recommendation systems, which is part of our research focus.

It is important to note that most of the theories presented do not deal exactly with the initiation of cooperation, but with participation in work teams, occurrence of social loafing in groups or volunteerism such as found in open source projects. As we are interested in factors that lead to the establishment of cooperative relationships, some of the concepts and ideas have been adapted to that effect.

3 An Integrative Model of Cooperation

Intentional cooperation usually implies overlap, which provides reasons to interact or cooperate. These commonalities may involve the problem itself, tasks, objects, interests or knowledge. Different levels of overlap may exist, such as one person performing a task that helps towards solving another person's problem or possessing knowledge that someone else needs. These similarities provide the informational context within which cooperation is to take place.

Our object of study is the engagement phase (the phase in which parties decide whether or not to engage in collaboration), contextualized by current individual work. Consider the following scenario: given two individuals, A and B, with similar problems, tasks or interests, and a system that provides them with information about this similarity (which may be a mutual friend who makes an introduction), what factors will influence the decision to (1) propose cooperation and (2) accept this proposal?

In this scenario we consider two roles: the *initiator*, who proposes cooperation and the *responder*, who receives the proposal and must decide whether or not to cooperate. There are, therefore, two situations: the *initiator* (A) has a problem to solve or a task to accomplish, and a choice to contact the responder or not regarding this problem; or the *responder* (B) has received a proposal from the initiator and must decide whether or not to cooperate. Factors influencing these two decisions are similar but not exactly the same.

Methods to detect similarities are also under study, but will not be discussed here, as a large body of research exists on recommendation systems and methods [16]. We assume similarities have already been detected and concentrate on the cooperation part of the equation. As mentioned before, this model is not exhaustive, as it doesn't include task [17] or resource related aspects, some of which may also be negotiated by the parties. These will be added as we refine and expand the model.

3.1 Individual Perception

As we are dealing with individual decisions, all factors relate to the individual's perception of the world. The model for the Responder is shown in Fig. 1, and the model

for the Initiator is shown in Fig. 2. The propensity for collaboration is shown at the center of the model. The factors represent how each person sees (a) the task or problem being undertaken or being proposed (top boxes); (b) him or herself (bottom boxes); (c) the person with whom he or she might collaborate (boxes on the right side) and (d) the interaction that is proposed (boxes on the left side). Plus an minus sign indicate whether the element’s influence is positive or negative.

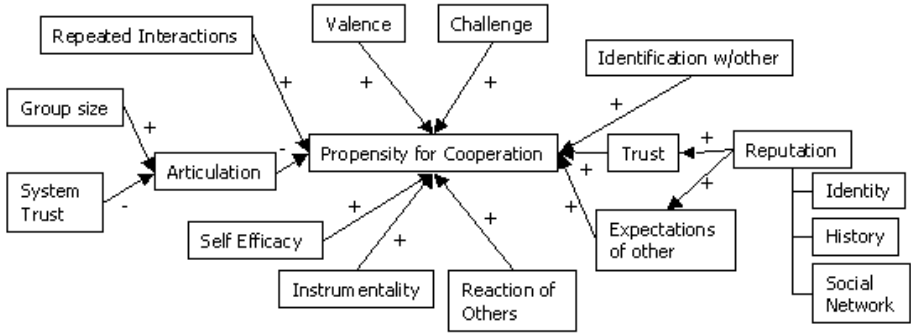


Fig. 1. Model of Cooperation for the Responder

An individual’s natural inclination towards collaborative or competitive behavior can be changed according to the situation, and factors such as the achievement of status, affection or behavioral confirmation affect the individual’s natural tendency to cooperate. Individuals decide to cooperate based on the long-term benefits they expect to obtain and the outcomes of the cooperation [7] [8]. Possible gains and losses are weighed by their likelihood [10] [13]. Some examples of possible gains are dividing the work, interacting with someone interesting or learning something new, and costs may involve time and effort, for instance.

Task. *Valence* is the degree of identification with the goal or task at hand. Motivation is directly proportional with goal or task valence [7] [10] [10] [13].

Challenge: challenging goals and complex tasks lead to increased participation [3] [13] [15]. The initiator may more readily perceive a need to call upon others when faced with a challenging task and the responder may enjoy the challenge or recognize the challenge at hand, and that it would be infeasible to undertake it alone.

Other. *Identification with other:* identification is an important factor for cooperation, and may lead to a sense of loyalty, commitment or camaraderie, greater effort and less loafing [3] [10] [13]. These effects are the result of recognition of interdependencies between individuals and the expectation of reciprocity [12]. It has a positive effect on a responder, when he or she identifies with the initiator’s situation (such as having faced the same problem or understanding the complexity of the tasks involved).

Reputation: reputation is based on three elements: identity, history and networks, all of which have been shown to have positive effects on cooperation levels. Being able to *identify* the partner leads to increased accountability and therefore to higher levels of cooperation[1]. Anonymity leads to an increased possibility of free-riding, since each individual’s actions cannot be identified [12]. Having information about

the partner's past behavior (*history*) can help determine whether this partner is trustworthy and if cooperation is desirable. When identity and history of interactions are distributed through a *network* or group, they can lead to the establishment of reputations, which can be used as a source of social information and control, as it generally matters to people what others say about them [1] [10].

