Gert-Jan de Vreede
Luis A. Guerrero
Gabriela Marín Raventós (Eds.)

LNCS 3198

Groupware: Design, Implementation, and Use

10th International Workshop, CRIWG 2004
San Carlos, Costa Rica, September 2004
Proceedings

Springer

Lecture Notes in Computer Science 3198

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

This page intentionally left blank

Gert-Jan de Vreede Luis A. Guerrero
Gabriela Marín Raventós (Eds.)

Groupware: Design, Implementation, and Use

10th International Workshop, CRIWG 2004
San Carlos, Costa Rica, September 5-9, 2004
Proceedings

Springer

eBook ISBN: 3-540-30112-7
Print ISBN: 3-540-23016-5

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.springerlink.com
and the Springer Global Website Online at: http://www.springeronline.com

Preface

This volume constitutes the proceedings of the 10th International Workshop on Groupware (CRIWG 2004). The conference was held in San Carlos, Costa Rica, and followed the traditional CRIWG spirit: a relatively small number of attendees, lively discussions and participation during and between paper presentations, and a constructive and friendly environment, which promoted a high level of learning and collaboration.

The previous nine 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), and Autrans, France (2003).

This 10th anniversary CRIWG exemplified the continuing interest in the groupware research area. Groupware researchers from 14 different countries submitted a total of 71 papers. Each of the 71 papers was reviewed by two to three members of an international Program Committee, using a double-blind reviewing process. Reviewers were carefully matched to paper submissions as much as possible to ensure expert assessments of the submitted research while avoiding potential conflicts of interest. Reviews were usually extensive and contained many constructive suggestions for improvement. 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 seven themes that represent current pockets of interest in groupware research: knowledge management, awareness, support for collaboration processes, collaborative applications, groupware infrastructure, computer-supported collaborative learning, and mobile collaborative work.

CRIWG 2004 would not have been possible without the work and support of a great number of people. First of all we extend our sincere appreciation for the reviewers' efforts that enabled both a strong selection of papers for the conference as well as provided useful and actionable feedback for the large majority of the submissions, even if they were rejected. We were grateful for the advice and support provided by the CRIWG Steering Committee. Last but not least we thank Alan Calderón and Gwendolyn Kolfschoten for their enthusiastic support.

We especially wish to acknowledge our sponsoring organizations: Universidad de Costa Rica, Grupo GBM de Costa Rica, and Universidad de Chile.

September 2004

Gert-Jan de Vreede
Luis A. Guerrero
Gabriela Marin

Conference Organization

Program Committee Chairs

Gert-Jan de Vreede and Luis A. Guerrero

Program Committee Members

Pedro Antunes, Universidade de Lisboa, Portugal
Beatriz Barros, UNED, Spain
Karin Becker, PUCRS, Brazil
Marcos Borges, Univ. Federal do Rio de Janeiro, Brazil
Patrick Brézillon, Laboratoire LIP6, Université Paris 6, France
Bertrand David, École Centrale de Lyon, France
Dominique Decouchant, Laboratoire LSR-IMAG, Grenoble, France
Pierre Dillenbourg, EPFL, Lausanne, Switzerland
Yannis Dimitriadis, Universidad de Valladolid, Spain
Jesus Favela, CICESE, Ensenada, Mexico
Christine Ferraris, Université de Savoie, Chambéry, France
Hugo Fuks, Catholic University of Rio de Janeiro, Brazil
David Fuller, PUC, Chile
Joerg M. Haake, FernUniversität Hagen, Germany
Andreas Harrer, University of Duisburg, Germany
Gerd Kortuem, Lancaster University, United Kingdom
Gloria Mark, University of California, Irvine, USA
Ana M. Martínez, CINVESTAV-IPN, Mexico D.F., Mexico
Jose A. Pino, Universidad de Chile, Chile
Jean-Charles Pomerol, Laboratoire LIP6, Université Paris 6, France
Wolfgang Prinz, Fraunhofer FIT, Germany
Mike Robinson, University of Jyvaskyla, Finland
Ana Carolina Salgado, Univ. Federal de Pernambuco, Brazil
Alfredo Sánchez, UDLA-Puebla, Mexico
Ivan Tomek, Acadia University, Nova Scotia, Canada
Aurora Vizcaino, Univ. de Castilla-La Mancha, Spain
Ana I. Martinez-García, CICESE, Mexico
César Collazos, Universidad del Cauca, Colombia
Martin Beer, Sheffield Hallam University, United Kingdom
Gerry Stahl, Drexel University, USA
Stephan Lukosch, FernUniversität Hagen, Germany
Suprateek Sarker, Washington State University, USA
Fred Niederman, Saint Louis University, USA
Jerry Fjermestad, NJIT, USA
Fiona Fui-Hoon Nah, University of Nebraska at Lincoln, USA
Doug Vogel, City University Hong Kong, China

Sajda Qureshi, University of Nebraska at Omaha, USA
Ann Massey, Indiana University, USA
Christer Carlsson, Åbo Akademi University, Finland
Jaco Appelman, Delft University of Technology, The Netherlands
Rod Jarman, Curtin University of Technology, Australia
Helena Holmström, Göteborg University, Sweden
Jackie Phahlamohlaka, University of Pretoria, South Africa
Bjorn-Erik Munkvold, Agder University College, Norway
Danny Mittleman, De Paul University, USA
Bruce Reinig, San Diego State University, USA
Robert Briggs, University of Arizona & GroupSystems.com, USA
Fran Ackermann, University of Strathclyde, UK
Duncan Shaw, Aston University, UK
Conan Albrecht, Brigham Young University, USA
Douglas Dean, Brigham Young University, USA
Jay Nunamaker, University of Arizona, USA
Donald Amoroso, San Diego State University, USA
Eric Santanen, Bucknell University, USA
Pak Yoong, Victoria University of Wellington, New Zealand
Saul Greenberg, University of Calgary, Canada
Els van de Kar, Delft University of Technology, The Netherlands
Gwendolyn Kolfschoten, Delft University of Technology, The Netherlands
Marielle den Hengst, Delft University of Technology, The Netherlands
Joyce Beumer, Delft University of Technology, The Netherlands
Mirjam Huis in 't Veld, Delft University of Technology, The Netherlands

Doctoral Colloquium Chair

Hugo Fuks, Catholic University of Rio de Janeiro, Brazil

Organization Committee Chair

Gabriela Marín, Universidad de Costa Rica, Costa Rica

Organization Committee

Gabriela Marín, Universidad de Costa Rica, Costa Rica
Vladimir Lara, Universidad de Costa Rica, Costa Rica
Marcelo Jenkins, Universidad de Costa Rica, Costa Rica

This page intentionally left blank

Table of Contents

Key Note

1. Knowledge Management

2. Awareness

3. Support for Collaboration Processes

4. Collaboration Applications

5. Groupware Infrastructure

This page intentionally left blank

On Theory-Driven Design of Collaboration Technology and Process

Robert O. Briggs

Delft University of Technology, University of Arizona
bbriggs@groupsystems.com

Abstract. The design and deployment of collaboration technology has, until lately been more of an art than a science, but it has produced some solid successes. Commercial groupware products now support millions of collaborations per year. Under certain circumstances teams that use Group Support Systems perform far better than groups that do not. However, as impressive as the achievements are in this field, we can do better. A rigorous theoretical approach to the design of collaboration technology and process can lead us to non-intuitive design choices that produce successes beyond those possible with a seat-of-the-pants approach. This paper explains the simple structure of a rigorous scientific theory and offers examples of theory-driven design choices that produced substantial benefits. It then differentiates rigorous theory from several classes of theory that have intuitive appeal, but cannot inform design choices. It then argues that the logic of the theory-driven design approach suggests that the most useful focus for collaboration technology researchers would be the technology-supported work process, rather than just the technology.

1 Collaboration Technology Design as an Art

Designing collaboration technology has, until lately, been an art, founded on common sense and intelligence, guided by heuristics derived from inspiration tempered by hard experience. This approach has given rise to some solid long-term successes – consider, for example Lotus Notes, NetMeeting, and Webex, each of which now supports millions of collaborations per year. A robust body of literature shows that, under certain circumstances, people who use Group Support Systems (GSS) can be substantially more productive than people who do not (see Fjermestad and Hiltz, 1999, 2001 for compendia of GSS lab and field research). In 1999, we surveyed 127 organizations that used GSS. They reported an average cost saving of $1.7 million per year on an average technology investment of $75,000 USD. A year-long field study of more than 60 groups of GSS users at Boeing (Post, 1992) showed an ROI of 689% on an investment of approximately $100,000. It is rare in any field to find returns at that level.

Such results are nothing short of spectacular, yet as good as they are, we can do better; much better. Brilliant, intuitive minds and seat-of-the-pants reasoning can only carry us so far.

As many successes as there have been in this field, there have been many more failures. Remember The Coordinator? In the early 1990's, this system was heavily funded and highly touted. It was one of the first attempts at an integrated virtual

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 1–15, 2004.

workspace, including team calendaring, document repository, and a handful of other seemingly useful technologies. Yet users hated the system. Some disparaged it as Nazi-ware because of the draconian patterns of collaboration it enforced (e.g. every inbound message shall receive a reply before other work could be continued in the system).

The Boeing GSS case was so successful that it was written up in the February, 1992 issue of Fortune Magazine. And yet, the same week the article appeared, Boeing disbanded their GSS facility and reassigned all its personnel to other projects. This pattern of significant success followed by sudden cessation has been repeated by many organizations that adopt GSS (Agres, Vreede and Briggs, 2004, Briggs, Vreede and Nunamaker, 2003).

In light of these examples, consider these questions:

- How can we account for the dramatic success of some collaboration technologies?
- More importantly, how can we repeat those successes elsewhere?
- As successful as some collaboration technologies have been, are they as successful as they could be?
- How would we know?
- What else could we try that has never been considered that would be all but guaranteed to work?
- How can we account for stunning failures of other collaboration technologies?
- More importantly, how can we avoid them elsewhere?
- Do we have to wait for inspiration to strike some genius in our field before we attain our next leap forward?

Good theory can address all these questions.

2 There Is Nothing So Useful as a Good Theory

There is nothing as useful as a good theory. This assertion may draw snorts of derision from skeptics who may regard theory as an excuse for not doing anything useful. Yet, a good theory can put people on the moon and return them safely to earth on the first try. What one theory can do for space travel, others can do and have done for collaboration technology. Rigorous theory can lead to designs for collaboration technology process that far surpass those produced by a good mind and a gut feel. This paper explains what is meant by *good theory,* and present several cases to illustrate how a good theory can drive the design and deployment of collaboration technology in non-intuitive ways to yield unexpected success. It also discusses several classes of theory that seem tantalizingly useful at first, but which cannot lead to useful results. It concludes by discussing implications for ongoing collaboration technology research.

2.1 Good Theory: Always a Causal Model

A good scientific theory is a *model* of *cause-and-effect* to explain some *phenomenon of interest.*

Every technology presumes a cause-and-effect. Every technology is built to improve some outcome. By definition, this presumes that mechanisms exist to cause

changes in the outcome of interest, and that technology can be used to invoke those mechanisms. A theory is a model of those causal mechanisms. The theory gives us a basis for understanding how we might use technology to attain the outcomes we want. If we have not taken the time to rigorously articulate the mechanisms that cause the outcome of interest, our technologies and our processes may miss the mark.

2.2 Phenomenon-of-Interest: Always the Effect; Never the Cause

In a good theory, the phenomenon-of-interest is *always* the effect; the outcome. The phenomenon of interest is the unchanging focus of a theory. The first step to successful theory-driven design is to explicitly identify, and then define the phenomenon of interest. It is not always obvious at first what outcomes a technology is meant to improve. For collaboration researchers, the possibilities are many – productivity, creativity, satisfaction, and so on. Suppose a technology designer wanted to improve satisfaction with group processes and with group products among stakeholders who were negotiating requirements for a new software development project. The phenomenon of interest would be satisfaction. It would not be group process, not group product, not requirements negotiation, not software development, and not project management. The research question would be, "What causes people to feel satisfied?"

However, labeling the phenomenon of interest is not sufficient. It must be explicitly defined. For example, the word, satisfaction, has many connotations in the English language. In one sense, satisfaction could be a *judgment* that goals have been attained or constraints have been met. In another sense, satisfaction could be an *emotional response* pertaining to goal attainment. These different satisfactions spring from different causes, and so have different theoretical explanations. Theory-driving design must therefore begin by not only identifying and labeling, but also by explicitly defining the phenomenon-of-interest.

Having identified and defined a phenomenon of interest, the next step is to challenge one's judgment, asking whether this outcome is truly the most useful or important target for improvement. If this outcome were improved, whose work might be more effective? Whose life might improve? Is there other outcome that could be improved instead to yield better results? When one can present an unbreakable case that a certain outcome is truly worthy of effort, it is then time to seek good theory.

2.3 Good Theory: Constructs Connected by Propositions

A good theory is a model of cause-and-effect that can account for variations in the outcome of interest. The logic of these models can be represented as collection of statements with a particular structure. These statements have only two components: axioms and propositions. For example:

If we assume that:

(Axiom 1) all individual actions are purposeful toward attaining goals,

Then it must be that:

(Proposition 1) the effort an individual expends toward attaining a group goal will be a function of goal congruence (the degree to which the group goal is compatible with the individual's private goals.)

An axiom is nothing more than an assumption about some mechanism that could affect the phenomenon of interest. An axiom doesn't assert Truth (with a capital T). It simply prompts one to ask, "If this assumption *were* true, would that be sufficient to explain how to get the outcome we want?"

A proposition is a functional statement of cause and effect. A proposition posits a causal relationship between two constructs. Constructs are different than variables. A variable is measurable. Productivity in the context of an automobile factory can be measured by the variable, "number-of-cars." Productivity in a brainstorming group can be measured in terms of the variable "number-of-ideas." But productivity itself is not a variable, it is an idea.

Propositions always posit that one construct causes another. A construct in a proposition is either a cause or an effect. In Proposition 1 above, the constructs are Effort and Goal Congruence.

There are a number of different ways a given proposition could be stated. The examples below express the same construct with different words:

- Individual effort toward a group goal is a function of goal congruence
- Goal congruence causes individual effort toward a group goal
- Individual Effort is determined by goal congruence
- The more goal congruence, the more effort.

All these phrasings can be summarized by a mathematical function like this:

$$E = f(G)$$

Where:

E = *Individual Effort*
G = *Goal Congruence*

Notice that this very simple expression is not specific about the nature of the function; it does not specify whether it linear, curvilinear, or discontiguous; it does not specify whether the function is bounded or infinite; whether it has constant or variable parameters. Such a young, incomplete theory would no doubt acquire more nuance and complexity as research progressed. Nonetheless, simple though it is, it is still useful to the collaboration technology designer. It suggests that E is a positive function of G, which means if you can figure a way to use technology to increase G, you should get more E as a result.

Theoretical propositions can also be illustrated as simple box-and-arrow diagrams. Figure 1 illustrates three propositions in a simple theory of group productivity. Each box-arrow-box combination constitutes a proposition. The direction of the arrow indicates the direction of causation.

In Figure 1, notice that Proposition 2, the Distraction proposition, posits an inverse relationship rather than a positive relationship. It could be interpreted, "The more distraction a group experiences, the less effort it will make toward a goal." Thus, if you could find a way to use technology to reduce distraction, effort toward the goal should increase, which should in turn increase productivity.

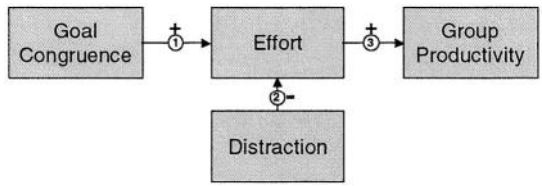

Fig. 1. A box and arrow diagram of several theoretical propositions to explain group productivity. The model posits Productivity as a positive function of goal-directed Effort. It posits Effort as a positive function of goal congruence, and as a negative function of distraction.

Box-and-arrow diagrams provide a way to run the Absurdity Test, a quick, useful way to smoke-test the basic logic of a proposition. Box-arrow-box combinations for improperly framed propositions will yield absurd statements when they are interpreted in the following form:

"The more of <Box X> we have, the more of <Box Y> will result."

For example, consider the two propositions in Figure 2.

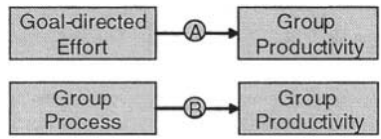

Fig. 2. Two Theoretical Propositions. Proposition A passes the Absurdity Test. Proposition B fails the test.

Proposition A yields a sensible statement when put in the form, "The more goal-directed effort group members make, the more productive the group will be." On the other hand, Proposition B, which has some intuitive appeal, (Group productivity must surely be a function of group process), nonetheless yields an absurd statement when put into the form, "The more process a group has, the more productive the group will be." The Absurdity Test quickly reveals a logical flaw in the proposition, signaling the need for additional attention.

Notice also that the axioms do not appear in the box and arrow diagram. Nonetheless, every proposition is based on one or more assumptions, whether or not they have been articulated. Until the axioms have been teased out and explicitly articulated, their validity cannot be judged, and so the model does not yet fully explain the phenomenon of interest, and the theory is not yet complete.

To summarize, then, a causal theory is a collection of statements that propose mechanisms that could cause a phenomenon of interest. These statements are composed of axioms (assumptions) and propositions (functional statements of cause and effect). They combine to form the logic of a causal theory, making arguments that take the form, "If we assume X, then it must be that Y is a function of Z. The propositions of a causal theory can be illustrated with a box-and-arrow diagram. Each box-arrow-box combination can be interpreted as some variation on the theme, "The more Z you have, the more Y will result."

3 Good Theories – Better Technologies

This section presents three examples that illustrate how a good theory can drive non-intuitive design choices that improve group outcomes.

3.1 Focus Theory and the Brainstorming Feedback Graph

The first example began with work started more than 50 years ago. In 1953, Osborn proposed a new group ideation technique that he called brainstorming. He conjectured that ideation could be improved if people followed a four-rule protocol:

- Do not criticize other people's ideas.
- Be open to wild or unusual ideas.
- Generate as many ideas as you can.
- Build and expand on other people's ideas.

Osborn's reasoning seemed sound, yet twenty subsequent studies were not able to demonstrate that people using the brainstorming protocol produced more ideas than nominal groups (Diehl & Stroebe, 1987; Valacich, Dennis, & Connolly, 1994).

Diehl and Stroebe (1987,1991) unraveled the mystery by demonstrating that brainstorming groups suffer from production blocking, evaluation apprehension, and free riding. A number of subsequent ideation studies demonstrated that production blocking and evaluation apprehension could be overcome by using Group Support Systems (GSS) that allowed participants to contribute their ideas simultaneously and anonymously over a computer network. People using GSS could all contribute to a brainstorm simultaneously, which eliminated production blocking. People using GSS could also contribute to a brainstorm anonymously, which eliminated evaluation apprehension. The results were dramatic; under certain circumstances, people using GSS produced thirty to fifty percent more unique ideas than did people using nominal group technique (e.g., Dennis and Valacich, 1993; Gallupe, Bastianutti, & Cooper, 1991; Gallupe, Dennis, Cooper, Valacich, & Bastianutti, 1992; Valacich, Dennis, & Connolly, 1994). As remarkable as those results appeared to be, the question remained whether they were as good as they could be.

More than one hundred years of social loafing research showed unequivocally that, regardless of task, people who were working anonymously tended to make less effort than people whose contributions were individually identifiable (e.g. Harkins and Jackson, 1985, Kerr and Braun, 1981, 1983). GSS users were working anonymously, which meant social loafing had to be occurring. The question was whether anything could be done about it.

The electronic brainstorming system we used in our research included a feedback graph (Figure 3 (left)). It plotted the cumulative number of contributions the team made to the brainstorm over time. However, research had not shown that use of the graph had any impact on brainstorming productivity. The Focus Theory of group productivity (Briggs, 1994) suggests that effort toward the group goal is a function of goal congruence (the degree to which the private goals of individual members are compatible with the public goal of the group). Social Comparison Theory (Goethels and Darley, 1987) suggested a goal congruence hook that could be invoked by modifying the feedback graph. The theory posited that, all else being equal, people want

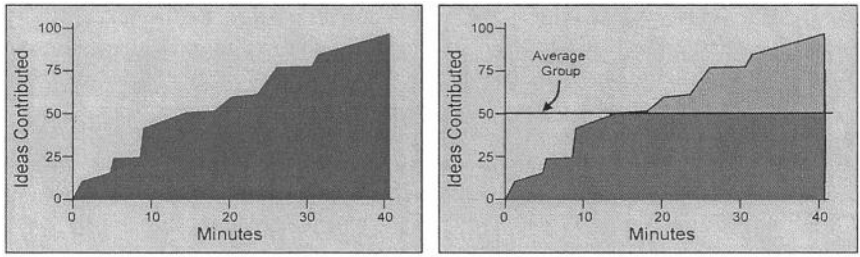

Fig. 3. Left: A feedback graph that provides no basis for social comparison in brainstorming groups. Right: A feedback graph that provides a way to invoke social comparison. Teams that viewed the right-hand graph during brainstorming produced approximately 60% more unique ideas than did teams using the other graph.

the status that accrues from being perceived as contributing fully to a group effort. They want to be seen as stars; they don't want to seem below average. They therefore tend to increase their efforts to at least match the performance of others.

We reasoned that we could add a single horizontal line to the middle of this graph (Figure 3 (Right)), and then tell people, "The average group produces about this many ideas during a brainstorming session... You don't to be below average, do you?" This, we reasoned, should invoke social comparison, causing anonymous brainstorming groups to make more effort, reducing the effects of social loafing. An experiment with 56 groups showed that the groups using the social comparison graph produced about 60% more unique ideas than did the groups using the groups using the standard graph. This was on top of the 50% gain they had already attained by moving from paper to electronic brainstorming. Thus, a good theory led us to a counter-intuitive design choice – the addition of a single horizontal line, which produced a significant improvement in group performance.

3.2 Cognitive Network Model and Creative Problem Solving Techniques

For many years, creativity researchers described creative people, creative environments, creative processes, and creative ideas. This research hinted at, but did not explain what caused creative ideas to emerge in the minds of people working together toward goals. It was not, therefore, possible to predict with confidence whether a new creative problem solving technique or technology might improve creativity. Creativity bordered on a mystical art.

Recently the Cognitive Network Model of Creativity (Santanen, Briggs, and Vreede, 2000) suggested mechanisms of the mind that could give rise to creative ideas. The model drew together standard axioms of cognitive psychology – long-term memory as a web of related concepts , limited working memory, and so on -- to argue that the creative solutions must emerge from novel juxtapositions in working memory of concepts from previously distant parts of the cognitive web. It further argued the number of novel juxtapositions was, among other things, a function of the variety of external stimuli. This theory was consistent with findings that teams using electronic brainstorming technologies could, under certain circumstances, produce substantially more ideas of greater creativity (e.g. Hender, Dean, Rodgers, and Nunamaker, 2002).

The variety of ideas proposed during brainstorming stimulated additional creativity on the part of participants. However, it also suggested that we could make teams more creative if we could find ways to use the collaboration technology that would increase the variety of external stimuli during brainstorming.

In an attempt to use this theory to increase the number of highly-creative ideas a team could produce, we created a new brainstorming approach called Directed Brainstorming. In standard brainstorming, the team responds to a single brainstorming question with a flurry of answers. With Directed Brainstorming, the team receives a stream additional prompts throughout the brainstorming activity. The prompts typically relate to criteria for judging the quality of an idea. For example, for a session on improving factory production quality, the prompts might be:

- *Now give me an idea that would be faster to implement than any you have seen so far.*
- *Now make a suggestion that would be less expensive than those already on the list.*
- *Now think of a concept that would reduce product defects more effectively than any of the ideas we already have.*

A series of experiments showed that, indeed, as the theory suggested, teams using Directed Brainstorming approach produced approximately three times as many unique ideas as those using an un-prompted electronic brainstorming approach, and that among those were a significantly larger number of highly-creative ideas (Santanen, Briggs, and Vreede, 2000). Thus, with a theory to explain our phenomenon of interest, we were able to make non-intuitive choices that carried us well beyond the gains we had already achieved by instinct and experience.

3.3 Technology Transition Model and GSS Architecture

In this case, a small theoretical insight led us to reverse the fundamental assumptions underpinning the architecture of a group support system (GSS), yielding to a new generation of GSS technology. GSS had proven useful in the field, mysteriously, however, despite unequivocal, measurable, profitable results, many GSS facilities fell into disuse within a year or two of being introduced into an organization (Briggs, Vreede and Nunamaker, 2003).

The Technology Acceptance Model (Davis, 1898) posited that people would use a technology to the extent that they found it useful, and that they found it easy to use. The model did not explain, however, why an organization might later discontinue using a technology that still met those criteria. A minor construct of the Technology Transition Model (TTM) (Briggs, et al, 1999) offered an insight that eventually unraveled the mystery. TTM considered ease-of-use in at least three dimensions: perceptual load, access load, and conceptual load. Perceptual load addressed the typical elements of user-friendliness – how easy is it to find and use the features and functions you need. Access load dealt with the fuss-factor – how much effort was required to gain permission and access to the features and functions you need. Conceptual load dealt with understanding what a system is *supposed* to do for the user. Perceptual load for GSS tended to be very low – people with no training could participate in electronic meetings very successfully. Access load was moderate for GSS. People had a

bit of fuss getting signed up for electronic meeting rooms and getting the network going.

However, GSS had very high conceptual load. The purpose of the GSS is to create useful patterns of collaboration, yet there was nothing on the screen of a GSS tool telling a user what pattern of collaboration might emerge if a team were to use the tool in a given way. Further, a given tool in a given configuration could be used to create a wide variety of useful patterns of collaboration. Although it took less than a week to learn how to work the system (low perceptual load), it typically took a year of apprenticeship to learn how to wield a GSS in service of group productivity (high conceptual load).

It was a fundamental assumption of GSS design that facilitators would run GSS workshops on behalf of the team, because the facilitators would know how to use the technology to help a group generate, organize, and evaluate ideas, and make informed decisions. However, successful GSS facilitators tend to be bright, articulate, technically-competent people-persons with a penchant for problem-solving. As such, they tended to get promoted away, leaving behind a GSS facility that others did not know how to use effectively.

TTM suggested that if we could find a way to cut the conceptual load for GSS, the technology might achieve wider, sustained use in the workplace. That led us to the concept of collaborative applications built for specific high-value recurring tasks. We reasoned that if we were to create a GSS application that moved practitioners step-by-step through a software requirements negotiation, the users would have almost no conceptual load. They would not have to guess which tool they needed, they would not need to know how those tools should be configured, nor would they need to know which pattern of collaboration might be useful for a given step. A master facilitator could make those choices at design time. At run time, the participants could simply go to work on one fully configured step, and then move to the next when they were ready.

We piloted the packaged-application approach first with paper-and-pencil methods, and then with guidebooks for how to run a particular task on a general-purpose platform. It was clear that this approach cut conceptual load substantially for practitioners of a recurring process. For the first time we observed non-facilitators who successfully conducted their own GSS processes after only a day or two of training, and who subsequently transferred those processes to others in the same organization as the standard way of doing business.

On the strength of these findings, we set about to create a new generation of GSS technology that would support rapid development of completely packaged step-by-step collaboration processes. As of this writing, the new technology has only been released into the workplace for a matter of months. Early results are promising. but it will be years before we know the outcome. It was a good theory, rather than instinct, that led to this radical shift in GSS architecture. If the logic of the theory holds, then we should see more self-sustaining and growing communities of users for purpose-built GSS applications than we did for the facilitator-conducted general-purpose approach.

4 Theoretical Temptations: Models That Do Not Inform

There is nothing so useful as a good theory. A model of cause and effect can suggest ways to design and use our technologies to cause the effects we need. However, all models are not created equal. Our literature is rife with models that yield no useful insight. Such models are seductive, because on the surface, they seem logical. However, in the end, they cannot drive our design choices for collaboration processes and technologies. This section discusses several classes of models that could tempt us into fruitless efforts.

4.1 Grand Theories of Everything

There is a class of theoretical offerings in our literature that propose a plausible umbrella construct, e.g. Group Process, as influencing many phenomena of interest (Figure 4).

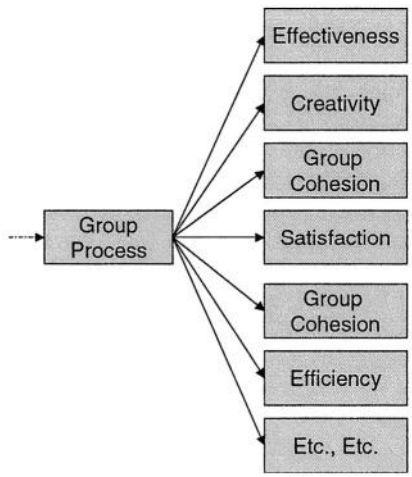

Fig. 4. Grand Theories of Everything posit some plausible umbrella construct as influencing all outcomes. However, such a model suggests no explanation of how to use technology to cause the outcome you need.

It seems logical that group process must, indeed, influence effectiveness, creativity, cohesion, satisfaction. However, the logic of this model breaks down in two ways. Firstly, experience shows that a technology used to improve one outcome will not necessarily improve another. For instance, that which enhances productivity does not necessarily cause creativity (consider the automobile assembly line). And that which produces the highest productivity does not necessarily produce the highest levels of satisfaction (e.g. Connolly, Jessup, and Valacich, 1990). If these outcomes have different causes, then a separate theoretical model is required for each of these outcomes. To produce both outcomes simultaneously, you must understand what causes each, and then seek technical solutions that instantiate the causal mechanisms of both models simultaneously.

Secondly, these models fail the Absurdity Test when one attempts to interpret the box-arrow-box combinations as statements of cause-and-effect, e.g.:

> "The more process a group has, more satisfied the group will be."
> "If we use technology to increase group process, then the group will be more effective, creative, cohesive, satisfied, and efficient..."

A moment of further reflection suggests that at least some of these outcomes may be caused by different mechanisms than others. It may be, for example, that some interventions to enhance efficiency would also enhance creativity, while others would interfere with creativity. Therefore, a single mechanism could not explain both productivity and creativity. A separate theory would be required for each. A model with an outward-fanning peacock-tail of effects should, therefore, at least be subjected to skeptical deliberation before attempting to use it as the basis for a collaborative intervention.

4.2 Grand Theories of Nothing

Other theories in our literature posit general antecedents to a vague abstraction, instead of attempting to explain the causes of a well-defined phenomenon of interest. On the surface, such models may seem plausible at first. However, like the grand theories of everything, these also yield absurd statements when tested, for example:

> "The more knowledge a group has, the more outcomes they will attain."

Models of general antecedents to vague abstractions can be used to discuss group outcomes ranging from brainstorming productivity to nuclear conflict, but they offer no insight about how to cause or prevent any particular outcome. They therefore cannot inform design choices for collaboration technology and process.

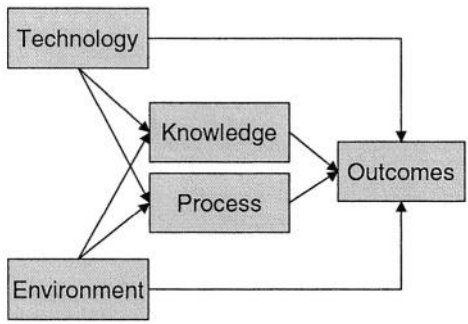

Fig. 5. Grand Theories of Nothing posit general antecedents to a vague abstraction instead of attempting to explain the causes of a well-defined phenomenon-of-interest. Such a model could be applied without fear of repudiation to effects as wide ranging as satisfaction-with-process and all-out armed conflict. However, it cannot articulate the cause of any particular outcome, and therefore cannot inform design choices for collaboration processes or technology.

4.3 Fit and Match Theories

The limitations of the models described in the previous two sections sometimes lead researchers and technologists to yet another blind alley of theory. Experience will reveal the inadequacy of a statement like, "the more technology a group has, the more productive it will become." In practice, sometimes a group with more technology is more productive, but sometimes it is less productive. Those who have the right tools for their task do well, while those who have the wrong task do poorly. It therefore becomes tempting to theorize that the phenomenon of interest (e.g. productivity) must be a function of the match between task and technology. The better the technical capabilities fit the needs of the group, the more successful the group can be. A box-and-arrow diagram of this proposition passes the Absurdity Test that the previous two classes of models fail. The propositions stand up to further thought experiments as a logical, credible assertion.

Match and Fit theories appear in a wide variety of literatures where people consider the use of technology to improve outcomes, and are regularly cited. However, despite their logical consistency, they offer nothing useful to a technology designer. A match theory alludes to patterns of cause-and-effect by implying that an outcome of interest might be caused by one technology but not by another. However, the model stops short of articulating the causes, and sometimes even the effects to which it alludes. It therefore offers the benediction, "Go forth and match," without indicating a basis for creating a match, or for knowing when a match has been achieved.

4.4 Descriptive Attribute, Characteristic, or Factor Theories

There are a number of theories in the literature that posit taxonomies of attributes or characteristics of some object as antecedents to a phenomenon of interest. It is not uncommon, for example to find models arguing that group productivity is influenced by characteristics of the technology, the group, and the environment. There are at least two problems attribute models. First, their propositions do not pass the absurdity test when they are framed as causal statements, for example:

> "The more attributes a group has, the more productive it will become."
> "The more characteristics a technology has, the more productive its users will become."

The logical fallacy of an attribute model is to confuse a category – attributes of the group – with a causal mechanism. Categories are not the same as causes. However, descriptive modes can be useful in a quest for causal mechanism. If research reveals that certain attributes, say, group cohesion and group history, seem to be connected with group productivity, one can ask "Why does cohesion matter? Why does history matter?" Sufficient questioning of this kind can lead to the discovery of underlying patterns of cause-and-effect, making models a useful tool on the quest for good theory.

On their own, however, attribute models suffer another fatal flaw: Infinite decomposability. Any category of attributes can be decomposed into sub-categories. Attributes of the Group, for example, might be decomposed into categories such as group history, group structure, group leadership, and so on. Group structure might be de-

composed into emergent patterns, imposed patterns, and other patterns limited only by the creativity of the researcher. As the models become more elaborated with sub-categories, they become increasingly difficult to wield, but they gain no additional logical utility as causal explanations, so their usefulness for guiding design choices does not increase.

5 Implications for Collaboration Technology Research

5.1 Theory Should Be Technology-Free

In the early 1990's, some in the collaboration technology reported that GSS tended to make people more productive and satisfied, while others reported that GSS made people more productive but less satisfied, and still others reported that GSS made people less productive and less satisfied (Fjermestad and Hiltz, 1998; 2001). None of these claims were justified. Just as Da Vinci's paintbrush could not cause a masterpiece in the hands of an unskilled painter, so, the outcomes of a GSS session depended on how the technology was used. We had the wrong research question. We were asking, "What is the effect of GSS on group productivity and satisfaction?" Our theoretical models dead-ended until we realized that we had to separate our research question into two questions, one scientific and the other engineering, to whit:

> **Science:** "What causes a group to be productive?"
> **Engineering:** "How can we use technology to invoke those causes of group productivity?"

In the quest for good theory to drive our technology design choices, we must guard against letting our technology creep into both our scientific questions and our theories.

If we ask scientific questions that include technology, we may attempt to build theories that include technology. If we build theories that include technology, then our theories will become obsolete when the technologies become obsolete. Further, our theories will not be able to explain the effects of other technologies, nor to suggest how to change the technology in the theory so as to enable even better outcomes. They will dead-end with the exact technology they embrace. However, if we understand what causes the phenomenon of interest, then we can think about ways to use technology to improve the outcomes we want, and to block the outcomes we do not want.

5.2 Change Research Focus from Technology to Technology-Supported Process

Scientific questions and theories that include technology also lead to unwarranted overgeneralizations that hinder technological progress. Our conclusions early nineties – that GSS caused productivity and satisfaction to increase (or decrease) – overlooked two key points:

a) Any technology that can be used in ways that cause good outcomes can also be used in other ways that cause poor outcomes. (Consider Da Vinci's paintbrush.)
b) An instance of technology is not the same as a class of technologies. Da Vinci cold not accomplish the same effects with a frayed stick that he could with a sable brush, although both could be classified as paintbrushes. In the same fashion, it may not be possible to create the same effects with a one-page brainstorming tool as with a multi-page brainstorming tool, although both could be properly classified as GSS.

Thus, we can never be justified in drawing conclusions about a technology apart from the work-process in which it is embedded. Nor can we defensibly draw conclusions about a class of technologies based on an instance of that class. We can only conclude,

> "When this specific instance of a technology is used in this particular work-process, it produces better (or worse) results than does a different combination of instance and work-process."

It may therefore important for collaboration technology researchers to quickly shift the focus of their research from collaboration technology to technology-supported-collaboration-processes. At this level, collaboration technology researchers can make valid comparisons that produce useful results which can lead to justifiable conclusions.

6 Conclusions

By driving our designs with rigorous theoretical models of cause-and-effect, the field of groupware technology can advance far beyond its already valuable achievements. If we understand the mechanisms that cause our phenomena of interest, we can use a technology in ways to deliberately cause better (or worse) outcomes. If we understand nothing of the causal mechanisms, then we can only achieve a given outcome by accident at first and by rote thereafter. Good theory can make us appear as wizards, able to make otherwise-unexpected design choices that yield better outcomes for our teams. With good theory we may be able to understand other choices don't work out as expected. The key value of the theory driven approach to groupware design derives from the way it discipline our thinking. When we clearly articulate the assumptions and logic that give rise to our technology choices, we will see more clearly than when we do not. We will discover possibilities we never considered. Sometimes it will turn out that our theories are flawed. Sometimes we will make flush out many bad ideas before they consume time, money, and passion.

By embracing technology-driven design, perhaps we can move groupware research from an art toward a science, toward a repeatable engineering practice.

References

1. Agres, Andres; Vreede, G.J.; and Briggs, R.O. "A Tale of Two Cities: Case studies of GSS Transition in Two Organizations." 37th Hawaii International Conference on System Sciences. IEEE. Kona Hawaii, January 5-8, 2004. On CD.

2. Briggs, R.O.: The Focus Theory of Group Productivity and Its Application to the Design and Deployment of Collaboration Technologies. Doctoral Dissertation, University of Arizona, (1994)
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. Briggs, R.O., Adkins, M., Mittleman, D., Kruse, J., Miller, S., & Nunamaker, J.F., Jr.: A Technology Transition Model Derived from Field Investigation of GSS use aboard U.S.S. CORONADO. Journal of Management Information Systems. Winter, 1998-99, 15(3), (1999) 151-193
5. Connolly, T., Jessup, L.M., & Valacich, J.S.: Effects of anonymity and evaluative tone on idea generation in computer-mediated groups. Management Science, 36 (6), (1990) 689-703
6. Davis, F. D., Bagozzi, R. P. & Warshaw, P. R.: User acceptance of computer technology: a comparison of two theoretical models. Management Science, 35(8), (1989) 982-1003
7. Diehl, M., & Stroebe, W.: Productivity loss in brainstorming groups: Toward the solution of a riddle. Journal of Personality and Social Psychology, 53, (1987) 497-509
8. Diehl, M., & Stroebe, W.: Productivity loss in idea-generating groups: Tracking down the blocking effect. Journal of Personality and Social Psychology, 61, (1991) 392-403
9. Fjermestad, J., Hiltz, S.R.: An assessment of Group Support Systems experimental research: methodology and results. Journal of Management Information Systems, 15(3), (1999) 7-149
10. Fjermestad, J., Hiltz, S.R.: Group Support Systems: A descriptive evaluation of case and field studies. Journal of Management Information Systems, 17(3), (2001) 112-157
11. Gallupe, R.B., Bastianutti, L.M., and Cooper, W.H.: Unblocking brainstorms, Journal of Applied Psychology, 76(1), (1991) 137-142
12. Gallupe, R.B., Dennis, A.R., Cooper, W.H., Valacich, J.S., Bastianutti, L.M., & Nunamaker, J.F.: Electronic brainstorming and group size. Academy of Management Journal, 35, (1992) 350-369
13. Goethals, G.R., Darley, J.M.: Social Comparison Theory: self-evaluation and group life. In Mullen, B., Goethals, G.R., (eds.), Theories of Group Behavior, New York: Springer-Verlag, (1987) 21-48
14. Harkins, S.G., Jackson, J.M.: The role of evaluation in the elimination of social loafing. Personality and Social Psychology Bulletin, 11, (1985) 457-465
15. Hender, J.M., Dean, D.L., Rodgers, T.L., & Nunamaker, J.F., Jr.: An examination of the impact of stimuli type and GSS structure on creativity: Brainstorming versus non-brainstorming techniques in a GSS environment, Journal of Management Information Systems, 18(4), (2002) 59-85
16. Kerr, N.L, Bruun, S.E.,: Ringlean revisited: Alternative explanations for the social loafing effect. Personality and Social Psychology Bulletin, 7, (1981) 224-231
17. Kerr, N.L, & Bruun, S.E.,: Dispensability of member effort and group motivation losses: Free-rider effects, Personality and Social Psychology Bulletin, 44, (1983) 78-94
18. Post, B. Q.: Building the Business Case for Group Support Technology. Proceedings of the 25th annual Hawaii International Conference on Systems Science, IEEE, (1992) 34-45.
19. Santanen, E.L., Briggs, R.O., Vreede, G.J. de,: The Cognitive Network Model of Creativity: A New Causal Model of Creativity and a New Brainstorming Technique. Proceedings of the 33rd Hawaii International Conference on Systems Sciences. (2000)
20. Valacich, J.S., Dennis, A.R., Connolly, T.: Idea generation in computer-based groups: A new ending to an old story. Organizational Behavior and Human Decision Processes, 57, (1994) 448-467

This page intentionally left blank

Divergence Occurrences in Knowledge Sharing Communities

Alicia Diaz[1,2] and Gerome Canals[2]

[1] Lifia, Fac. Informatica-UNLP, CC 11, 1900 La Plata, Argentina
alicia@sol.info.unlp.edu.ar
[2] Loria, Campus Scientifique, B.P. 239, 54506 Vandoeuvre-les-Nancy cedex, France
canals@loria.fr

Abstract. While knowledge-intensive communities are actively interacting, divergent knowledge positions appear as a natural consequence of the knowledge-sharing activity. Although this feature can look like an unfavorable situation, we argue that maintaining the coexistence with conflicts and following their evolution allows the community to understand how new knowledge emerges. In this paper, we discuss the knowledge sharing process where divergences occur and we propose a technological approach that allows communities to coexist with conflicts.

1 Introduction

This paper seeks to present an approach to supports divergent knowledge positions in the context of a knowledge intensive community that collaboratively develop it own memory.

Communities of practice have gained a particular interest in Knowledge Management due to their knowledge-intensive nature. Communities of Practice, as Wegner states in [9], are groups of people who share a concern, a set of problem, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis.

People find value in meeting this kind of communities because they become bound by the value they find in learning together. Because of their knowledge sharing activity, communities accumulate knowledge and develop a unique perspective on their topic as well as a body of common knowledge, practices, and approaches. However, before reaching a unique perspective, divergent positions appear as a natural consequence of the act of sharing knowledge. Divergence means the generation of alternatives, arguments and different point of views about a topic of interest. Divergences are generally considered as conflicts at the common understanding. The community can take different decisions to solve the conflict, but we are more interested in the situation where the community coexists with the conflict. In spite of this situation can be seen as unfavorable, it exactly describes how the agreed knowledge naturally emerges in the community by the simple act of sharing knowledge. Although the achievement of a consensus may or not happen, the main thing is the process that takes place while

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 17–24, 2004.

the community persists with a conflict. This process represents the discussion in which the participants are involved. It is an evolutionary process based on sharing knowledge.

From a general point of view, our approach centers on community technological supports that allow communities to accumulate knowledge at the same time they suitably share knowledge, and we specially focus the attention on allowing the community to deal and coexist with conflicts while it shares knowledge. In particular, we conceptualize this problem in a community that collaboratively develops its own knowledge repository. We also put forward a knowledge sharing workspace that allows: knowledge externalization through representing the shared knowledge by ontology formalism [6]; representing private and shared knowledge context, supporting publishing through bringing a contribution from the private to the shared workspace; and facilitates to express divergences and follow the discussion thread.

This paper is organized as follow. In section 2, we discuss the knowledge sharing activity paying special attention to the knowledge sharing process and identifying the nature of the shared knowledge. Next, in section 3, in the context of the knowledge sharing process, we introduce the problem of divergence appearance. Finally, in section 4, we will describe our approach for supporting knowledge divergences.

2 The Knowledge Sharing Activity

To achieve to a suitable support of divergence occurrences in a knowledge sharing community, first of all it is important to have a complete understanding of what is the knowledge that the community shares and a significant attention must be paid to the process that allows communities to share knowledge in a coherent manner. In the following, we will briefly describe the kind of knowledge the community share and we will analyze the features of the process throughout the community shares knowledge.

The Shared Knowledge. The shared knowledge is the knowledge that the community accumulates while the knowledge sharing process takes place and it represents the community's common understanding. The nature of the shared knowledge is varied. Communities do not only accumulate knowledge about a topic of interest, they also share knowledge about who are participating in the community, who knows what, who are interested in, level of expertise, perspectives, and more. All of them together are the shared knowledge. We classify the shared knowledge in:

Domain Knowledge. This is knowledge about the domain of interest or competence. It consists of conceptual elements and facts that conceptualize the domain. Community's domain knowledge also represents the consensual knowledge and the shared common language.

Social Knowledge. This is knowledge about members and their organization. Member's knowledge is knowledge about who is each member and their relationships. Members can be individuals or groups.

Members Profile. This knowledge describes interests, capabilities, and expertise of the community members. This knowledge is described in terms of the relationships that exist between the knowledge and people.

Community's knowledge could be seen as conceptual network that is made up of conceptual knowledge artifacts (domain and members knowledge) linked by associations between knowledge and people expressed by member profiles.

The Process. The knowledge sharing process is an iterative and incremental process, similar to the one described by Nonaka in [7], where knowledge goes emerging in each cycle. Knowledge sharing involves from the community point of view, individual and collaborative learning and from the knowledge point of view, knowledge evolution. This process begins when one member contribute with some knowledge, and continues when other members realize this, and begin to contribute with comments, and additional information that allows the community to have a more complete idea of the subject of the initial contribution.

The knowledge sharing process consists of four steps, externalization, submission, internalization and reaction.

Externalization means to make explicit some knowledge. Externalization is a private activity, which is carried out it in isolated manner at the individual knowledge context. Some knowledge representation system it is needed to make explicit the private knowledge. This knowledge representation can be informal or formal, going to informal systems (emails or document writing) to semi-formal systems that mixes a formal and a informal system classifying document in based on a ontology; or even to formal systems to develop a formal specification (using ontologies to design a knowledge conceptualization).

Submission/Publication is the act of making public some knowledge. Submission means to transfer some knowledge from the individual knowledge context to the community knowledge context. Publication has externalisation as pre-condition. The submitted element generally is called a contribution and the submitted knowledge is named the contribution subject. Communities can use different media for publishing their knowledge.

Internalization is an individual process, which takes place when someone realises and appropriates a new contribution - individual learning. At this moment, the contribution subject becomes part of the individual knowledge context. Internalization it is not easy to detect, but we can say that internalization took place if a reaction was manifested.

Reaction is the act of giving some kind of response to a contribution. Any reaction is an externalisation of an individual position in face of a new contribution. Reaction always gives an "augmented" version of the original knowledge subject because it is improved with new knowledge and even new point of view. Reactions are interesting to observe because they imply that internalization has taken place.

Although the four steps of the knowledge sharing process are interesting, we will pay a special attention to submission and reaction because they are the key to maintain the community learning together. Meanwhile the community is sharing knowledge; its knowledge context is constantly growing and evolving.

Each new contribution to the community knowledge is a step forward to a new community knowledge state.

3 Divergences in the Knowledge Sharing Process

While knowledge-intensive communities are actively interacting, divergent knowledge positions appear as a natural consequence of the knowledge-sharing act. In knowledge sharing communities, it is not so realistic to think that everybody is agree with everything that is told; whereas, it is very often to observe people that express different positions or argumentations in the context of the same knowledge subject. Therefore, to coexist with knowledge divergences is very natural in any knowledge intensive community.

Occurrences of divergence are consequence of reaction, where each contribution by reaction represents an "augmented" version of a initial contribution. Reactions always are tied to an initial contribution. A sequence of reactions corresponds to a sequence of contributions triggered by an initial contribution. This sequence begin with an initial contribution and follows by a set of contributions by reaction.

Adapting the Ibis model [3] to our needs, we define different kinds of contributions by reaction: complementary contribution, alternative contribution and argumentations.

- *Complementary contributions* always add more knowledge to the original one and do not imply any divergence.
- *Alternative contributions* are contributions created with the intention of replacing the original one. They introduce another point of view on the knowledge subject. An alternative contribution manifest always conflicts.
- *Argumentations* give a personal opinion that supports or object any given contribution. Argumentations are always attached to some contribution.

Contributions are organized in the *discussion thread.* A sequence of contributions, triggered by an initial contribution, represents the discussion thread at one particular moment in the knowledge sharing process. Discussion threads represent the history of the reactions tied to an initial contribution. Threads act as the continuous link of the discussion. As reaction can occur over any kind of contribution, we define the thread of discussion as an aggregation of complementary and/or alternative contributions. Alternative contributions correspond to different branches in the thread structure. Each branch can be seen as a sub-thread of the original contribution. The discussion thread also holds the argumentations that are attached to contributions. Therefore, a thread looks like a tree where the root represent the initial contribution an each branch represents an alternative in the knowledge discussion. Although given the thread definition allows one to imagine the thread structure as a deep tree it is not so realistic to think that in the real life the thread structure can grow in depth so much, because of going in depth in the tree means to follow the discussion on a subject that has not be reached by consensus.

4 Supporting Divergence in a Knowledge Sharing Workspace

There are many approaches to support knowledge sharing. Wenger enumerated in [9] many technologies already used by on-line communities like home pages, on-line discussion groups, collaborative-shared workspaces, document repositories. There are also many technologies for supporting community's knowledge sharing where the community develops it own group memory. However, there are not many systems explicitly oriented to communities of practice that support divergence. Existing ones only focus on one or more aspects of the whole picture. For example, systems based on Ibis model, like G-Ibis [2] and currently Questmap [3] or even, WebGuide [8] may be considered as an approximation to this problem, but they emphasized more in modeling the discussion, that in supporting suitably the occurrence and evolution of divergences.

In particular, we are interested in those that allow the community to develop its own community's memory with the capability of expressing divergences. The community memory is a knowledge repository where the shared knowledge is stored. For achieving this, the community needs a knowledge sharing workspace that supports the knowledge sharing process with following requirements:

- *Knowledge representation formalism.* For externalization, it is mandatory to define a mechanism that allows one to make explicit the knowledge. This formalism is embedded in the knowledge sharing workspace and define the type of the allowed actions.
- *Representation of private and shared knowledge context.* People need to differentiate between private and shared knowledge. Knowledge externalization is a private activity, whereas publication and community's memory development are public activities.
- *Knowledge discussion thread.* The discussion thread is the result of expressing conflicts. Conflicts are characterized depending on the knowledge representation formalism, therefore the knowledge sharing workspace needs mechanisms that facilities appropriately the expression of them.
- *Discussion awareness.* Internalization facilities are needed to have a suitable awareness about knowledge changes and discussion evolution.

In our approach, we focus in a community technological support that allows a community to share and make explicit its accumulated knowledge. In particular, we conceptualize this problem in a community that collaboratively develops its own memory through the design of the ontologies [6] that represent the shared knowledge. Following we will discuss this requirement in the context of a community that develop its memory where knowledge is represented by ontologies.

4.1 Supporting Divergence in a Collaborative Ontology Design

Currently, there are some approaches for design ontologies collaboratively, like Protege approach [5], but they lack of facilities to represent the private and public

knowledge context and to follow the knowledge discussion and its evolution. Here we will present our approach that take into account the requirement presented above.

Using Ontologies for Externalizing Knowledge. Although there are different systems to represent the knowledge, we had chosen ontology formalism, because ontologies allow developing a conceptualization of the domain of interest and describing common language among communities' members. Then, this knowledge can be browsed, queried and used to make deduction about the community shared knowledge.

When the community externalizes its knowledge with ontologies, it makes a conceptualization of its shared knowledge. This conceptualization is based on objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that are held among them [1]. A conceptualization is an abstract, simplified view of the world, which is be specified for some purpose.

Ontologies to represent any shared knowledge, but the community only focuses on the collaborative developing of the *domain knowledge ontology,* since it is the knowledge the community must to conceptualize. Whereas, other knowledge, like social or member profile, has already a predefined conceptual level. Therefore, the domain ontology is the result of the collaborative processes to build the community memory. This ontology is the core of the community's memory, and it is the shared objects among community's members while they share knowledge.

Therefore, a contribution, in the knowledge sharing workspace is a conceptualization of the knowledge subject externalized in terms of ontology primitives, that we will call ontological contributions.

Knowledge Sharing Workspace. Knowledge sharing workspace consists of two workspaces: a private knowledge workspace and a shared knowledge workspace, where participants can alternate between both of them.

The *private knowledge space* is a non-public space that is only accessible by its owner and is useful to represent the private knowledge context and allows participants to externalize any knowledge in a private fashion. Private knowledge is also articulated with personal view of the shared knowledge space. A private knowledge becomes public by publishing it from the private to the shared workspace.

The *shared knowledge workspace* is a public space that is accessible to any community member and is useful to represent the shared knowledge context. It holds the shared knowledge. It allows user to publish some private externalization as any contribution type, and beginning or following a discussion thread.

Knowledge Discussion Thread. As ontologies are use to externalize the knowledge, in the following, we discuss how the collaborative ontology development is carried out on the top of the shared knowledge workspace. In particular,

we explain how divergences are manifested in terms of ontologies and when they happen.

The collaborative ontology development occurs through the edition of ontologies at the private workspace (knowledge externalization), and their publication at the shared workspace. Edition of ontologies occurs at private level and is carried out through directly manipulation of ontology primitives; whereas, ontological contribution means to publish a piece of an ontological representation of the knowledge that is held at private knowledge context.

A new contribution has to be *compatible* with the shared version; otherwise, it may be a potential conflict. Conflict may appear when there are at least two versions of the same shared-knowledge, this means there is a private version that is not compatible with the shared version, and it would be published.

In order of detecting conflicts we have follow a conflict detection approach that categorizes the edition operations. This categorization is based on a set of conservation rules. These rules allows us to determine if an edition action can provoke a conflict, if it is the case, participants must publish it as an alternative contribution. Notice that this rules are dependent on the knowledge representation system, each knowledge representation system has its own set of rules.

Therefore, an ontological contribution, in the context of the discussion thread, can be a complementary conceptualization of an initial ontological contribution if it does not provoke any conflicts, or an alternative conceptualization to an existing one if it provokes conflicts. Arguments that support or object some previous ontological contribution are also part of the discussion thread.

Discussion Awareness. In short, we can say the activity on a community is summered to externalizations and contributions, but it is also important to maintain the context where they take place, since it defines if it is a reaction. To determine if a contribution is a reaction is very useful since it gives information to follow the discussion thread, because of identifying if a contribution follows or not discussion thread allows providing the users with awareness information about the flow of the discussion.

There are some cases where to identify if a contribution is a reaction is very easy because it is explicitly expressed (for example the user decides to submit them as an alternative contribution when the system detect the conflict). However, there are other cases where to determine the occurrence of reaction can be more complicated are not explicit like complementary contribution.

We propose the design of a thread manager component that allows users to act more free without managing the discussion thread but feeling it. To reach this goal is necessary to determine the contribution context, this means to understand if a contribution is a trigger of a new thread or not. For determining if a new contribution is a trigger or not, it is necessary to understand if it is related to a previous contribution in terms of: the involved knowledge element (it touches some of the more recent contributions, the performer (it was carried out by the same member), the type of contribution (argumentations are always attached to a previous contribution), or may also be the submission/publication time.

5 Conclusions and Future Works

In this paper, we have introduced the problem of sharing knowledge in a knowledge intensive on-line community, and in particular, we have centered on the problem of divergence occurrences. We have introduced technological requirements to support to develop the community memory that support conflict expression. In particular, we conceptualize this problem in a community that collaboratively develops its own knowledge repository through the collaborative design of the ontology. On the top of a collaborative workspace for developing a knowledge repository, we put forward a knowledge sharing workspace that allows knowledge externalization through representing the shared knowledge by ontology formalism; represents private and shared knowledge context; and where is able to alternate between them; supports publishing through bringing a contribution from the private to the shared workspace; and facilitates to follow the discussion thread.

At the moment of writing this paper we are implementing a prototype system on the top of Protege, that supports previous requirement. Next steps, it is evaluate the usability of the system.

References

1. Chandrasekaran, B., Josephson, J. and Benjamins V. What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems, Vol. 14, No. 1, (1999) 20-26
2. Conklin, J. and Begeman, M.L. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transactions on Office Information Systems, 4, 6, 1988, 303-331
3. Conklin, J. Selvin, A. Buckingham Shum, S. Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS. Proceedings of Hypertext'01 Conference (2001)
4. Diaz, A., Canals G. Improving CoP Knowledge Sharing: a CSCW Approach Based on Awareness. Caise'03 Forum, Velden / Kangefurt, Ostrich, (2003)169-172
5. Grosso, W., Eriksson, H., Fergerson, R., Gennari, J., Tu, S., Musen, M. Knowledge Modelling at the Millenium: the design and evolution of Protégé-2000, Technical Report SMI-1999-0801, Stanford Medical Informatics, Stanford University (2000)
6. Gruber T. R. Toward principles for the design of ontologies used for knowledge sharing. In Formal Ontology in Conceptual Analysis and Knowledge Representation, edited by Nicola Guarino and Roberto Poli, Kluwer Academic Publishers (1994)
7. Nonaka, I. A dynamic theory of organizational knowledge creation. Organization Science, 5(1)(1994)14-37
8. Sthal, G. WEBGUIDE: Guiding Collaborative Learning on the Web with Perspectives. AERA '99, Montreal, Canada (1999)
9. Wenger E., McDemott R., Snyder W.: Cultivating Communities of Practice. Harvard Business School Press (2002)

On the Convergence of Knowledge Management and Groupware

Sajda Qureshi[1], Vlatka Hlupic[2], and Robert O. Briggs[3]

[1] Department of Information Systems and Quantitative Analysis
College of Information Systems &Technology
University of Nebraska Omaha, The Peter Kiewit Institute
Omaha, NE 68182-0392
squreshi@mail.unomaha.edu
[2] Brunel University, Department of Information Systems and Computing
Uxbridge, Middlesex UB8 3PH, UK
Vlatka.Hlupic@brunel.ac.uk
[3] Delft University of Technology, The Netherlands
CMI, University of Arizona, USA 85721
bbriggs@groupsystems.com

Abstract. As successful organizations recognize that they need to convert their intellectual resources into customized services, the value of electronic collaboration has increased. Current efforts in managing knowledge have concentrated on producing; sharing and storing knowledge while business problems require the combined use of these intellectual resources to enable organizations to provide innovative and customized services. This paper argues that knowledge management and collaboration have common, mutually interdependent purposes and practices. It develops a framework that demonstrates this interdependence, mapping collaboration technologies to knowledge management activities. It concludes with a call for the convergence of these two streams for the benefit of researchers, practitioners, and organizations.

1 Introduction

Organisations increasingly see their intellectual assets as strategic resources that must be harnessed and managed effectively to achieve competitive advantage and to survive. An organisation's intellectual assets consist of the knowledge held in the minds of its members, embodied in its procedures and processes, and stored in its (non)-digital media that could be useful for achieving its strategic ends [26]. It is or the sum of everything everybody in a company knows that gives it a competitive edge [22]. With its strategic intellectual resources, an organisation can minimise its costs, create innovative products, improve production procedures, improve quality, respond to dynamic market conditions, and improved customer service.

The effective performance and growth of knowledge intensive organisations requires integrating and sharing knowledge that is otherwise highly distributed [26]. Distributed knowledge is often personalised, residing in isolated pockets and communities within and outside of the organisation. It has been suggested that problems which stem from traditional business environments that hoard knowledge is an obstacle which is preventing knowledge management efforts from being a complete suc-

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 25–33, 2004.

cess [11]. In addition, Vance [24] suggests that the reason information and knowledge may not be easily transferred from the holder to the person needing it may be because much of it is tacit, ingrained in the holder's mind, but difficult to articulate.

Nunamaker *et al.,* [16] and Qureshi *et al.,* [18] suggest that an organisation's potential to create value through the use of its intellectual capital is a function of the extent to which people can understand data, information, knowledge, and wisdom, and can collaborate. Technologies for knowledge management may enable improved capture and conveyance of understanding that might otherwise be inaccessible in isolated pockets; technologies collaboration may enable communication and reasoning among people who use knowledge to create value. It may therefore be that a convergence of knowledge management and collaboration technologies could increase an organizations ability to create value with knowledge.

Developments in collaborative technology are increasingly focusing on technology for geographically distributed teams. This means that instead of bringing groups together in a meeting room equipped with computers, people can accomplish some kinds of tasks online in virtual workspaces. This type of electronic collaboration has become a powerful means of capturing, exchanging, exploiting, and managing knowledge. In this way, electronic collaboration becomes instrumental in capitalising on an organisation's intellectual capital.

This paper is to bring together key concepts in groupware and knowledge management to develop a framework that demonstrates the interdependence of knowledge management and collaboration. The framework may also provide guidance to researchers and practitioners seeking to tap into diverse, yet disparate knowledge resources. This framework may be a useful basis for developing or choosing groupware tools specifically designed for knowledge management activities.

2 Methodology

To develop the framework, we examined groupware literature: 1) that had well-grounded theoretical foundations; 2) that had rigorous empirical findings, and 3) that explored pragmatic, practical organizational applications. We analyzed the knowledge management literature to identify the basic activities to which knowledge management technology was applied. We then mapped modes of collaboration to the knowledge management activities, and mapped collaboration onto the intersections of collaboration and KM to illustrate how these technologies might improve KM performance. We focused on practical application of groupware and knowledge management activities; philosophical and theoretical notions of knowledge were beyond the scope of this paper.

3 Knowledge Management in the Context of Collaboration

The concept of knowledge management (KM) is still in its formative stages. Until recently, the KM literature tended to have strongly technical focus [23] More recently, researchers have begun to focus not only on knowledge management (KM) technology [e.g. 2,7, 17,20], but on the human practices and activities of knowledge management. [e.g. 9,10].

At this point, different KM authors use the same terms to label different concepts, and different labels for the same concept. Nonetheless, common understandings of

KM activities have emerged. For example, according to Angus and Patel [1], *knowledge gathering* refers to:

- bringing in of information and data
- organising related concepts to ensuring that the knowledge is easily accessible by giving it context through linking items to subjects
- adding value to knowledge by identifying new relationships, abstracting, synthesising and sharing.

Yet Kramer [13] limits the concept *knowledge gathering* to the process of collecting knowledge, and posits *knowledge organizing* as a separate concept that involves classifying knowledge to give it meaning so that it can be easily located by those searching for it. Kramer [13] defines *knowledge distribution* as yet another separate KM activity. Nonetheless, there is significant overlap in the concepts identified by these and other authors. Table 1 synthesizes generic KM activities from exemplars in the literature on KM practices. These are defined as follows:

- **Create.** Develop new understandings and procedures from patterns, relationships, and meanings in data, information, and prior knowledge.
- **Collect.** Acquire and record knowledge in a medium from which it can later be retrieved.
- **Organize.** Establish relationships among knowledge items through synthesis, analysis, generalization, classification, or affiliation. Create context so that collected knowledge can be easily accessed and understood by those who need it.
- **Deliver.** Grant access to people who should have access to knowledge, while blocking access to those who should not. Search for and share knowledge. Present knowledge in a medium and format that minimises cognitive load while maximising understanding of those who need it.
- **Use.** Bring knowledge to bear on a task that creates value for an organization.

Table 1. Knowledge Management Activities identified in KM literature

Source	Knowledge Management Activities				
Synthesis of Literature	Create	Collect	Organize	Deliver	Use
Ruggles [19]	Generation	Codification		Transfer	
Angus and Patel [1]		Gathering	Organizing/ Refining	Disseminating	
Kramer [13]		Gathering	Organizing	Distributing	Collaboration
Ferran-Urdaneta [6]	Creation		Legitimisation	Sharing	
Jackson [12]		Gathering /Storage	Synthesis	Dissemination	Communication
Macintosh [14]	Developing	Preserving		Sharing	

The KM literature tends to conceive of the activities in Table 1 in terms of individuals interacting with a KM system. Yet, each of the activities corresponds closely to activities used by teams to achieve their mutual goals. It is therefore likely that collaboration could improve KM activities, and that KM could improve collaboration activities. Indeed, it may be that collaboration activities and KM activities are the same.

Nunamaker *et al.* [16] suggest that *"we are moving towards an age of any time any place collaboration"*. Fuelled by the exponential growth of the Internet, the World Wide Web, and local area networks, there are various communication technologies that enable this flexible form of collaboration. These include combinations of electronic mail, real time conferencing, and multicast audio and video used to support, for example, internet-based concerts and presentations [8, 21].

Any time any place collaboration can also be achieved through information sharing technologies such as digital whiteboards, computer bulletin boards and threaded discussion groups, document management systems that provide for the creation and reuse of documents as well as the control of access, concurrency, and versioning [5,25]. Such suites of collaborative technologies are now in use in organizations and universities around the world. Such advanced collaboration environments can be used for multiple tasks that cross temporal, spatial and social distance.

4 Groupware for Knowledge Management Activities

The generic activities of knowledge management are closely intertwined with collaboration concepts. Briggs & Vreede, [4] argue that regardless of task, there are five patterns of collaboration that generally characterize a team interactions:

1. *Diverge*: To move from having fewer concepts to having more concepts. The goal of divergence is for a group to identify or create concepts that have not yet been considered. The Generate KM activity would be an instance of divergence.
2. *Converge:* To move from having many concepts to having a focus on, and understanding of, fewer concepts worthy of further attention. The goal of convergence is for a group to reduce their cognitive load by reducing the number of concepts they must address. The Gather KM activity would be an instance of convergence.
3. *Organize:* To move from less to more understanding of the relationships among the concepts. The goal of organization is to increase understanding reduce the effort of a follow-on reasoning. The Organize KM activity is an instance of such a process.
4. *Evaluate:* To move from less to more understanding of the benefit of concepts toward attaining a goal, considering one or more criteria. The goal of evaluation is to focus a group's discussion or inform a group's choices.
5. *Build consensus:* To move from having less to having more agreement among stakeholders on courses of action. The goal of consensus building is to let a group of mission-critical stakeholders arrive at mutually acceptable commitments.

There is substantial correspondence among the patterns of collaboration identified by Briggs and Vreede [4] and the KM activities identified in this paper. *Diverge* has to do with brainstorming and idea generation, and corresponds closely with *Create,* which has to do with generating new knowledge. The *Converge* and *Organize* collaboration patterns correspond closely to the *Organize* KM activity. The *Evaluate* and *Build Consensus* patterns may be part of the *Organize* activity, and would clearly be part of the *Use* activity, as would all the other patterns.

Briggs [3] argues that there are three cognitive processes underlying group interactions: communication, deliberation, and information access. Communication refers to conceiving, delivering, receiving, and decoding communication stimuli. These same cognitive processes underlie the Deliver KM activity. Deliberation refers to goal-directed thinking and reasoning, which is congruent with the Organize and Use KM activities. Information access refers to finding, acquiring, and coming to understand

information. This is the essence of the Gather and Organize, and Deliver KM activities. Given that KM technology is mean to communicate knowledge and information in support reasoning

Nunamaker et al. [16] suggest that there are three modes of collaboration that may be made more effective through the use of various collaborative technologies: 1) *collective effort,* people work on their own and group productivity is simply the sum of individual efforts; 2) *coordinated effort,* people make individual efforts, but productivity depends on the level of individual effort and on the coordination among those efforts; 3) *concerted effort* all members must make their effort in synchrony with other members and the performance of any member directly affects the performance of the other members. The generic KM activities can be conducted in all of these modes. Further, people working in all these modes require knowledge to support their deliberations that they could derive from KM technology. Support for co-ordination among individuals carrying out a collaborative work process requires a different combination of technologies than do concerted collaboration efforts. Collaborative (or group) task is defined as the behaviour requirements needed to accomplish stated goals (i.e. create value), via an explicit process, using given information [27].

5 A Framework of Interdependence for KM and Collaboration

Various authors have suggested taxonomies for the classification of groupware applications and products; see e.g. [5,8,15]. This section presents a framework based on those taxonomies to demonstrate the interdependence of KM and Collaboration. The framework achieves this purpose by mapping collaboration technologies to KM activities against several dimensions of collaboration. Because the framework is multidimensional, it is presented here as a series of tables.

Table 2 maps collaboration technologies to KM activities by mode of collaborative work. The first column of the table lists the five generic KM activities synthesized from the KM literature in Table 1. The top row of the table lists the three modes of collaboration. Each cell of the table contains exemplars of one or more collaboration technologies that that could be used by teams in the work mode represented by the column to support the KM activity represented by the row.

Table 2. Collaboration technologies mapped to KM activities by Collaborative Work Mode

KM Activities	Collaborative Work Mode		
	Collective	Coordinated	Concerted
Create	Individual Productivity Suites (e.g. MS Office)	Web-based forms	GSS brainstorming and convergence tools
Collect	Individual productivity suites (e.g. MS Office), team document repositories	Team Database, Forms, GSS discussion tools, Team Calendaring, Project management	GSS brainstorming, discussion, and convergence tools, Simulation Modelling
Organize	Statistical analysis packages, spreadsheet, database queries.	GSS Classification & outlining tools, team calendaring, Project Management	GSS Classification and outlining tools
Deliver	Team document repositories, shared workspaces	Multi-user databases, notice boards, newsgroups, E-mail, shared workspace	GSS, on-line discussion tools, simulation and modelling tools
Use	All of the above	All of the above	All of the above

Table 3. Collaboration Technology for KM activities organized by pattern of collaboration

KM Activities	Diverge	Converge	Organize	Evaluate	Build Consensus
Create	GSS Brainstorming Tools; Online News Group; E-mail	GSS classification tools; Electronic polling tools	Shared Outlines, GSS classification tools	Online Polling tools; Structured discussion tools	Electronic polling tools, GSS assumption surfacing tools
Collect	GSS Brainstorming tools, Online database forms	GSS classification tools, electronic polling tools	GSS outlining tools, GSS classification tools	Online polling tools, structured discussion tools	GSS polling & discussion tools used with goal alignment & conflict resolution methods
Organize	Statistical analysis packages, spreadsheet, database queries	GSS classification & outlining tools, team calendaring, Project Management	GSS classification & outlining tools	GSS classification tools, structured reading methods, simulation & modelling tools	GSS classification & outlining tools used with clarification and review methods
Deliver	Document repositories, shared workspaces,	Multi-user database, notice boards, newsgroups, E-mail, shared workspace	GSS, on-line Discussion Tools, simulation and modelling tools	Relevance-weighted AI Collaborative Query Tools (Quantitative and Qualitative)	Collaborative query tools & GSS discussion tools used with argumentation & review methods
Use	All of the above	All of the above	All of the above	All of the above	All of the above

Regardless of which KM activities a team performs, the activity will require that the team engage in one or more of the five patterns of collaboration. Table 3 illustrates the interdependence of KM and Collaboration by mapping collaboration technology to KM activity by the pattern of collaboration the technology can foster.

Regardless of which KM activity a team performs, they must divide their limited attention resources among the three cognitive processes required for collaboration: communication, deliberation, and information access. Table 4 illustrates the interdependence of KM and collaboration by mapping collaborative technologies to KM activities organized by cognitive process.

6 Discussion

The multidimensional framework presented in the tables above demonstrates the interdependence of knowledge management activities and collaboration concepts. The correspondence between collaboration and KM is not surprising when you consider that a) the purpose of an organization is to create value for its stakeholders that the stakeholders cannot create for themselves as individuals; b) the purpose of collaboration is to achieve a goal through joint effort, thus, creating value; and c) the purpose of Knowledge Management is to make relevant knowledge available to people who seek to create value for organizations. Organizations consist of people working together toward value-creation goals; both KM and collaboration technology exist to make them more effective and efficient.

Table 4. Collaboration Technology for KM Activities organized by Cognitive Process

KM Activities	Cognitive Process		
	Communication	Deliberation	Information Access
Create	Voice, Video, IM, Chat, E-mail, Online News Group, GSS discussion tool	GSS classification and outlining tools, Shared diagramming tools, group decision support tools	Team Database, Online Document Repository, GSS transcript repository; collaborative query capability
Collect	All of the above	All of the above, plus Online forms, Document and transcript repositories, GSS Classification and outlining tools, Multi-user database, Notice boards, Newsgroups, E-mail, Shared workspace	All of the above
Organize	GSS Shared Outline Tools, GSS Concept Classification Tools, supported by the tools mentioned above	GSS classification and outlining tools, Shared diagramming tools, group decision support tools; Collaborative simulation and modelling tools	All of the above
Deliver	Virtual Workspace, Document & transcript repositories, Team Databases, Remote Presentation Capabilities	Virtual Workspace, Document & transcript repositories, Team Databases, Remote Presentation Capabilities, Collaborative simulation and modelling tools	All of the above
Use	All of the above	All of the above	All of the above

6.1 Implications for Research

There are several implications of this framework for researchers. First, KM and collaboration are currently separate research streams with few points of overlap. The framework we offered argues the interdependence of collaboration and KM, which suggests that KM and collaboration are two views of a larger underlying concept. The framework offered here does not illuminate the nature of the underlying concept, nor does it articulate a rigorous theoretical foundation for integrating these two lines of research. More research is required to reveal the nature of the underlying concept, and to explain how and why KM and collaboration technologies could or should be integrated. This research must identify and articulate the phenomena and outcomes of interest that the newly integrated research stream could or should address.

6.2 Implications for KM and Collaboration Technology Designers

Collaboration technology typically focuses on group process – sense-making, alternative identification and evaluation, decision making, planning, action, and after action review. Collaboration technologists typically deliberately exclude considerations of content. KM technology typically focuses on content – understanding and delivery of data, relationships, information, patterns, procedures, and knowledge. KM technologists typically do not focus on group process. Yet, to achieve their goals, teams and organizations must have effective and efficient collaboration processes, and they must be able to bring the intellectual capital of an organization to bear on their task. It might therefore be useful for KM and Collaboration technologists to find ways to

integrate both kinds of capabilities into a single process-and-knowledge system to support joint effort toward a goal.

6.3 Implications for Practitioners

The framework offered here was developed to illustrate the interdependence of collaboration and KM. It is also useful for organizational practitioners. First, it provides as a useful way to understand the variety contributions a given collaboration or KM technology could make to team and organizational productivity. A practitioner could, for example, consider whether the technology were better suited to collective, coordinated, or concerted work, and whether it offered support for communication, deliberation, or information access. The practitioner could consider the variety of collaboration patterns that one could evoke with a given technology.

In like manner, the framework might be useful for comparing two or more KM/collaboration technologies to one another, and for positioning collaboration technologies in the market space. Current technology comparisons are usually based on feature checklists. However, feature comparisons are perhaps less informative than comparisons of mode, pattern, and cognition support.

Finally, practitioners may find the framework for choosing which mix of technologies might be must useful for addressing a particular collaboration/KM need. The need could be characterized in terms of the dimensions of the framework (what work modes are required, what patterns are required, what cognitive processes are required, what KM activities are necessary?). That characterization could then become a basis for selecting the mix of technologies to address the need.

7 Conclusions

The framework offered in this paper illustrates the interdependence of KM and collaboration. These domains are clearly closely related to each other and to organizational value creation, but the nature of those relationships is not ye plain. The theoretical arguments that link them have yet to be articulated. The framework has implications beyond the purpose for which it was originally developed, serving useful purposes for researchers, technologists, and practitioners. However, it cannot substitute for a rigorous theoretical foundation. The development of that theoretical foundation is the next logical step in an effort to improve value creation in organizations by integrating KM and collaboration initiatives.