Trust: interpersonal trust plays an important role in the establishment of cooperative relationships [7] both for initiator and for responders. It is the expectation that one's efforts will be reciprocated [10] and is influenced by interaction history, communication [10] and pre-existing relationships [21]. Reputations and intermediaries from a network can also help establish trust in relations [7] [13].

Expectations of other: if an individual believes the other to be incompetent or unable, he or she will be less likely to want to work with them [13]. Also, if an individual is expected not to perform the assigned tasks (considered untrustworthy), it is unlikely that there will be an initiation of contact or a response to it. These expectations are affected by the individuals' reputations.

Self. Instrumentality is the perceived importance or indispensability of one's contributions for the group outcome: the more noticeable the effect of a contribution on the outcome, the more inclined to cooperate the individual will be [1] [3] [10] [12] [13]. This impacts positively on the responder, as the individual will perceive his contribution as important to the joint venture.

Self-Efficacy: an individual's perceived capability of performing the required activities can greatly impact his level of cooperation. If he perceives himself as unable to accomplish his part, his motivation will be low [10] [10]. For the initiator, self-efficacy has a negative effect, as he may believe in his ability to perform the task alone. It has a positive effect on responders, as they perceive they can accomplish the task proposed.

Reaction of others: expected reactions of significant others (friends and family members) make a difference in one's actions. The more positive the feedback and the more valued the significant others, the higher the likelihood of cooperation [10].

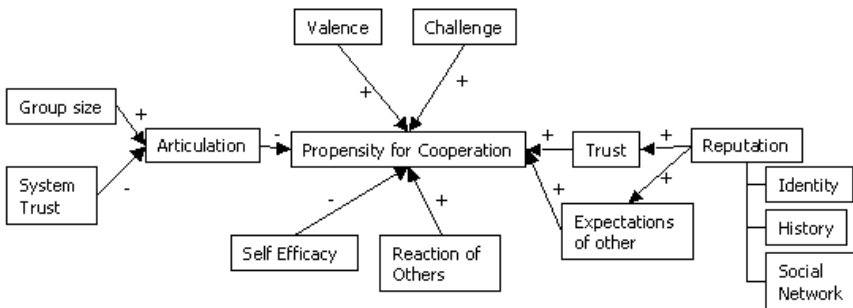


Fig. 2. Model of Cooperation for the Initiator

Interaction. *Repeated Interactions*: the existence or expectation of continued interaction leads to more cooperative strategies than when individuals meet only once or for the last time [1]. Repeated interactions enable sanctions and rewards, as well as reciprocity and identification of individual's actions [7]. Individuals will take into account

probable future interactions when deciding whether or not to cooperate [8]. This might influence respondents, who may be more inclined to cooperate if they perceive that they will have to interact with this person again in the future.

Articulation: in a group, additional effort must go into secondary activities to control and mediate relationships and to articulate interdependent activities. The added effort required communicating and coordinating actions affects cooperation negatively, as individuals may find it difficult to organize the group or perceive too much extraneous work to be necessary [5] [22]. Larger groups (*Group Size*) require greater articulation effort. Reduced visibility and influence of each member's actions and the added difficulty of identification lead to lower participation in larger groups [1] [12] [13]. System support may facilitate the articulation effort. The existence of a system leads to the expectation that team processes will work reliably (*System Trust*). Well-defined processes and technology can reduce the added complexity incurred due to the necessity of articulation work [10].

4 Discussion: Possible Applications

Many recommendation systems have been constructed to bring users together. However, the majority of systems are concerned with how well the matching algorithm works or how well levels of expertise are determined [16]. Many of these studies focus on help-seeking behavior [9] [20], but few evaluate levels of and reasons for engagement after recommendations are made. It seems that there is an underlying assumption that once made, a recommendation will be followed. While this may be the case in groups where all the individuals know each other or in companies where a strong cooperative culture already exists, it may not be so in larger groups, or when individuals do not know each other. The decision to engage in collaboration may seem like a simple one (when parties are in close contact or know each other), but it becomes more complicated in cases where there is no acquaintance or history of interaction, as when recommendations are expanded through social networks (an approach frequently adopted by recommendation systems).

We wish to explore opportunities for cooperative learning, work division and information exchange [2] [6], and how and why these happen. Using a model for collaboration would enable a designer (or the system) to emphasize different elements to achieve different results (for instance, providing information on a user's history of interaction with a common acquaintance or on a system to facilitate their interaction).

The model presented provides an integrated view of some of the elements that influence the decision to engage in cooperation with others. Even though an individual may be aware that another is available, sometimes communication may not be started due to a lack of confidence that the other party will be receptive. Using the aforementioned elements, a system could provide additional information that might motivate the initiation of cooperation. In any case, by providing additional information, the users will be able to make more informed decisions, whether to cooperate or not.

This is not a one-size fits-all model, and each user assigns different values to each of the elements involved (e.g. how important is what someone else thinks?) These preferences will strongly influence the final outcome of the decision making process. This model is meant as a generic view, which can be adapted to each user through explicit parameterization or inference from continued observation. With a stable and

expressive model to represent each user, one could conceivably calculate the likelihood of engagement into cooperation and add this information to a recommendation (e.g. A may be the topmost expert, but B is more likely to respond positively).