References

1. Angus, J., Patel, J.: Knowledge Management Cosmology. InformationWeek, March 16, 59. (1998)
2. Borghoff, U.M., Pareschi, R. (eds.): Information Technology for Knowledge Management. Springer. (1998)
3. Briggs, R.O.: Focus Theory of Group Productivity, Doctoral Dissertation, University of Arizona. (1994)
4. 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

5. Ellis, C.A., Gibbs, S.J., Rein, G.L.: Groupware: Some Issues and Experiences. Communications of the ACM, (1991). 39-58
6. Ferran-Urdaneta, C.: Teams or Communities? Organizational Structures for Knowledge Management. Proceedings of the 1999 ACM SIGCPR conference on computer Personnel Research. (1999) 128-134
7. Gamble, P. R., Blackwell, J.: Knowledge Management: A state of the art guide. Kogan Page. (2001)
8. Grudin, J., Palen, L.: Why Groupware Succeeds: Discretion or Mandate? In Proceedings of ECSCW'95, Kluwer. (1995) 263-278
9. Gupta, A. K., Govindarajan, V.: Knowledge Management's Social Dimension: Lessons From Nucor Steel. Sloan Management Review. (2000) 71-80
10. Hansen, M.T., Oetinger, B., von.: Introducing T-shaped Managers: Knowledge Management's Next Generation. Harvard Business Review. March (2001) 107-116
11. Hibbard, J. and Carillo, K.M.: Knowledge Revolution – Getting employees to share what they know is no longer a technology challenge – it's a corporate culture challenge. InformationWeek. (1998) 663
12. Jackson, C. Process to Product: Creating Tools for Knowledge Management. [www document] URL http://www.brint.com/members/online/120205/jackson/. (1999)
13. Kramer, M.: Knowledge Management Becomes Catch Phrase but Eludes Easy Definition. PC Week, December, (1998). 95.
14. Macintosh, A.: Knowledge Management. [www document]. URL http://aiai.ed.ac.uk/~alm/kamlnks.html. (1999)
15. Nunamaker, J. Briggs, R., Mittleman, D., Vogel, D., & Balthazard, P.: Lessons from a Dozen Years of Group Support Systems Research: A Discussion of Lab and Field Findings, Journal of Management Information Systems. Winter 1996-97, 13 (3), (1997) 163- 207
16. Nunamaker, J.F. Jr., Briggs, R.O., Vreede, G.J. de,: From Information Technology To Value Creation Technology, in: Dickson, G.W., DeSanctis, G. (eds). Information Technology and the Future Enterprise: New Models for Managers, New York: Prentice-Hall. (2001)
17. Quinn, J.B.: Intelligent Enterprise. Free Press. New York (1992)
18. Qureshi, S., van der Vaart, A., Kaulingfreeks, G., Vreede, G.J., de, Briggs, R.O., Nunamaker, J.: What does it mean for an Organisation to be Intelligent? Measuring Intellectual Bandwidth for Value Creation, The Thirty Fifth Hawaii International Conference in Systems Sciences. IEEE Computer Society Press (2002)
19. Ruggles R.: Knowledge Tools: Using Technology to Manage Knowledge Better, URL: http://www.businessinnovation.ey.com/mko/html/toolsrr.html (1997)
20. Skyrme, D. J.: Knowledge Networking: Creating the collaborative enterprise. Butterworth Heinemann, Oxford. (2000)
21. Sproull, L Kiesler, S.: Connections: New ways of working in the networked organization. The MIT Press. Cambridge, MA. (1991)
22. Stewart, T.A.: Intellectual Capital: The New Wealth of Organisations, Nicholas Brealey Publishing Limited, London. (1997)
23. Sveiby, K.E.: The New Organizational Wealth: Managing and Measuring Knowledge-based Assets. San Francisco CA, USA: Berrett-Koehler Publishers. (1997)
24. Vance, D. M.: Information, Knowledge and Wisdom: The Epistemic Hierarchy and Computer-Based Information System . Proceedings of the 1997 America's Conference on Information Systems. http://hsb.baylor.edu/ramswoer/ais.ac.97/papers/vance.htm. (1997).
25. Whitaker, R.: Computer Supported Cooperative Work (CSCW) and Groupware: Overview, Definitions, and Distinctions. www.informatik.umu.se/~rwhit/CSCW.html. (1996)
26. Zack, M.: Developing a knowledge strategy. California Management Review, (1999) 125-145
27. Zigurs, I., Kozar, K.A.: An Exploratory Study of Roles in Computer-Supported Groups, MIS Quarterly, (1994) 277-294

Applying Group Storytelling in Knowledge Management

Raphael Perret[1], Marcos R.S. Borges[1], and Flávia Maria Santoro[1,2]

[1] Post-Graduate Program in Informatics, NCE&IM, Universidade Federal do Rio de Janeiro
PO Box 2324, CEP 20001-970, Rio de Janeiro, Brazil
rperret@ufrj.br, mborges@nce.ufrj.br
[2] School of Applied Informatics, Universidade Federal do Estado do Rio de Janeiro
Av. Pasteur, 458, Rio de Janeiro, Brazil
flavia.santoro@uniriotec.br

Abstract. Knowledge management and, specifically, organizational memory have become vital for organizations' life. Documenting tacit knowledge used to perform daily activities, such as, discussions and decisions is a complex task. Another challenge is dealing with collective knowledge, because an important part of organizational work is executed in a cooperative mode. In this paper, we present stories as an important tool to externalize tacit knowledge. We describe TELLSTORY, a system that supports the collaborative construction of stories. Based on the characteristics of traditional literary and journalistic narrative structure, TELLSTORY helps teams in developing stories to make explicit tacit knowledge elements.

1 Introduction

The need to manage and share knowledge with partners, suppliers, and customers - let alone with employees - is increasingly becoming part of today's organizations. According to a recent report from Gartner Group, as enterprises better understand Knowledge Management (KM), it will become a more-practical and important part of doing business [1]. However, there are many aspects of KM which taken together represent a significant change in the way organizations manage people, processes and information. KM involves taking a more *holistic* view of information, not only combining internal and external information, but also consolidating informal ('soft') and formal ('hard') information [2].

Lotus state that 80% of corporate knowledge flows through informal groupings – these are normally networks that businesses have little awareness of and zero control over. Expressive communication often occurs in informal settings including hallway conversations, informal meetings, stories, notes scrawled on napkins, and non-business e-mails. Organizations clearly rely on instrumental communications, but the role of expressive communications is less clearly understood, with perceived value varying from irrelevant to vital to the accomplishment of work [3].

Every day, team workers need to accomplish goals and perform tasks in a cooperative mode. Although the knowledge management is a prominent issue, in many cases the documentation and the historical context of the actions are left behind. This scenery is partially explained by the lack of specific support to teamwork. Had groupware been used to support group interaction, part of communication would be naturally captured and would become part of the organizational knowledge.

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 34–41, 2004.

Registering the information related to the execution of the tasks, even after its end, is essential to successful KM. Studies about organizations indicate that the real process of work differs a lot from the way they are described in practice manuals, proceedings, task reports and tables [4].

One possibility of registering that real way of work can be made through stories. The tradition of an oral narrative history that records and hands down learning, insight or collective revelation still thrives in social communities and is particularly effective in helping change our business mindset and improve our knowledge practice. Storytelling can be considered as a means of communicating about business. The power of a good story well told can inspire innovation and professional breakthrough. Stories can encourage us to change, to think 'out of our boxes', and to seek the help of others in leveraging our own efforts [5].

The goal of this paper is to discuss how a groupware, specifically a group storytelling tool, can support the externalization of the group tacit knowledge within a project, making it available for future retrieving and reuse. We claim that this approach is preferable when informal knowledge is to be recovered from past projects. While stories can be considered a nice way to report past experiences, it can also be an essential part of the organization knowledge.

The story of this paper can be told as follows. Section 2 tells the story of the importance of knowledge management to projects. Section 3 will tell the importance of stories as it is reflected in applications – single user or groups. Section 4 reports the development of TELLSTORY, a groupware aimed at supporting group story telling. Finally, Section 5 is where this part of the story ends, summarizing what has happened in this episode and announcing the subsequent events.

2 Knowledge Management in Projects

The increasing importance of the knowledge component is stimulating the companies to develop resources that facilitate its administration. Thus, Knowledge Management became adopted in wide scale, supporting the organization of the procedures, practices and tools that aim at capturing, storing and disseminating the knowledge among the employees of the organization, taking advantage of the existent technological resources.

One of the most important practices related with KM is the Organizational Memory (OM) that, intuitively, can be classified as the registration of useful data, information and knowledge. OM can be retained in the culture, transformations, ecology, external files, corporate manuals, databases, stories and the individuals within an organization [6].

In fact, we cannot assert that the entire knowledge is registered in documents. The experience of the organization members, their ideas and decisions cannot be left out of the OM. Nonaka and Takeuchi define these elements as *tacit knowledge* [7]. 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 share and easy to articulate in clear terms.

One of the main challenges of KM and OM is exactly to capture the tacit knowledge. Because it is not logical and strictly documented, it is difficult to maintain. Nevertheless, it should be disseminated among the members of the organization be-

cause it represents ideas that were drifted, decisions that were taken, reasons that led to the rejection of another decisions, options and doubts before choices in certain projects and other essential information.

One possible solution for this problem is to transform the tacit in explicit knowledge, so that it can be registered and documented easily. This process is called *externalization.* This includes benefits for the organization since that process is based on the contribution of the individual knowledge, represented by the personal experience to be shared by the organization. The gain is, therefore, for the whole company.

In spite of all the benefits, the process of converting tacit knowledge into explicit is an expensive and complicated task. The main reason for the externalization complexity is the difficulty of formalizing tacit knowledge, once it is consolidated in the individual's mind, through its know-how.

There are very few tools able to capture the institutional experience and to disseminate the tacit knowledge. Some appealing techniques, such as elicitation among employees, best practices documentation and external consulters are flaws [8]. In this scenery, the development of new tools to support the externalization seems to be very relevant.

Besides the metaphors and analogies, other interesting speech techniques would be capable to help the externalization, such as the stories [8]. Next section will come up with the narrative structure, its importance for the human being and how they can be used to aid the externalization process.

3 Stories Applications

A story can be defined as "a narrative of an event chain told or written in prose or verse", while the word narrative comes from the Latin *narrere* that means "to pass knowledge" [8]. Therefore, story is a possible mechanism of knowledge transmission.

Telling stories is as old as the human being history. It has been used as an important technique of knowledge propagation. But stories are not just means of communication; they also have a jolly aspect. The reader of a narrative hopes to finish his reading reaching a state of emotional balance, something similar to an easy spirit [9]. People are usually interested intensely in the uncoiling of a very well told story.

We can also compare the stories with another element that is part of the individuals' daily life: the journalistic text. The news is defined as "the report of a series of facts starting from the most important or interesting fact; and each fact starting from the most important or interesting aspect" [10]. There is a very close link between story and news.

The popularity and the importance of the stories for the individuals have made the organizations adopt them as a tool to support knowledge management. Since people like to read and hear stories, the storytelling practice works attractively over the members of the institution in the organizational memory construction, without necessarily meaning an additional workload. The stories have the capacity to build communities, to facilitate the communication, to accelerate organizational changes, to stimulate the innovation and to transmit knowledge. It is an old human ability applied to a new context: knowledge management [11].

Besides purely transmitting the teller's knowledge to the listener or reader, the story can bring some tacit elements of its embedded knowledge. Many times it is a

vague and disjoint description of elements or, in the best case, a sample of the knowledge that can be made explicit, combined with the personal expression of emotional, physical and informational aspects that the individual associates with the knowledge [12].

The stories help to humanize the environment, creating favorable scenery for the knowledge sharing and to the new communities' construction [13]. The narratives involve emotions, thus they also provoke the personal commitment and stimulate the externalization [12]. The dissemination of stories technique also presents obstacles inside the company. The most important is the lack of a common and acceptable language for all the members, which usually originate from different cultural backgrounds.

In spite of certain enthusiasm on the use of stories in KM, methods for the development of narratives by capturing knowledge are not found in the literature. The studies remain on how to *use* the stories, but few on how to *create* the stories. Thus, the two more important methods are the reports and the learning stories. Both can be considered peculiar cases of narratives, each one with its specific characteristics.

A very traditional method is the report, which is a written work that describes the accomplishment of a task, independently of the number of agents involved, period of execution or degree of difficult. Nevertheless, they are not effective in the externalization of tacit knowledge, once they do not allow the same creative freedom that the stories do. The reports are objective, hindering that the authors tell little of themselves, the characters involved in the story and existent details of the fact, blocking the transmission of tacit elements.

Another methodology is the learning stories, which are written narratives of a recent group of critical episodes of the company: some cases of corporate change, a new initiative, some diffuse innovation etc. [11]. The learning stories use the storytelling technique in group in a peculiar way. Individuals that did not participate in the work told develop them. It means that, in spite of the learning story be born from a collective effort, there is little collaboration among the performers of the task. Besides, the structure of the learning stories is always strictly defined.

In this context, we propose the group storytelling technique. An individual or a group can tell a story. In the later case, members of a team, distributed or in the same place, contribute to create a story collectively, synchronously or asynchronously, using different media [8]. We believe that the knowledge generated at the end of this process is a combination of the tacit knowledge from each participant. This technique is called group storytelling.

The group storytelling is a more appropriate process than the individual storytelling in contexts where there are several people involved in the execution of a project. The group will build collectively a story about a work done by its members. Each participant performed a role in the project, which story will be "told". Therefore, the stories written by a team will probably contain more valuable details because everybody has the opportunity to present their viewpoint on what happened during the project. In other words, we are inducing team members to expose tacit elements of their knowledge.

Nevertheless, discussions and disagreements will certainly arise. Thus, the group needs support to express their thoughts and to solve the conflicts in order to produce a real, interesting and useful story.

4 The TellStory Groupware

The growth of cooperative work needing in the organizations stimulated the development of a type of support tool used by teams, the groupware. It facilitates several activities traditionally performed in the group work, providing coordination, communication and awareness.

Although we haven't found any consolidated system that supports collaborative construction of stories, some tools present useful resources to this job: Computer-Supported Collaborative Learning environments, that stimulate collective knowledge building [14][15] and also the Collaborative Authoring Tools [16][17][18][19].

Based on some characteristics of those systems and the analysis of the literary narratives and journalistic texts [10] [20], we developed a groupware for collaborative stories building, which supports the method of group storytelling. Tellstory is a web application implemented under the Zope platform [21].

Any registered member of TellStory can create a story and invite new participants. An individual can participate on the story performing one of the following roles: (i) moderator: creator of the story and responsible for the coordination of the actions inside of the story; (ii) user (common to all members): able to contribute with the story; (iii) teller: 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 persons.

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 [13]. Tellstory used that definition to model the construction of the story in group. Each user can insert an event, that is, a fact happened during the accomplishment of the task, which he remembers (Fig. 1).

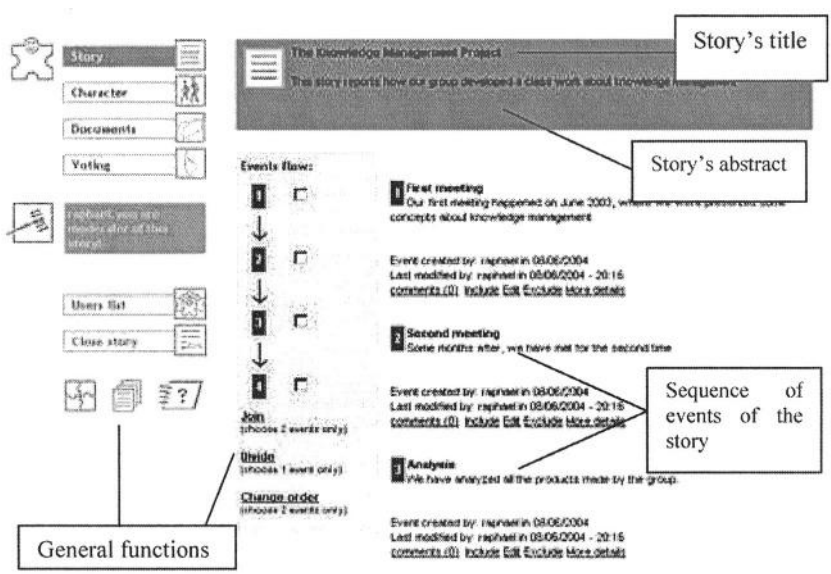

Fig. 1. Flow of a story

The possible 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, as well as the fragmentation of the event divides it in two, when necessary. The criteria that indicate if a fact is an event or a sub-event or 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.

Once the users included the events, they can discuss, through comments in a forum format, and, eventually, to decide certain subjects through voting organized by the moderator. One example of subject discussion is event truthfulness. In case there is no consent about the existence of a certain fact, the tool allows the story to have two versions, each one considering the hypothesis of the event to have happened or not. However, duplicity of versions should only be used if there is not majority on decisions related to a subject.

One of the most important issues of TellStory is that it offers a possibility to use a template in order to address the elaboration of the story through the typical characteristics of a narrative structure. For example, the template shows to the users that an event should always have a cause relationship and consequence with its successor or predecessor, according to the causality principle [11]. The template also has a module in which the users can define and configure the characters, activity that aids much the externalization. The template works as a guide for the tellers, stimulating their memories and helping them to better structure their thoughts.

When group understands that the story already provides a sufficient flow of events, moderator can conclude the task. At this time, teller gathers the events and writes a final text based on the sequence. Finally, commentator searches tacit elements that can be identified in the story, which are registered in a module enclosed to the final text (Fig. 2).

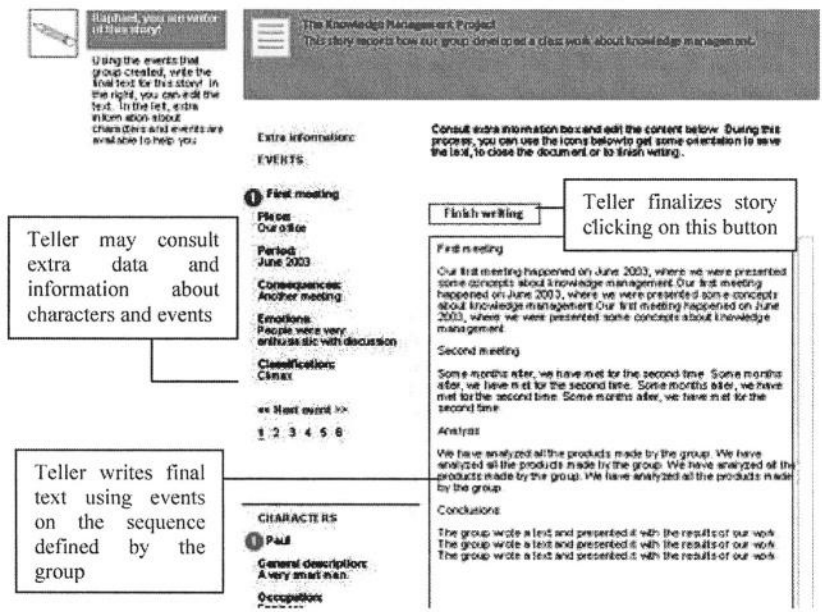

Fig. 2. Writing final text of the story

5 Conclusions and Future Work

One of the challenges of the KM is to capture the individuals' tacit knowledge that participated together on the accomplishment of tasks. We believe that the stories, narratives with beginning, middle and end, are an appropriate way of telling what happened and, at the same time, can externalize the tacit knowledge of the group. Therefore, we developed TellStory, a groupware that supports the collaborative stories building, where all the team members can contribute.

The tool allows a group to tell a story, starting from the contributions of each one of the members. Even if they are introduced in a disordered way, the environment determines roles that guarantee the coordination and the organization of the events, discussing and, if necessary, voting to decide which direction will be taken. Tellstory is also useful in the construction of stories on the other contexts described in this paper, such as in the educational or journalistic domain.

The next steps of our research include the execution of a case study. Thus, it will be possible to analyze how the tool helps in the construction of stories in real groups and to validate if it stimulates the externalization. The study will happen at the City hall of Rio de Janeiro, where a group will tell the story of the preparations of a seminar.

References

1. Gartner Group: KM in 2004: Just Part of Doing Business. http://www.gartner.com. Accessed on 7 April, 2004 (2004)
2. Corrall, S.: Knowledge Management: Are We in the Knowledge Management Business?. Ariadne Issue 18. http://www.ariadne.ac.uk/issuel8/. Accessed on 10 April, 2004 (1998)
3. Thomas, J.C., Kellogg, W.A. and Erickson, T.: The knowledge management puzzle: Human and social factors in knowledge management. IBM Systems Journal. 40 (4) (2001)
4. Raybourn, E.M., Kings, N.J. & Davies, J.: Adding Cultural Signposts in Adaptive Community-Based Environments. Interacting with Computers: the interdisciplinary journal of human-computer interaction. Special Issue on Intelligent Community-based Systems. Elsevier Science Ireland Ltd. 15/1 (2003) 91-107
5. The OPAL Team: Story-telling in Shell: Managing Knowledge through New Ways of Working. http://www.knowledgeboard.com/library/Stories_extern_l5.pdf. Accessed on 5 April, 2004 (2004)
6. Ackerman, M.S.: Augmenting the Organizational Memory: A Field Study of Answer Garden. In: Proceedings of CSCW'94. (1994) 243-252
7. Nonaka, I. Takeuchi, H.: The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, Oxford (1995)
8. 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)
9. Scholes, R. e Kellogg, R.: The Nature of Narrative. Oxford University, New York (2003)
10. Hills, R.: Writing in general and the short story in particular. Houghton Mifflin, Boston (1997)
11. Kleiner, A. and Roth, G.: How to Make Experience Your Company's Best Teacher. In: Knowledge Management. Harvard Business Review 75, no. 5 (1997)
12. Ruggles, R.: The Role of Stories in Knowledge Management. Storytelling Foundation. http://www.storytellingfoundation.com/articles/business/stories-km.htm. Accessed on 9 August 2003 (2003)
13. Lelic, S.: Fuel Your Imagination - KM and the Art of Storytelling. Knowledge Management (2001)

14. Suthers, D., & Hundhausen, C: Learning by Constructing Collaborative Representations: An Empirical Comparison of Three Alternatives. In: Proceedings of the First European CSCL, Universiteit Maastricht, Maastrict, the Netherlands, (2001) 577-584.
15. Stahl, G. A model of collaborative knowledge-building. In: Fourth International Conference of the Learning Sciences. MI, USA (2000)
16. 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.
17. Decouchant, D., Martínez Enríquez, A.M., Martínez González, E.: Alliance Web: Cooperative Authoring on the WWW. In: SPIRE/CRIWG, Mexico. IEEE Computer Society (1997) 286-295.
18. Dimitriadis, Y.A., Martínez, Al., Rubia, B., Galleg, M.J.: Cooperative Learning in Computer architecture: an Educational Project and its Network Support. In: IEEE Proceedings of Frontiers in Education. FIE, USA (2001)
19. Santoro, F. M., Borges, M. R. S., Santos, N.: Experimental Findings with Collaborarive Writing within a Project-Based Scenario. In: Computers and Education: Towards a lifelong learning society. Kluwer Academic Publishers, London (2003)
20. Holloway, J.: Narrative and structure: exploratory essays. Cambridge University Press, New York (1979)
21. Zope.org: http://www.zope.org, Accessed on 20 April 2004 (2004)

Ranking the Web Collaboratively and Categorising It to Produce Digital Collections

Vidal A. Rodríguez and David A. Fuller

Computer Science Department
Pontificia Universidad Católica de Chile,
Vicuña Mackenna 4860, Santiago 6904411, Chile
{vidal,dfuller}@ing.puc.cl

Abstract. In this paper, we discuss the need of groups of people working towards a common goal, requiring reliable information. It would be helpful for these groups to have digital collections that would provide them with the information they need to do their work. We propose a tool that allows them to build their own collections using the Web. Each group may work in different projects. Each project requires a digital collection. The group members can rank the resources obtained by them from the Web for the project in a collaborative way. In addition, specialised cataloguers can categorise the resource collections, using a controlled language. We propose an interaction model, the architecture of the software system, and we show our actual implementation.

1 Introduction

Nowadays, most research on the creation of trustworthy digital libraries is being done in the automation field [21, 22, 23, 24, 25, 26, 27]. Several advances based on human capabilities to produce quality digital libraries and to get better results in information retrieval, have been achieved [i.e. 4, 7, 28, 30]. However, we believe that still there is considerable human effort being wasted, which can be taken advantage for the sake of persons. Our interest in this work is to reuse and foster the efforts deployed by humans in the search of information in the Web and the creation of digital libraries, mainly when it comes to small groups of peer persons working in a common project.

1.1 Finding Information

The Web is a big digital library without an engineered architecture [14]. To reach the different pieces of information available in it, we use concepts such as "electronic bookmarks", hypermedia links, search engines, directories, experts' recommendations and publicity. Despite of the numerous ways and services available to retrieve information from the Web, finding useful and relevant information in it, is a very difficult task [9, 16] and doing it in a timely fashion can be very frustrating [8]. The search engines, which are the unquestionably most used services to find information

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 42–51, 2004.

in the Web, have many limitations. They index only a tiny part of the Web [17], cover all topics with a relatively small coverage of any particular topic [2], index resources in a few number of formats, and many of them examine only certain parts of the pages [2]. Generally the results contain lots of irrelevant suggestions, caused by the susceptibility to search engine persuasion and keyword spamming [18], sponsored results, deficiencies of the indexing and ranking policies and algorithms, among others.

In consequence, persons after all have to check the content of the found resources to verify their relevance and usefulness. As Conklin [5] states, at least for the near future, human intelligence and effort will remain a key component for retrieving meaningful and relevant information from large repositories.

Furthermore, with current Web tools, sharing their findings and outcomes between several persons may become awkward. Even more, people in a group working together with a common goal requiring to share their findings, will most probably process (visit, decide relevancy and use or discard) the same resources, which in many occasions is an unwanted waste of efforts.

1.2 Ranking the Resources

Many authors and institutions [e.g. 1, 6, 29, 31, 33] recommend evaluating the sites before using their information and hence avoid the bad ones. In this respect, several efforts have been done to rank or evaluate the Web resources and thus get a quality measure of every resource. There are quality indicators, independent from the users, like those kept or calculated by some search engines that measure the importance of the resource for the search engine. This is the case of Google that uses PageRank [10], which can be viewed through the Google Toolbar [11]. Many other sites do not publish their rankings for resources they index. Anyway, this method does not take into account that the quality of a resource depends on the target audience. Thus, a specific page can be excellent for a kid's homework, but too basic or useless for a scientist. Moreover, there are resources' quality indicators associated to individual users, like those mentioned in [3] and those kept in many Web sites that as a requirement users must register themselves before using the sites' services.

One problem we detect in the previous approaches is that they ignore the use a person will give to the resources. A page can be useful for a person in some projects or goals but not in others, even though it may look as if it was. Furthermore, current tools don't support the marking of unwanted resources so that we do not come across them, or if we find them somewhere else, some kind of warning could be given.

1.3 Building Digital Collections

To find useful information in the Web, several studies and publications [6, 9, 13, 15, 32, 33] recommend consulting professors, colleagues, friends; to review printed articles, journals about Internet, directories, etc. In other words, to locate helpful information in the Web, a good alternative is through the seeking and categorizing work performed previously by other persons. Handcrafted digital collections are valuable

[2] and very desirable, especially for the academic community [19]. It is clear that the generation of digital collections by hand is expensive, does not allow scalability and cannot cope with the huge amount of available and continuously growing information [2]. But, can't be fostered and reused the human work invested in searching, reviewing and evaluating Web resources? We believe it is possible and to accomplish it, we propose a collaborative ranking system that allows groups to build their own collection of web resources, for each project in which they are involved. Additionally, the collections can be categorised, with a controlled language, by a specialised entity interested in producing a sort of library of digital collections.

Our proposal is organised as follows. In the next section, we look into the problem of groups working together to pursue a common goal in need of reliable information to support their work. Based on the use of Web, we propose an interaction model, as well as the required activities and roles. In section 3 we discuss our implementation, and we finish with a discussion, conclusions and further work.

2 Design of a Solution

2.1 Interaction Model

Our proposal attempts to face the previously exposed problems, based on a group of peer persons working toward a common goal. The main idea is to allow the group to collaborate in ranking resources from the Web to form a collection, then categorising the resources collection as a whole to integrate them in a digital library and, at all times, using the generated information to improve the results for queries of the group members in search engines. We use the phrase project collection to refer to all the information contributed in the processes of ranking and categorising. A graphical representation of this interaction model is presented in figure 1.

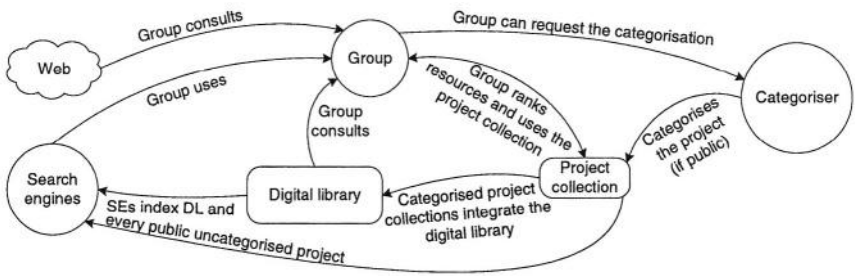

Fig. 1. Interaction model. Arrows indicate the direction of the main flow of information

Now a detailed description of the different stages is presented.

2.2 Creation of the Project Collection

The project collection is described with metadata, such as a name, a description, a privateness character (public or private), date of creation and last update, and a rank-

ing method. Moreover, the project collection must be associated with the group members. Project collections marked as private won't be categorised and hence won't be integrated in the digital library.

In theory, the ranking method could be any that can be expressed with a mathematical series and represented with a graphical widget. For the moment, we are working with binary rankings. This method accepts two possible values (e.g. relevant and irrelevant, useful and useless, good and wrong, etc.), and can be represented with a drop down list or a radio button. The ranking method must be decided and chosen by the group members on the creation of the project before starting to rank resources because later, when there are resources that are already ranked, changing the ranking method could be difficult.

2.3 Collaborative Ranking

Once the project collection is defined, the members of the group can use a collaborative system to navigate the Web and share their advances in one or more project collections. So, when a member finds a relevant resource and selects it for the project collection, it becomes visible and usable for the rest of the group. The same happens with irrelevant resources, but the aim with these ones is to warn or keep all the group members away from them. A special situation happens when a person finds a resource but s/he is not sure about its relevance for the project collection or perhaps s/he does not have enough time to review the resource. In this case, the resource can be marked as "To be reviewed". Resources are referenced by their URL.

All the selected (ranked and deferred to be ranked) resources can be commented at any moment and by every group member. If necessary, a discussion forum per resource can be established by group members to debate about and document the relevancy of the resource. Users do not have to rank every resource they find while being active in a project collection. They surely will like to rank every relevant resource they find, but they do not have to rank every irrelevant resource. They just could (and are recommended to) rank as irrelevant those resources that pretend to be relevant for the project but they are not.

In our current design, a project collection maintains only one ranking per resource and any group member can change the ranking of any resource at any moment, regardless of who had ranked the resource previously. This was decided in that way to ease the use of the tool and the development, but this can be enhanced even to allow every member to rank every resource.

The outcome of the ranking process is a list of ranked (good and bad) and commented resources related to the group's project collection. It should be noted that the ranking assigned to a resource in a project collection could be different to the ranking for the same resource in other project collection.

Group members can receive fresh notifications about project collection changes via a communication system, e.g. email. They also get appropriate awareness about project collection changes by accessing a project collection management and report subsystem. When navigating a Web page, group members get proper awareness about ranked resources (for the active project collection), which appear inside the page. For

example, if a page was ranked as irrelevant and a link to that page appears inside any other page, then the link appearance is modified, to warn the user about its irrelevancy for the active project collection.

2.4 Categorising the Project Collections

At any moment after the creation of public project collections, the categorising entity can categorise them or modify a previous categorisation. It is expected that the categorisation of a project collection will be more accurate near the end of the ranking process, when there are enough ranked and commented resources. Under our conditions, librarians of the Department of Cataloguing at the Library of the University do this work. The cataloguing of the project includes: (i) Assigning principal and secondary entries of a standard classification, for example the Dewey Decimal Classification. (ii) Adding controlled keywords. There is no need to categorise the individual ranked resources, but the project collection as a whole.

With the categorising of public project collections, the categoriser is creating a digital library formed by sets of digital resources. This digital library can be browsed through a directory based in the standard classification and searched with the controlled keywords, taking into account the age of project collections.

As can be deduced, with the work of ranking and categorising, it is possible to obtain a hand-made digital collection with reduced human efforts.

2.5 Improving Search Results of Search Engines

Similar to the categorising stage but at any time after the creation of a project collection, search engines could use all the information contributed to the project collection to improve the results of requests made by members of the group when they are active in the project collection.

The creation of user profiles and the use of rule-based and filtering techniques [3] can be applied to improve the search results. But it is necessary to consider that users belong to defined groups collaborating towards a common goal within a project. Hence, search engines must take into account all the information in the project collection to really improve the results. We recognise that this is a complex problem, which requires deeper research. However, there are some actions that can be applied immediately. Up to the moment, we identified the following actions that can be applied to resources in lists of results for a query.

Possible actions to perform with irrelevant resources are: (i) Remove the resource (ii) Change the appearance to attenuate or diminish as much as possible (iii) Move down in the list of results.

Possible actions to perform with relevant resources are: (i) Move up in the list of results (ii) Changing the appearance to highlight or augment over the others (iii) Remove the resource. This action over relevant resources can be useful when the group or user wishes at some moment in the project to gather as much relevant resources as

possible, so that any already ranked as relevant may obstruct the way to find more relevant ones.

With these simple actions, which can be regulated according to preference rules of the project collection or users, group members can be helped to avoid repeating much work done by every other group member, and as a consequence get a better productivity in the entire group.

3 Implementation

3.1 Main Architecture

To implement our design, we developed a system based on a client–server architecture. See figure 2 for an illustration. A Web server hosts the ranking and categorising system, which is accessed from the users' browser. The ranking and categorising system includes three modules: an administration module, a ranking module and a categorising module. The administration module allows the registration of users and the management of project collections, including reports about them. The ranking module serves the ranking tool, which is used in the client window. The categorising module is accessed from any Web browser and provides information and tools needed by the categorisers to catalogue the public project collections.

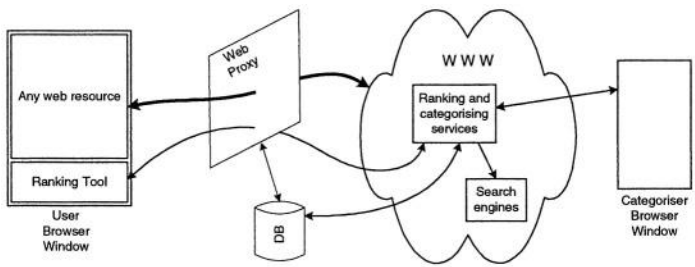

Fig. 2. Main architecture of the system

In order to monitor the users' navigation and modify Web resources in accordance with the active project collection, a Web proxy is used. So, it is mandatory for the user of the ranking tool to set the system to use the Web proxy. This setting is not necessary for the categorisers.

The ranking and categorising system is supported by a database, which stores all the needed data. The database is also accessed from the proxy to retrieve the required information to enhance the user navigation.

3.2 Collaborative Ranking Tool

The collaborative ranking tool is embedded as part of the favourite user's Web browser and implemented as a browser plug-in. To facilitate this, we restricted our prototype to the MS Internet Explorer browser and created a Microsoft Explorer Bar

[20] with a Web page as the content. This content is generated dynamically based on the current navigating user, the active project collection (if any) and the resource (usually a page) that is opened in the browser main window. Figures 3, 4 and 5 show some screenshots of typical browser windows during navigation with the ranking tool open. The link to a resource ranked as relevant is modified with the green ✓ sign and the irrelevant one is modified with the red × sign.

The ranking tool can be implemented in some other ways, for example, modifying the content of the resource in navigation by adding the ranking tool to the resource. This is possible in the same proxy that we are using to monitor the user navigation.

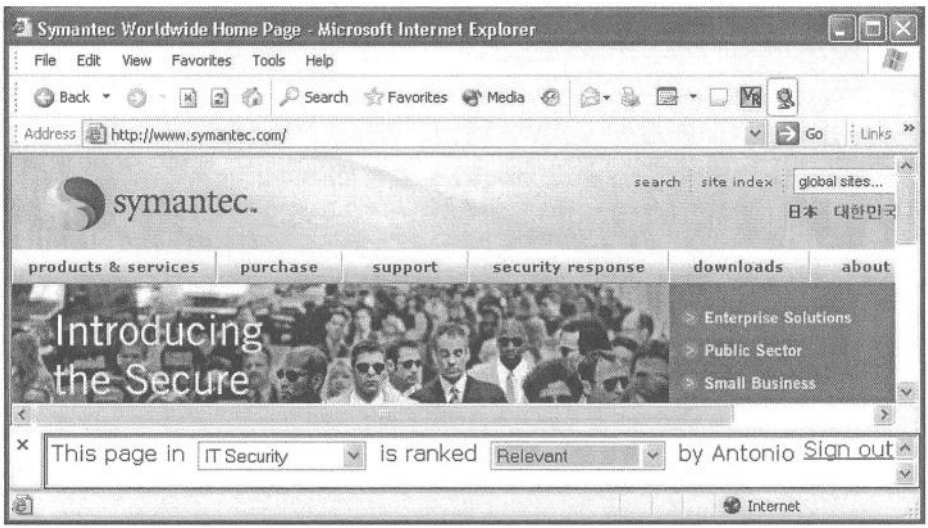

Fig. 3. A user is working for his/her project collection named *IT Security*. Another group member named *Antonio* ranked the page that is navigating as *Relevant* in that project collection. The ranking tool can be shown or hidden pressing a button in the browser toolbar

Fig. 4. A user is working for his/her project collection named *Human illnesses* and is browsing through the same page of the figure 3. Another group member named *Vidal* ranked the page as *Irrelevant* in that project collection

3.3 Extending Search Engines

In figure 5 it can be noted that the user searched the query "biological virus" for its project collection "Human illnesses" and a page related to computer viruses was proposed by Google in the first results page. If Google had had access to information of the user active project collection, then probably the page about computer viruses wouldn't be in such a privileged position of the results page.

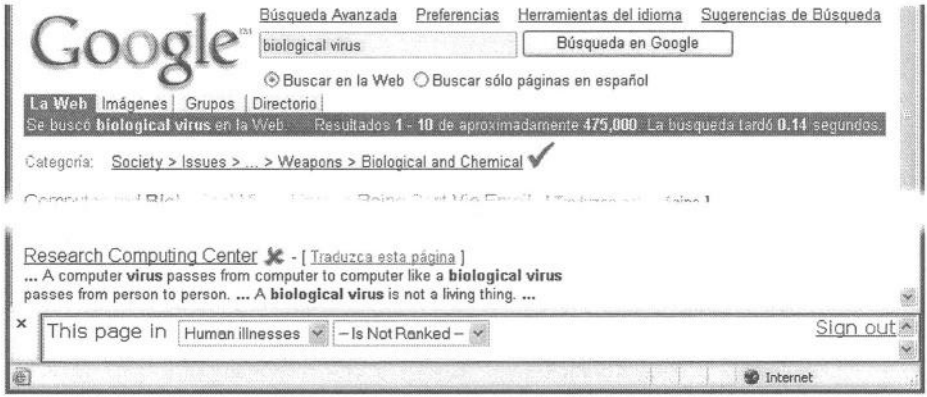

Fig. 5. A user is working for his/her project collection named *Human illnesses.* The page that is navigating *is not ranked* in that project collection but it contains two links to ranked resources

Besides the ranking and categorising system, we also attempt to demonstrate the possibility of improving the search engines results. In the diagram of figure 2, search engines are represented forming part of the WWW because they are accessed via Web by the users, and in communication with the ranking and categorising services, to retrieve information about the active project collection of the user searching with a query. This communication is not direct to the database, because we believe that search engines must use a higher-level standard protocol to obtain this information.

At this moment we are not developing a full search engine. Instead we have built a prototype based on Google Web APIs [12]. This prototype wraps Google search engine with a module to filter the results aiming for improving the results obtained for a query. It sends the query to Google, then before showing the results to the user, it sends them through the module that tries to improve the results.

4 Discussion, Conclusions, and Future Work

In this paper, we discuss the need of groups of people working towards a common goal, requiring reliable information. We propose a tool that allows them to build their own collections using the Web. Each group may work in different projects. Each project requires a digital collection. The group members can rank Web resources for the project collection in a collaborative way. The Library cataloguers can categorise the project collections, using a controlled language.

We implemented an environment to support our proposed interaction model. While a group is working on the construction of its digital collection, the system modifies the ranking of the search engine according to the information already ranked by the group members, which makes the search even more effective. Additional features are also planned to be added to the environment, such as the extension of the filtering of irrelevant sites to include those known publicly regardless of the active project collection, as well as allowing the system to use multiple search engines.

Up to moment we have tested the system only in small groups, in reduced experiments. We plan in the near future to do extensive tests of the system in different organisations, with different age ranges and educational background. We have a special interest in testing the system in groups of school children in search of relevant material in support of their school group projects. We believe that the exhaustive testing of the system will allow us to fully understand the user's required functionality. We expect to report those results in the full paper.

References

1. Alexander, J. E. and Tate, M. A.: How to Evaluate and Create Information Quality on the Web. Published by Lawrence Erlbaum Associates (1999)
2. Bergmark, D.: Collection synthesis, In: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, Portland, Oregon, USA (2002)
3. Buono, P. et al.: Integrating User Data and Collaborative Filtering in a Web Recommendation System. Lecture Notes in Computer Science, Vol. 2266. Springer-Verlag (2002) 315-321
4. Calado, P. P. et al.: The Web-DL Environment for Building Digital Libraries from the Web, In: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, TX, USA (2003)
5. Conklin, J. Designing organizational memory: Preserving intellectual assets in a knowledge economy. Electronic publication by CogNexus Institute, http://cognexus.org/dom.pdf (2001)
6. Curtis, T.: Top 10 Tips for Using the Web More Effectively, http://airsafe.com/netuse/web/effectiv.htm (2002)
7. Di Giacomo, M. et al.: MyLibrary, A Personalization Service for Digital Library Environments, In: Proceedings of the 2nd DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries, Ireland (2001)
8. Dodge, M. and Kitchin, R.: Atlas of Cyberspace. Addison Wesley (2001)
9. Franco, A. and Palladino, R. L.: Finding Quality Information on the World Wide Web, http://www.iona.edu/faculty/afranco/iima/webliog.htm (1999)
10. Google Technology, http://www.google.com/technology/
11. Google Toolbar, http://toolbar.google.com/
12. Google Web APIs, http://www.google.com/apis/
13. Hirst, K. K.: Mining the Web: Techniques for Bridging the Gap between Content Producers and Consumers. First Monday, 2(10), (1997)
14. Kleinberg, J. and Lawrence, S.: The Structure of the Web. Science, 294(5548), (2001) 1849-1850
15. Lager, M.: Spinning a Web Search, In: Proceedings of the Conference Sponsored by the Librarians Association of the University of California, Santa Barbara and Friends of the UCSB Library, http://www.library.ucsb.edu/untangle/lager.html (1996)
16. Lam, S.: The Overview of Web Search Engines http://citeseer.nj.nec.com/lam01overview.html (2001)
17. Lawrence, S. and Giles, C. L.: Accessibility of information on the Web. Nature, 400(8), (1999) 107-109
18. Lempel, R. and Moran, S.: SALSA: the stochastic approach for link–structure analysis. ACM Transactions on Information Systems, 19(2), (2001) 131–160

19. Maxon, J., et al.: INFOMINE: promising directions in virtual library development. First Monday, 5(6), (2000)
20. Microsoft Corporation: Creating Custom Explorer Bars, Tool Bands, and Desk Bands. http://msdn.microsoft.com/library/default.asp?urWlibrary/en-us/shellcc/platform/Shell/programmersguide/shell_adv/bands.asp
21. Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, Portland, Oregon, USA (2002)
22. Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, TX, USA (2003)
23. Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries, Tucson, AZ, USA (2004)
24. Proceedings of the 5th International Conference on Asian Digital Libraries, Singapore (2002)
25. Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries, Rome, Italy (2002)
26. Proceedings of the 6th International Conference on Asian Digital Libraries, Kuala Lumpur, Malaysia (2003)
27. Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries, Trondheim, Norway (2003)
28. Renda, M. E. and Straccia, U.: A Personalized Collaborative Digital Library Environment, In: Proceedings of the 5th International Conference on Asian Digital Libraries (2002)
29. The Sheridan Libraries of The Johns Hopkins University: Evaluating Information Found on the Internet, http://www.library.jhu.edu/elp/useit/evaluate/index.html (2002)
30. Twidale, M. B. and Nichols, D. M.: Designing Interfaces to Support Collaboration in Information Retrieval, Interacting with Computers, 10(2), (1998) 177-93
31. University Libraries at Virginia Tech: Bibliography on evaluating Web information, http://www.lib.vt.edu/research/evaluate/evalbiblio.html
32. University of New England: Locating information on the Web, http://www.une.edu.au/library/infolit/locatinginfo.htm (2001)
33. Wesleyan University Library: Finding Useful Internet Resources, http://www.wesleyan.edu/libr/tut/rst10.htm

Understanding and Supporting Knowledge Flows in a Community of Software Developers

Oscar M. Rodríguez[2], Ana I. Martínez[2], Jesús Favela[2],
Aurora Vizcaíno[1], and Mario Piattini[1]

[1] Alarcos Research Group, University of Castilla-La Mancha
Escuela Superior de Informática, España
{Aurora.Vizcaino,Mario.Piattini}@uclm.es
[2] CICESE, Computer Science Department, México
{orodrigu,martinea,favela}@cicese.mx

Abstract. Knowledge sharing is a collective process where the people involved collaborate with others in order to learn from them. This effort creates communities where each member cooperates by sharing knowledge about a common domain. An example of these kinds of communities is software maintenance groups, where their members must collaborate with others, and share their knowledge and experience in order to complete their assignments. This paper presents a study carried out in two software maintenance groups to understand how the knowledge flows through these groups, that is, how their members share their knowledge when they perform their activities. The approach used to model the flows of knowledge and to identify the problems that affect that flow are described, as well as the main problems detected, and how we are trying to solve them with an agent-based knowledge management system.

1 Introduction

Nowadays, knowledge has becoming a very important factor for organizations' competitive advantage. In fact, intellectual capital is one of the most important assets for many organizations [9]. Therefore, it is important to increment the sharing of knowledge through their communities, since this can increase their intellectual capital and their performance [10].

Knowledge sharing is a collective process where the people involved collaborate with others in order to learn from them [9]. This effort creates communities where each member cooperates by sharing knowledge about a common domain. These are called communities of practice [9, 10].

An example of communities of practice is software maintenance groups. Software maintenance is a knowledge intensive work where maintainers must collaborate with other members of the team, and share their knowledge and experience in order to complete their assignments. Even though, maintainers mainly use their own experience, generally they do not have enough knowledge to complete their work and need to consult other sources of information such as other members of the team in order to finish some assignments [21]. However, this could be a difficult task because those sources are often limited, inaccessible, or unknown [17].

Knowledge management provides methods and techniques that can help software organizations to increment the collaboration of their members; for example, by sup-

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 52–66, 2004.

porting the sharing of knowledge between them. Even though this could bring many benefits to software organizations, little work has been done to apply it in the software maintenance process [16].

We think that providing software maintenance groups with tools that facilitate their members to collaborate and share their knowledge, the performance of these communities can be incremented. However, before these tools can be developed, some questions must be answered [1], such as: what kinds of problems could be solved?, what is the knowledge involved in the activities performed by the maintenance groups?, and how they share that knowledge?.

In order to answer the above questions, we conducted two case studies in two software maintenance groups. The case studies were focused on identifying how the knowledge flows through the teams, what are the problems that affect these flows and how we can address them with a knowledge management tool. This paper presents this work. The content is organized as follows: section 2 introduces some theoretical background about knowledge and *knowledge flow*. After that, section 3 presents our approach for modelling *knowledge flows* in cooperative communities, and how scenarios can be used to show the problems that affect the flow of knowledge. In section 4 some findings of the study are discussed. Then, the work related to this research is in section 5 and section 6 presents a multi-agent knowledge management system that was designed and implemented from the results of the case studies to promote collaboration between the members of a maintenance group. Finally the conclusions of this work are in section 7.

2 Theoretical Background

There are different conceptions of the types of knowledge; most authors agree that knowledge could have two forms, explicit and tacit. Explicit knowledge can be expressed in words or numbers and shared in form of data, scientific formulate, specifications, manuals, audio, video, etc. Thus, explicit knowledge is easy to share. Tacit knowledge is more personal and difficult to formalize, therefore is harder to communicate and share to others [15].

On the other hand, knowledge creation is a process where tacit knowledge becomes explicit and vice versa. These transfers between the two kinds of knowledge generate four conversion mechanisms: *socialization,* when tacit knowledge creates more tacit knowledge by people communicating with others sharing experiences; *externalization,* when tacit knowledge becomes explicit by formalizing it in documents, reports, etc.; *combination,* when explicit knowledge creates more explicit knowledge by combining information that resides in formal sources like documents, reports, etc.; and *internalization,* when explicit knowledge creates tacit, for example by consulting formal sources of information to obtain knowledge [15].

When people need to do some activity, they mostly use their own knowledge and experience, but when this is not enough, they use different techniques to consult other sources to obtain information that will help them to make the decisions required [4]. These sources could be documents, or other members of the community.

As we stated before, these communities of people that collaborate to share their knowledge and learn together are called communities of practice [9, 10]. In order to observe how the knowledge flows through the communities of practice of an organi-

zation, it is important to identify the knowledge they need and how they use it and obtain it while they perform their activities and make decisions. Next we describe how this process was done in the case studies carried out.

3 The Study

We conducted two case studies in two software maintenance groups. The first group was from a department in charge of the development and maintenance of the software systems used to the management of a scientific research institution. This department maintains applications of 5 domain areas: finances, human and material resources, academic productivity, and services for students. The second group was from a company that develops and maintains software for the management of telephone services. This company has more than 4000 clients to whom it provides services.

The studies were performed based on interviews, observation and analysis of documentation. All the interviews were recorded for later analysis. We also performed a bibliographic research to compare our findings with those that have been described in the literature.

The methodology followed in the study has four main steps: in stage 1 the main sources of information were identified and classified (documents and people); then, in stage 2, the knowledge that those sources had was defined and classified; in the third stage the processes and activities performed by the group were modelled to identify the people involved, how they collaborate to fulfil their tasks, and how the knowledge and sources interact in those activities; finally, in stage 4 the main problems that can affect the flow of knowledge were highlighted through the definition of scenarios. In the next section, the approaches used to identify the *knowledge flows* and the problems that affect these flows are described.

3.1 Identifying *Knowledge Flows* in a Software Maintenance Group

The identification of the *knowledge flows* in the software maintenance groups was based on the generic model showed in Figure 1. The model considers that, when a maintainer must perform some activity, s/he uses her/his knowledge and experience to analyze the problem and find possible solutions. Frequently, maintainers do not have all the knowledge they need, so they consult other sources of information to obtain it [21]; then, they use all these knowledge to decide which solution to implement. The processes of analyzing the problem and making decisions generate new knowledge and experience for the maintainers that can be shared later with the rest of the team. They obtain and share knowledge by using the conversion mechanisms of the knowledge creation process [15].

The conceptual model enables the identification of the knowledge involved in the activities performed by the members of the team. Here, the use of a graphic process modelling technique could be very useful [13]. An example of the latter is presented in Figure 2, which shows the main activities performed in the definition of the modification plan performed for one of the groups studied. The model also shows the people involved in those activities, the knowledge they have together with their relevance to the activities modelled, and the main sources used, created or modified in the activities.

Fig. 1. This shows the generic conceptual model of the *knowledge flow* in a process decision making.

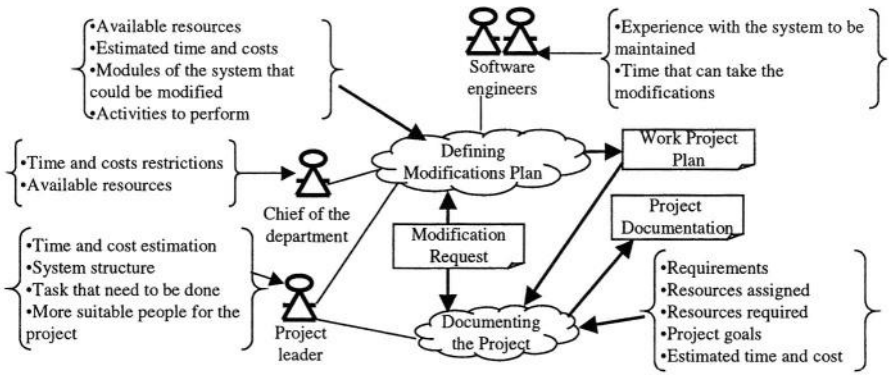

Fig. 2. This is an example of a model of activities performed by members of a maintenance group.

Once the activities have been modelled, the next step is to define the decisions that must be made by the people involved. To do that, we used the schema showed in Table 1. This schema helps to identify the knowledge that people in charge of the activities must have to make the decisions needed to fulfil their activities, and the sources of information they consult to help them making those decisions. At this step, it is important to identify the mechanisms that people can use to consult the sources, as well as those used to share the knowledge generated from the development of the activities; for example, the documentation of the plan of modifications of Figure 2.

The analysis of the activities performed by the maintainers, using the graphical model and the information from the tables, facilitates to identify when they need to consult other members to obtain information, and when they must collaborate and share their knowledge. The identification of the mechanisms that the community uses to share knowledge helps to understand how the knowledge actually flows through the community. Finally, all these can be used to identify what kinds of tools might be useful to increment the flow of knowledge in the team.

Table 1. Schema used to identify knowledge in decision making.

Role	Project leader	
Activity	To define modification plan	
Decision	To define required resources	
	To define main tasks to perform	
	To assign tasks to the participants of the project	
	To estimate the time the project would consume	
Knowledge	Previous projects experiences	
	Requirements and restrictions of the project	
	Abilities and experience of each of the possible participants of the project	
Sources of information		
Name	**Information**	**Consulted at**
Chief department	Available resources; time and cost restrictions	Telephone, Physical address, Email
Software engineers	Experience with the system that will be modified; time that could consume the modifications; time availability	Telephone, Physical address, Email
Previous projects documentation	Resources required by previous projects	Documents files, modifications logbook

3.2 Using Scenarios to Show *Knowledge Flow* Problems

The main goal of the identification of the *knowledge flow* in communities is to understand how this is occurring in order to provide mechanisms that help to increment it. Scenarios can be useful to do this, since these enable the identification of design requirements for software systems. Also, they make feasible the participation of users during the requirement specification stage [3].

A scenario is a textual description (like a history) about the activities that people might engage in while pursuing a particular concern [2]. Particularly we identified and defined scenarios that illustrate the problems that affect the flow of knowledge in the groups studied. These scenarios were classified in two main problems. The first one is related to the management of information and knowledge sources; for example, finding experts, document management, etc. The second one is related to the management of lessons learned; for example, experiences reuse to estimate the resources that a project might consume, to identify the modules and source files that might be changed in a modification request, to identify other modules of the system that could be affected by the changes, etc. Next we present two examples of scenarios, one of each kind of problem.

Expert Finding (Information Sources' Management). Mary is a software engineer that must make some changes in the finances system. Since her knowledge in the domain of finances is not good enough, the changes to the system are taking more than a week of the estimated time. At the end of the week, Susan, the chief of the department, while she was checking the advances of the project, she detects the delay and asks Mary the reasons of that delay. Mary tells Susan the problem and since Susan has experience with finances, she tells Mary how the problem could be solved. Finally, Mary solves the problem the same day.

Identifying Files Affected by the Changes (Lessons Learned). While Tom was performing changes in the structure of a table of the data base of one of the systems maintained by the team, he observed that there were some files of the system that

would also have to be changed. Because of this, Tom wrote a report in order to not forget what modules he would have to change if he had to modify that table in the future. Later, Tom left the team, and Mike took his place. One day, Mike needed to modify that table but, because he did not know about the report made by Tom, he did not consult it, and when he made the changes and tested them, he thought that everything was fine. Then, when the system was executed, two modules started to fail, as Tom had specified in the report before.

Discussion. As can be seen from the scenarios, there was knowledge in the team that was not used because of ignorance of the people who could have benefited with it. This fact has already been commented on by other authors, such as Szulanski [18] who found that the number one barrier to knowledge sharing was "ignorance": the sub-units are ignorant of the knowledge that exists in the organization, or the sub-units possessing the knowledge are ignorant of the fact that another sub-unit needs such knowledge. Thus, it is important to solve this problem in order to increase the *knowledge flow* in the community group. In next section, some of the findings obtained from the analysis of the models and scenarios defined in the case studies carried out are discussed.

4 Findings' Discussion

Our interest has focused on understanding how the knowledge flows in the community in order to provide mechanisms to increment that flow. In the study, we found tree main aspects that we consider important to the flow of knowledge. These findings are discussed in this section; also some fragments of the interviews are presented to support our discussion.

1) Team Work. Team work is a good technique to promote the sharing of knowledge in the group. When people must collaborate to accomplish one task, they share their knowledge by discussing and making decisions together.

> *when I started working here, I had to work with a [partner], he knew about the tools, and I knew about the procedures in the domain area we were working on, I had more experience in some programs and he in others, so we share the work in that way*

When the result of the job of one member of the community is needed by others, they have to find ways to share information.

> *I used to use the logbook because my work was related to others. The secretary took the calls and wrote the problems in the logbook, then she sent it to me, I solved it, and I documented it in the logbook.*

In contrast, when people work alone, they usually do not share knowledge with others.

> *but now, ... my work is not related to almost anybody, so, I am not documenting in the logbook... as I now remain almost isolated!*

These remark the importance to promote team working in order to increase the collaboration and sharing of knowledge through the members of a community. An ex-

ample of how to do this in software maintenance communities is by implementing pair programming techniques [14].

2) Experience Reuse. Lessons learned are a good way for knowledge and experience reuse. We found that lessons learned could be useful, for example, to identify the modules that need to be changed and those that could be affected by the changes. Moreover, past experiences help the members of the team to make estimations about the resources that the modifications can consume. Therefore, it is important to identify and capture these lessons learned in order to make them available to the team.

> *the logbook helps me to do changes that aren't done every day, but once or twice a year. If I don't know very well what to do, then, I look at the logbook, I see how was done before, and then I do it*

3) Know What the Team Knows. Knowing which is the knowledge of the group is important for knowledge sharing, if people do not know what knowledge is available in some way to consult, this will be lost and never used.

> *but, how she could know (what I knew) it if I didn't tell her before*

It is important to provide tools that enable to the members of the team to know what sources of information they can consult and what kind of knowledge or information they can obtain from those information sources. This is one of the main problems that knowledge management tools must solve [16]. One approach that can help to solve this problem is to use software agents. Agents can act like a personal assistant that knows the engineer's profile and identifies the needs of knowledge and search for sources that can help the engineer to fulfill his/her job [11]. Based on this, we have designed a multi-agent based knowledge management system for software maintenance. This is focused on supporting experience reuse and helping to know what the group knows enabling the collaboration between the members of the team. In the next section we will describe briefly the related work in this area, to continue next with the description of the system.

5 Related Work

Agents have characteristics that make them a good alternative for developing systems for supporting knowledge and information management in collaborative communities [19]. For example, Guizzardi et al. [7, 8] have proposed the use of agent oriented methodologies for analyzing and designing systems for supporting collaborative learning communities such are communities of practice; they have designed an agent-based architecture to support collaborative learning in education [6]. There are other approaches that can be useful to tackle some of the problems identified in the case studies carried out for this work; for example, Jasper [5], a multi-agent system which agents can generate profiles of their users while they consult information, and inform to other agents about sources found which can be relevant to the users of these last agents. However, these approaches are too general and do not address some specific problems of software maintenance groups; for instance, to identify which knowledge is needed by the maintainer for solving a particular maintenance request. To address this problem, it is not sufficient to know the engineer's profile; it is also required to know about the context of the activities the engineer must perform in order to search

for information sources that can help the maintainer to fulfill his/her job. To accomplish this, they should not be used only as an external application for searching into a knowledge or information repository when the maintainers decides s/he needs information about something, agents must be integrated into the work environment of the maintainers, for example, providing support for managing the projects and tasks where maintainers are working on.

On the other hand, there are few works that use agents to support knowledge management in software maintenance; Vivacqua [20] developed an agent based expertise locator which has been applied in a software maintenance domain, in this system users must define their knowledge profiles; this is one of the problems why these kinds of systems are not well accepted for software organizations [1]; we think it is important that these profiles can be generated and managed automatically by the agents. Mercer and Greenwood [12] propose a multi-agent architecture for knowledge sharing focused on helping to develop good programming practices.

As can be seen, there are different works that address different aspects of knowledge management needs in collaborative communities. However, they are not directly oriented to support *knowledge flows* for a particular community need as the found in our case with software maintenance groups. In the next section we describe the system developed.

6 An Agent-Based Knowledge Management System for Software Maintenance

There are several reasons why agents are good technical alternative for the development of knowledge management software [19]. First of all, agents are proactive; this means they act automatically when it is necessary. One of the obstacles to implementing knowledge management in software organizations is that employees do not have time to introduce or search for knowledge [16]. During their daily work, different kinds of knowledge and experiences are created, but not captured in a formal representation; they are only in the engineer's head. To increment the *knowledge flow* in software maintenance, it is important to address this problem without increasing the maintainer's work. Agents can capture and manage information automatically; for example, acting like a personal assistant [11].

Moreover, agents can manage both distributed and local knowledge. This is an important feature since the software maintenance knowledge is generated by different sources and often from different places.

Another important issue is that agents can learn from their own experience. Consequently, the system is expected to become more efficient with time since the agents have learnt from their previous mistakes and successes [11].

Finally, in a multi-agent system each agent may use different reasoning techniques depending on the situation. For instance, they can apply ID3 algorithms to learn from previous experiences and case-based reasoning to advise a client how to solve a problem.

Based on the problems detected in the case studies, and on the agents' characteristics to facilitate the solution of some of the problems, we have developed a prototype of a multi-agent knowledge management system. The preliminary tests made to the prototype had showed that it can help to promote the collaboration and sharing of knowledge between the members of a maintenance group; this is by informing to the

maintainers who has knowledge that can be relevant to the activities they are performing. Next the multi-agent architecture of the system is presented; also a scenario that shows how the system can help to promote collaboration and sharing of knowledge is described.

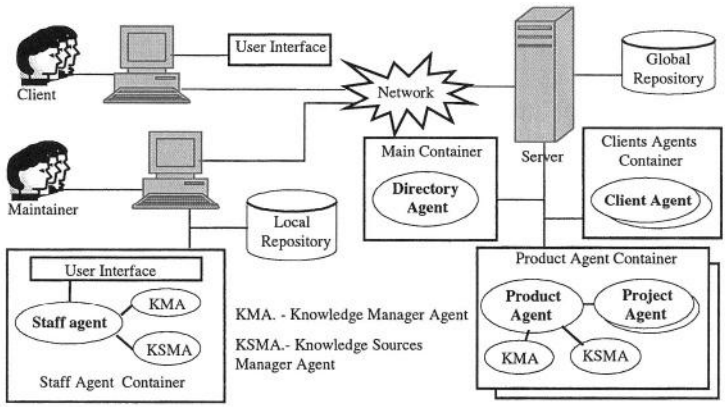

Fig. 3. Multi-agent architecture of the system.

6.1 Multi-agent Architecture

The system's architecture has five main types of agents (see Figure 3). Four of them represent the main elements identified in the case studies carried out. Next we describe each agent.

The *staff agent* is a mediator between the maintainer and the system. It acts like an assistant to him. This agent monitors the maintainer's activities and requests the KMA to search for knowledge sources that can help him to perform his/her job. This agent has information that could be used to identify the maintainer profile, such as which kinds of knowledge s/he has or which kinds of sources s/he often consults.

The *product agent* manages information related to a product, including its maintenance requests and the main elements that integrate the product (documentation, source code, databases, etc.). The main role of this agent is to have updated information about the modifications carried out in a product and the people that were involved in it.

Each maintenance project is managed by a *project agent,* which is in charge of informing the maintainers involved in a project about the tasks that they should perform. This is done through the staff agent. The project agents also control the evolution of the projects.

The *client agent* manages information related to the maintenance requests performed by a client. There is one agent of this kind per client. Its main role is to assist them when they send a maintenance request, directing it to the corresponding product agent.

The *directory agent* manages information required by agents to know how to communicate with other agents that are active in the system. This agent knows the type, name, and electronic address of all active agents. Its main role is to control the different agents that are active in the system at each moment.

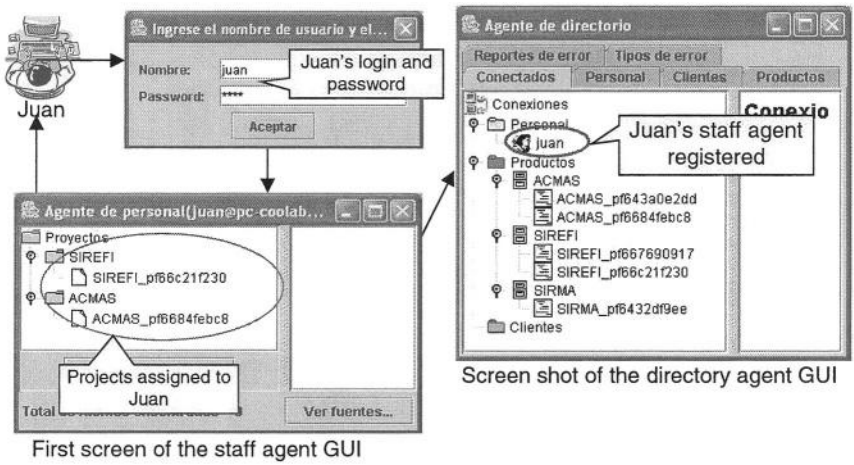

Fig. 4. Screen shots showing the user registration in the system.

The architecture also has two types of auxiliary agents: The Knowledge Manager Agent (KMA) and the Knowledge Sources Manager Agent (KSMA). The KMAs are in charge of the management of the knowledge base. These kinds of agents provide support by capturing and searching for knowledge in the knowledge base. The KSMAs have control over the knowledge sources, such as electronic documents. These agents know the physical location of those sources, as well as the mechanisms used to consult them. Their main role is to control access to the sources.

6.2 Promoting Collaboration and Sharing of Knowledge: A Scenario of Use

To exemplify how the tool helps to promote the collaboration and sharing of knowledge in the maintenance groups, we are going to describe a scenario of use.

Juan is a maintainer assigned to two modification projects. When Juan starts the system and provides his login and password, his staff agent is registered by the directory agent, and then it shows a window with the list of projects assigned to Juan (see Figure 4). This information is provided by the projects' agents in charge of those projects.

Juan decides to solve one of those projects which is a problem reported by a client. When Juan presses the corresponding button, the staff agent shows a window with the information of the problem report and generates some rules to request the KMA to search for knowledge sources that can help Juan to solve that problem. To create the rules, the staff agent tries to identify the knowledge that the engineer would need to carry out the assignment. Also the agent considers the types of sources the engineer consults, assigning more relevance to the sources that the engineer consults most frequently. When the search has finished, the KMA sends a message to the staff agent informing it about the sources found. The staff agent displays a message with the number of sources found in order to inform the engineer (see figure 5).

Juan decides to look for the sources found, so he chooses the corresponding button in the staff agent GUI, and the agent displays a window with the list of sources grouped by categories. Then, Juan selects an item of the sources list which corre-

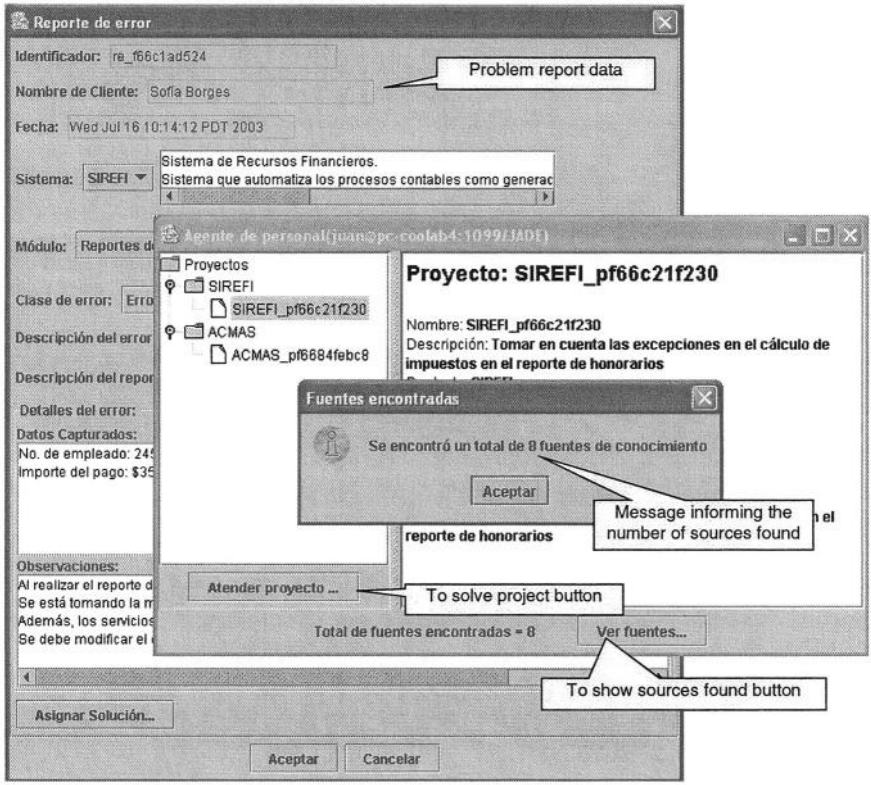

Fig. 5. Screen shot that shows how the system informs to the user the total of knowledge sources found.

sponds to a staff member called Felipe, and the agent shows some information related to Felipe: knowledge that he has, location, etc. (see Figure 6). Juan looks for the knowledge that Felipe has, and finds out that Felipe could help him to solve the problem, since he has experience in the domain area of the system where Juan must do the changes. Because of this, Juan decides to consult Felipe calling him to the telephone number provided by the tool. Finally, Felipe brings Juan some advices that help him to solve the problem.

As can be seen in the scenario just described, the tool provided support to Juan to find a colleague that had the knowledge to help him solve the problem. As a consequence Juan consulted Felipe, who gave Juan some advices that facilitated him to solve the problem. Thereby, the tool helped to promote the collaboration and sharing of knowledge between Juan and Felipe. A similar scenario was identified in the case studies carried out, but, in that case, the maintainer did not consult his colleague at time because he did not know that the colleague had knowledge that could be useful to help him to do the changes.

The tool not only shows information about other colleges, it also presents different kinds of sources of information, such as documents, previous problem reports or maintenance requests which could be related with the project the maintainer must

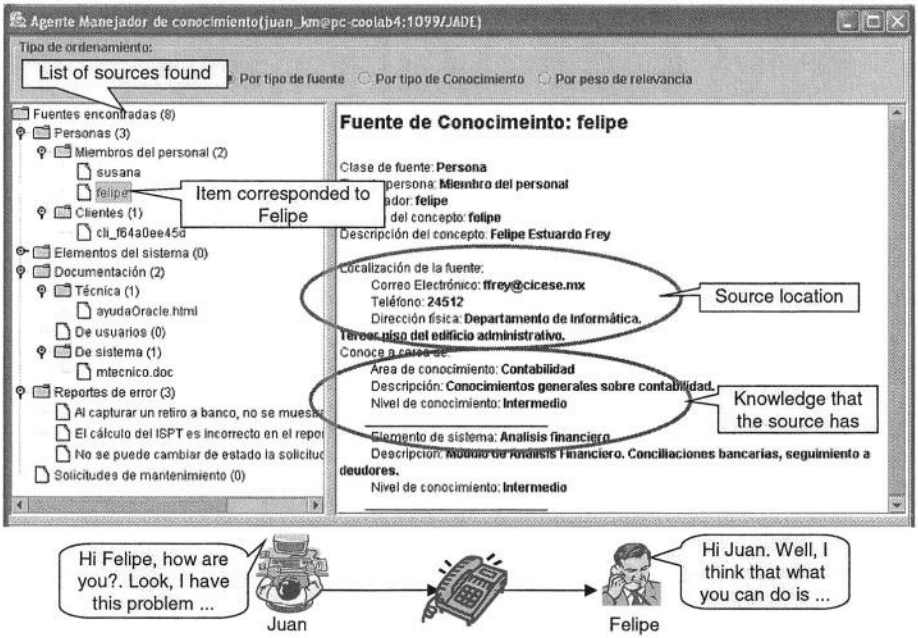

Fig. 6. Screen shot showing the list of sources found; we can observe that Felipe is one of these knowledge sources.

solve. Therefore, the tool helps maintainers to consult sources where s/he can obtain information or knowledge to help him/her to make decisions required by the activities s/he must carry out. Moreover, the tool provides an option for capturing the solution or the changes made to the system by the maintainer, thus, these information can be later used for others to solve similar problems. In other words, the tool enables maintainers to consult information sources to obtain knowledge that can be helpful to make decisions about the tasks they must carry out, and also to share the knowledge generated by making those decisions by capturing the solutions and changes made by maintainers (see Figure 7), in order to make them available for others; a process similar to the *knowledge flow* model illustrated in Figure 1.

7 Conclusions and Future Work

In this paper we presented a study carried out to understand how the knowledge flows through two communities of practice in software maintenance. The study focused on identifying the knowledge needed by the members of the group while they perform some tasks and make decisions, and how they obtain and share that knowledge. To do this, we used some qualitative techniques and proposed an approach for modeling the flow of knowledge in software maintenance teams.

On the other hand, the study has provided some aspects that must be considered to increase the flow of knowledge in the groups studied; for example, help them to know the knowledge they can consult and where they can find it. We think that software

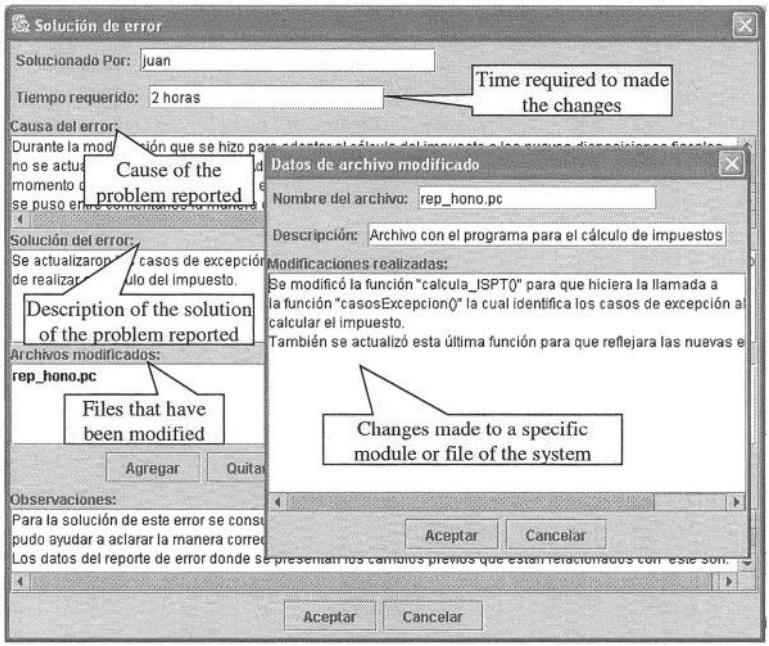

Fig. 7. Example of a solution of a problem reported and the description of the changes made in a modified file.

agents can help to solve some of these problems. Thus, we have designed an agent based knowledge management system for software maintenance that helps to solve some of the problems detected in the studies, and at the same time, helps to increment the flow of knowledge in the software maintenance groups, by promoting the collaboration and sharing of knowledge between their members. The basic characteristics of this system have been obtained from the results of the study. A first prototype has been developed and tested with scenarios that show how a tool with these characteristics can address some of the problems detected in the case studies carried out.

At the moment we are working on improving the methodology to provide more and better information for designing agent-based knowledge management systems for supporting *knowledge flows* in communities of practice in software organizations. With the information obtained by applying the improved methodology, we are planning to improve the system's prototype, in order to perform another case study to evaluate how the tool is perceived by software maintainers. To do this, some aspects must be considered first, such as how the tool can be integrated into the maintainer's environment.

Acknowledgements

This work is partially supported by CONACYT under grant C01-40799 and the scholarship 164739 provided to the first author, and the MAS project (grant number TIC2003-02737-C02-02), Ministerio de Ciencia y Tecnología, Spain.

References

1. Aurum, A., Jeffery, R., Wohlin, C., Handzic, M., Preface, in Managing Software Engineering Knowledge, Aurum, A., et al., (eds.), Springer, Berlin, (2003), ix-xv.
2. Carroll, J. M., Rosson, M. B., Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario, ACM Transactions on Information Systems, 10(2), (1992), 181-212.
3. Chin, G. J., Rosson, M. B., Carroll, J. M., Participatory Analysis: Shared Development of Requirements from Scenarios, in Conference on Human Factors in Computing Systems (CHI97), Atlanta, GA, USA, ACM/SIGCHI, (1997), 162-169.
4. Choo, C. W., La Organización Inteligente: el Empleo de la Información para dar Significado, Crear Conocimiento y Tomar Decisiones, Oxford University Press, Oxford, USA, (1999).
5. Davies, J., Weeks, R., Revett, M., Jasper: Communicating Information Agents for WWW, presented at the 4th International Conference on the World Wide Web, Boston, USA, (1995), available at http://www.w3j.com/1/davies.180/paper/180.html, last visited 23-06-2004.
6. Guizzardi, R. S. S., Aroyo, L., Wagner, G., Agent-oriented Knowledge Management in Learning Environments: A Peer-to-Peer Helpdesk Case Study, in Agent-Mediated Knowledge Management, Heidelberg, Springer, (2003), 15-22.
7. Guizzardi, R. S. S., Perini, A., Dignum, V., Providing Knowledge Management Support to Communities of Practice through Agent-oriented Analysis, in Proceedings of the 4th International Conference on Knowledge Management (I-KNOW'04), Granz, Austria, (2004), available at http://wwwhome.cs.utwente.nl/~souza/publications/guizzardi-perini-dignum-iknow04.pdf, last visited 23-06-2004.
8. Guizzardi, R. S. S., Perini, A., Dignum, V., Using Intentional Analysis to Model Knowledge Management Requirements in Communities of Practice, CTIT Technical Report, 03-53 University of Twente, (2003), available at http://www.ctit.utwente.nl/library/techreports/ tr03.doc/ index.html, last visited 23-06-2004.
9. Huysman, M., Wit, D. d., Knowledge Sharing in Practice, Kluwer Academic Publishers, Dordrecht, (2000).
10. Lesser, E. L., Storck, J., Communities of practice and organizational performance, IBM Systems Journal, 40(4), (2001), 831-841.
11. Maes, P., Agents that reduce work and information overload, Communications of the ACM, 37(7), (1994), 31-40.
12. Mercer, S., Greenwood, S., A Multi-Agent Architecture for Knowledge Sharing, in Third International Symposium From Agent Theory to Agent Implementation at EMCSR 2002, Vienna, Austria, (2002), available at http://www.ai.univie.ac.at/~paolo/conf/at2ai3/, last visited 23-06-2004.
13. Monk, A., Howard, S., The Rich Picture: A Tool for Reasoning about Work Context, Interactions, 5(2), (1998), 21-30.
14. Natsu, H., Favela, J., Morán, A. L., Decouchant, D., Martínez-Enriquez, A. M., Distributed Pair Programming on the Web, in Fourth Mexican International Conference on Computer Science, Tlaxcala, México, IEEE Computer Society, (2003), 81-88.
15. Nonaka, I., Takeuchi, H., The Knowledge-Creating Company, Oxford University Press, (1995).
16. Rus, I., Lindvall, M., Sinha, S. S., Knowledge Management in Software Engineering: A State of the Art Report. Data & Analysis Center for Software: ITT Industries: Rome, NY. (2001), available at http://www.dacs.dtic.mil/techs/kmse/kmse.html, last visited 23-06-2004.
17. Seaman, C., The Information Gathering Strategies of Software Maintainers, in Proceedings of the International Conference on Software Maintenance, (2002), 141-149.