The model still needs to undergo appropriate validation, and experiments are being designed to that effect. Furthermore, there are influential elements regarding the work itself that need to be factored in, such as expected quantity and division of work, types of tasks [17], individuals' resources and availability and task interdependencies.

In distributed work, the existence of systems that provide the information necessary to enable collaboration is important, as it may not be readily available. Knowing what to communicate and to whom is a question we are trying to address. Models help us understand what and why things happen and serve as a guidance to systems development [4]. We hope that, with appropriate validation, this model can provide some useful insight for group support and recommendation systems.

References

- [1] Axelrod, R. *The Evolution of Cooperation*. Basic Books, New York (1984)
- [2] Becks, A., Reichling, T. Wulf, V. Supporting Collaborative Learning by Matching Human Actors. *Proceedings of HICSS 03.IEEE* (2002)
- [3] Beenen, G., Ling, K., Wang, X., Chang, K., Frankowski, D., Resnick, P., Kraut, R. Using Social Psychology to Motivate Contributions to Online Communities. In *Proceedings of CSCW 04*. Chicago, IL (2004)
- [4] Briggs, R.O. On Theory-Driven Design of Collaboration Technology and Process. In *Proceedings of the 10th International Workshop Groupware, CRIWG 2004*. San Carlos, Costa Rica. Springer-Verlag GmbH (2004)
- [5] Carstensen, P., Schmidt, K., Computer Supported Cooperative Work: New Challenges to Systems Design. In Kenji Itoh (ed.). *Handbook of Human Factors*, Tokyo (1999)
- [6] Constantino-González, M., Suthers, D. Automated Coaching of Collaboration Based on Workspace Analysis: Evaluation and Implications for Future Learning Environments. *Proceedings of HICSS 03. IEEE* (2002)
- [7] Diekmann, A., Lindenberg, S. Cooperation: Sociological Aspects. In *International Encyclopedia of the Social and Behavioral Sciences* (4). Oxford: Pergamon-Elsevier (2000)
- [8] Galance, N., Huberman, B. Organizational Fluidity and Sustainable Cooperation. In Carley, K. and Prietula, M. (eds.), *Computational Organization Theory*. Lawrence Erlbaum Associates, Hillsdale, New Jersey (1994) 217-240
- [9] Groth, K., Bowers, J. On Finding things Out: Situating Organizational Knowledge in CSCW. *Proceedings of ECSCW 01*. Kluwer Academic Publishers (2001)
- [10] Hertel, G., Geister, S., Konradt, U. Managing Virtual Teams: A review of current empirical research. *Human Resource Management Review* 15. Elsevier (2005) 69-95
- [11] Hertel, G. Niedner, S., Herrmann, S. Motivation in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel. *Research Policy, Special Issue on Open Source Development* (2003)
- [12] Kollock, P. Social Dilemmas: The Anatomy of Cooperation. *Annual Review of Sociology* 24 (1998) 183-214
- [13] Kraut, R.E. Applying Social Psychological Theory to the Problems of Group Work. In Carroll, J. (ed.) *Theories in Human Computer Interaction*. Morgan Kaufmann, NY (2002)

- [14] Kraut, R.E., Fussell, S.R., Brennan, S.E., Siegel, J. Understanding the Effects of Proximity on Collaboration: Implications for Technologies to Support Remote Collaborative Work. In Hinds, P., Kiesler, S. (eds.) *Distributed Work*. MIT Press, Cambridge, MA (2002) 137-164
- [15] Locke, E.A., Latham, G.P. Building a Practically Useful Theory of Goal Setting and Motivation: A 35-Year Odyssey. *American Psychologist* 57 (9) (2002) 705-717.
- [16] McDonald, D. Supporting Nuance in Groupware Design: Moving from Naturalistic Expertise Location to Expertise Recommendation. University of California, Irvine. Ph.D. Thesis (2000)
- [17] McGrath, J. *Groups: Interaction and Performance*. Prentice Hall, Inglewood, NJ (1984)
- [18] Nissani, M.: Ten Cheers for Interdisciplinarity: A Case for Interdisciplinary Knowledge and Research. *Social Science Journal*, 34 (2) (1997) 201-216
- [19] Paulus, P.B.: Groups, Teams and Creativity: The Creative Potential of Idea-Generating Groups. *Applied Psychology: An International Review*, 49 (2) (2000) 237-262
- [20] Pipek, V., Wulf, V. Pruning the Answer Garden: Knowledge Sharing in Maintenance Engineering. *Proceedings of ECSCW 03*. Kluwer Academic Publishers (2003)
- [21] Rocco, E. Trust Breaks Down in Electronic Contexts but Can Be Repaired by Some Initial Face-to-Face Contact. *Proceedings of CHI 98*. Los Angeles, CA. ACM Press (1998) 496-502
- [22] Schmidt, K. and L. Bannon: "Taking CSCW Seriously. Supporting Articulation Work" in *Computer Supported Cooperative Work*, Kluwer Academic Publishers (1992) 7-40.
- [23] Shubik, M. 13th International Conference on Game Theory, quoted by Wired: <http://www.wired.com/news/business/0,1367,54131,00.html>
- [24] Turocy, T., von Stengel, B. *Game Theory*. CDAM Research Report LSE-CDAM-2001-09. London School of Economics (2001)