18. Szulanski, G., Intra-Firm Transfer of Best Practices Project, in American Productivity and Quality Centre, Houston, Texas, (1994), 2-19.
19. van Elst, L., Dignum, V., Abecker, A., Agent-Mediated Knowledge Management, in International Symposium AMKM 2003, Stanford, CA, USA, Springer, (2003), 1-30.
20. Vivacqua, A. S., Agents for Expertise Location, in Proceedings of the AAAI Spring Symposium on Intelligent Agent in Cyberspace, Stanford, USA, AAAI Press, (1999), 9-13.
21. Walz, D. B., Elam, J. J., Curtis, B., Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration, Communications of the ACM, 36(10), (1993), 63-77.

A Framework for Asynchronous Change Awareness in Collaboratively-Constructed Documents

James Tam and Saul Greenberg

Department of Computer Science, University of Calgary
Calgary, Alberta, T2N 1N4 Canada
{tamj,saul}@cpsc.ucalgary.ca

Abstract. *Change awareness* is the ability of individuals to track the asynchronous changes made to a collaborative document or surface by other participants over time. We develop a framework that articulates what change awareness information is critical if people are to track and maintain change awareness. Information elements include: knowing *who* changed the artifact, *what* those changes involve, *where* changes occur, *when* changes were made, *how* things have changed, and *why* people made the changes. The framework also accounts for people's need to view these changes from different perspectives: an *artifact-based* view, a *person-based* view, and a *workspace-based view.*

1 Introduction

People often collaborate for the purpose of creating and developing a work artifact over time. This happens when people co-author papers, or iterate designs from conception to final form, or negotiate a plan through an evolving blueprint. The participation of all partners is vital, perhaps due to the group's particular combination of skills and expertise, or because all participants are interested stakeholders, or because there is too much work for one person to do by themselves, or because involvement is required if participants are to buy into the final outcome.

While many episodes of collaboration often occur in face to face meetings over the work artifact, others frequently occur asynchronously. Asynchronous collaboration and evolution of the artifact can happen in several ways.

- People may explicitly pass the artifact back and forth for comments and revisions (i.e., 'it is your turn, give it back to me after you have worked on it').
- Individuals may work on the artifact as time and opportunities arise, without explicitly coordinating this with the other participants.
- The group may drift in and out between collocated and asynchronous work (e.g., a group may begin work in an extended face to face meeting, but members may leave and return to the meeting over its course).

In same-time collaborations, we already know that people use *workspace awareness* not only to follow actions of others, but to understand and respond to any changes others make to the workspace artifact [5] (see Section 3). The problem is that when people interact asynchronously this awareness disappears; changes are only understood if one person tells the other what they have done (e.g., through prior coordinat-

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 67–83, 2004.

ing talk, on-going emails, or notes attached to the artifact), or if the person can somehow understand the changes made by inspecting the artifact. If people cannot understand what has changed, collaboration can quickly spiral out of control. Missed changes made by one person can unintentionally wreak havoc on the work of others or even the entire project.

Within this context, our research interest is on *asynchronous change awareness of artifacts* (which we call change awareness for short), defined as the ability of individuals to track the asynchronous changes made to a collaborative document or surface by other participants. Our research concentrates primarily on how people maintain change awareness by inspecting the changed document, rather than on how change awareness is transmitted by other communications e.g., verbal or textual dialog that occurs directly between people outside the confines of document.

Because change awareness is a broad subject, our immediate goal in this paper is to contribute a framework of the critical information people need if they are to maintain change awareness. First, we set the scene with several examples of failures that arise when people have inadequate change awareness, and then describe how current systems try to provide this information. Second, we summarize Gutwin's earlier framework for workspace awareness for real time interactions [5], for it acts as a theoretical precursor to our own work. Third, we introduce and describe in detail our framework for change awareness. We close by discussing several implications this framework has to practitioners and implementers of change awareness systems.

2 Motivations and Related Work

Several true incidents give an example of the consequences of missed changes.

The Town of Canmore in Canada has an administrative office that oversees all subdivision plans submitted by developers. In this process, the developers and the administrative office negotiate the plan through many back and forth exchanges. The administration would ask for changes so that the subdivision fit the needs of the town, and the developers would respond by incorporating these (and perhaps other) changes in a modified plan of the subdivision development. In one on-going subdivision plan, the developers added a gate to the main road entrance, a security measure that inhibits 'outsiders' from entering the grounds. The administrative office did not notice this addition, and approved that particular version of the plan. This oversight was only seen after the subdivision was built with the gate in place. The gate generated widespread controversy, where it made the front page of the local newspaper and became a heated issue for the town council, the town population, and the developers. Because Canmore was a small town fostering community values, the townspeople felt that this gated community created a private enclave that violated the sense of community held by its population, and that it would also set a bad precedent for future developments. The developers, on the other hand, believed that they had followed the planning process correctly, and because the plan had been approved they had the right to implement their vision. The developers were adamant that they had not tried to 'slip in' this gate amongst the other changes in the plan, and indeed had told a staff member about it. The administrative staff said that the key staff members did not see the addition of

the gate; it was a visually small change in a complex plan that had many other changes within it. The town council stated that "they didn't know about the plan and wouldn't have approved it had they known" (reported in the Rocky Mountain Outlook, March 11, 2004). What happened was that the administrators lacked awareness of what had changed between documents, and thus missed a small but critical detail.

Another true example focuses on missed changes within TeamWave Workplace, a commercial room-based groupware environment [4]. Community members could enter a virtual room at any time and change its contents, i.e., a collection of persistent groupware applications and documents scattered on the wall of the room. The only difference between synchronous and asynchronous work was whether people happened to inhabit and make changes within the room at the same time or at different times. Independent evaluators of TeamWave found that its major usability problem was that people had difficulty maintaining awareness of whether anything had changed since the last visit, and what those changes concerned [16]. "Participants reported that they were unable to effectively and efficiently detect modifications to artifacts made by others in the rooms they were using. TeamWave users devised [email workarounds] to provide the information that they needed" (pp. 5). The problem was that TeamWave users had to resort to manual techniques to monitor changes. To see if anything had changed, they had to start and login to the application, navigate to a room, and visually inspect it. Because this is a heavyweight and error-prone process, people did not visit rooms often enough to notice if anything had changed, and when they did visit a room some changes were easily missed. In turn, this meant that people were increasingly reluctant to leave time-critical information in a room, for they knew that others were unlikely to notice it until too late. The usefulness of the entire system suffered as a consequence.

Our third example illustrates the effort people go through if a system does not explicitly provide a mechanism revealing change awareness information. A typical strategy is to compare the two document versions – by memory or visual inspection – and to try and spot changes and decipher their meaning. This is both a daunting and error prone task. For example, Figure 1 (a) and (b) show a before and after version of a UML class diagram respectively. Although a person may be able to eventually determine all the differences, it requires much time and effort to do so, and changes are easily missed. Try this for yourself: see how many differences you and your collaborators can spot in (say) 30 seconds. Have a first person try it when both images are side by side (for direct visual comparison). Have a second person try it when the images are on opposite sides of the same piece of paper (for short-term memory comparison). Have a third person look at the first image on the first day for 30 seconds, and then have them examine the second image on the second day for changes (for long-term memory comparisons).

Of course, the document itself can help reveal changes. For example, the version of the UML diagram depicted in (c) highlights specific blocks of changed items by bolding them and by muting unchanged items (coloring would be preferred, but this is not possible in this black and white reproduction). While limited, even this simple technique clearly allows people to spot changes faster and with fewer errors than by using manual comparisons.

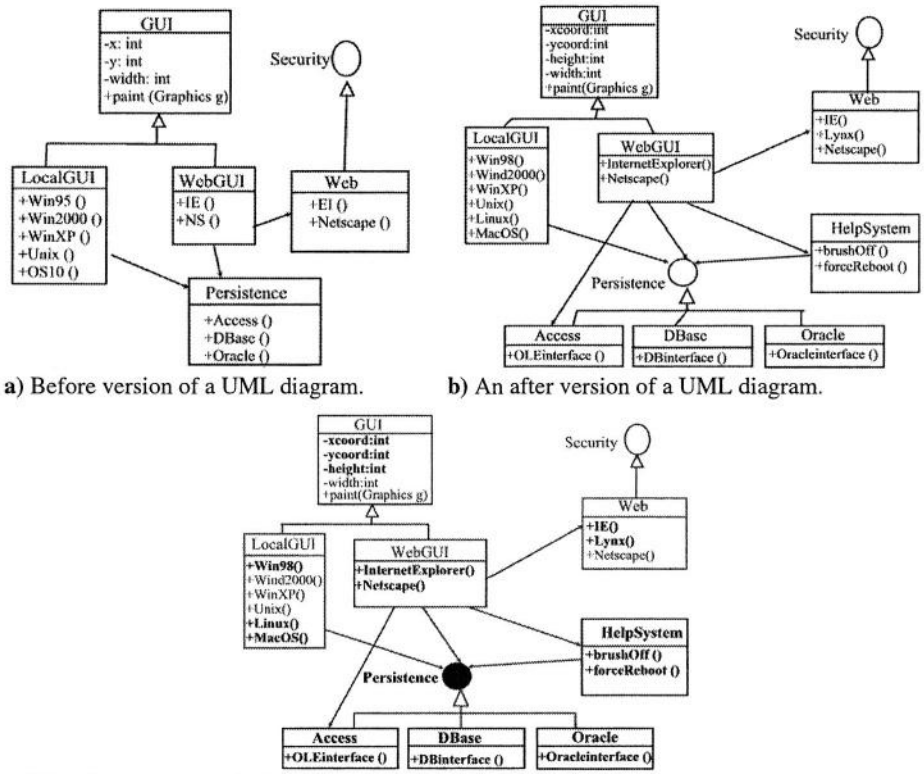

Fig. 1. A UML diagram: a) original, b) changed version, and c) visually enhanced changes

These three examples deal with scenarios where people are already interested in collaborating within a shared workspace or document. [13] also describe the benefits of asynchronous awareness for potential future collaborations.

Developers are not blind to the importance of change awareness. Many commercial applications concerning sequential documents (e.g., collections of reports, papers, and source code) can detect and show change information between new and old versions. A sampling of standard techniques is listed below.

The first class of change awareness techniques contains several ways that differences between two versions of a document are visually displayed on the screen.

- *Sequential deltas,* as exemplified by the UNIX *Diff* system [7] inserts instructions after lines that are deemed different. Instructions describe how the line(s) in the previous document can be transformed into the new document, i.e., instructions to add or delete new lines, or instructions that transform an old line into a new form.
- *Annotations and markups* display differences as explanatory notes that points to the parts of the document that has changed. Microsoft Word's *track changes* capabilities, for example, optionally displays changes as margin notes [11]. Flexible Diff [14] accompanies the source text with multiple columns of annotations. The first and second columns contain the original and modified text. The third column

shows only the differences (where thresholds can be specified to mute small differences), while the fourth shows explanatory annotations added by the author. Another example is how Rational Rose shows changes in object-oriented source code [8]. In one of its views, it displays objects within a hierarchical class browser, and marks items that have changed by special symbols positioned next to the item.

- *Highlighting* displays the differences within the document itself by directly marking up the appearance of the document contents. One example is the UML diagram in Figure 1c. Another is Microsoft Word's ability to show inserted text in a contrasting color, and deleted text as crossed out (changed text is treated as a combination of deleted and new text) [11].
- *Overviews* communicate changes made to an entire document at a glance. They usually provide a graphical miniature of the document, and mark regions that have changed through highlighting. Attribute-mapped scroll bars [6] provide indications of edit-wear (i.e., the areas where people have repeatedly edited the document) as marks within a scroll bar. Seesoft provides a zoomed out view of source code scattered over several files [3]. Each column of the overview represents a file; each character is compressed into a pixel. Line coloring indicates change to the code, such as its age and the developer responsible for adding it. State TreeMaps uses a Treemap overview to emphasize changes made within multiple documents [12].
- *Graphical playback* replays changes over time by showing all fine-grained editing actions, or by storyboarding major changes into easily comparable scenes [9,17].

The second class of change awareness techniques describes how document versions are maintained or specified, and how changes within them are detected or tracked over time. Each technique may use one or more of the change display mechanisms described above to reveal its information to the end user.

- *File differencing* occurs when a user manually specifies two files and asks the system to compare them for differences [7].
- *Real time differencing* lets the author turn on a facility to continually track changes. The document itself stores these changes internally. Microsoft Word serves as a good example, where text is marked up as the author types [11]. All change information is kept until an author decides to accept or reject them.
- *Version control systems* automate and enhance the process of manually tracking progressive versions of documents and their changes [18,1,15,10]. The first version is typically created by freezing the current document. The frozen version can no longer be changed, and so editing it implicitly creates a new revision [18].
- *History systems* track all incremental changes made to a document, typically so they can be played back at a later time or so actions can be undone [9,17].

Most of the above are *ad hoc* solutions for sequential documents, and even then there is much they leave out. They typically neglect change awareness in two-dimensional graphical documents [17]: figures, photos, blueprints, concept maps, graphs, UML diagrams, and collaborative workspaces containing spatially scattered artifacts. While 2D documents are widespread, techniques for displaying change awareness within them are undeveloped and are likely non-trivial [17]; we do not even know what change information they should show to the end user. Our goal is to fill this void.

Table 1. Elements of WA relating to the present, from Gutwin [5]

Category	Element	Specific questions
Who	• Presence • Identity • Authorship	• Is anyone in the workspace? • Who is participating? • Who is that? • Who is doing that?
What	• Action • Intention • Artifact	• What are they doing? • What goal is that action part of? • What object are they working on?
Where	• Location • Gaze • View • Reach	• Where are they working? • Where are they looking? • Where can they see? • Where can they reach?

Table 2. Elements of WA relating to the past, from Gutwin [5]

Category	Element	Specific questions
How	• Action history • Artifact history	• How did that operation happen? • How did this artifact come to be in this state?
When	• Event history	• When did that event happen?
Who (past)	• Presence history	• Who was here, and when?
Where (past)	• Location history	• Where has a person been?
What (past)	• Action history	• What has a person been doing?

3 Theoretical Foundations

The prerequisite to understanding change awareness is to determine what information is necessary if people are to comprehend change in the collaborative workspace. These informational elements of knowledge verbalize, categorize and explain what information should be tracked and captured by an application as a change occurs and how this information could be useful to an end user. Once the informational elements have been specified, we can then design an interface that captures and display this information in a meaningful and useful fashion. The details of the first step - the information elements – are the focus of the remainder of this paper. Our theoretical foundations of these information elements have their roots in Gutwin's framework for workspace awareness [5], summarized in this section and in Tables 2 and 3.

3.1 Workspace Awareness for Real Time Interactions

Gutwin focused on *workspace awareness* within real time groupware environments. He was concerned with people's continuous maintenance of awareness of others in a visual workspace and how others were interacting with the artifacts held by that space. He articulated a broad set of awareness elements, where each element consists

of the information that people need to track events in the workspace as they occur. Table 1 shows Gutwin's elements of knowledge contained within a "who, what and where" category of questions asked about workspace events in the present.

For each of these categories of questions, there is a unique set of *informational elements* that provides answers to those questions. These informational elements are the specific pieces of information that a person would require in order to keep up with events as they occur in a collaborative real time workspace.

For example, knowledge of the 'who' category simply means that you know the number of people who are present in the workspace (if any), their identity, as well as being able to attribute a specific person to each action that is taking place (Table 1, first row). In the 'what' category (second row), awareness of action and intention means that you know what a person is doing both at a rudimentary level (e.g., typing some text) and at a more abstract level (e.g., creating a title). Awareness of artifact is knowing what object another person is currently working on. Location, gaze, view and reach are all inter-related in the 'where' category (third row): location refers to the part of the workspace where a person is currently working; gaze is the part of the workspace where a person is currently looking at; view is where they can potentially be looking (i.e., their field of vision), and reach includes the parts of the workspace that this person can potentially change [5].

3.2 Workspace Awareness for Past Interactions

Gutwin does mention elements for maintaining awareness of asynchronous changes in his framework, required for people to catch up with events that have already taken place in the workspace [5]. Table 2 lists this second collection as elements of knowledge contained within the "who, what, where, when and how" categories of questions that may be asked of workspace events in the past.

Determining 'how' the workspace has changed involves two elements: *action history* and *artifact history* (first row, Table 2). Action history describes the unfolding of events that changed the workspace. Artifact history includes details about the process of how an object was changed over time. Information about 'when' something has occurred (second row) is described by the *event history* of the workspace. This element indicates the time at which things occurred in the workspace. 'Who' provides a *presence history* of people in workspace, that is, of knowing who has visited a particular location and when this visit occurred (third row).

Although there are potentially many aspects to the 'where' category of questions e.g., where did events take place, where have artifacts been etc., Gutwin mentions only the *location history* of other people, which indicates where a person has been in the workspace. Finally the 'what' category of questions lists the *action history* of a person, which describes what actions have another person engaged in (last row).

Gutwin's framework is a good beginning. However, because he was mostly concerned with elements relating to the present, he did not elaborate on past elements beyond this initial list. The remainder of this paper tries to continue where he left off in developing a framework of change awareness.

4 Information Elements for Change Awareness

In this section, we extend and elaborate the elements Gutwin identified for workspace awareness to create a more comprehensive change awareness framework for different-time asynchronous work. In this situation, a person has been away from the workspace for a period of time (hours, days, weeks) and must be brought up-to-date on what has changed in the interim.

When trying to catch up with changes the first piece of information that a person needs to know is "Is anything different since I last looked at the work?" Obviously a change awareness system must provide the answer to this question in a very lightweight fashion. Afterwards, the person can then probe for further details by trying to find out the specifics of a change. The specific information that a person may require in order to track changes will vary from situation to situation. It will depend upon the task that is being performed, the person who is carrying out the task, as well as the surrounding environment.

In a manner similar to how Gutwin constructed his framework for workspace awareness, we can describe at a high level the questions that may be asked. This set of questions includes:

1. Where have changes been made?
2. Who has made the changes?
3. What changes were made?
4. How were things changed?
5. When did the changes take place?
6. Why were the changes made?

However, this does not suffice by itself. A change awareness framework must account for the fact that people may need to view aspects of the workspace in different ways at different times, i.e., from different perspectives. In particular, a person may query the workspace for changes in terms of:

- the artifacts that exist within it *(artifact-based view),*
- the people who work within it *(person-based view),* or
- the workspace may be viewed as one locale or as a collection of related locales (*workspace-based view*) where a person is interested in the changes and events that have taken place in one or more locales. This relates to [l]'s artifact metaphor.

The specific perspective that a person has of the workspace will have an impact on the information that he or she is interested in and the way that the information is requested and represented. In terms of the artifact-based view, the person will be interested in changes made as they relate to particular workspace artifacts, and will make various queries about those changes. Examples include: how has this item changed, and what has been done to this item? From the person-based view, an individual wishes to know about the changes that were made by another collaborator. Queries about changes will therefore be focused on this person e.g., what did he do while I was away? On the other hand, someone with a workspace-based view would be interested in and inquiring about the events that have taken place in a specific location e.g., what changes and events took place in this space? This location can

consist either of the workspace as a whole, or portions of the space that are somehow logically related e.g. a specific room in a room-based groupware system or a particular spatial region.

Of course there is a strong relation between the three workspace perspectives and the six categories of awareness questions. An individual that holds a person-based perspective will focus heavily on the 'who' category of questions. Someone that holds an artifact-based view of the workspace may focus instead on the 'what' category of questions and try to determine what changes were made to specific objects. Yet another person that holds a workspace-based view of the workspace may focus on the 'where' category of questions. Alternatively the person with a workspace-based view may focus on 'what' was done in the part of project that he or she holds an interest in. The main point is that the person's particular view of the workspace will influence the value that he or she attaches to each category of question. As will be seen, however, the specific example questions that are unique to each of the six high level categories awareness questions can be asked from any of the three workspace perspectives.

The following subsections will describe in detail the informational elements associated with each category of question as well providing some specific example questions that a person might ask about changes.

4.1 Where?

Location in a 2D graphical workspace could be a simple Cartesian spatial region, or a direct digital analogue of physical demarcations, e.g. the rooms in a room-based system such as TeamRooms [4] or locations may be more abstract in relating workspace entities to each other, e.g., the different logical or conceptual parts of a collaborative project. In all cases, the location of a change provides valuable clues regarding its context, which in turn guides people towards further exploration. For example, a person may ask if a given change was part of the project component that is undergoing extensive rework, that is, if the change occurred is in the same place where many other changes are also occurring.

Table 3 shows the specific questions that may be asked to learn 'where' changes have occurred with respect to each of the three workspace perspectives. As already mentioned, the difference is how queries about changes made to the workspace are formulated within each perspective. With the artifact-based view, the questions could be asked in terms of a specific object in the workspace. Where is it now? Where was it before? Where has it been since I have been away? From the person-based view, the example questions may be asked about a specific collaborator. Where has this person visited or looked in the workspace? Where did this person change things? From the workspace-based view, the questions asked would inquire about the different events that have taken place in the space since a person has been away. Where in the workspace have people visited? Where were the artifacts moved?

The informational elements that will answer the 'where' category of questions include Gutwin's location history (described in Section 3), and the new categories of gaze history and edit history. Location history refers to the parts of the workspace

that have been visited by a person. Gaze history includes the parts of the workspace that a person has looked at. The difference between location and gaze history is that while a person may have been present in a general location, he or she may not have actively attended to everything that went on there. Although location and gaze history do not directly provide information about changes that have been made, they do indicate the parts of the workspace that have been visited or viewed and the frequency of these visits [6]. This provides strong clues as to where one should look for changes. Edit history, on the other hand, explicitly deals with the changes that were made. Awareness of 'where' edits occurred is vital to routine project management as it provides strong clues as to the progress that has made towards satisfying project-level goals. By this very fact, the location of changes provides strong cues to the answers to other "who, what, why, and how" category of questions.

Table 3. Information elements and workspace questions related to 'where'

Where			
Information elements	Specific questions		
	Artifact based view	Person based view	Workspace view
Location history Gaze history Edit history	Where was this artifact (when I left)? Where is the artifact now? Where has this artifact been during the time that I have been away?	Where in the workspace has a person visited? Where in the workspace has a person looked at? Where in the workspace has a person made changes?	Where have people been in the workspace? Where were artifacts in the workspace? Which parts of the workspace have people looked at? Which parts of the workspace have people made changes in?

4.2 Who?

Answers to questions concerning 'who' are important. In collaborative environments, knowing who made a change becomes an opportunity to query that person directly for further information. Also, people may attend to changes differently depending upon who made them. For example Neuwirth et. al. [13] described how collaborators could be more interested in seeing changes made by co-authors that he or she has known for a long time and less interested in seeing changes made by less trustworthy co-authors.

In Table 4, we provide a more detailed breakdown of the 'who' questions from the different workspace views. To Gutwin's concept of *presence history,* we add *identity, readership history* and *authorship history.* While presence history tracks if anyone was present, identity indicates the specific individual associated with a change or event. Establishing identity is needed to provide the context for answering the person-based view of workspace changes. For example, a team member may be responsible

for auditing and upgrading different parts of the project. If his or her identity is associated with a particular change, it may better help other team members (who may prefer their original version) accept that change. Readership history carries identity even further by listing all the people who have viewed a particular artifact, or conversely, listing all the items that a particular person has seen. This is important as knowing that someone has viewed the changes without raising any objections or making any further changes suggests an implicit acceptance of the current work. By a similar vein, authorship history can list the people who have made changes to an artifact, or list all the artifacts that a particular person has changed. Tracking readership and authorship history can, for example, be used to gauge progress of the project through a process-oriented lifecycle. In such a case knowing who has seen an object and 'signed off' on it is an important part of workflow management and document routing.

Table 4. Information elements and specific workspace questions related to 'who'

Who			
Information Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
Presence history Identity Readership history Authorship history	Who has looked at this artifact? Who has changed this artifact?	Who has this person interacted with? Who made changes with this person?	Who has been in the workspace? Who has looked at the workspace? Who has made changes to the workspace?

4.3 What?

The 'what' category of questions leads to answers that produce a picture of the *action history* of the workspace (Table 5). Gutwin described two ways that action history can answer these questions [5]. First, it can be used to track all low level actions that a person has done, e.g., creating, labeling and positioning a circle in a diagram. Knowledge about actions that people have engaged in while one was away is important. When people are asked to describe what is new in the workplace it is frequently in terms of the actions and events that have taken place. Sometimes there is, of course, a need to put all of these lower level actions in the context of the higher goals. So Gutwin also described action history from a higher-level perspective of the low level changes in a way that considers the goals that motivated these actions, e.g., the labeled circle was created in order to represent a new person in an organizational chart.

Yet the low-level questions and answers presented in Table 5 are often the only information that developers can hope to capture when they add change awareness support to an application. The problem is that it is difficult to ascertain and store the motives behind a series of low level changes. One could use spatial proximity (i.e.,

changes located near to each other) or temporal proximity (i.e. changes that occur at the same time) as predictors of inter-relatedness, but often these methods will fail because the sub-steps for achieving several different high level goals may often be interleaved. Thus, we will postpone discussing the higher-order view of changes until Section 4.6, and instead focus here on only the significance of low-level changes. It is important to point out, though, that people are able to relate and combine several low-level actions to derive a higher-level action if they are given enough contextual information to understand the relationships between these lower-lever actions.

The specific questions associated with the 'what' category varies depending upon the perspective that a person has of the workspace. These questions are shown in table 5. For the artifact-based view, inquires are made about the changes that have been made to a particular artifact. From the person-based view, the questions ask about what actions has a person undertaken. With the workspace-based view, the questions ask about the actions that were undertaken within the workspace or actions that were carried out on the artifacts in the workspace.

Table 5. Informational elements and specific workspace questions related to the 'what'

What			
Information Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
Action history	What changes have been made to the artifact?	What artifacts has a person looked at? What artifacts has a person changed? What activities has a person engaged in?	What changes have occurred in the work-space? What artifacts were viewed? What artifacts were changed?

4.4 How?

The 'how' category asks how the current workspace differs from the way that it was before (Table 6). The answers to these questions can be integrated to derive one of two historical views of changes. The first is in terms of the *process history* of the workspace, which indicates incrementally how the workspace evolved from its previous state (i.e., the state that it was in when one last looked at it) to its present state. This is useful when a person is interested in the mechanical means (the intermediary steps) that produced a change or group of changes as well as the end result. Thus process history is tightly coupled with action history. The difference is that the action history consists of all the actions that have occurred while one was away, while process history relates and abstracts a subset of all actions into a series of steps in a process. The process view is important for, as Gutwin described, people may have trouble interpreting instantaneous changes [5]. Thus describing all the sub-steps involved in a change may help to clarify what happened. Also, the process-oriented view describes

the *evolutionary context* of changes (i.e., the specific details of the circumstances faced by the person who made the change at the time that the change occurred), and thus can provide valuable insight on 'how' and 'why' things came to be in their present state. Of course, a person may only be interested in the final result. This is the second historical view of 'how' a workspace has changed, i.e., the *outcome history*. The outcome history presents only a 'bottom line' understanding of a change where it highlights only those things that differ between the initial and the final state.

The choice of process *vs*. outcome history will depend largely upon the task at hand. For example, a graphic artist may be interested in the technique used to produce some visual effect. In this case, this person will want to see (and thus learn) the process history of the workspace. On the other hand, a newspaper editor reviewing an article submitted by a reporter is far too busy to be concerned with the rough drafts produced by this person, and would thus be interested only in the outcome history of the article. Consequently, it is important that software support for change awareness provide the ability to discover both the process and outcome history of a workspace.

Table 6. Informational elements and specific workspace questions related to 'how'

How			
Information Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
Process history Outcome history	How has this artifact changed?	How has a person changed things?	How has this workspace changed?

Table 7. Informational elements and specific workspace questions related to 'when'

When			
Information elements	Specific questions		
	Artifact based view	Person based view	Workspace view
Event history	When was this artifact changed?	When did a person make changes?	When were changes made to the workspace?
	When was a particular change to this artifact made?	When did a person make a particular change?	When did a particular change in the workspace occur?
	In what order were changes made to this artifact?	In what order did this person make changes?	In what order did changes to the workspace occur?

4.5 When?

The timing and ordinality (sequential order) of changes is revealed by the answers to the questions of 'when' changes occurred as listed in Table 7. The time when a change occurred, particularly if it overrides an earlier change by another person, is often of great significance and affects the perceived importance of a change. For

example, a person may only be interested in recent workspace events, or a person may only be interested in changes that occurred within a specific period of time.

The timing and ordinality of changes constitute the *event history* of the workspace, and it provides the *chronological context* for understanding and interpreting changes giving clues to the 'where', 'who', 'what', 'how' and 'why' categories of questions.

4.6 Why?

Knowing the thought and motives behind a change can be important for accepting the changes that others have made. The questions that a person will ask to discover 'why' changes were made are summarized in Table 8. A historical view of 'why' changes were made includes both the *cognitive history* and the *motivational history.* Cognitive history describes the logic or reasoning that may be behind a change, which is a rational reconstruction of the person's goals and plans. Motivational history deals more with the impulses or desires that are the impetus for making a change, which is the actual reason why a person did something at a moment in time. The reason that they are separate elements is because a change may be based upon a well thought out and carefully conceived plan or it may be more of a spur of the moment thing as one reacts to the current situation. Also, some changes are completely unintended accidents.

Although it is not always needed, knowing 'why' a change was made is obviously an important step for coming to understand and accept it. For lower-level changes that are the parts of a grander higher-level change, the motivating factors may be painfully obvious. In this way providing the motivational history for simple changes may be too effortful (and distracting from the main task) to explain. Also, describing all the motives behind a change and detailing all the reasoning behind each event is extremely difficult for computers to do automatically. This is because understanding the 'why' often draws upon a person's accumulated technical expertise and implicit or 'hidden' cultural information relating to group priorities, work practices, and short and long term goals. Today's computing systems lack the ability to sense these technical and cultural factors that motivated a change. They also lack the intelligence needed to produce a truly comprehensive picture of the cognitive history of the workspace. Consequently, most 'why' information will likely be generated explicitly by authors, e.g., as annotations added to changes or as design rationales.

Table 8. Informational elements and specific workspace questions related to 'why'

Why			
Information Elements	Specific questions		
	Artifact based view	Person based view	Workspace view
Cognitive history Motivational history	Why was this artifact changed?	Why did a person make that change?	Why was that change made in the workspace?

5 Discussion

We have described in detail the information that can be used by a person to track changes made by others over time in a collaborative project. The informational elements were classified according to several categories of change awareness questions. These categories are inter-related and inter-dependent. When a person is tracking changes he or she may start with the highest-level question, 'Has anything changed?' From that point the person will make inquiries about changes from one or more of particular perspectives - artifact, person, or workspace-based - that make the most sense to him or her and the work context. The inquiries can be directed towards a specific collaborator, 'What has this person done?' Alternatively, the process of inquiry can take the form of an examination of a particular artifact, 'How does this object differ from before?' or it can take the form of an inspection of a select portion of the workspace, e.g., 'What has happened in this corner of the project?'

Within the bounds of the chosen perspective, a person can ask specific questions from these categories to probe for further details of changes. The specific category that a person begins with (where, who, what, how, when or why) is not fixed. As mentioned previously, it will be influenced by the workspace perspective that is held e.g., if a person is making inquiries from a person-based view then the 'who' category may be of the most pressing urgency. Also, as we have shown in the previous sections, queries made in one category of question are not isolated from the other categories. The process of inquiry can occur in parallel as someone delves for answers in more than one category at once. For example, take the case of a project manager who is reluctant to have certain parts of the project undergo anymore changes, or who has severe misgivings about the work of specific team members. When the manager returns to the project after a period of absence and discovers that many changes have been made, he may immediately try to determine exactly where changes were made and specifically who made those changes.

The answers to the questions from one category may also inspire additional inquiries in another category. For example, a person who is tracking the historical context of the changes to a software system may start by asking about 'when' most of the changes occurred. Upon discovering this information she notes that the code was most volatile during a port between operating systems. Since she knows that there is a methodological way to do this, her queries then focus on the process history of the software as she tries to determine exactly what the programmers did during the port.

Furthermore, the answers to the questions that a person asks about one category of question may directly provide him with further information. For example when a person knows exactly where changes occurred, she then knows who made those changes (because she knows who is responsible for which portions of the project). Or the person may be able to make predictions about the answers to the other categories of questions based upon the information that she gets from one category. When a person learns that a specific team member made a change, she can guess as to how the change was made. These guesses are based upon her personal knowledge of the person who made the changes and the techniques that he has employed in the past.

Although a single answer may provide information about multiple categories of questions, the main point of the framework is to ensure that designers of change awareness systems actually consider what change information should be captured if the system is to provide its end-users who are tracking changes with the information they need to answer these questions. At the very least, the designer can use this framework to priorize what change information is needed, and to focus on those with the highest priority. The framework ensures that the designer will not neglect certain categories out of ignorance.

6 Conclusions

In this paper we have introduced a theoretical framework that can be used in several ways. First, designers can use it as a high-level guide for determining what change information should be tracked and displayed to participants, and what perspectives of viewing this information are relevant to the end user. Of course, we don't claim that this determination is always easy, for the needs of the collaborators can be quite situation specific. Second, evaluators can use the framework to critique existing systems: even using it as a simple 'checklist' will quickly reveal what change information is captured and displayed by the system, and what is left out. We have already done this critique to reflect on a change awareness system we had created before forming the framework, and it revealed many oversights and deficiencies [17].

The next challenge is to use the framework to build an exemplar of how a two-dimensional graphical groupware application can support asynchronous change awareness. While the framework states what is needed, creating good interaction and display techniques for change awareness remains a significant challenge.

References

1. Berlage, T., and Sohlenkamp, M.: Visualizing Common Artefacts to Support Awareness in Computer Mediated Cooperation, CSCW Vol. 8 No. 3 (1999)
2. Brown, H.B.: *The Clear/Caster System.* Proc NATO Conf. Software Engineering (1970)
3. Eick, S., Steffen, J. and Sumner, E.: Seesoft—A Tool for Visualizing Line Oriented Software Statistics. In Card, S., MacKinlay, J. and Shneiderman, B., Ed., Readings in Information Visualization, Morgan Kaufmann Publishers Inc. (1992) 419–430
4. Greenberg S. and Roseman, M.: Using a Room Metaphor to Ease Transitions in Groupware. In M. Ackerman, V. Pipek, V. Wulf (Eds) Sharing Expertise: Beyond Knowledge Management, 203-256, January, Cambridge, MA, MIT Press (2003)
5. Gutwin, C.: Workspace Awareness in Real-Time Groupware Environments. Ph.D. thesis, Department of Computer Science, University of Calgary, (Calgary Canada), (1997)
6. Hill, W.C. and Hollan, J.D.: *Edit Wear and Read Wear.* Proc ACM CHI'92, (1992) 3–9
7. Hunt, J., W. and McIlroy, M.D.: An Algorithm for Differential File Comparison. Computing Science Technical Report No. 41, Bell Laboratories (1975)
8. IBM Corporation: Rational Rose. //www-306.ibm.com/software/rational/
9. Kurlander, D.: Graphical Editing by Example. Proc ACM CHI'93, (1993)

10. Magnusson, B. and Asklund, U.: Fine Grained Version Control of Configurations in COOP / Orm. Proc Symposium on Configuration Management, SCM6 (Berlin, Germany) (1996)
11. Microsoft Inc.: Microsoft Word, as included in Microsoft Office Professional 2003
12. Molli, P., Skaf-Molli, H. and Bouthier, C: State Treemap: an Awareness Widget for Multi-Synchronous Groupware. 7th Intl Workshop on Groupware - CRIWG'2001 (2001)
13. Morán, Favela, Martínez-Enríquez, and Decouchant: (2002), Before Getting There: Potential and Actual Collaboration Proc Springer-Verlag CRIWG (2002)
14. Neuwirth, C.M., Chandhok, R., Kaufer, D.S., Erion, P., Morris, J. and Miller, D.: Flexible Diff-ing in a Collaborative Writing System. Proc ACM CSCW '92, (1992) 147–154
15. Rochkind, M.J.: The source code control system. IEEE Trans Software Engineering Vol. 1 No. 4, (1975) 364-370
16. Steves, M.P., Morse, E., Gutwin, C. and Greenberg, S.: A Comparison of Usage Evaluation and Inspection Methods for Assessing Groupware Usability. Proc ACM Group '01 (2001)
17. Tam, J.: Supporting Change Awareness in Visual Workspaces. Unpublished M.Sc. thesis, Department of Computer Science, University of Calgary, Alberta, February (2002)
18. Tichy, W. F.: RCS – A System for Version Control. Software – Practice and Experience, Vol. 15 No. 7, (1991) 637–654

Increasing Awareness in Distributed Software Development Workspaces

Marco A.S. Mangan[1,2], Marcos R.S. Borges[3], and Claudia M.L. Werner[1]

[1] Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ, Brazil
{mangan,werner}@cos.ufrj.br
http://www.cos.ufrj.br/~odyssey
[2] Faculdade de Informática/PUCRS, Brazil
mangan@inf.pucrs.br
[3] Núcleo de Computação Eletrônica and Instituto de Matemática/UFRJ, Brazil
mborges@nce.ufrj.br

Abstract. This work presents a middleware for collaborative applications that increase product and workspace awareness information available to users of computer-aided software engineering tools. This middleware-based approach helps application developers to construct enhanced tools, adapted to specific needs, reusing software components and existing applications. These enhanced tools must be designed to overcome some of the technical difficulties of collaboration in distributed software development scenarios, like the need of monitoring changes in remote workspaces. This paper describes the middleware architecture and intended usage, presents examples of enhanced tools, and proposes future case studies.

1 Introduction

In the last decades, many organizations have adopted remotely located facilities and outsourcing in software production. Global Software Development (GSD) must deal with the strategic, technical and cultural issues of participants and teams dispersed over time and physically distant [5]. Distributed Software Development Environments (DSDEs) try to provide software developers with facilities that help to overcome some of the difficulties imposed by the separation over time and distance. For instance, communication and awareness breakdown inside virtual development teams are difficulties that are dealt with such environments. Collaboration facilities range from shared repositories, on-line and off-line communication channels to coordination and awareness mechanisms.

This work deals with the difficulties of constructing such environments, and, in particular, on obtaining and managing awareness information. The most challenging problems are that the enhanced environment should provide (a) adequate support for software development activities and, conversely, (b) useful awareness information for software developers. We propose a middleware-based approach that enables the creation of shared workspaces on top of pre-existent software

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 84–91, 2004.

tools and groupware. Team participants must assess tools and the correspondent application events they want to share before the collaboration activities take place.

Once configured, the middleware is able to provide services that are similar to those found in groupware applications. Depending on the degree of modification of the tool or of its execution environment, and the participant's objectives, it is possible to provide a set of collaboration services. These range from simple presence information to tracking of development history, product and workspace awareness information, and even collaborative editing.

Our assumption is that collaboration can be enhanced with small modifications in real working environments. Information gathered from the environment can be used to identify opportunities for collaboration that participants may not be aware of.

The remainder of this article presents an overview of the class of collaborative environments (Sec. 2) we plan to develop by using the proposed middleware architecture (Sec. 3). We summarize roles and a process to develop awareness enhancements in Computer-Aided Software Engineering (CASE) tools (Sec. 4) and also present some examples of enhancements (Sec. 5). Finally, we present the conclusion of this article and the next steps in this ongoing work (Sec. 6).