Towards an Ontology for Context Representation in Groupware

Vaninha Vieira, Patrícia Tedesco, and Ana Carolina Salgado

Center for Informatics, Federal University of Pernambuco, Brazil
C.P. 7851, Recife, PE, Brasil – CEP 50732-970
{vvs, pcart, acs}@cin.ufpe.br

Abstract. An important issue in groupware is how to improve interaction and collaboration among participants. Through the analysis of the context a user is in or the context that surrounds an interaction, groupware systems can provide users with useful information in that situation. A relevant issue when using context is how to represent context information. Ontologies constitute an interesting method for representing context, since they enable information sharing and reuse. They can also be used by existing inference machines to reason about various contexts. In this paper we propose an ontology to formally represent context in groupware systems. We also present an example where this ontology is used by a logic-based reasoning mechanism for tool recommendation based on the current context of group members. We believe that this ontology could help to understand the role of context in collaboration and thus make the development of context-aware groupware systems easier.

1 Introduction

Context is defined as any information used to characterize the situation of an entity where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application [7]. Context-aware systems are those able to understand the context of users and anticipate their needs, in terms of services and/or information. Moreover, context can play an important role in the communication and interaction between humans as well as between humans and machine, since it diminishes ambiguity and conflicts, increases the expressiveness of dialogues, makes applications more friendly, flexible and easy to use, and consequently raises user's satisfaction.

When people collaborate, it is essential that they perceive and understand things that are happening or have happened in the context of their group which are relevant for the accomplishment of their activities [14]. This concept is known as awareness in Computer Supported Cooperative Work (CSCW). The lack of awareness can provoke several problems, such as conflicts, duplicated or inconsistent work and unmotivated participants.

The definition of *awareness* associates this knowledge with the *context* of the group. However, these two different but related concepts are not yet well defined and

explored by CSCW researchers. The use of the concept of context by groupware systems is still an open issue. Some attempts in doing so are presented in [1, 5, 10, 12]. Through the analysis of the context a user is in or the context that surrounds an interaction, a groupware system can provide the users with information that is valuable in that situation.

An important issue in reasoning about context is the acquisition of relevant information. Generic context models are of interest since many applications can benefit from these. Most of the work done thus far [13], indicate that ontologies are a very interesting approach to represent context. This is because they are a technique that enables knowledge sharing between human and software agents, as well as knowledge reuse between systems. Moreover, ontologies can be used by existing inference machines to reason about various contexts.

This paper presents a formal context model based on ontologies using OWL (Web Ontology Language [2]) that describes context information in order to support the use of context in groupware systems. The idea is to identify which information presented in groupware applications could be classified as context and which kinds of context we should represent. Through this model it is possible to compose inference rules that enable the identification of high-level implicit context from low-level explicit context. To clarify matters, we present a scenario exemplifying how this ontology could be used by logic-based reasoning mechanisms to recommend tools based on the current context of group members.

This paper is organized as follows: Section 2 introduces some concepts related to context and context representation; Section 3 discusses some related work; Section 4 describes our proposal and an example of its use; and Section 5 presents our conclusions and future work.

2 Context and Context Representation

Context is a well established concept in everyday life. We commonly use the word context to restrict the situation we are talking about. However, when we think about context as a computational concept, things become a little less straightforward. First it is necessary to define what exactly we mean by context, identifying the focus of the context and the information needed to describe it. Next, we need to choose a technique to represent the context. Then it is necessary to define ways to acquire and process the context, reasoning over basic context information to generate high-level complex context.

There are several attempts to define and use context for computational purposes (e.g.[4, 7, 8]). In fact, with the advance of context-aware computing, there is an increasing need for developing formal context models to facilitate context representation, sharing and semantic interoperability of heterogeneous systems [15].

In human or service interactions, it is always desirable that each agent shares the same interpretation of the exchanged data. Ontologies define common vocabularies for information sharing in a domain including machine-interpretable definitions of concepts and relations among them [11]. There are several reasons for developing ontology-based context models [11, 15], namely: to share a common understanding of the structure of information; to enable reuse of domain knowledge; to make domain

assumptions explicit; and to enable the use of existing inference engines to reason about context.

The use of location context is well known and widely propagated in ubiquitous computing applications [6]. However, there is an increasing interest in using context in other application domains, such as groupware systems. In these systems the concept of context is already present albeit associated to issues such as awareness information and group memory [3]. This research trend is described in the next section.

3 Related Work

There are some ontologies for context representation proposed in the literature such as CONON (Context Ontology) [15] and CoBrA-Ont [6]. Similarly to our work both are constructed using OWL. However these ontologies are used to model context in the ubiquitous computing domain and consider only the concepts of location, time, people and devices. They do not consider the organizational context as well as the context related to group work and collaborative interactions.

There are some studies on how context can be modeled and managed in collaborative applications [1, 3, 5, 9, 10, 12]. Brézillon et al. [5] consider three levels of specificity for describing context in group work: individual, group and project context. The conceptual framework proposed in [12] points out some contextual elements present in groupware applications related to the group members, the group itself, the scheduled and the completed tasks, the interaction that led to the concluded tasks and the environment where the interaction took place. Kirsch-Pinheiro et al. [10] propose a context-based awareness mechanism which filters the information delivered to the user according to a context description. It takes into account concepts such as group and role definition, activities and work process, and uses an object-oriented representation. Alarcon et al. [1] propose a preliminary taxonomy for context in groupware based on a survey of the work done in groupware. They define group context in terms of three main components: people, task or project, and resources.

There is some consensus among researchers that context information for groupware should include entities such as project, group, people, tasks and resources as well as the whole situation that surrounds the interaction among members. However, the modeling and use of context information to improve group work is still an open issue. Our ontology considers concepts such as the physical context concept (based on CONON and CoBrA) and the organizational context (similar to some concepts defined in [1] and [10]). However we differ in the classification for context information and we contribute with the use of the ontology for inference. The next section describes our proposal and an example of its use.

4 A Context Ontology for Groupware

This section describes our first steps in the construction of an ontology for representing context information in groupware systems. The ontology is constructed using the Web Ontology Language (OWL) [2]. OWL is a semantic web language

proposed by the W3C that enables the definition of domain ontologies and sharing of domain vocabularies. A domain is formalized through the definition of classes, properties and instances of these classes. To edit the ontology and axioms we used the Protégé 3.1¹ and to implement the rules to reason about the context we used the Jeops² inference machine. The following sections present the main concepts defined and an example of their use.

4.1 Main Concepts

Figure 1 shows a subclass/superclass upper level view of the main concepts defined in the context ontology. These concepts, their properties and the classification proposed were constructed based on a survey of the concepts related to context and collaboration found in the technical literature [1, 6, 10, 12, 15]. These concepts are explained below.

Context: this is the main class for all the context elements. It is divided into three subclasses: *physical context*, *organizational context* and *interaction context*.

PhysicalContext: contains information about the physical elements that characterize the situation a user is in at a specific time. This indicates the whole environment that surrounds the user, the physical and virtual space s/he is located and relations such as proximity, distance, presence, and absence (*LocationContext*); the time when the interaction occurs and information about the context of the time such as time zone (*TimeContext*); the physical and electronic devices available, such as printers, computers, microphones, webcams (*DeviceContext*) and physical conditions such as temperature (*ConditionContext*).

OrganizationalContext: this concept represents the contextual information related to the whole structure that identifies the *user*, the *group* s/he belongs to and the *role* s/he is playing in the execution of a *task* which is part of a process in a specific *project*. Thus, the *ProjectContext* contains elements that identify the context of the project that is being carried out by the group members such as objectives and schedules. The *GroupContext* represents information such as the objective, abilities and interests of the group. The *AgentContext* contains elements related to the members of the group. We consider that a group can be composed of human members (*HumanAgentContext*) and also of software members (*SoftwareAgentContext*), which can act collaboratively with other software or human agents, in an intelligent collaborative application. Contextual information for the *HumanAgentContext* includes identification, interests, abilities and availability. The *SoftwareAgentContext* has information such as objective, intentions and utility function. The *TaskContext* concept represents elements related to the context of the tasks (and subtasks) that are being or have been executed including the task objective and deadline. The *RoleContext* refers to contextual elements related to the roles that members of a group can play in the execution of a task, identifying, for instance, if the user plays the role of coordinator.