2 Distributed Software Development Environments

Distributed Software Development requires the coordination among teams and individuals dispersed over time or physical distance. Herbsleb et al. [5] believe that we have to understand distribution as a more ample term. Even colleagues that work a few rooms down the corridor or in different floors of the same building may suffer from some sort of breakdown in communication, coordination, and awareness that are characteristic of global and distributed software development scenarios.

Recently, some software development environments were proposed to support GSD [2–4,6,7,1]. These environments aim to combine software development task support with some sort of teamwork support. Altmann and Pomberger's [2] ample survey of cooperative software development practices lead to a proposal of an environment aimed at large software projects organizational support. Distributed process coordination is present at Serendipity [4] component-based environment. Palantir [1] and Gossip [3] propose environment enhancements based on improved product awareness information over artifact changes. Real-time, collaborative editing support is applied on distributed extreme programming in both MILOS [6] and Tukan [7] environments.

This diversity of approaches in CSCW support and software development scenarios is an indicator that GSD support is a complex problem. Software development task support must be fitted to meet particularities in software development process, technology, and methodology. It is not economically viable, if not virtually impossible, to provide a different environment for every particular combination of these factors. In practice, these collaborative environments offer

software development tools that are not as effective as their single-user counterparts. As a result, these environments neither evolve to follow new trends in software development nor are commercially supported. Clearly, a successful DSDE must support high levels of customization and be built on top of existent, evolving, supported tools and applications.

DSDEs are usually developed using one of two approaches. In the first one, the starting point is a working software development environment on top of which some enhancements are produced to create a more collaborative environment. Typically a collaborative library or toolkit is adopted, e.g., Tukan is developed over a Smalltalk environment and COAST [7]. In the second approach, the starting point is a distributed programming platform (e.g message passing, distributed objects, or shared memory) or a collaborative library. The software engineering support is completely programmed from the scratch, e.g., MILOS. Both approaches require a large programming effort. We propose a third approach, based on the existence of a collaborative middleware.

3 Middleware Architecture

The proposed middleware architecture (Fig. 1) is composed of three types of elements connected by an event notification service. The first element is the CASE tool, or application, that supports a specific set of operations and deals with end-user input and output. The CASE tool provides a rich context for software engineering activities that is explored as a source of awareness information.

The second element, the collaborative extension, also provides end-users with input and output. Mainly, the extension provides some visualization about events in remote workspaces. We have successfully implemented collaborative extensions that provide diverse awareness information: telepointers, radar-views, group and task awareness widgets.

The third element, the collector, monitors operational events in the CASE tool and its run-time events. The collector is activated by some application operation, captures data from the application current state, and places new objects in the event notification system. The collector should be a simple program, so that it should be easy to maintain and develop. We have successfully implemented collectors using different approaches: monitoring the application windowing system information, input devices, and persistence mechanisms, and also using tool extension mechanisms and aspect-oriented techniques [8].

The event notification system offers storage, distribution, and query facilities. The current implementation is based on tuple spaces, a distributed programming paradigm [10]. A high-performance tuple space server [14] can hold a large number of independent objects (called tuples) and offers three basic primitives: write, read, and take. The write primitive places a new tuple in the tuple space. Read and take are query primitives that receive a query template as a parameter and return a tuple from the space that satisfies the template. Take is a destructive query, which means that the tuple is removed from the space.

We summarize the flow of information in the architecture through a description of four supported activities: capture, secure, analyze, and distribute

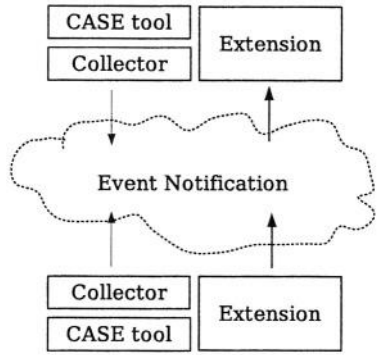

Fig. 1. Middleware architecture.

awareness information events. Since the CASE tools chosen by the team participants already define the work environment, the main problem is how to capture the necessary application events. Current DSDEs are programmed specifically to provide points of capture for the necessary events. In order to provide loose coupling of applications, the architecture use collectors, software programs that are programmed to collect significant application and environment events. Collectors are activated by subscription of one or more collaborative extensions. The selection of collaborative extensions requires the existence of appropriated collectors. On the other hand, the availability of collectors constrains the applicability of extensions to a given CASE tool.

An event is defined by the awareness information need of some extension element. Events can range from simple input and output activity in the end-user workstation to more elaborate information about operations over the objects present in the workspace. For instance, a real-time shared workspace extension may provide a telepointer implementation that relies on mouse move events collected from the CASE tool run-time environment.

The main concern in this architecture proposal is how to program collectors without interfering with the application perceived performance. A sensor may not be able to capture accurately every relevant event and its programming can be a challenging task, even if the application source code is available. On the other hand, we assume that risk because programming sensors is less complex than programming a new DSDE.

Event occurrences are expected to be very high. A database is assumed to secure all event occurrences for posterior analysis. On a distributed setting, more than one database may be necessary for different groups and organizations to control their own data. Event analysis is to be made based on causality of actions and common operation of artifacts. Some analysis results may be feed back into the work environment (distribute). A sideshow window may be adopted to notify team participants of relevant information, thus, reducing the need for another interference in the applications. Collaborative editing is the more challenging extension because we need to feed remote events back into the application, which is not always possible.

4 A Process to Increase Awareness in DSDEs

This process aims to organize guidelines to developers producing a DSDE. The process definition prescribes four different roles. Each role requires a set of skills and attributes some responsibilities. Process enactment requires the identification of which roles each participant will have. We expect these roles to be performed by different persons. This allows work specialization and reduces the complexity of a DSDE.

We summarize the role descriptions as follows. **CASE tool developer** is responsible for the development of a particular CASE tool development. This role may be responsible for providing generic tool extension mechanisms. **Enhancement developer** is responsible for a particular collaborative extension. **Integration developer** is responsible for develop adapters between a particular CASE tool and a particular extension implementation. This role requires a highly skilled programmer. In some cases, this role may need the assistance of the CASE tool developer. **Environment composer** is responsible for the selection and composition of a CASE tool and collaborative extensions in order to produce an enhanced tool, that is adequate for a particular scenario.

We suggest the following steps to enact the proposed approach: **People Assessment**, the environment composer role lists involved people, roles, and individual goals; **Tool Assessment,** for each person, obtain a list of the tools she uses; **Context Assessment,** for each tool, obtain a list of the actions types and object types each person wants to share and be aware of; **Extension Selection,** to select extensions that present context information for the tool set. Extension developers may be activated in this step to produce new extensions. **Collector Selection,** to select collectors that collect context information for the tool set. CASE tool developers and integration developers roles may be activated on this last step to produce new collectors. Currently, we are working on the production and adaption of documents and tools to support the proposed process.

5 Enhancement Examples

In order to evaluate the proposed approach, we are developing some prototypes that try to reproduce the collaboration support of DSDEs on top of pre-existing CASE tools. We have selected two CASE tools implementations: the Odyssey SDE's [9] Unified Modeling Language (UML) diagram editor (Fig. 2(a)) and the Eclipse IDE's text editor (Fig. 2(b)). All tools involved are Java applications.

For the first tool, we initially proposed the following scenario: two developers need to peer-review a software model. They agree to work together at the same time, one developer exposes the design rationale and the other places questions and tries to point out problems while both review the system requirements. One designer is a senior designer that will take notes during the session. They are not in the same place, but both have the same CASE tool and access to the same shared repository. The session will last for about one hour, at an agreed upon date and time, and communication will be held over a voice chat.

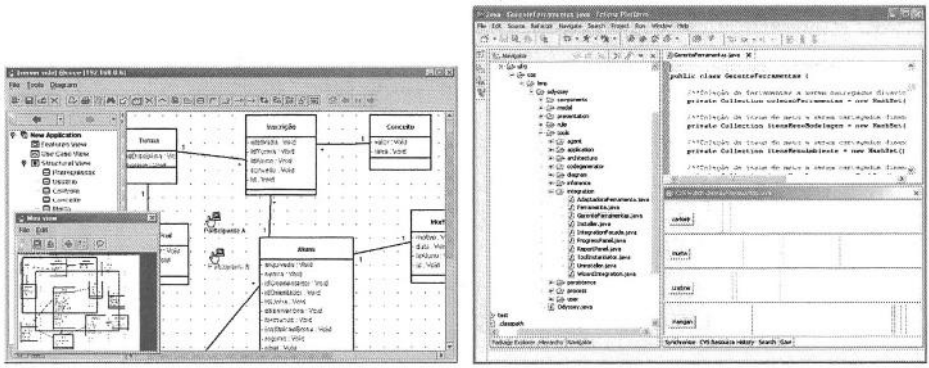

Fig. 2. (a) Radar view (small window at left hand side) and telepointers (cursor hands at center) and (b) Group awareness widget extensions.

This setup could be enacted without further support, although the participants would need a communication overhead to give each other indication of which diagram and model element they are presently reviewing. The application of strict-WYSIWIS screen sharing (as present in MILOS and commercial applications) is not adequate to this scenario because the participants need to access other applications in their workstations (e.g. a text editor to see the requirements document and to take notes) and may need to cross-check information of one diagram in another diagram. To overcome such problems, DSDEs such as Tukan and Serendipity provide support for relaxed-WYSIWIS application sharing. Unfortunately, both Tukan and Serendipity do not support UML diagrams and navigation support that are present in the Odyssey SDE.

The scenario indicates that one participant needs to be aware of the current positioning of the other participant. Relevant application events are mouse, window, and viewport resizes and moves. Two classic widgets for real-time collaborative authoring were implemented as collaborative extensions: a telepointer and a radar view software components. A collector was designed to capture the relevant events directly from the application platform's windowing system. The extension implementation is a refactoring of support available in groupware frameworks and kits [12] [11].

For the second tool, we propose the enhancement of awareness information on the following scenario: a developer is currently making changes to software code. The changes will take some days to be completed and she may need to ask some questions to other developers about the code. She is concerned with being aware of: (a) who is making changes in the same file that she is working on and (b) someone who had made contributions in the same files in the past.

The information she needs is available from the configuration management repository's log file. She would have to pull information and interpret it. Log entries are not user friendly and she would need to browse the file to organize the data. We come out with the refactoring of an awareness widget that provides

group awareness information [13]. The widget represents a timeline for each developer reported in the log file and uses colored bars to represent activity. The developer is able to quickly find out that three other developers have worked with the file, and that one of them is working in parallel. The collector monitors changes in real-time, therefore there is no need to poll the repository to see recent activity logs. The extension is integrated as a view in the Eclipse environment, promoting greater availability and organization of awareness information. The view can easily be hidden and displayed with a simple mouse click.

Available extensions can be applied in both CASE tools, i.e., the telepointers can be applied in the Eclipse tool and the group awareness can be applied to the Odyssey SDE. An appropriate collector implementation is the main requirement. In the examples, the first collector is generic to Java GUI applications, the second collector must be redefined to the Odyssey SDE, changing the monitoring object from file changes to model changes.

6 Conclusion

This article presents a middleware-based approach to the enhancement of awareness information available in current software development workspaces. The approach is a guideline to DSDE developers that want to build collaborative environments on top of existing CASE tools. This work proposes a new process to develop DSDEs because current environments are not being applied in real software development scenarios. We argue that DSDEs will not be adopted because a software team has the need for specific tools that must be enhanced and not replaced by collaborative counterparts, which are usually not adequate for software development. At some extent, the example enhancements demonstrate the feasibility of this new development approach.

There are some limitations on our approach. Example enhancements are limited to side-show windows and graphical elements superimposed to the application window. The collection of events is easier when the tool already has a native extensibility mechanism. In general, these extensibility mechanisms offer quite good flexibility because they must support demanding tools, such as, compilers, reverse compilers, profilers, and model analyzers. Besides these technical limitations, we have found that the collaborative work process in the enhanced tool is constrained by the individual work process in the original application. Therefore, the changes in the application and work processes are conservative.

Currently, we have new collaborative extensions under development that explore the structure of software engineering models and related metrics. In this way, we can provide a richer set of awareness information. We expect to provide a catalog of extensions, tools and collectors along with guidelines about the applicability of extensions.

Further work will propose two evaluations: a developer evaluation and an end-user evaluation. The first evaluation will have the process enacted by a team of developers trying to propose enhancements to a specific scenario. They will be oriented to adapt current extensions or suggest new extensions. The second evaluation will observe the reaction of developers actually using the enhancements in specific activities.

Acknowledgments

This work is partially supported by CNPq and CAPES grants.

References

1. Sarma, A., Noroozi, Z., van der Hoek, A.: Palantir: Raising Awareness among Configuration Management Workspaces. Proc. of Twenty-Fifth Int. Conf. on Software Engineering, May, Portland, Oregon (2003) 444–454
2. Altmann, J., Pomberger, G.: Cooperative Software Development: Concepts, Models, and Tools. Proc. Tech. of Object Oriented Languages and Systems, Santa Barbara, Aug. (1999) 194–277
3. Farshchian, B. A.: Integrating Geographically Distributed Development Teams Through Increased Product Awareness. Information Systems Journal, 26(3), May (2001) 123–141
4. Grundy, J.C., Hosking, J.G.: Serendipity: Integrated Environment Support for Process Modeling, Enactment and Work Coordination. Automated Soft. Eng., Jan., Kluwer Academic Publishers (1998) 27–60
5. Herbsleb, J.D., Moitra, D. (eds.): Global Software Development. IEEE Software, March/April (2001)
6. Maurer, F., Martel, S.: Process Support for Distributed Extreme Programming Teams. Proc. Int. Conf. on Soft. Eng., Int. Workshop on Global Software Development, Orlando, Florida (2002)
7. Schümmer, T., Schümmer, J.: Support for Distributed Teams in Extreme Programming. Succi G., Marchesi M.(eds.), Boston, MA: Addison Wesley (2001) 355–377
8. Kiczales, G.: Aspect-Oriented Programming. ACM Comp. Surveys (1996) 28(4es): 154
9. Werner, C.M.L. et al.: OdysseyShare: an Environment for Collaborative Component-Based Development. Proc. Information Reuse and Integration Conference, Las Vegas, Nevada, Oct. (2003)
10. Gelernter, D.: Generative Communication in Linda. ACM Trans. Program. Lang. Systems (1985) 7(1): 80–112
11. Gutwin, C., Greenberg, S.: Effects of Awareness Support on Groupware Usability. ACM Trans, on CHI (1999) 6(3): 243–281
12. Begole, J., Rosson, R., Shaffer, C.: Flexible Collaboration Transparency. ACM Trans, on CHI (1999) 6(2): 95–132
13. Kreijns, K., Kirshner, P. A.: The Social Affordances of Computer-Supported Collaborative Learning Environments. Proc. 31th ASEE/IEEE Frontiers in Education Conference, Oct., Reno (2001)
14. GigaSpaces Inc.: GigaSpaces Server. http://www.gigaspaces.com (2004)

Ariane: An Awareness Mechanism for Shared Databases*

Vaninha Vieira[1], Marco A.S. Mangan[1,2], Cláudia Werner[1], and Marta Mattoso[1]

[1] Computer Science Department, COPPE, Federal University of Rio de Janeiro – Brazil
C.P. 68511, Rio de Janeiro, RJ, Brazil – 21945-970
{vaninha,mangan,werner,marta}@cos.ufrj.br
[2] Faculdade de Informática, PUCRS - Rio Grande do Sul, RS, Brazil

Abstract. Awareness is an essential requirement in collaborative activities. This paper presents Ariane, a generic and reusable awareness infrastructure, independent of a specific application or DBMS. Ariane improves the availability of awareness information to different cooperative applications by monitoring the application persistence mechanism. A prototype of Ariane was developed using the Java Data Objects (JDO) persistence mechanism and aspect-oriented programming techniques, which were employed in order to increase the potential reusability of the solution. A preliminary evaluation of the prototype, applied in an environment for cooperative software development based on components, confirmed that no additional code is necessary to monitor JDO complaint applications. Besides, Ariane proposes a multidimensional data structure for awareness information, the awareness cube. On-line analytical processing tools can be employed to perform queries to retrieve aggregated value from small grained awareness information.

1 Introduction

Awareness in cooperative work means the knowledge and the understanding of things that happen or have happenned in the context of the group which are relevant for the accomplishment of the activities of the participants. The lack of awareness could result in several problems, such as unmotivated people, conflicts, duplicated or inconsistent work [1] [2].

Most of the awareness mechanisms proposed in the technical literature develop specific solutions for particular problem domains. These approaches are hard to generalize in different situations, compelling cooperative application designers and developers to recreate awareness solutions in every new application [3].

In cooperative applications, users commonly interact through shared artifacts manipulation [4]. In order to guarantee the artifacts sharing and durability, database systems are regularly used for persistence. The knowledge of the changes performed in the application database helps to clarify the interactions occurred in the group and further improve the group awareness.

This paper presents Ariane, an awareness mechanism based on the monitoring of application databases. The main innovation of Ariane is that it is a generic and flexible mechanism, independent of any specific application or database system. To

* This work was partially funded by CAPES and CNPq agencies.

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 92–104, 2004.

achieve flexibility and independence, Ariane was developed using Java Data Objects (JDO) [5], a standard proposal for transparent persistence in Java applications, combined with Aspect Oriented Programming (AOP) techniques [6]. These technologies enable the selective activation or deactivation of the awareness mechanism, according to the application needs. In addition, extensions to the mechanism can be developed with reduced effort. A contribution of Ariane is the proposal of an awareness cube, a multidimensional structure to store awareness information which enables OLAP and data mining tools to be used over past awareness information to analyze group interaction and to discover hidden knowledge about the work of the group.

Ariane is a component of the OdysseyShare project [7] [8]. OdysseyShare is a collaborative version of the Odyssey Software Development Environment (Odyssey SDE) [9], which supports component-based software development. OdysseyShare aims to provide a set of tools for group interaction support to be used in Odyssey SDE. The main contribution of Ariane to OdysseyShare is the monitoring of its persistence mechanism, providing collections of awareness information. These collections are explored by awareness widgets of OdysseyShare, which are responsible for data filtering and organization in order to produce appropriate information to different user categories of the OdysseyShare SDE.

This paper is organized as follows: the next section introduces the awareness problem in cooperative applications; Section 3 describes Ariane, its awareness process and architecture; Section 4 presents the Ariane prototype and its use in OdysseyShare; related work is discussed in Section 5; and, finally, Section 6 concludes this paper with some final considerations.

2 Awareness in Cooperative Applications

A cooperative application should produce awareness information about things that happen or have happened in the group context, reporting this information to group members, in order to improve the interaction between members of a group and to enable them to coordinate their own activities. The awareness information can be related to: the group composition (Who are the group participants? What are their skills? Are they available?), the group objectives (What are the activities that should be executed? How will each participant contribute?), and the group activities execution and coordination (What activities have already been finished? What needs to be done? Are the members having problems?). Awareness mechanisms are defined as techniques implemented by a system that aim to support the generation of awareness information through the monitoring of the overall group activities, and the distribution of that information to interested group members.

Through awareness information, participants can coordinate their own activities relating them to activities of other participants, discovering and solving problems such as conflicts, duplicated or inconsistent work. Besides, the group coordinator can benefit from awareness information to identify and resolve high levels of conflicts, premature decisions and lack of participation [10]. If the coordinator had mechanisms that gave him relevant information about events, he probably would find it easier to lead the group to a successful accomplishment [10].

The role a participant plays in the group is important to determine the kind of awareness information he/she is interested in. Ariane considers three main roles: operators, coordinators and analysts. *Operators* are those who execute the group activi-

ties working in a cooperative way through a shared workspace. They are interested in information about occurrences related to their own activity, as for example actions over artifacts they are or were working on. Most awareness mechanisms proposed in the technical literature are designed to help this kind of users, specially when they are working synchronously.

Coordinators are users who are responsible for activities related to make the group work well and focused on their tasks [10]. They need summarized and aggregated information about planned and executed activities to identify situations where their intervention might be necessary, such as when a user has a low level of participation, when there are high level of conflicts or low level of interaction between participants. Borges and Pino [10] have identified these activities and related awareness information to propose awareness tools to support this kind of users.

Analysts are advanced users who are responsible for analyzing the overall work of different groups in an organization in order to discover things about the group that help them making decisions or defining strategies for the group. High level analysis queries might be "Which users are the most participatives?", "Which users have abilities to play the coordinator role?", "How can I improve the productivity of my groups?", "Which users work better with whom?" and so on. To answer these kind of questions is not easy. The technical literature lacks discussions and solutions to support this kind of user in cooperative applications.

The specific objective of Ariane is to support the gathering and distribution of awareness information related to asynchronous interaction for coordinators and analysts. The next section describes the proposed mechanism.

3 The Ariane Awareness Mechanism

The Ariane awareness mechanism was designed to provide awareness information in a flexible and non-intrusive way taking advantage of the application persistence process to collect awareness information. Thus, no change is required in the application code since persistence is a functionality that is implemented in most applications.

Monitoring the application database can provide a great deal of information about the actions performed by the group members. However, some actions can not be monitored. To be monitored the action must generate communication between the application and the database. Thus, actions such as mouse moving or bar scrolling are not captured by Ariane.

Ariane is based on event notification. Events are structured messages containing information needed to promote awareness. There are four event types: **Session Event,** refers to actions of opening and closing a connection to a database, **Transaction Event,** indicates that a begin transaction, commit or rollback action occurred in the database, **Change Event,** reports create, update and delete actions, and **Query Event,** concerns retrieve actions. Change and query events can refer to either the database data or the database schema.

The event structure in Ariane follows the 5W+1H format, which indicates **who** executes **what** action, the date/time **when** the action occurred under an artifact (**where**) for what reason (**why**) and **how** it happened. The **why** and **how** questions are very difficult to answer in an automatic way, because the former implies a knowledge about what the user was thinking when he/she executed the action, and the latter implies a knowledge of the application model. These questions were considered in the

event structure of Ariane, but they still lack appropriate answers. The next subsections will describe the awareness process used by Ariane and the Ariane architecture.

3.1 Awareness Process

The Ariane awareness process (Fig. 1) consists of four phases: event production, distribution, consumption and analysis, and ten activities described in the following.

The **event production** phase takes place with the following activities: (i) the monitoring of the communication between the application and the database to capture actions performed by application users (awareness producers); (ii) the generation of event messages containing the 5W+1H information, which are gathered from the actions captured; and (iii) the storage of the events in the event repository.

In the **event distribution** phase, there is only one activity: (iv) the events are transported to awareness components previously registered as event listeners.

The **event consumption** phase has an activity where (v) visual awareness components prepare and present the awareness information in an appropriate representation to final users (awareness consumers). These awareness consumers might be playing the role of an operator or a coordinator, as defined in the previous section.

The **event analysis** phase aims to prepare the generated events for use in tools based on the On-Line Analytical Process (OLAP) model. To achieve this, the first step is to prepare the awareness information and store it in a special storage structure based on multidimensional modeling techniques called *awareness cube.* To populate the awareness cube the events must be (vi) extracted from the event database , (vii) transformed into the multidimensional format, and (viii) loaded in the awareness cube.

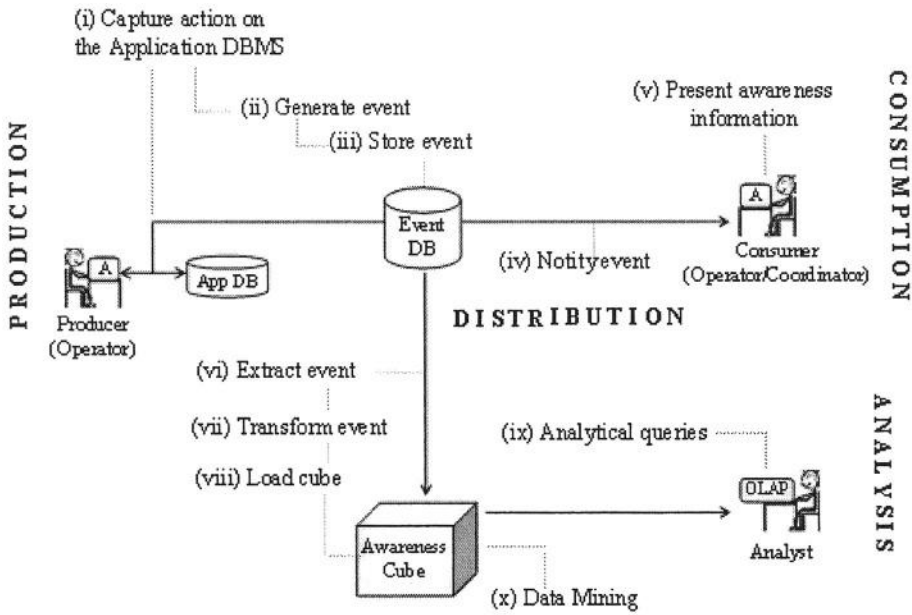

Fig. 1. The Ariane Awareness Process

Using the awareness cube an analyst can: (ix) use any OLAP tool to execute analytical queries, or (x) apply data mining techniques to discover knowledge about the group interaction.

The analysis phase in our awareness process and the proposal of a multidimensional structure is a first step to support the analyst role. As we have said in the previous section, support for analyst users is not stressed in previous awareness mechanisms. The multidimensional model enables flexible visualization of the information, since the user can freely choose and change the axis and rotations for data visualization. Users can choose the granularity they want to visualize the data and the way they want to see the information (pivot table, graphics, etc).

Awareness filters must be applied in every phase to decrease the information overload. In the event production phase, the actions collected are filtered by their type, so that only actions related to persistent artifacts are monitored. Additional filters are not considered in this phase because the focus is to monitor the overall communication between the application and the database. Therefore, the sequence of actions are kept and can be rebuilt afterwards by the same, or another, application. The distribution phase uses event types to filter events that should be delivered. Events are classified by their type and the visual awareness components must register their interest in specific event types, so that they only receive events from the type that they are registered for.

In the consumption phase, filters should be created according to the fined user profile and must be implemented by the visual awareness components. Finally, in the analysis phase, filters are provided by the OLAP tools.

3.2 Architecture of Ariane

Fig. 2 illustrates the architecture of Ariane. The components designed and implemented by Ariane are represented by ellipses, and they are the frames named *client* and *awareness server.* Visual awareness components appear to emphasize their role in the awareness solution. However, cooperative application developers must design and implement specific visual components that match the awareness needs of the application users.

The general operation of the architecture occurs as follows: first, *operators* (producers and consumers) interact with instances of a cooperative application, which uses a persistence mechanism to store their artifacts in a database (this flow is represented in Fig. 2 by dotted arrows). A *sensor* is plugged in to the persistence mechanism and listens to the communication between the application and the database, reporting all collected information to the awareness server. In the *awareness server,* the *event handler* component checks the event type, creates an event message and sends the event to all registered components, including the *storage handler* component. The *storage handler* is responsible for storing the event in the *event DB,* reading the events already stored, and answering queries performed by visual awareness components. The *visual awareness components* receive events from the event handler (in a synchronous mode related to the action occurence) and from the storage handler (in an asynchronous mode). They extract the awareness information from the event messages, prepare and present them to the consumer users, considering their roles and profile. Periodically, *ETL (Extract-Transform-Load) processor* component catches

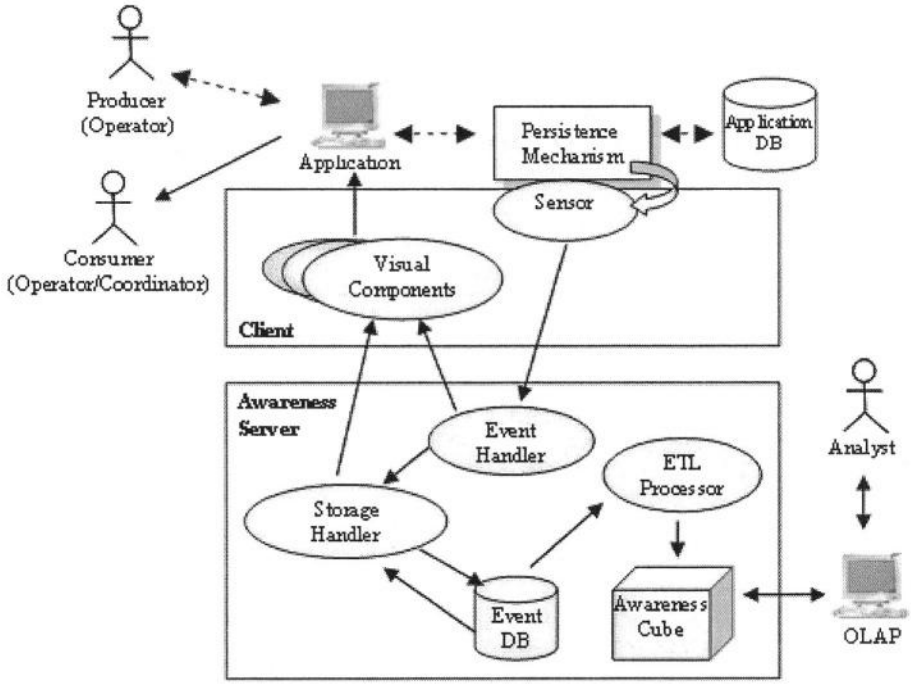

Fig. 2. Architecture of Ariane

events from the *event DB* and loads them in the *awareness cube, where* the analysts can use OLAP tools to execute *ad-hoc* analytical queries.

4 Ariane Prototype Development

In order to validate the feasibility of the Ariane approach, a prototype of the mechanism was developed in the Java platform. Communication between clients and the awareness server uses Remote Method Invocation (RMI) [11]. The next sections discuss details related to the development of the main components of Ariane.

4.1 Sensors

Ariane is proposed as a flexible and generic awareness mechanism, independent of a specific Database Management System (DBMS) or application. The first concern in the development of the prototype was to define where exactly the Sensor should be placed in such a way that it could monitor and collect the actions in a non-intrusive way, requiring no changes in the application and in the DBMS. To accomplish this, we analyzed the communication between a generic application and a DBMS...

Java was our development platform, therefore, we decided to monitor Java applications using a standard interface for persistence: Java Data Objects (JDO) [5], that

specifies a set of interfaces that define transparent persistence. Using JDO, the application is capable of persisting artifacts using any JDO compliant DBMS. Currently, JDO has many implementations (commercial and open source) and is being considered as a standard for persistence in Java desktop applications. Also, is crucial to make sure that the collection of awareness information is done over the object-oriented application data model and not the underlying persistence relational data model. This issue is fundamental in the event consumption phase when the awareness information gathered can easily be understood by the application users, since its semantic is in the same level of the application semantics.

The Sensor must be connected to a JDO-compatible persistence mechanism (JDO implementation) to monitor the events. The second main issue concerning the prototype development was how to implement the Sensor without changing a specific JDO implementation "by hand"? We must consider that better and more robust JDO implementations are commercial and their source codes are not available to be changed.

To solve this issue, Ariane implements the Sensor as an *aspect.* Aspects constitute the programming unit in the software development paradigm called Aspect Oriented Programming (AOP) [6]. Aspects describe and implement application crosscutting concerns, clearly separating them from the application base code. A weaver mechanism merges the aspect code with the application base code so that they are compiled as one single unit. Ariane uses AspectJ [12], that extends the Java language with AOP constructs. AspectJ enables weaving over bytecodes, therefore it is not necessary to access, to know or to understand the JDO implementation source code, since the Sensor, as an aspect, specifies the monitoring code based on JDO standard interfaces . Fig. 3 ilustrates the overall idea of how AOP works in Ariane.

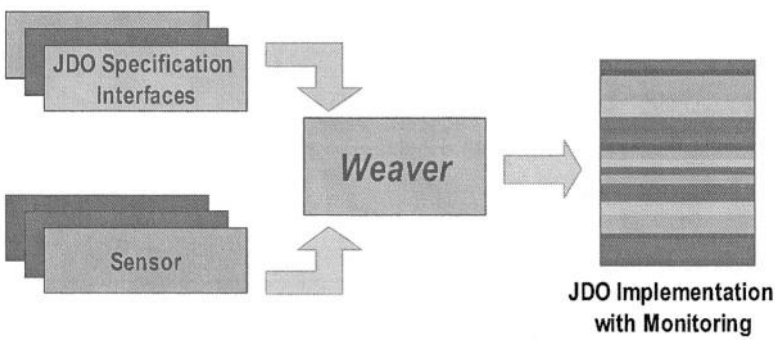

Fig. 3. Weaving of sensor code and JDO Integration interfaces using AOP

One important advantage in using the AOP approach is that it makes possible to implement additional sensors. Thus, different event types could be defined (as, for example, mouse moving or key pressing) or even the monitoring of different persistence mechanisms or the application itself could be implemented is a similar way.

4.2 Event Handler

The event handler receives, from the sensor, information about the action and creates correspondig events according to the 5W+1H format. These events are propagated to

the storage handler and to all visual awareness components that have registered interest in the event type. This propagation was developed following the model used in the Java Abstract Windowing Toolkit (AWT) and Java Beans [13]. The events are classified by their types and each event has a corresponding event class and listener interface. The visual awareness components should implement the listener interface related to their desired event types and should register themselves as interested in the event handler. When a new event is created, the event handler verifies all registered listeners for that event type and sends them a notification message.

4.3 The Awareness Cube

Multidimensional modeling is a discipline that structures data with the purpose of analysis and performance and is a common format in OLAP tools [14]. The modeling of the awareness cube is based on two structures proposed in the technical literature: the CRUD Cube [15] and the DataWebHouse [14]. The awareness cube was modeled using the star scheme, a widely used form of multidimensional modeling. This scheme consists of one central table, named the fact table, which generally contains a huge amount of data and is linked to several peripheral tables, named dimension tables, which qualify the data. In Ariane, the fact table stores the events and the dimension tables store the 5W+1H information, increased with as much information as the Sensor can capture.

A standard ETL processor was used to load the awareness cube with data extracted from the event database using an ETL procedure.

4.4 An Example of Use of the Ariane Prototype

Some tests were conducted with the Ariane prototype using the OdysseyShare SDE. OdysseyShare SDE is developed in Java and can use several persistence mechanisms. JDO compliant or, partially compliant, mechanisms include the JDO Genie [16] and the persistent object manager GOA [17]. GOA is currently the OdysseyShare main persistence mechanism due to its capability of manipulating distributed and mediated databases [18], and XML documents [19].

We choose to monitor the JDO Genie persistence mechanism since it is considered one of the best and more popular JDO implementations available. To connect the Ariane sensor and the JDO Genie, the AspectJ weaver was executed over the main .jar file of the JDO Genie generating a new .jar file, weaved with the Sensor functionality. The new JDO Genie jar file started to be used by OdysseyShare instead of the old one. No changes had to be done in OdysseyShare in order to include the monitoring functionality. To OdysseyShare users, the monitoring occurs in a transparent way and they proceed using the application as before.

Two widgets were constructed to present the awareness information gathered from monitoring OdysseyShare: (i) *ConnectionReport* (Fig. 4) that shows Session Events, reporting which user has connected to which database, open connection time (open column) and the time they have disconnected (close column), and (ii) *EventMonitor,* a tabular interface that shows all events, and all 5W+1H information gathered, acting as the application log.

Widget ConnectionReport

User	Database	Open	Close
vaninha	jdbc:microsoft:sqlserver://l...	Tue Oct 21 15:09:0...	
vaninha	jdbc:microsoft:sqlserver://l...		Tue Oct 21 15:09:2...
mangan	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:00:4...	
mangan	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:10:4...
mangan	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:13:1...	
mangan	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:18:4...
mangan	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:21:0...	
mangan	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:21:4...
vaninha	jdbc:microsoft:sqlserver://l...	Tue Oct 21 17:26:2...	
vaninha	jdbc:microsoft:sqlserver://l...		Tue Oct 21 17:34:3...

Fig. 4. Visual Awareness Component displaying server connection events

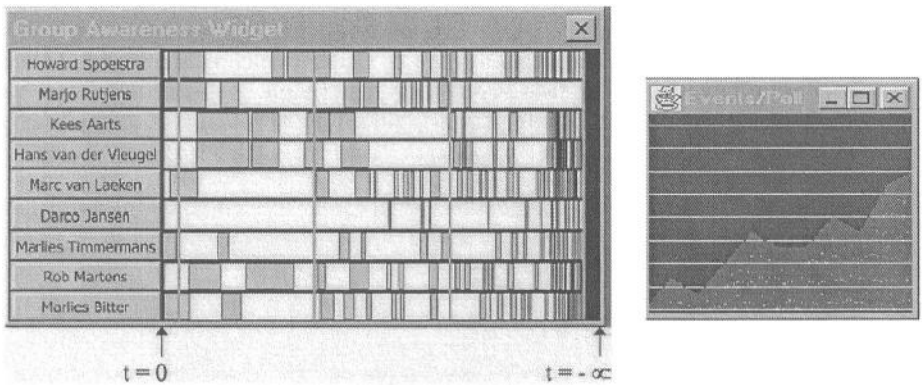

Fig. 5. Visual Awareness Components (a) GAW [20] and (b) Awareness Gauge [21]

The events produced by Ariane can be used as data sources to facilitate the creation of additional visual awareness components that match the needs of the application users. The same information can be visualized in different ways as, for example, using a Group Awareness Widget (GAW) [20], a visual component that shows users connected to a system during a period of time (Fig. 5a), or the Awareness Gauge Events/Poll [21], a component that presents an amount of events occurred in periods of time (Fig. 5b). Therefore, Ariane helps to reduce the effort of visual awareness component developers, since they can just concentrate their efforts in presentation issues.

An additional possibility for event visualization is provided by OLAP tools (Fig. 6). These tools provide functionalities that help analysts to understand patterns and trends in large collections of data. An analyst aware of the group objectives and activities can benefit from this information to evaluate group performance and detect situations where an intervention is needed.

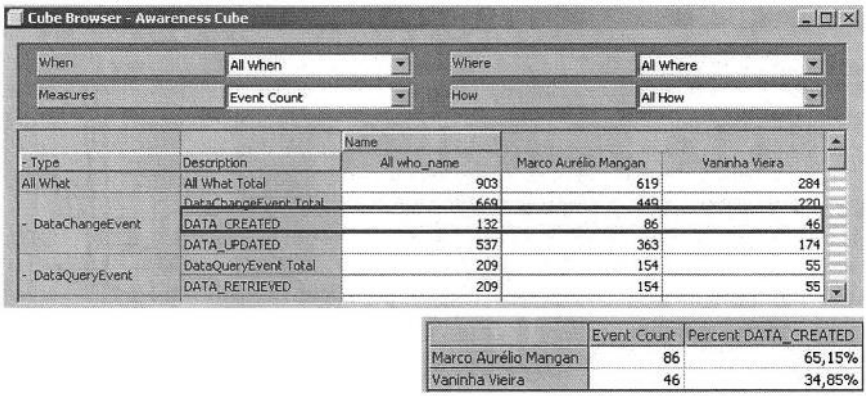

Fig. 6. Awareness Cube – Example of query using an OLAP tool

5 Related Work

There are several awareness mechanisms proposals. Most of them are strongly coupled to a specific cooperative application. Few approaches propose generic and reusable awareness mechanisms, such as SISCO [22], NESSIE [23] and BW [1]. Also, none of the works found reports the use of OLAP to improve awareness in cooperative applications.