¹ <http://protege.stanford.edu/>

² <http://www.di.ufpe.br/~jeops/>

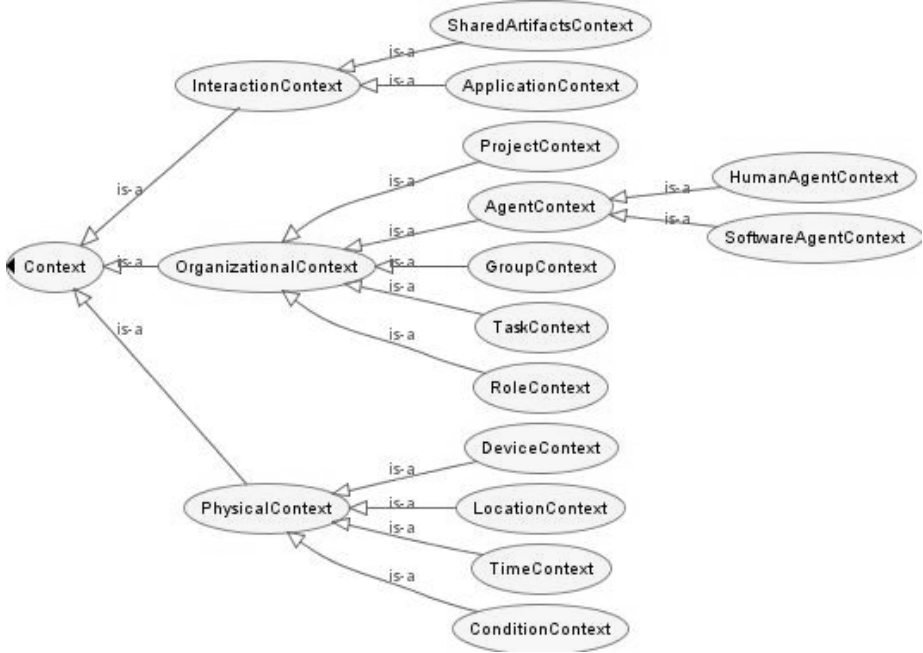


Fig. 1. The Main Class Hierarchy Partial View

InteractionContext: identifies the information related to the context of an interaction that is happening (synchronous) or that has happened (asynchronous) during group work. It is divided into two subclasses: *SharedArtifactsContext* and *ApplicationContext*. The *SharedArtifactsContext* contains elements related to the context of the shared artifacts used in the interaction, such as class objects in a collaborative class diagram or pieces of text in a collaborative text editor. The *ApplicationContext* includes information related to the context of the application being used or available to be used in the interaction, such as the purpose of the application (for communication, coordination or awareness) and the type of interaction it supports (synchronous/asynchronous).

According to the OWL specification, each concept has *datatype* properties, which define its attributes, and *object* properties, which define its relationships to other concepts. For example, as shown in Figure 2, the *HumanAgentContext* has *datatype* properties such as *hasName*, *isAvailable*, *hasVisualDisability*, and *object* properties such as *isMemberOf* (*GroupContext*), *performsRole* (*RoleContext*), *isLocatedIn* (*LocationContext*), *hasDeviceContext* (*DeviceContext*) and *executesTask* (*TaskContext*). For the sake of space we will not describe the properties of all classes in this paper.

4.2 Context Inference

One of the main advantages of using a formal model, such as ontologies, to represent context is that logical reasoning mechanisms can be used for checking context

consistency and inferring new complex information from existing basic context. For instance, the system could suggest to users participating in an interaction which communication tool should be used for them to interact more easily and efficiently with their peers.

To explain the use of context reasoning in the ontology proposed consider a scenario where a context-aware groupware system can make recommendations based on the current context of each member in the group or in an interaction. Suppose that Mary, Joseph and Rose work together in tasks related to a project for a new module on a computing course. The first two are the teachers and the latter is the module’s teaching assistant. At some point, Rose needs to contact the teachers to discuss her ideas about the module. Rose and Joseph are in their respective homes, while Mary is at her office. Rose wishes to know which communication tool is better suited to contact and interact with both professors, considering their current context.

Table 1 summarizes the current context of each human agent and shows that the three of them are available and have the following hardware devices installed: a video

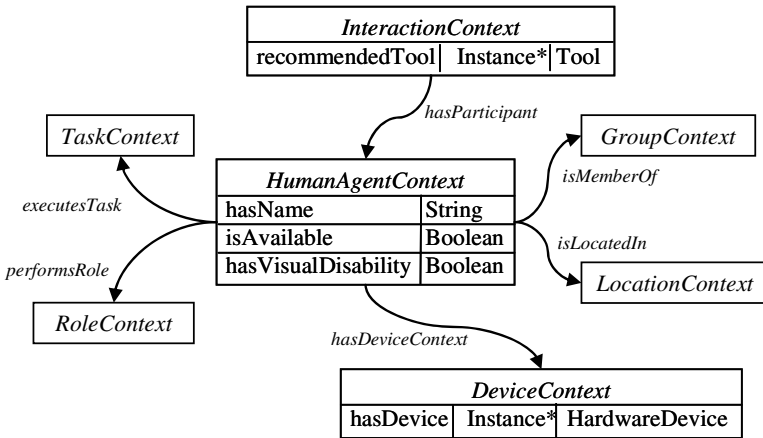


Fig. 2. Partial View of the Properties of *HumanAgentContext* Concept

Table 1. Examples of Instances for the *HumanAgentContext* Class

Context	Mary	Joseph	Rose
isAvailable	True	True	True
hasVisualDisability	False	True	False
hasAudioDisability	False	False	False
hasDeviceContext	VideoCard \wedge SoundCard \wedge Microphone \wedge Keyboard	VideoCard \wedge SoundCard \wedge Microphone \wedge Keyboard	VideoCard \wedge Webcam \wedge SoundCard \wedge Microphone \wedge Keyboard
isMemberOf	ContextModuleGroup IntegraGroup	ContextModuleGroup IntelligentAgentsGroup	ContextModuleGroup
executesTask	PrepareContextClasses DefineModuleTopics	PrepareContextClasses DefineModuleTopics	DefineModuleTopics
isLocatedIn	Office	Home	Home

card, a sound card, a microphone and a keyboard. Rose also has a webcam, and Joseph has some kind of visual disability. None of them have audio disability. They belong to a common group (*ContextModuleGroup*), are involved in a same task (*DefineModuleTopics*) and the three of them are located in different physical places.

Video conferencing, audio conferencing, chat and email are examples of communication tools that peers can use to directly contact each other. These tools have specific hardware device requirements to be used and offer different services that could make the communication more or less effective for discussion purposes. Considering these characteristics we have classified these tools from the most to the least effective: Video conferencing as the most effective followed by audio conferencing, chat, and email as the least effective. However, this order must not be too rigid as the tool priority could change according to the agent context. For example, if a user has a hearing disability s/he cannot use a tool that requires audio, such as audio or video conferencing.

Table 2 shows some inference rules defined for the recommendation of communication tools, taking into account the context of the human agents that wish to establish the communication, based on the example above. The rules check the context of each user for each tool and set new context information to the interaction context, represented by the object property *recommendedTool*. The recommended tool will be the first one (according to the sequential order in Table 2) that matches the correspondent rule.

The rule for the *video conferencing* tool indicates that the *HumanAgentContext* of the three involved people must indicate that they are available, have no hearing disability and have microphone, sound card, video card and webcam as hardware devices. The *audio conferencing* rule establishes that they must be available, have no hearing disability and have microphone and sound card installed. The *chat* rule indicates that they must be available, have no visual disability and must have at least a keyboard. Lastly, the *email* rule indicates that only a keyboard device is necessary to establish the communication but the user must have no visual impairment.

These rules are validated by an inference machine that will use them to answer queries such as “which communication tool the users in the context described in Table 1 should use to interact?”. According to the information provided in the example the answer is an *audio conferencing* tool. A video conferencing tool cannot be used because only one of the users in the interaction has a webcam, which is a required device defined in its inference rule. Also email and chat are not appropriate since one of the participants has visual disability.

Table 2. Inference rules defined for communication tools recommendation

<i>Tool</i>	<i>Inference Rule (for each HumanAgentContext h in InteractionContext i)</i>
Video Conferencing	$isAvailable(h) \wedge \neg hasAudioDisability(h) \wedge hasDeviceContext(h, Microphone \wedge SoundCard \wedge VideoCard \wedge Webcam) \Rightarrow recommendedTool(i, VideoConferencing)$
Audio Conferencing	$isAvailable(h) \wedge \neg hasAudioDisability(h) \wedge hasDeviceContext(h, Microphone \wedge SoundCard) \Rightarrow recommendedTool(i, AudioConferencing)$
Chat	$isAvailable(h) \wedge \neg hasVisualDisability(h) \wedge hasDeviceContext(h, Keyboard) \Rightarrow recommendedTool(i, Chat)$
Email	$\neg hasVisualDisability(h) \wedge hasDeviceContext(h, Keyboard) \Rightarrow recommendedTool(i, Email)$

This is a small example of how the use of a context ontology could help improve group work. It demonstrates how we can use different kinds of basic contextual information (Physical, Interaction and Human) to produce new high level context information. Other facts and predicates could be considered in the rule composition to recommend the tool, such as the task the users are involved in, the mode of the interaction (synchronous/ asynchronous), the place they are located or the users' preferred tools. We can extend the tools recommendation to consider not only communication tools, but coordination or collaborative editing tools. Another promising use is the recommendation of expertise, helping users that have common interests and complementary abilities to opportunistically be in touch and create, for instance, communities of practice.

5 Conclusions and Further Work

This paper presents an ontology for context representation in groupware systems. The context information is classified in three main categories: physical, organizational and interaction context. The paper also presents an example of the use of this ontology for context inference, recommending tools for communication among users based on the current context of each user.

We expect that this ontology will be used by developers of groupware systems to identify, model and represent context in their applications. The ontology could also be used by intelligent software agents to manage and infer contextual information. Inference rules based on the ontology can support collaborative interactions by recommending resources to participants or filtering awareness information notification.

We are currently working on refining the concepts and their properties, as well as on identifying scenarios where the ontology could be applied and the definition of inference rules for context reasoning in these scenarios. In the near future, we plan to implement a prototype of an intelligent groupware system that uses the context ontology to recommend tools and expertise.

Acknowledgments. The authors would like to thank CNPq for their financial support. The first author also thanks the UFBA for their support.

References

1. Alarcon, R. A., Guerrero, L. A., Pino, J. A. Groupware Components as Providers of Contextual Information. In: CONTEXT'05, Paris, France (2005).
2. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L. "Web Ontology Language (OWL) Reference, W3C Recommendation" (2004), Accessed in 03/2005.
3. Borges, M.R.S., Brézillon, P., Pino, J.A., Pomerol, J.Ch. "Bringing Context to CSCW". In: Proc. of the 8th Int. Conference on Computer Supported Cooperative Work in Design, Xiamen, P.R. China, v. II, International Academic Publishers, Beijing World Publishing Corporation, IEEE Press (2004), pp. 161-166.
4. Brézillon, P. "Modeling and using context: Past, present and future". Rapport de Recherche du LIP6, Université Paris 6, France, In: <http://www.lip6.fr/reports/lip6.2002.010.html> (2002), Accessed in 03/2005.