SISCO is a cooperation filter for a multi-user generic object-oriented database (OODBMS). Awareness information is gathered by trapping all accesses made to the shared objects from both cooperation-aware and unaware database clients. The filter is responsible only for the cooperation mechanisms, therefore the underlying database is left unaltered. However, the SISCO prototype implementation was done through extensions to a specific OODBMS, the Oggetto. Thus, it is restricted to applications that use that OODBMS. Also, SISCO imposes changes in applications that must connect to the filter instead of the DBMS. Since Ariane monitors a standard persistence mechanism, there are no restrictions related to the DBMS used by the application and no changes in the application are necessary.

NESSIE is an awareness infrastructure for cooperative environments. Key elements of NESSIE are an application independent generic infrastructure, an open and extensibleprotocol including dynamic event types, and a set of sensors and configurable indicators, both for discrete and contextual event notifications. However, the events captured by NESSIE sensors come from changes occurred in a specific NESSIE client or the events are explicitly communicated by the applications using HTTP and CGI scripts. In Ariane, events are generated in an automatic and non-intrusive way, being gathered from the current persistence process of the application.

Finally, BW is a framework that supports asynchronous awareness of past events and has been designed to be used when developing new groupware applications and also to improve existing ones. By being a framework, BW does not implement all functionality needed to promote awareness, demanding extensions to be done in the groupware application in order to use its services. Also, BW forces applications to communicate directly event ocurrencies. Ariane can be used as a plug-in component, which can be connected to or disconnected from the application, without changing its behavior.

6 Conclusion

This paper presents a generic and reusable awareness infrastructure independent of a specific application or DBMS. Ariane improves the availability of awareness information to different cooperative applications by monitoring the application persistence mechanism. This infrastructure can be reused in the construction of many cooperative applications and awareness components.

Ariane monitors application actions performed over its persistent artifacts and consequently over the application database itself. Ariane uses an awareness process based on four tasks: production, distribution, consumption and analysis of events. Events are produced by sensors, coupled to the application persistence mechanism. Event data can be easily consumed and interpreted by different visual awareness components. Events are stored in a database, therefore allowing the building of group memory. The proposal of an awareness OLAP cube, a multidimensional structure to organize and mine the event database, may help work group specialists to analyze group interactions. OLAP tools allow ad-hoc queries, appropriated to decision making and analysis of group interaction.

A prototype of Ariane was developed using the Java Data Objects persistence mechanism and aspect-oriented programming techniques, which were employed in order to increase the reusability potential of the solution. A preliminary evaluation of the prototype applied in OdysseyShare SDE, an environment for cooperative software development based on components confirmed that no additional code is necessary to monitor JDO complaint applications. Ariane helps cooperative application developers in the construction of new awareness solutions, since they can focus on information processing and presentation issues.

The Ariane prototype opens up many many possibilities of research. Currently, we are evaluating three different applications. First, a future work in the CSCW field is to use this approach to create *context-aware applications* [24], using sensors to help identify context elements. Second, in the software engineering field; another possibility would be the development of *Personal Software Process (PSP) applications* [25], monitoring patterns that would reveal current practices used by a software team. Another application in software engineering would be the construction of operation-based configuration management applications, that need to control all the artifact manipulation occurred in a given period of time.

Current work tries to development of sensor implementations for different persistence platforms and applications and is concerned with the organization of large collections of application event data in different application domains. These data will be explored in future analysis.

Ackowledgments

The authors would like to thank CAPES and CNPq for their financial support. The first author also thanks the UFBA for their support.

References

1. Pinheiro M. K., Lima J. V., Borges M. R. S.: A Framework for Awareness Support in Groupware Systems. In: Proc. 7th International Conference on CSCW in Design, Rio de Janeiro, Brasil, (2002), 13-18
2. Sohlenkamp M., Prinz W., Fuchs L.: POLIAwac: Design and Evaluation of an Awareness Enhanced Groupware Client, AI & Society Journal, v. 14, (2000), 31-47
3. Gutwin C., Greenberg S.: A Descriptive Framework of Workspace Awareness for Real-Time Groupware. In: Computer Supported Cooperative Work, v. 11(3-4), Special Issue on Awareness in CSCW, Kluwer Academic Press, (2002), 411-446
4. Preguiça N., Marting J. L., Domingos H., Duarte S.: Data Management Support for Asynchronous Groupware. In: Proc. of the 2000 ACM Conference on Computer-Supported Cooperative Work, Philadelphia, PA, USA, (2000), 68-78
5. Russell C.: Java Data Objects (JDO) Specification - Final Release. In: http://jcp.org/aboutJava/communityprocess/final/jsr012/index.html, Access in 06/2004
6. Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes C.V., Loingtier J.M., Irwin J: Aspect Oriented Programming. In: Proc. of the European Conference on Object-Oriented Programming, v. 1241 of LNCS, Springer-Verlag, (1997), 220-242
7. Mangan M. A. S., Araújo R. M., Kalinowski M., Borges M. R. S., Werner C. M. L.: Towards the Evaluation of Awareness Information Support Applied to Peer Reviews of Software Engineering Diagrams. In: Proc. of the 7th International Conference on CSCW in Design, Rio de Janeiro, Brasil, (2002), 49-54
8. Werner C. M. L. et al.: OdysseyShare: an Environment for Collaborative Component-based Development. In: IEEE International Conference on Information Reuse and Integration, Las Vegas, USA, (2003), 61-68
9. Braga R. M. M., Werner C. M. L., Mattoso M. L. Q.: Odyssey: a Reuse Environment Based on Domain Models. In: 2nd IEEE Symposium on Application-Specific System and Software Engineering Technology, Richardson, USA, (1999), 50-57
10. Borges M. R. S., Pino J. A.: Awareness Mechanisms for Coordination in Asynchronous CSCW. In: 9th Workshop on Information Techonologies and Systems, Charlotte, North Carolina, (1999), 69-74
11. Sun: Java Remote Method Invocation (RMI), In: http://java.sun.com/products/jdk/rmi/, Access in 06/2004
12. AspectJ: AspectJ Project Home Page. In: http://www.aspectj.org, Access in 06/2004
13. Sun: JavaBeans Specification. In: http://java.sun.com/products/javabeans/docs/spec.html, Access in 06/2004
14. Kimball R., Merz R.: The Data WebHouse Toolkit, New York, USA, John Wiley & Sons, Inc., (2000)
15. Sulaiman A., Souza J. M., Strauch J. C. M.: The Crud Cube. In: Technical Report ES-616/03. COPPE/UFRJ, (2003), http://www.cos.ufrj.br/publicacoes/reltec/es61603.pdf. Access in 06/2004
16. Hemisphere: JDO Genie. In: http://www.hemtech.co.za/jdo/index.html, Access in 06/2004
17. GOA: GOA Home Page. In: http://www.cos.ufrj.br/~goa/, Access in 06/2004
18. Souza R. P., Costa M. N., Braga R. M. M., Mattoso M. L. Q., Werner C. M. L.: Software Components Retrieval Through Mediators and Web Search, Journal of the Brazilian Computer Society, v. 8, n. 2, (2002), 55-63
19. Vieira H., Ruberg G., Mattoso M. L. Q.: Xverter: Querying XML Data with ORDBMS. In: Web Information and Data Management. In: Fifth International Workshop on Web Information and Data Management. ACM Press., New Orleans, USA, (2003), 37-44
20. Kreijns K., Kirschner P. A.: The Social Affordances of Computer Supported Cooperative Learning Environments. In: 31th ASEE/IEEE Frontiers in Education Conference, Reno, NV, (2001), 12-17

21. De Souza C. R. B., Basaveswara S. D., Redmiles D. F.: Using Event Notification Servers to Support Application Awareness. In: IASTED International Conference on Software Engineering and Applications, Cambridge, MA, (2002), 691-697
22. Mariani J. A.: SISCO: Providing a Cooperation Filter for a Shared Information Space. In: Proc. of the International ACMSIGGROUP Conference on Supporting Group Work: The Integration Challenge, Phoenix, Arizona, USA, New York: ACM Press, (1997), 376-384
23. Prinz W.: NESSIE: An Awareness Environment for Cooperative Settings. In: Proc. of the Sixth European Conference on Computer Supported Cooperative Work, Copenhagen, Denmark, (1999), 391-410
24. Dey A. K.: Understanding and Using Context. In: Personal and Ubiquitous Computing Journal, v. 5(1), (2001), 4-7.
25. Humphrey W. S.: The Personal Software Process (PSP). In: Technical Report CMU/SEI-2000-TR-022, http://www.sei.cmu.edu/publications/documents/00.reports/00tr022.html, Access in 06/2004.

Design of Awareness Interface for Distributed Teams in Display-Rich Advanced Collaboration Environments

Kyoung S. Park[1], Jason Leigh[2], Andrew E. Johnson[2], and Yongjoo Cho[3]

[1] Digital Media Laboratory, Information and Communications University
517-10 Dogok-dong, Gangnam-gu, Seoul, Korea
park@icu.ac.kr
[2] Electronic Visualization Laboratory, University of Illinois at Chicago
851 S. Morgan, Chicago, IL, 60607, USA
cavern@evl.uic.edu
[3] Division of Media Technology, Sangmyung University
7 Hongji-dong, Jongno-gu, Seoul, Korea
ycho@smu.ac.kr

Abstract. This paper discusses design issues for enhancing awareness among distributed teams in the display-rich advanced collaboration environments. We conducted an exploratory design study of nine groups performing a set of collaborative tasks using a variety of advanced collaboration and display technologies. The result showed that group interaction and awareness was improved by the use of multiple public displays due to information visibility to all members. While maximized visibility supported work activity awareness, it also revealed a need to support shared resource awareness and more importantly task awareness for effective group coordination.

1 Introduction

With the emergence of collaboration technologies, people can easily communicate with one another and accomplish complex tasks even though they are distantly located. Major corporations launch global teams, expecting that technology will make virtual co-location possible. While these technologies are valuable, there has been a reassessment of the basic value of working co-located in physical spaces. A recent field study conducted at several corporate sites investigated the work of teams who were maximally co-located, i.e. working in war rooms [8]. One of the key features in war rooms is awareness. In a war room, a group of team members not only share the office space but also work together synchronously in all phases of the project. In some cases productivity in war rooms can be enhanced far beyond the corporate average [13].

An Amplified Collaboration Environment (ACE) is a distributed extension of a war room that aims at augmenting the traditional concept of the war room with technologies to permit distributed teams to make use of its problem solving benefits [7]. The Continuum is an ACE specifically targeted for supporting collaborative scientific

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 105–120, 2004.

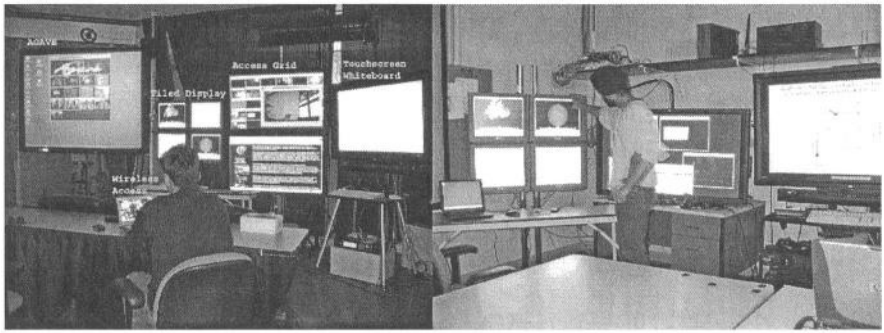

Fig. 1. Two Continuum Amplified Collaboration Environments built at the Electronic Visualization Laboratory at UIC, to facilitate an explorative usability study.

investigation connected via high-speed networks to high-performance computer and data resources [10]. Current off-the-shelf collaboration tools such as NetMeeting® do not support the kind of interaction that occurs in real science campaigns. Scientists want more than just being able to videoconference and share spreadsheets with each other. They want to collaboratively query, mine, view and discuss visualizations of enormous data sets (in the order of terabytes) in real time. The visualization systems that are capable of displaying data sets of this size require more than desktop PCs.

Fig. 1 shows the displays that comprise fully constructed Continuum spaces at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago. Two Continuum spaces are built at EVL to develop technologies and to facilitate an explorative usability study. The Continuum consists of following modular technologies: conferencing, content sharing, collaborative annotation, and wireless interaction. The Access Grid (AG) supports group-to-group collaboration in which a group of people at different locations can see and talk with one another simultaneously [1]. The high-resolution tiled displays provide shared content views of text documents, web pages, spreadsheets, graphs and charts, and scientific visualizations. The collaborative annotation is supported by shared touch-screen whiteboard on which collaborators may jot down notes and sketch diagrams. Also, remote access interface is supported to encourage users to work on these displays collectively.

We are currently investigating how the Continuum's tiled displays can be used in enhancing group awareness between distributed participants during intensive collaborative work. Awareness has been extensively studied in computer supported cooperative work (CSCW) research and identified as a key feature for collaborative systems [5]. Yet, maintaining awareness over the distance has been shown to be difficult because a lot of this information is not easily conveyed to remote users with today's technology. Tiled displays are typically used to project a single, extremely large, high-resolution visualization. It is our belief, however, that for collaboration, a better way to use a tiled display is to treat it as a large distributed corkboard that allows meeting participants to pin up information artifacts for all to see.

A few recent collaborative systems such as Notification Collage [4] and Semi-Public Displays [6] have used similar approaches. They use a large public display to foster awareness among distributed or co-located workgroups, but mostly focus on asynchronous collaboration. We aim to create shared workspaces where individuals can work in parallel while maintaining group awareness by giving the individual the ability to casually glance at others work over the distributed corkboard tiled display.

This paper presents a set of iterative design studies conducted to examine the system configuration of Continuum technologies for distributed teams tackling scientific problems. The study involves placing a group of collaborators in two separate Continuum spaces and asking them to perform a variety of information discovery tasks. The goal of this study was to explore design issues and group's needs on various collaborative tasks to facilitate war room like interaction for distributed teams. In this paper we describe the design changes of the Continuum technologies, the findings and lessons learned from the study with an emphasis on awareness issues, and ideas for future research directions.

2 Participants

Nineteen computer science graduate students volunteered as subjects in this study. All students had a high level of experience with computers and collaboration technologies such as email and instant messaging. Some students have used Microsoft's NetMeeting® or other commercial or research online meeting room systems. Some had experiences with information visualization tools, but none of them had prior experience with the XmdvTool tool [14] that was used as a part of this study. Most students had little to moderate experience with correlation statistics. All students expressed fairly high interests in collaborative work using the Continuum technologies.

3 Method

Table 1 shows the changes of group and system configuration over the iterative design studies. The study consisted of a pilot study followed by four iterative studies. For each study, the system configuration was varied, and we evaluated mainly on the configuration of the tiled display. The system configuration was changed in response to the participants' feedback and our observations. The iterative studies were conducted over three-week intervals due to the time required to reconfigure the systems for the each study. The pilot group was formed with three students, and two groups of four students were assigned in the following four design studies. All students participated in the two design studies, and each time the groups were given different question sets with similar difficulties, to reduce the learning effects from previous exposure. In the third and fourth study, we regrouped the members to see if they broadened their ideas best about the best way to use this technology.

Table 1. System configuration changes over iterative design studies.

Study	Group	System Configuration				
		Conference	Visibility	Seamlessness	Proximity	Size
Pilot Study	Pilot Group	Full-AG & mini-AG	Public visibility display	Discrete display		
Study 1	Group 1 Group 2	Full-AG & mini-AG	Public visibility display	Seamless display		
Study 2	Group 3 Group 4	Full-AG & Enhanced mini-AG	Public visibility display	Seamless display	With close-up display	
Study 3	Group 5 Group 6	Full-AG & Enhanced mini-AG	Less public visibility by full-screen	Discrete display	With close-up display	Full-screen
Study 4	Group 7 Group 8	Full-AG & Enhanced mini-AG	More private visibility display	Discrete display	With close-up display	Full-screen

The group was given a pre-test survey (e.g. technology familiarity, comfort, interest, and domain knowledge) and received one hour training about the technologies and some basic knowledge required in their first exposure to the tasks. The group members were then separated in two Continuum spaces and asked to solve a set of collaborative tasks (approximately 3-hour sessions). A post-test survey and interview was followed shortly at the end of the tasks for feedback about the usability of Continuum technology.

The tasks given to the groups were information query and gathering (45 minutes), information analysis and pattern detection of multivariate data (60 minutes), and collaborative brainstorming and design (30 minutes). In the information querying and gathering task, the group was asked to search and gather information on the web to answer specific questions. The group was given two focused questions and one trend question. In the focused questions, group members would need to simultaneously gather as much information as possible from the web. In the trend question, members would need to make a group decision based on their combined findings. In the information analysis and pattern detection task, the group was asked to perform an exploratory data analysis on a given dataset using the XmdvTool information visualization tool to answer questions. There were five specific questions given to the group where members would find evidence to verify or refute hypotheses. Two trend questions were given where members would need to search for inquired trends or patterns in the dataset. In the collaborative brainstorming and design task, the group was asked to brainstorm, prioritize, and summarize design ideas for Continuum technologies.

All groups were recorded using video cameras. An evaluator in each room also recorded group behaviors, which were taken into the observation notes. All activities of group members on the computers were also captured into log data files such as mouse

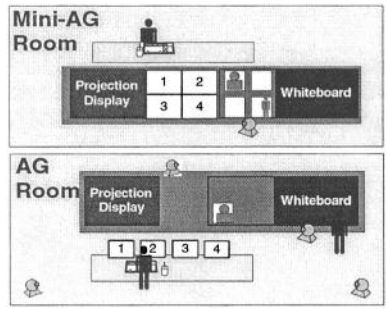

Fig. 2. The system configuration of evaluating the distributed corkboard in the pilot study.

movements across the screens. We also measured group performance such as work quality and group work process which included members' participation and contribution, group interaction for decision-making, information exchange, awareness and attention, influence, and team coordination. We also used post-test survey to measure user satisfaction for working with technology and work process.

4 System Configuration

In the pilot study, we evaluated the initial system configuration of the Continuum. The Continuum uses the tiled display as a distributed corkboard where information is visible to all members and users can casually glance over at other's work. This pilot study was tested to understand how the distributed corkboard would help group awareness and parallel work for distributed group's collaborative work. Each Continuum space had an Access Grid, a shared whiteboard, a projection screen, and 4-node tiled display (see Fig. 2). On the left a projection display was provided to allow users to project one of the tiled display screens in a large format. Next to it was the tiled display (1 x 4 table mounted in the full AG room; 2 x 2 wall mounted in the mini-AG room). The Switcher program allowed any user to grab the remote keyboard and mouse control for any of the tiled display screens [7]. It provided a way to quickly switch user's input control from a laptop or tablet PC to any tiled display nodes. Next was a plasma screen that was used for AG multi-site video conferencing. To the right was the plasma touch-screen that was used for shared whiteboard. The whiteboard was connected between two sites via NetMeeting. Only one keyboard and mouse was provided in each site, e.g. the two co-located members had to share one input control.

In the first study (as shown in the left image of Fig. 3), we evaluated the seamless distributed corkboard that was designed to give users an illusion of one continuous display. The main technology addition was SpaceGider, a software interface that allowed users to navigate their mice across four tiled display screens seamlessly [2]. We tried to replicate the display setting as much as possible in both sites: the tiled display (2 x 2 layout) on the left wall, AG display in the middle (4 cameras and 2 microphones in full AG setting, and one camera and one microphone in mini-AG setting), the shared touch-screen whiteboard on the right, and four keyboards and

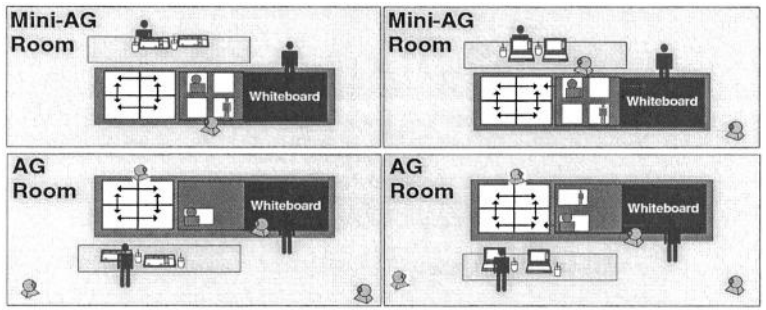

Fig. 3. The system configuration of evaluating the seamless distributed corkboard using SpaceGlider in Study 1 (on the left) and the seamless distributed corkboard with personal displays in Study 2 (on the right).

mouse (one input control for each user). Full-sized keyboards and mice were given to the participants to encourage them to casually look at others' work on the tiled display. The shared whiteboard between rooms were connected using the NetMeeting's shared whiteboard application.

In the second study (as shown in the right image of Fig. 3), the system configuration was changed to add personal displays and improve the mini-AG setting. The personal displays, such as table PCs, were provided to give users a close-up view. In mini-AG setting, we added another microphone and a video camera with a magnifying filter on the close-up camera to help casual interaction between distributed participants. In addition, we used SpaceGlider to connect four tiled display screens and the whiteboard.

In the third study (as shown in the left image on Fig. 4), the system was configured to provide a discrete flexible tiled display to support an easy transition between individual work and group work. It allowed users to view either four individual screens or one screen maximized over the entire tiled display. Any user could turn on or off a full-screen option (to maximize his/her workspace over the entire tiled display) at any time. This desktop sharing was implemented by using Aura [11]. The tablet PC was used to mirror one of the tiled display with each tile screen having a distinct background color for easy identification of workspace. We provided Switcher for users to access the tiled display and the whiteboard which allowed the users to jump to any of the displays. To improve the display layout, we swapped the location of AG and the tiled display so that the tiled display was centered and next to the whiteboard.

In the fourth study (as shown in the right image on Fig. 4), the system was configured to provide a presentation-style display. The presentation-style display allowed only one individual's private workspace, i.e., tablet PC to be visible at a time on the public group display, i.e., tiled display. With this configuration, group members could work individually on their tablet PCs and anyone could choose to show his/her workspace onto the tiled display. Also, they could hide their workspace if they did not want to show it to others. They could show their individual workspace on the tiled display as either one large full-screen or four identical (cloned) small screens. Individual workspaces had the same distinct background colors given in the third study.

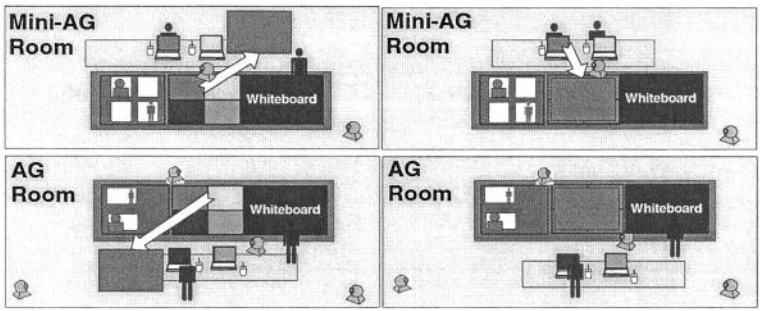

Fig. 4. The system configuration of evaluating the discrete flexible display with personal displays in Study 3 (on the left) and the presentation-style display with personal displays in Study 4 (on the right).

This time, participants were only allowed to switch between his/her workspace and the whiteboard on the tablet PCs. Besides the configuration changes on the tiled display, the other settings were the same as the third study.

5 Observations

5.1 Communication

Similar to war rooms [8, 13], overhearing pattern was also observed over the AG in this design study – when one member was explaining something to others, remote members could overhear and interject clarifications and corrections. We argue that the sufficient quality of audio conferencing is necessary to capture all conversations to support overhearing and easy communication. The first study showed the need to improve mini-AG setting. We believe this was because the group members talked to remote collaborators freely at various places in the room. Also, Group 3 had audio problems during the task, but it was not easily overcome by the use of other mediums such as text chat.

The result of this design study shows the necessity for additional video cameras for group member close-up view and additional video images of shared resources to convey members' spatial reference. In fact, participants often gazed at the video image of the remote collaborator's close-up face during the course of a discussion. This close-up view of the collaborator's face helped them get some forms of deictic reference or small feedback signal (e.g. nodding, murmuring, or facial expressions from the listener). The video window of remote collaborator's view placed close to meeting participants also seemed to help casual interaction among remote participants.

The additional camera and microphone in mini-AG setting introduced since the second study seemed to help increase remote interaction. For example, the read-and-write collaboration occurred between two distributed members over the AG. The read-and-write collaboration is a pattern of collaboration between two members to

Table 2. The frequency of read-and-write collaboration pattern

	Read-and-write collaboration between local participants		Read-and-write collaboration between remote participants	
	Information query and gathering	Information analysis and pattern detection	Information query and gathering	Information analysis and pattern detection
Study 1	2	7	0	0
Study2	9	7	0	2
Study 3	3	2	2	3
Study 4	4	2	0	0

transfer data across displays via voice channel. That is, one person read text (e.g. answers) from the tiled display while the other wrote it down on the whiteboard. Table 2 shows the total number of frequency of groups' read-and-write collaboration pattern. This read-and-write collaboration between remote members was observed in the second and third study, but it disappeared in the fourth study.

In the fourth study, no particular audio and video problem occurred, but the groups tended to have less interaction over the AG. The participants indicated that the decrease in the number of remote interactions was due to less visibility. They said that during the first or second study, they could maintain group awareness by seeing each other's work and talking to each other to ask questions and discuss ideas when they were given the distributed corkboard; whereas, they had to rely on strictly on overhearing to obtain such awareness information.

Interestingly, it was observed that the participants in the second and third study used the video of whiteboard view (or overall view) for shared resource awareness, i.e., to get information about who was using the shared whiteboard. In particular, Group 5 showed a large amount of whiteboard usage, which was similar to that of Group 2. But, unlike Group 2, Group 5 participants quickly adopted using the whiteboard view which reduced further possible conflicts. We expected frequent use of the video sources for shared whiteboard awareness by group members, but they opted for verbal communication to announce their intentions prior to using the whiteboard or being done using the whiteboard. These actions helped resolve possible conflicts, but they were burdensome. This result indicates that we should provide simple awareness tools (such as, beep sound indicating someone's using the whiteboard when another member is also approaching the whiteboard) to help ease the burden of turn taking among the group members on shared resources.

5.2 Visibility

The distributed corkboard supported high visibility of all members' work activities. The group members were aware of what others were doing by glancing over their tiled screens. They could also easily refer to the contents on the screens. Such visibility supported by the distributed corkboard was useful when the group worked together to solve problems sequentially. While working together, one participant con-

trolled the mouse on tile screens while the others could see this manipulation and added their insights into the analysis. It was also useful when the group moved frequently between individual work and group work. In this case, individual work on tile screens could become a focus of group attention when the group wanted to work together. However, visibility seemed to be less useful when the group divided the task and then simply combined individual work. In this case, the group members worked largely independently, and interaction was limited to sharing task progress or asking for assistance.

It was observed that visibility became more important as the groups showed increased interaction and collaboration between remote members, e.g. remote instruction. When a Group 4's member helped her remote collaborator, she wiggled her mouse cursor to point out what she referred to on her screen. Visibility helped implicit peer learning since it allowed users to observe how others tackled the same problem. Some participants reported that they got ideas and learned from others by observing what others were doing so that they did not have to ask questions, e.g. how to select all variables in the data set. Some also said they shared searching strategies and built new strategies from those of other group members by looking at the other's work. Visibility also supported the immediacy to access to information and experts. For example, members found useful information or answers from collaborators' work, and one's difficulty could be shown to remote collaborators. Overall, the participants indicated that they did not pay much attention to the distributed corkboard tiled display but it was helpful to have information visible on the tiled display all the time.

Another important pattern was that the participants often asked others to take a look at one of their screens, to point out their interests. They also used this pattern to get group attention bring them into group discussion by showing one's finding to others. Table 3 showed the pattern of "look at (this)" or "see (my screen)" to draw attention when a member wanted to indicate his or her finding to the collaborators. This is a typically associated pattern when one wants to bring other user's attention to a certain part of the display during the collaborative work session. The frequency of "look at" pattern seemed to be varied by the group's working style, but it appeared consistently throughout the studies.

The presentation-style display in the fourth study disallowed a user's casual glancing over at another's work but instead allowed information to be visible by presenting one's workspace to the tiled display so that the whole group members could see. However, such reduced visibility resulted in degrading group performance and interaction, as compared to the groups in the third study. Unlike the previous studies, the "show me" pattern appeared due to less visibility (See Table 3). This pattern was an explicit request to make information visible – for example, one group member asked the other to show his/her workspace or another member offered to show his/her own workspace in order to share information with others. This pattern was observed when one wanted to present something to others or to solve the problem together when someone had a problem.

Having experience with the distributed corkboard, the participants disliked going back to a classical round-robin style of Power Point Presentation model of collabora-

Table 3. The frequency of Look at and Show me pattern

	Look at		Show me	
	Information query and gathering	Information analysis and pattern detection	Information query and gathering	Information analysis and pattern detection
Study 1	6	13	0	0
Study2	14	38	0	0
Study 3	6	13	0	0
Study 4	13	1	7	11

tion. They wanted to see all of the information on the tiled display and compare to each other. The fourth study's participants stated that they could easily see information over the distributed corkboard when they wanted to see and needed to share, whereas they had to request to see information over the presentation-style display. This request became a source of delay when the group wanted to share information.

5.3 Awareness

Some participants benefited from using the distributed corkboard to maintain group awareness. The group members seemed to use various channels to be aware of their partner's activities. According to post-test user surveys, the participants maintained group awareness by listening to conversations, asking what people were doing, looking at the AG video windows, and looking at other's work over the distributed corkboard. While the distributed corkboard seemed to help overall group awareness, the participants often checked task progress over the AG, to get task awareness. That was done by asking the remote collaborators which question they were working on, or by informing them about what they had done and what they were going to do next. This pattern of synchronization was observed constantly throughout the studies and more frequently observed when the group showed mixed focus collaboration, i.e., the members worked independently on their individual workspaces most of the time and worked together from time to time on one of their individual workspaces to verify or discuss findings.

The presence of tablet PCs (close-up display) introduced in the second study helped the group members focus on their individual work, but it seemed to result in reducing the number of users' glancing over at other's work on the distributed corkboard tiled display. The overall subjective rating showed that the participants focused on their own work using close-up personal displays and then checked other's work once in a while over the distributed corkboard or by quick glimpse to local collaborator's personal display. The participants in the third study also showed this similar behavior, i.e. working mostly on their close-up displays and occasionally checked the tiled display to see others' activities. Interestingly though, even though they looked at other's work occasionally, the participants perceived the importance of having the distributed corkboard because it supported the immediacy of information – that is,

when they needed to work together, they could easily refer to "look at" the particular screen or maximize the screen so that all of them could see and discuss.

The presentation-style display provided users with more private workspaces with limited information sharing capability among group members. It only allowed one member's private workspace to be publicly visible over the tiled display. It also did not allow users' casual glancing over at collaborator's workspace. As it was discussed earlier, the groups in the fourth study showed less remote interaction. Group 7 showed the divided work pattern, and a large percentage of their conversation over the AG was devoted to task awareness (about 60% during information query and gathering and about 88% during information analysis and pattern detection). Group 8, in contrast to Group 7, showed the pattern of group's working together in the information analysis and pattern detection task; however, a fairly large amount of interaction over the AG had been shown for task awareness and the "show me" pattern. The participants in these groups commented that they did not look at the tiled display in casual manner unless the work was presented.

5.4 Seamless or Discrete Display

The seamless distributed corkboard seemed to introduce users to feel more continuity of the tiled display. For example, the participants even tried to move the windows (e.g. a web browser) from one tile screen to another. While SpaceGlider gave users the illusion of tiled display as one big continuous display, the multiple mouse pointers presented in the same screen made the individual mouse identification more difficult. This problem was identified by user's mouse accidentally entering the adjacent collaborator's screen, i.e., when they tried to adjust the window size at the screen corner.

Similarly, the second study groups also had mouse sharing and identification problems due to SpaceGlider. The groups stated that SpaceGlider was intrusive to their work by causing unnecessary mouse conflicts. These groups also wanted to move windows from one screen to another. However, with SpaceGlider connecting the tiled display and the whiteboard, the groups said they felt no continuity of the workspace because of the AG plasma display. This led us to change the physical layout of the displays in the third study, i.e. swapping the location of the tiled display and the AG display, to increase the continuity when moving a mouse across the displays among the tile screens and the whiteboard.

To solve this mouse sharing problem with SpaceGlider, some groups suggested multiple independent mouse pointers to support simultaneous access to displays. Other groups suggested a solution of providing awareness tools (i.e. distinguishable mouse pointers to indicate who owns a certain screen) and locking mechanism where another mouse could not enter the workspace that was already owned by other members. Since the seamless display presented mouse sharing problems, we changed to provide the tiled display as discrete display using Switcher in the third study. When compared to the seamless display, the pattern of wanting to move windows from one screen to another disappeared with the discrete display. The number of mouse conflicts was also reduced, and the participants felt this was less conflicting. Similar to

the third study, the pattern of desiring to move windows from one screen to another was not observed in the fourth study because of the discrete display. There was no mouse conflict over the tiled display since this configuration did not permit mouse sharing over the other's workspace.

The groups commented that Switcher was preferred to SpaceGlider because of the responsiveness to the user's action for a display – i.e. better to press a key to switch screens then to move a mouse. Switcher was also limited to one individual workspace and did not allow users to accidentally move a mouse to the next workspace. While the groups preferred Switcher for multiple users' collaborative work, they also stated that they would prefer SpaceGlider against Switcher if a single user used the tiled display.

5.5 Resolution, Proximity to Display, and Display Size

The proximity to display was also an important factor in the design of display-rich environments. In the first study, those participants who were assigned to use the upper tile screens had to frequently stand up for a close look at the screen due to the distance of the screen. This result led us to provide the close-up view display such as laptop or tablet computers in the following studies. The presence of tablet PCs helped resolve the proximity to display issue but raised the size issue. The second study groups thought perhaps increased distance from the tiled display would help them focus on monitoring all the displays, but the fonts had to be big enough to read at such distance. They also suggested maximizing the window to fill all screens of the tiled display to help their group discussion.

To address the size issue, we provided the flexible tiled display that allowed users to maximize one individual workspace into the entire tiled display. Group 6 used this full-screen option for group discussion where all members worked together to verify one individual's finding during the information query and gathering task. This group also used this option for personal use (and subgroup discussion) during the information analysis and pattern detection task, e.g. to make a scatter plot graph bigger to see the patterns easier. However, this full-screen tiled display for personal use did not interfere with other members since they could still work on their tablet PCs. Group 6 found this option useful because, sometimes, the image on the tiled display was not big enough. Some participants also indicated that the full screen was used for grabbing other's attention.

The presentation-style display did not allow the group to share information side by side, because it only allowed sharing one individual screen at a time. This affordance created a resolution problem when the groups needed to see two or more views together for a comparison. For example, when Group 8 worked together solving problems one after another during the information analysis and pattern detection task, the group immediately realized the need for multiple screens on the tiled display after the group requested "show me your screen" followed by "show you my screen".

6 Discussion

In the everyday world people are aware of events around them whether that event be just other people, conversations or objects. In the group collaborative work, awareness information is always required to coordinate group activities. In this design study, we have experienced several dimensions of awareness problems such as the mouse identification and conflict problem using SpaceGlider, the whiteboard conflict and resolution between remote participants, and task awareness. Unfortunately, the participants spent much time negotiating these problems and explicitly informing or asking each other their intentions and activities. The use of multiple video sources helped them be more aware of remote participants and it was somewhat useful for resolving the group resource sharing problems. But such problems can be further reduced by the provision of awareness information such as audible or visual cues to indicate who is using shared resources and the coordination tool such as the group activity history tool to manage task progress.

The notion of awareness was defined as "an understanding of the activity of others, which provides a context of your own activity [3]." A huge body of work on awareness has been done both within computer supported cooperative work and outside of it. There were various types of group awareness: activity awareness, availability awareness, process awareness, and many more [12]. Activity awareness is knowledge about the activities of other members on the shared workspace. This includes workspace awareness, i.e., an understanding of others' interaction with the shared workspace [5], and artifact awareness, i.e., information about what (artifact) has changed, in asynchronous systems. Availability awareness is a general sense of who is around and whether those people are available to meet or participate in an activity – for example, instant messenger status cues provide this awareness information. Process awareness is a sense of where members' tasks fit into the stages of the project, what the next step is, and what needs to be done to move the process.

In this design study, the participants constantly but subconsciously gathered awareness information of remote collaborators through various channels, such as overhearing conversations, glimpse of video windows and of the distributed corkboard and the whiteboard. With the high quality Access Grid conferencing, the participants could overhear when problems arose and they helped each other or worked together even though they were remotely located. Casual glancing over at other's work over the distributed corkboard and the shared whiteboard helped them be aware of activities of others. Even though our participants said they did not pay much attention to other's work, the results showed that glancing helped maintain group awareness between remote collaborators. With the presentation-style display that did not support glancing, there was less interaction between remote participants and a greater degree of explicit notification to share individual findings and strategies.

Awareness information can be gathered through overhearing conversation, paying attention to general level of activity, casually looking over the shoulder, and monitoring progress of work. In this design study, overhearing and a general sense of progress monitoring among remote participants was supported through high quality audio and video. Casual glancing was also supported through the maximally visible

Table 4. The frequency of attention calls to group (i.e., all members), subgroup (i.e., the remote site members), and individual locally and remotely.

	Attention call to group, subgroup, and individual	Attention call to individual over the AG remotely	Attention call to group
To get task awareness	56	26	15
To ask remote user to do	54	39	4
To ask for help	55	34	7
To inform or discuss findings	29	7	16
To inform or discuss strategies	36	19	9
To initiate work	32	18	8
To get shared resource awareness	23	19	2
To get user activity awareness	25	16	5
To look at	20	13	4
To talk about problems	8	6	1
To talk about mouse sharing problem	14	1	10*
To talk about screen sharing problem	5	4	1
To talk about whiteboard sharing problem	3	3	0
To talk about audio problem	3	4	0
To indicate reference	3	0	0

**Note that nine attention calls to group occurred during Group 1's information query and gathering due to mouse conflicts.*

distributed corkboard. However, our system did not effectively support conveying user's attention over the distance. Interestingly, a few participants did try to use non-verbal gesture (e.g. looking at the camera and raising his hand towards it) to draw the remote partner's attention. Most participants showed that they just spoke out loud to get others' or group attention. Some participants used the 'full-screen' option or 'show' screen on the tiled display to get group attention as well.

Table 4 shows the total number of attention calls to group (i.e., all members), subgroup (i.e., the remote site members), and individuals located locally and remotely. Most attention calls were made to individual located remotely (57%) followed by group (24%), subgroup (12%), and individual located locally (7%). We expected more of these attention calls to be about problem discussion and conflict resolution or initiating group discussion, but the result showed that attention calls overall were initiated mainly to get task awareness and to ask remote users for assistance or ask them to do a certain action. Most attention calls to individual remotely were triggered to ask for remote help or action. Most attention calls to group were to get task awareness or inform/discuss findings and strategies. We also observed that calling out for group attention usually led to a transition from individual work to group focus work.