5. Brézillon, P., Borges, M.R.S., Pino, J.A., Pomerol, J.Ch. "Context-Awareness in Group Work: Three Case Studies". In: Proc. of the 2004 IFIP Int. Conf. on Decision Support Systems, Prato, Italy (2004), pp. 115-124.
6. Chen, H., Finin, T. "An Ontology for a Context Aware Pervasive Computing Environment". In: IJCAI Workshop on Ontologies and Distributed Systems, Acapulco, MX (2003).
7. Dey, A. K., Abowd, G. D. "Towards a Better Understanding of Context and Context-awareness". In: CHI 2000. Workshop on The What, Who, Where, When, Why and How of Context-awareness, ACM, The Hague, The Netherlands (2000), pp. 1-6.
8. Dourish, P. "Seeking a Foundation for Context-Aware Computing", *Human Computer Interaction*, v. 16, n. 2 (2001), pp. 229-241.
9. Gross, T., Prinz, W. "Awareness in Context: A Light-Weight Approach". In: Proc. of the Eight European Conference on Computer-Supported Cooperative Work (ECSCW 2003), Kluwer Academic Publishers, Dordrecht, NL (2003), pp. 295-314.
10. Kirsch-Pinheiro, M., Gensel, J., Martin, H. "Representing Context for an Adaptive Awareness Mechanism". In: Proc. of the X International Workshop on Groupware (CRIWG'2004), v. LNCS 3198, San Carlos, Costa Rica, Springer-Verlag (2004).
11. Noy, N.F., McGuinness, D.L. "Ontology Development 101: A Guide to Creating Your First Ontology". In: <http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html> (2001), Accessed in 03/2005.
12. Rosa, M.G.P., Borges, M.R.S., Santoro, F.M. "A Conceptual Framework for Analyzing the Use of Context in Groupware". In: Proc. of CRIWG'03, v. LNCS 2806, Springer-Verlag Berlin, Heidelberg (2003), pp. 300-313.
13. Strang, T., Linnhoff-Popien, C. "A Context Modeling Survey". In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp2004, Nottingham, England (2004).
14. Vieira, V., Mangan, M.A.S., Werner, C.M.L., Mattoso, M.L.Q. "Ariane: An Awareness Mechanism for Shared Databases". In: Proc. of the X International Workshop on Groupware (CRIWG'2004), v. LNCS 3198, San Carlos, Costa Rica, Springer-Verlag (2004), pp. 92-104.
15. Wang, X. H., Zhang, D. Q., Gu, T., Pung, H. K. "Ontology Based Context Modeling and Reasoning using OWL". In: Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom2004), Orlando, FL, USA (2004), pp. 18-22.

Author Index

- Antunes, Pedro 168, 308
Avgeriou, Paris 49
- Baloian, Nelson 341
Barsotini, Claudia 248
Barthès, Jean-Paul 359
Borges, Marcos R.S. 216
Bosquet, Fabien 121
Brézillon, Patrick 232
Bu, Jiajun 137
- Capponi, Francisca 263
Carminatti, Naiana 216
Chang, Sheng-Ho 192
Chen, Chun 137
Collazos, Cesar 284
- Dabholkar, Akshay 17
Dean, Douglas L. 325
Decouchant, Dominique 33
de Farias, Cléver R.G. 105
de Souza, Jano Moreira 359
de Vreede, Gert-Jan 325
Domingos, Henrique 89
Duarte, Sérgio 89
- Eim, Kristin 292
Ellis, Clarence A. 184
Englert, Roman 153
- Favela, Jesus 33
Ferreira Pires, Luís 105
Fruhling, Ann L. 325
Fuller, David A. 255
- Gomes, José Orlando 216
Gonçalves, Carlos E. 105
Guerrero, Luis A. 284, 351
Guicking, Axel 49
- Herrera, Oriol 255
Husby, Øyvind 292
Hwang, Gwan-Hwan 192
- Jeffery, Clinton 17
Jiang, Bo 137
- Kim, Kwanghoon 184
Kim, Yosep 17
Koneri, Pushpa G. 325
- Lagos, María Ester 263
Lee, Yung-Chuan 192
Lukosch, Stephan 73
- Madariaga, Milko 284
Marquès, Joan Manuel 57
Martins, J. Legatheaux 89
Martínez Enríquez, Ana María 33
Mendoza, Sonia 33
Morán, Alberto L. 33
Mourão, Hernâni 168
Munkvold, Bjørn Erik 292
- Navarro, Leandro 57
Neyem, Andrés 351
Nussbaum, Miguel 263
- Ochoa, Sergio F. 284, 351
- Paul, Jean-Claude 121
Pinkwart, Niels 145
Pino, José A. 284, 351
Pipek, Volkmar 153
Preguiça, Nuno 89
- Ramires, João 308
Raposo, Alberto B. 121
Reis, Luciano P. 121
Respício, Ana 308
Rosatelli, Marta C. 105
- Salgado, Ana Carolina 367
Santoro, Flávia Maria 232
Schümmer, Till 73
Slagter, Robert 73
Stahl, Gerry 1, 271
- Tachtevrenidis, Kosta 17
Tandler, Peter 49
Tedesco, Patrícia 367
Tramontina, Gregório Baggio 208

van Sinderen, Marten 105

Vieira, Vaninha 367

Vivacqua, Adriana S. 359

Wainer, Jacques 184, 208, 248

Wolcott, Peter 325

Won, Markus 153

Wulf, Volker 153

Xhafa, Fatos 271

Yang, Jianxv 137

Zemel, Alan 271

Zurita, Gustavo 341