As shown in Table 4, attention calls were also made to get shared resource awareness and user activity awareness. We observed a few occasions of two remote participants trying to access the shared whiteboard at the same time which resulted in con-

flicting actions, i.e., one tried to move to the next page while the other tried to write down notes. Obviously, this is not a surprised finding. Previous work have already discussed conflicts on shared resources caused by the lack of shared input feedback, such as unexpected changes on group shared workspace by other user's input [2, 9]. In our design studies, the turn-taking pattern emerged to compensate this shortcoming: always asking others to check the availability of these resources before using them or informing others after using them. This was the reason that such many attention calls were observed for shared resource awareness.

7 Conclusions and Future Work

In this paper we presented iterative design studies to explore design issues for enhancing group awareness and performance among distributed teams in the Continuum. The Continuum is an Amplified Collaboration Environment designed to support intensive collaborative work among distributed teams using advanced collaboration, computation, and visualization technologies. The study involved placing a group of collaborators in two separate Continuum spaces and asking them to perform a set of collaborative scientific tasks, information querying and gathering, information analysis and pattern detection, and collaborative brainstorming and design, while varying the technology configurations. Group interactions and their use of the available tools were observed to determine which tools assisted them in accomplishing the tasks.

The value of the Access Grid is that it enables distributed team members to be brought together into common spaces (called as virtual venue in AG) for group interaction. Through the high-quality AG, distributed members get a sense of awareness and overhear conversations, which enable casual interaction. The study findings confirmed the value of having all information artifacts of group member's work and their activities visible for collaborative work. The distributed corkboard tiled display afforded all work visible at a glance, and this feature helped group members be aware of progress and problem that others faced during collaborative work sessions. Although the participants stated that they did not pay much attention to the distributed corkboard tiled display especially with the close-up displays, we believe this display was used as a peripheral group display. When this visibility was taken out such as in the presentation-style display, it was fairly obvious to see the awareness problem where members explicitly showed or asked to show information from their private close-up display to the public group display.

While the distributed corkboard helped group awareness by being able to casually glance over at other's work, we need to address more awareness issues observed in our studies, such as shared resource awareness and more importantly task awareness among group members. Distributed participants needed more shared resource awareness as the number of their interactions on these resources increased. Thus, we need to provide awareness tools that indicate who currently owns which resources so as not to have resource conflicts or have members explicitly ask for or notify the use of shared resources. We also need to support tools that help group awareness of the overall state of the task progress such as to-do list or action items. This is particularly

necessary for the group whose members work simultaneously in parallel but share work process and results during the course of collaborative work.

Moreover, we observed that the participants often gazed at the video image of remote collaborators during discussion and did lots of hand gestures over the displays to point at different interests. User's hand gesture, eye gaze, and physical proximity to shared resource were powerful indicators of attention, but our participants suffered from a lack of this information. We believe the system showing simple attentive cues would encourage more natural interactions. Further investigations will be made to evaluate and improve the features discussed in this paper, and we will continue to work on improving the Continuum technologies to support real world distance collaboration among distributed teams.

References

1. Childers, L., Disz, T., Olson, R., Papka, M. and Stevens, R.: Access Grids: Immersive Group-to-Group Collaborative Visualization. In Proceedings of IPT Workshop (2000)
2. Chowdhry, V.: SpaceGlider: An intuitive approach for controlling multiple remote computers, M.S. Thesis, University of Illinois at Chicago (2003)
3. Dourish, P. and Bellotti, V.: Awareness and Coordination in Shared Workspaces. In Proceedings of CSCW'92, 107-114 (1992)
4. Greenberg, S. Rounding, M.: The Notification Collage. In Proceedings of CHI 2001, 514-521 (2001)
5. Gutwin, C. and Greenberg, S.: A Descriptive Framework of Workspace Awareness for Real-Time Groupware. Journal of Computer Supported Cooperative Work, 11(3-4):441-446, (2002)
6. Huang, E. M., Mynatt, E. D.: Semi-public displays for small, co-located groups, In Proceedings of CHI 2003, 49-56, Ft. Lauderdale, FL (2003)
7. Leigh, J., Johnson, A., Park, K., Nayak, A., Singh, R. and Chowdry, V.: Amplified Collaboration Environments. In Proceedings of VR Grid Workshop, Daejun, Korea, 77-85, (2002)
8. Olson, G. M. and Olson, J. S.: Distance Matters. Human-computer Interaction 15(2 and 3):139-179 (2000)
9. Park, K., A. Kapoor, and J. Leigh. Lessons Learned from Employing Multiple Perspectives In a Collaborative Virtual Environment for Visualizing Scientific Data. In Proceedings of Collaborative Virtual Environments 2000, 73-82, San Francisco, CA (2000)
10. Park, K., Renambot, L., Leigh, J. and Johnson, A.: The Impact of Display-rich Environments for Enhancing Task Parallelism and Group Awareness in Advanced Collaborative Environments, In Workshop on Advanced Collaboration Environments 2003, Seattle, WA (2003)
11. Renambot, L. and Schaaf, T.V.D.: Enabling Tiled Displays for Education. In Workshop on Commodity-Based Visualization Clusters, in conjunction with IEEE Visualization (2002)
12. Steinfield, C., Jang, C., and Pfaff, B.: Supporting Virtual Team Collaboration: The TeamSCOPE System, In Proceedings of GROUP 1999. Phoenix, AZ, 81-90 (1999)
13. Teasley, S., Covi, L., Krishnan, M. S., and Olson, J.: How does radical collocation help a team succeed? In Proceedings of CSCW 2000. Philadelphia, PA (2000)
14. Ward, M. O.: XmdvTool: http://davis.wpi.edu/xmdv/

Combining Communication and Coordination Toward Articulation of Collaborative Activities

Alberto Barbosa Raposo[1], Marco Aurélio Gerosa[2], and Hugo Fuks[2]

[1] Computer Graphics Group (Tecgraf)
Computer Science Department – Catholic University of Rio de Janeiro (PUC-Rio)
R. Marquês de São Vicente, 225, Gávea, Rio de Janeiro – RJ, 22453-900, Brazil
abraposo@tecgraf.puc-rio.br
[2] Software Engineering Laboratory (LES)
Computer Science Department – Catholic University of Rio de Janeiro (PUC-Rio)
R. Marquês de São Vicente, 225, Gávea, Rio de Janeiro – RJ, 22453-900, Brazil
{gerosa,hugo}@inf.puc-rio.br

Abstract. In this paper, we present a proposal for the articulation of collaborative activities based on communication and coordination representation models. Articulation is essential in any kind of collaboration and involves pre-articulation of the tasks, their management, and post-articulation. For the representation of pre- and post-articulation phases, conversation clichés (communication) are used. For the coordination phase, a model separating the tasks and their interdependencies is used. The articulation schema, which is especially suited to e-business applications, is then applied to a business-web example.

1 Introduction

Planning is an essential activity to develop collaboratively any major task, since it ensures that the desired goal will result from individual tasks. In Computer Supported Cooperative Work (CSCW), the notion of planning is realized by articulation, a set of activities required to manage the distributed nature of collaborative work [18].

The articulation work involves the pre-articulation of the tasks, their management and post-articulation. Pre-articulation involves the actions that are necessary to prepare the collaboration, normally concluded before the collaborative work begins, like the identification of the objectives, the mapping out of these objectives into tasks, the selection of the participants, the distribution of tasks among them, etc. The post-articulation phase occurs after the end of the tasks and involves the evaluation and the analysis of the tasks that were carried out and the documentation of the collaborative process.

The management of the carrying out of the tasks, or coordination, is the part of the articulation work with a distinct, dynamic nature, needing to be renegotiated in an almost continuous fashion throughout the collaboration period. It can be defined as the act of managing interdependencies between tasks that are carried out to achieve an objective [13].

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 121–136, 2004.

Articulation is essential to any kind of collaboration. In spite of that, coordination does not need to appear explicitly in some kinds of computer-supported collaborative activities–called loosely integrated collaborative activities [15]–such as those realized by means of chats or audio/videoconferences. These activities are deeply associated with social relations and generally are satisfactorily coordinated by the standing social protocol, which is characterized by the absence of any explicit coordination mechanism among the activities, trusting users' abilities to mediate interactions (the coordination is culturally established and strongly dependent on mutual awareness). By coordination mechanism, we mean "a specialized software device, which interacts with a specific software application so as to support articulation work" [19].

On the other hand, there is a large group of activities (tightly integrated collaborative activities) that require sophisticated coordination mechanisms in order to be efficiently supported by computer systems. In this kind of activity, tasks depend on one another to start, to be performed, and/or to end. Workflow management systems are examples of computational support suited for this kind of activity.

Business-webs (b-webs), which are partner networks of producers, suppliers, service providers, infrastructure companies, and customers linked via digital channels [23], is an example of process realized by tightly integrated collaborative activities, demanding computational support for coordination in the workplace. In this context, this paper presents an articulation schema based on conversation clichés and a coordination model separating the tasks and their interdependencies.

It is important to mention that this work follows a collaboration model based on Communication, Coordination and Cooperation (the 3C Model [6], [10]). According to this model, in order to collaborate, people communicate. During this communication, commitments may be generated. These commitments imply tasks that will be necessary to have the job done. These tasks are managed by coordination that organizes the group and guarantees that the tasks are accomplished in the correct order, at the correct time and according to the imposed restrictions. During cooperation, which is the joint operation in a shared space, the members of the group accomplish tasks through the interaction of the individuals and the artifacts of the work environment. However, while working, the necessities of renegotiating and of taking decisions appear. Then, a new round of communication is demanded, which in turn may modify and generate more commitments that will need coordination to reorganize the tasks that are performed during cooperation.

In the following section, the representation models are explained. Section 3 shows how those models may be applied to b-webs. In Section 4, the articulation schema combining both representation models is discussed in the light of related work.

2 Representation Models

This section presents two representation models embracing the whole spectrum of articulation. For the pre- and post-articulation phases, we propose the use of conversation clichés for negotiating, structuring and evaluating the collaborative work [12]. For the coordination phase, we propose a model where the commitments generated

during the conversation (pre-articulation) define the collaborative activity, and coordination mechanisms are created to manage the interdependencies between the tasks carried out by the members of the group [15].

2.1 Conversation Clichés

Commitments are the results of a conversation for action [25], representing an informal contract with other people involved in that conversation. They may indicate a new responsibility, an obligation, a restriction or a decision, guiding the people's actions [8].

There are other notions of commitment in the computing literature [2], [4], [11], [27]. Our notion of commitment could be seen as a proper subset of Bond's [2] concept of commitment. Although we agree on the behavioral aspects of commitment, we have different views about retracting commitments. Moreover, we provide a calculus for dealing with commitments [9].

As the commitments originate the group tasks, which will be managed by the coordination, the representation of commitments has an important role in the collaboration as a whole. However, since these commitments come from the natural language communication, it is difficult to capture them. In spite of that, some kinds of negotiation, such as those involving a well-known domain (such as an e-business setting), tend to be structured and repetitive. Therefore, it is possible to identify previously the parts of the discourse structure that are repetitive–the clichés–and their effects on the commitments stores of each group member.

Clichés may be seen as state transition machines that control the sequencing of dialog events between two participants in a conversation. A cliché restrains the unfolding of a conversation. Hence, it is possible to use specific clichés for reaching specific results in conversations with a well-known goal.

The structure for dialog representation called ACCORD [12] is used to represent the clichés in the context of this work. In this representation, a dialog is a sequence of events. Each participant has a commitment store containing the commitments undertaken by her during the conversation. The individual stores are accessible to other participants and are continuously updated while the conversation occurs.

According to the dialog primitives of ACCORD, a dialog event is a tuple consisting of a locution, and the speaker and the hearer of that locution. A locution is a locution modifier applied to one or more sentences. In the following subsections, the constructors of ACCORD conversation schema are presented.

Locution Modifiers and Dialog Events

Locutions consist of a statement and a modifier, represented as *locution modifier(statement).* Statements are constructed in a propositional language, which includes negation, conditional, disjunction and conjunction of statements. Locution modifiers are as follows.

Assertions, to be read as "it is the case that s*tatement",* notationally ***asserts***(s*tatement*).

Questions, to be read as "is it the case that *statement*?", notationally ***questions*** (*statement*).

Withdrawals, to be read as "I am not sure that *statement*", or "no commitment to *statement*", notationally ***withdraws***(*statement*).

Each locution modifier affects the commitment stores in a proper way, as will be seen later. There are four other locution modifiers that are not used in this paper [9].

Dialog events are represented by a notation in the form *<P1toP2, Locution>*, where *P1* and *P2* are participants. A dialog is formed by a sequence of dialog events, which is stored in an event register.

Commitments

Commitments are placed in commitment stores, which are represented in the following way: *Participant*(*C*(*<Set of sentences>*), *D*(*<Set of sentences>*)). For the sake of clarity, each participant will not only have a commitment store for its positive and negative commitments–C part of the commitment store–but also a D part of the commitment store as a place for the statements which the participant is not committed to.

For example, $(P_1(C(s1,s2)) \wedge P_2(C(s1), D(s3))$ indicates that sentence *s1* is in part *C* of the commitment stores of participants P_1 and P_2, that sentence *s2* is in part *C* of P_1's commitment store and that sentence *s3* is in part *D* of P_2's commitment store. It is possible to read from that, that P_1 is committed to both *s1* and *s2,* while P_2 is committed to *s1* and is not committed to *s3.*

Commitments may be explicit or implicit. Explicit commitments are those generated in the dialog process, while implicit commitments are those inferred by the commitment calculus. Commitment stores are graphically represented according to Fig. 1.

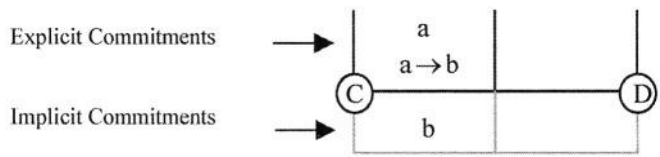

Fig. 1. Graphical representation of a commitment store (a and b are commitments)

Commitment Axioms

In this work, dialogs are represented in the following format:

$Pre \rightarrow [P_i toP_j, Locution]^n Post$

where *i* and $j \in \{1, 2\}, i \neq j; n \geq 1$, and *Pre* and *Post* are the conditions before and after the dialog. *Pre* and *Post* are commitments, and when they are instantiated as *healthy,* the current state of the dialog is valid.

Commitment axioms define the changes in the commitment stores caused by each locution modifier. Below commitment axioms are presented for some locution modifiers. P_1 and P_2 are participants and *s* is a *<statement>*.

$healthy \rightarrow [P_1 toP_2, \textbf{asserts}(s)]\ P_1(C(s)) \wedge P_2(C(s))$

Given that *Pre* is a healthy state, after P_1 uttered asserts(s) to P_2, both participants are committed to s (this statement is included in part *C* of the commitment stores of both participants).

$P_1(C(s)) \rightarrow [P_1 toP_2, \textbf{withdraws}(s)]\ P_1(D(s))$

Given that P_1 is committed to *s* prior to the assertion, she withdraws her commitment with such statement. This causes the removal of s from the *C* part of her commitment store and its inclusion in the *D* part.

$healthy \rightarrow [P_1 toP_2, \textbf{questions}(s)]\ healthy$

Given that *Pre* is a healthy, after P_1 uttered *questions(s)* to P_2, *Post* is a healthy state. The uttering of a question does not affect the commitment stores.

In this work, clichés will be used as a means to realize the pre- and post-articulation phases. The coordination phase of articulation will use the model presented in the following.

2.2 Task/Interdependency Coordination Model

In order to realize the coordination process it is necessary to have a clear definition of tasks, collaborative activities and interdependencies. In the model adopted in this paper, a collaborative activity is a set of tasks carried out by several members of the group in order to achieve a common objective (commitments) [15]. Tasks are the building blocks of the collaborative activities and are connected by interdependencies. Tasks can be atomic or composed of sub-tasks. A group of sub-tasks can be considered a task on a higher abstraction level when it does not present interdependencies with tasks that are external to this group. This ensures the modeling of collaborative activities on several levels of abstraction.

Interdependency is a key concept in the coordination theory–if there are no dependencies between tasks to be performed in a collaborative effort, there is nothing to coordinate. (Note that this does not mean that there is no need for the pre- or post-articulation.) The approach task/interdependency is a step toward giving flexibility to coordination mechanisms, which is crucial to further use of this kind of mechanism. This approach is based on a clear separation between "the work devoted to activity coordination and coordinated work, i.e., the work devoted to their articulated execution in the target domain" [21].

One of the advantages of the separation task/interdependency is that interdependencies and their coordination mechanisms may be reused. It is possible to characterize different kinds of interdependencies and identify the coordination mechanisms to manage them, creating a set of interdependencies and respective coordination mechanisms capable of encompassing a wide range of collaborative applications [14]. An example of a set of coordination mechanisms that uses Petri Nets to model tasks and the treatment of interdependencies is found in [16].

The coordination can take place on two levels—the activities level (temporal) and the object level [7]. On the temporal level, the coordination defines the sequencing of

the tasks that make up an activity. On the object level, the coordination describes how to handle the sequential or simultaneous access of multiple participants through a same set of cooperation objects.

Temporal Interdependencies

Temporal interdependencies establish the relative order of execution between a pair of tasks. The set of temporal interdependencies of the proposed model is historically based on temporal relations defined by J. F. Allen [1]. He proved that there is a set of seven primitive and mutually exclusive relations that could be applied over time intervals. The adaptation of Allen's primitives to the context of collaborative activities takes into account that any task *T* will take some time (from *i* to *f*) to be performed.

A time interval is characterized by two events, which in turn are associated to time instants. The first event is the starting (initial) time of an interval *A,* denoted here *ia.* The other event is the ending (final) time of the same interval, denoted *fa,* always with *ia*<*fa.* Allen's relations between time intervals, *A* and *B,* are: **equals** (*ia*=*ib* and *fa*=*fb*), **starts** (*ia*=*ib* and *fa*<*fb*), **finishes** (*ia*>*ib* and *fa*=*fb*), **before** (*fa*<*ib*), **meets** (*fa*=*ib*), **overlaps** (*ia*<*ib, ib*<*fa* and *fa*<*fb*), **during** (*ia*>*ib and fa*<*fb*).

However, the merely descriptive characteristic of Allen's temporal relations allows for different interpretations of a single interdependency. For example, suppose that tasks *A* and *B* are associated by the **equals** relation. In this situation, what should the coordination mechanism do when task *A* is ready to begin, but not task *B*? Should it block the execution of task *A* until task *B* is ready (passive interpretation), or should it force the start of task *B* to guarantee that the interdependency will be respected (active interpretation)? In a different situation, if it is said that task *A* occurs before task *B,* what should be done when task *B* is ready but not task *A*? Should the coordination mechanism block task *B* until the end of task *A,* or should it allow the execution of task *B,* blocking future executions of task *A* (which would violate the relation)? For all of these reasons, it was necessary to make some adaptations to Allen's basic relations.

The first adaptation deals with active and passive interpretations by means of two operators: **enables** and **forces.** The **enables** operator represents the passive interpretation, while **forces** represents the active one. These operations may be applied on the initial and final instants of each interdependent task. Additionally, these extreme points have two states, *ready* and *concluded,* indicating, respectively, that the task is ready to start (or finish) and that it has already started (or finished). These states are used in the first operand, indicating that it will enable or force the second operand before *(ready)* or after *(concluded)* its own execution.

Consider, for example, two tasks *Ta* and *Tb,* with initial and final points *ia, ib, fa* and *fb*. The interdependency *Ta* **starts** *Tb,* which establishes that both tasks start simultaneously, may be extended into different interpretations:

ia (ready) **enables** *ib* AND *ib (ready)* **enables** *ia* – this statement indicates the passive situation, in which the tasks will start their execution only when both are ready (i.e., *Tb* will be enabled to start only when *Ta* is ready to start, and vice-versa), but neither will force the execution of the other.

ia (ready) **forces** *ib* – in this situation, when *Ta* is ready to begin, *Tb* is forced to start, indicating a master/slave active interdependency (similarly, *Tb* could be considered the master if *ib (ready)* **forces** *ia*).

In spite of the operators **enables** and **forces,** there are undefined situations remaining. Such a situation occurs, for example, in *Ta* **before** *Tb.* After *Ta* and *Tb* have been finished, how should the coordination mechanism proceed if *Tb* wants to start again? Should it allow its execution, since *Ta* has already been executed (one to many), or should it make *Tb* wait until *Ta* is executed again (one to one)?

In order to deal with such situations, it was necessary to include an optional parameter for the **enables** operator. This parameter indicates the number of times a condition (first operand) enables the event (second operand).

For example, to define that *Ta* **before** *Tb* allows the maximum of three executions of *Tb* for each execution of *Ta,* the following statements are used *fa (concluded)* **enables(3)** *ib,* indicating that, after each execution of *Ta, Tb* is allowed to execute up to three times. It is also possible to define that there is no restriction on the number of times a task may be executed after or during another.

In order to enhance the flexibility of the model, it is also necessary to create the **blocks** and **unblocks** operators that, respectively disable and re-enable the execution of an event (second operand) when the state of the first operand is reached. The use of these operators, for example, allows for a new interpretation of *Ta* **before** *Tb*:

ib (*concluded*) **blocks** *ia* – in this case, there is a restriction in the execution of *Ta,* which may not be executed anymore if *Tb* has already started its execution. There is no restriction on the execution of *Tb (Tb* does not have to wait for the execution of *Ta,* as would happen with the situation given by *fa (concluded)* **enables** *ib).*

Resource Interdependencies

Resource-related interdependencies may be represented by combinations of temporal relations. For example, if two tasks, *Ta* and *Tb,* may not use the same resource simultaneously, it is possible to define a "not parallel" dependency as the following statement, *ia (ready)* **blocks** *ib* AND *fa (concluded)* **unblocks** *ib* AND *ib (ready)* **blocks** *ia* AND *fb (concluded)* **unblocks** *ia.* However, besides being prone to deadlocks, this possibility ignores the notion of resource, which is quite important in the context of workflows and collaborative activities. Therefore, it is not sufficient to treat the problem of task interdependencies as a temporal logic problem. Considering resource management dependencies independently of temporal ones, a more flexible model is created, allowing the designer to deal with each kind of dependency separately.

Resource management interdependencies in the proposed model are complementary to temporal ones and may be used in parallel to them. This kind of interdependency deals with the distribution of resources among the tasks. Three basic resource management dependencies are defined.

Sharing – a limited number of resources must be shared among several tasks.

Simultaneity – a resource is available only if a certain number of tasks request it simultaneously. It represents, for instance, a machine that may only be used with more than one operator.

Volatility – indicates whether, after the use, the resource is available again. For example, a printer is a non-volatile resource, while a sheet of paper is volatile.

Each of the above interdependencies requires parameters indicating the number of resources to be shared, the number of tasks that must request a resource simultaneously and/or the number of times a resource may be used (volatility).

3 B-Web Example

B-webs are defined as "distinct systems of suppliers, distributors, commerce providers, infrastructure providers, and customers that use the Internet for their primary business communications and transactions" [23]. Different companies are increasingly reaggregating their values in these b-webs, which are becoming one of the driving forces of the digital economy.

A b-web is an adequate environment to apply the proposed articulation schema because it provides deeply interdependent tasks, which need computer-supported coordination. Moreover, the pre-articulation phase demands a normally bureaucratic conversation, which is suited to be represented by clichés.

The b-web typology comprises agoras, aggregations, alliances, value chains and distributive networks. Zooming into the value chain type of b-web, its main organization is called context provider, which "structure and direct the b-web network to produce a highly integrated value proposition" [23]. Other participants of this type of b-web may do everything else: manufacturing, delivering, on-site customer services, etc.

Value chains are further differentiated between routine production and shop production. While the former is product-centric and goods are designed for mass markets and production efficiency, the latter supports custom solutions where activities are not routine and are driven entirely by demand, that is, the end-customer is the one that triggers the value-creating process.

Cisco Systems is the quintessential example of shop production value chain. Using Cisco's Configuration Tool on the company's web page, the end-customer receives guidance to prepare its order while all kinds of discounts and other services are being offered. Only after selling the good does Cisco make it. However, in reality, Cisco will coordinate the production process instead of actually making it.

Cisco plays the coordinator role in this shop production value chain b-web. Using the available configuration tool customers communicate their orders, triggering the cooperation cycle among manufacturers, assemblers, distributors, component suppliers and the sales channels. The cooperation object itself is the computer (or solution) that will be shipped to the end-customer. Fig. 2 illustrates this process.

From the coordination point-of-view, it is possible to model the b-web workflow using the task/interdependency previously presented. An example of a simple transaction in a hypothetical value chain b-web is presented, involving four independent participants, a customer, the context provider, a producer, and a distributor. The workflow of this environment is represented in Fig. 3. In this figure, tasks are described in the ovals and their relations in the hexagons. Dotted arrows indicate inter-

dependencies and normal arrows, workflow transitions. The word *or* in the workflows indicates alternative paths (only one of them is followed). The absence of *or* indicates parallel paths. Letters *a* and *b* near the inter-related tasks are used to identify them in the directives inside the hexagons.

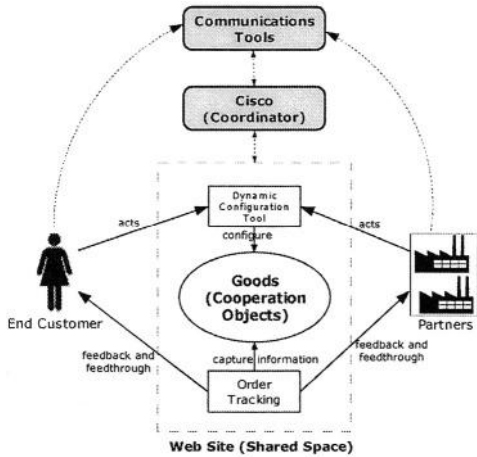

Fig. 2. Cisco Systems value chain b-web

In the workflow of Fig. 3, the customer must start the process by ordering the product in the context provider web site. This task starts the context provider's workflow (the end of the customer's task forces the beginning of the context provider's workflow). The customer then waits for the product or cancels the order. However, the order can only be cancelled if the product has not been delivered yet. This defines the relation between tasks deliver product (in the context provider workflow) and cancel order (in the customer workflow). The relation establishes that the beginning of the delivery blocks the order's cancellation.

After receiving an order, the context provider triggers two parallel activities: *confirm payment* and *order product.* This task starts the producer's workflow (*fa*(*concluded*) **forces** *ib*). After the payment confirmation and the ordering, the context provider must remain waiting until the product is ready. This determines the relation that when the producer concludes the product, task *organize delivery* in the context provider's workflow is enabled. We do not use the forces operator here because it is possible that the product becomes ready before the payment confirmation, and in this case, the delivery must wait, although enabled by the producer.

When the context provider is ready to organize the delivery, it starts the distributor's workflow, which has a close relationship to the rest of the context provider's one. This relationship is expressed by means of the *equals* relationship *(ia(ready)* **enables** *ib AND fa(ready)* **enables** *fb),* meaning that the tasks occur almost simultaneously in the distributor and in the context provider.

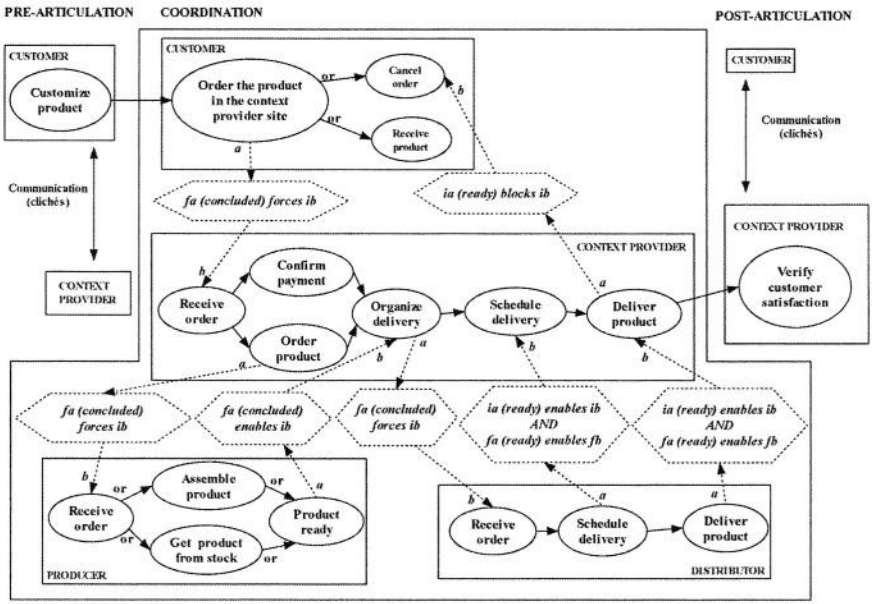

Fig. 3. Workflow of a typical transaction in a value chain b-web

The above example used only temporal interdependencies between tasks, but resource management dependencies could also be used in a straightforward way. For example, the context provider could have a stock, which defines an alternative route to that of contacting the producer. The task of getting the goods from the stock would have a volatility dependency, indicating that the stock would eventually finish.

The pre-articulation phase serves as an input to the workflow described above. The pre-articulation is part of the customer task *customize product*. During the performance of this task, the customer interacts with the company's web site and possibly to an operator. During this interaction, both the customer and the company assume commitments and, at end of the interaction, these commitments will define the product's configuration and delivery date. To illustrate this, consider the following situation.

The customer enters the web site and registers herself. To be registered, the customer accepts the terms and conditions of the company. After that, she enters the Dynamic Configuration Tool to order a product. She chooses product number AB123 and then, the Dynamic Configuration Tool offers configuration options. Then she chooses blue for the color and 220 for the voltage of the product. Then, the customer calls the site operator to negotiate the delivery date. She needs the equipment in three days. The operator says that it will be possible only for the green product. The customer accepts it, checks the price and orders the product.

The conversation cliché above is represented by the following sequence of dialog events and commitment stores.

I. Registration Dialog

healthy → [COMPANYtoCUSTOMER, **questions**(t)]
[CUSTOMERtoCOMPANY, **asserts**(t)]
CUSTOMER (C(t)) ∧ COMPANY (C(t))

II. Configuration Dialog

CUSTOMER (C(t)) ∧ COMPANY (C(t)) →
[CUSTOMERtoCOMPANY, **questions**(p)]
[COMPANYtoCUSTOMER, **asserts**(p)]
[CUSTOMERtoCOMPANY, **questions**(b)]
[COMPANYtoCUSTOMER, **asserts**(b)]
[CUSTOMERtoCOMPANY, **questions**(v)]
[COMPANYtoCUSTOMER, **asserts**(v)]
CUSTOMER (C(t,p,b,v)) ∧ COMPANY (C(t,p,b,v))

III. Delivery Date Negotiation Dialog

CUSTOMER (C(t,p,b,v)) ∧ COMPANY (C(t,p,b,v)) →
[CUSTOMERtoCOMPANY, **questions**(d)]
[COMPANYtoCUSTOMER, **withdraws**(b)]
[CUSTOMERtoCOMPANY, **withdraws**(b)]
[COMPANYtoCUSTOMER, **questions**(g)]
[CUSTOMERtoCOMPANY, **asserts**(g)]
[COMPANYtoCUSTOMER, **asserts**(d)]
CUSTOMER (C(t,p,v,g,d),D(b)) ∧ COMPANY(C(t,p,v,g,d),D(b))

IV. Ordering Dialog

CUSTOMER (C(t,p,v,g,d),D(b)) ∧ COMPANY (C(t,p,v,g,d),D(b)) →
[COMPANYtoCUSTOMER, **questions**(y)]
[CUSTOMERtoCOMPANY, **asserts**(y)]
CUSTOMER (C(t,p,v,g,d,y),D(b)) ∧ COMPANY (C(t,p,v,g,d,y),D(b))

Legend:
t – respect the company terms and conditions
p – manufacture the product AB123
b – manufacture the product in blue
v – manufacture the product for 220V
d – delivery the product in three days
g – manufacture the product in green
y – pay for the product AB123

At the end of the dialog, the customer is committed to respect the company terms and conditions and pay for the product. On the other hand, the company is committed to respect the terms and conditions, manufacture the product in 220V and in green color, and deliver it in three days for the stipulated price. These commitments serve as input to the workflow: pre-articulation. In a similar way, not shown here, it is possible to define a post-articulation cliché to verify customer satisfaction.

It is necessary to reinforce that this example deals with a single part of the whole b-web model. We did not stress all details for the modeled scenario (for instance, the consequences of the customer's cancellation, such as refund, devolution of the goods

to the producer, and so on). Its main goal is to show how the representation models may be used in a practical situation.

One of the advantages of using the coordination model to represent the workflow of b-webs is that, by means of a formal mathematical analysis (for example, using Petri Nets [16], [24]), it is possible to anticipate and test the behavior of the interorganizational environments before their implementation. From the primitives of the coordination model, it is also possible to define a direct mapping to construct the adequate software coordination mechanisms [15].

4 The Articulation Schema

The articulation schema presented is illustrated in Fig. 4. In the pre-articulation phase, conversation clichés are used to generate user commitments regarding the tasks definition, tasks allocation, and other aspects necessary in the negotiation of the collaborative activity. These commitments serve to model the activity workflow, in which the interdependencies among tasks are coordinated. Finally, the execution of the activity leads to a post-articulation phase, in which the users may document and inspect the process, assess the quality of tasks execution, and realize other evaluations that generate inputs to following activities.

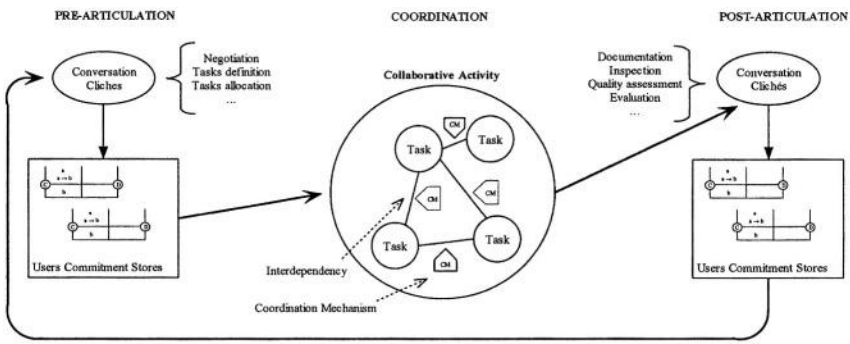

Fig. 4. The articulation schema

One of the important characteristics of this schema is the combination of features of different articulation approaches, namely those relying on communication–here represented by the conversation clichés– and those relying on explicit coordination mechanisms. The proposed articulation schema is discussed in the following sections in the light of some related works.

4.1 Articulation Dimensions

Carstensen and Nielsen, in a study comparing oral and artifact-based coordination, defined some dimensions that characterize articulation approaches [3]. In the light of such dimensions, we see that the presented articulation schema combines advantages of both coordination types:

Degree of Automation: It is possible to implement the articulation schema presented here [12], [15].

Persistency: As an approach with high degree of automation, all the interactions may be stored.

Dedicated and Non-exclusive Coordination Support: The pre- and post-articulation phases involve tasks that serve both the work and coordination, since it is more related to communication (non-exclusive coordination support). In the coordination phase, the work tasks are completely separated from coordination tasks (dedicated coordination support). Actually, this separation is the main characteristic of the task/interdependency model.

Direct and Indirect Referencing: The conversation clichés provide support for both direct referencing (i.e., telling someone what to do) and indirect referencing (i.e., providing information to assess what to do). This is also represented in the clichés by explicit and implicit commitments. The coordination model, on the other hand, provides only direct referencing, since it explicitly manages tasks interdependencies.

Dynamic and Static: The pre- and post-articulation phases, once based on communication, tend to be dynamic, in the sense they may reflect changes in the world around the collaboration. The coordination model is essentially static.

Coupling and Detachment: The pre- and post-articulation phases are detached from the field of work, in the sense that there is no straight connection between a state change in the coordination means and a state change in the work. The coordination mechanisms, on the other hand, are coupled to the field of work (tasks), in the sense that their state changes may affect the tasks (for example, when using the **forces** operator).

Reduction of Coordination Workload: The articulation schema augments the coordination workload during the pre- and post-articulation phases, since they use a structured conversation model. The task/interdependency model, on the other hand, reduces the coordination workload, because it provides the coordination mechanisms to regulate the tasks execution.

Flexibility: This coordination dimension needs further explanation. Therefore, it will be discussed separately, in the following section.

4.2 Articulation Flexibility

The necessity of coordination mechanisms to regulate interactions in collaborative systems has been the center of a heated discussion (e.g., [22], [26]). At one side, there are normative models that try to regulate the collaboration by restricting the interaction between participants and their tasks. A classical system that follows this approach is The Coordinator [25]. The criticisms on such normative approaches may be roughly summarized by the fact that their rigidly defined protocols applies only to very specific scenarios, limiting the flexibility of the collaborative systems. Eventually, there would be situations not predicted by the specified protocols, restraining the application of the defined mechanisms.

At the opposite site, are those advocating that collaborative systems should take flexibility to the extreme, leaving the articulation burden to the users, who become extremely dependent on mutual awareness. The critics on this kind of approach are that they augment the coordination workload, since users must deal with the complexity of articulating their tasks. Moreover, giving the coordination responsibilities for the users does not guarantee that activities are performed according to any prescription.

Actually, the discussion makes sense because there are different kinds of collaborative activities. As previously mentioned, tightly integrated collaborative activities need coordination mechanisms to regulate interactions. On the other hand, the coordination workload and the limited flexibility imposed by such approaches are completely awkward for loosely integrated collaborative activities, which are more suited for the awareness-based approach.

In spite of this discussion, there is a trend to conciliate both ideas, arguing that both kinds of activities are "seamlessly meshed and blended in the course of real world" [20]. Moreover, it is argued that facilities should be provided so that the users may interpret and exploit predefined coordination standards, deciding to use, change or reject them [17].

Deiters et al., in a study about flexibility in workflow management systems, defined some flexibility dimensions and solutions proposed to achieve them [5]. Among their flexibility dimensions there are two that relates to the presented articulation schema:

Process Flexibility: Related to situations when exceptions from the previously defined activities occur, and when there are parts of the activity that cannot be defined in advance. Regarding the presented articulation schema, the fact of having pre- and post-articulation phases based on communication attenuates the rigidity of the coordination phase, because exceptions and undefined situations may be renegotiated and reconsidered in a following coordination situation.

Flexible Task Allocation: Related to the reallocation of tasks at runtime. Deiters et al. [5] cited that one of the requirements for flexible task allocation is the negotiation of assignment rules and decisions. Moreover, the negotiation itself may be modeled as a workflow. The clichés-based pre-articulation phase presented here is an adequate means for such negotiation.

The articulation schema presented in this paper is definitely more related to tightly integrated collaborative activities, such as those that realize b-webs. However, the schema has some degree of flexibility, represented both by the separation between tasks and interdependencies in the coordination model and by the use of conversation during the pre- and post-articulation phases. Conversation, even in the form of clichés, is more flexible than coordination in the form of workflows.

5 Conclusion

This paper proposed an articulation schema based on two representation models, namely conversation clichés and task/interdependency model. Regarding the 3C

model, the main contribution of this paper is to provide a step toward establishing an implemental linking between the structured aspects of communication and coordination. E-business, exemplified by b-webs, captures the structured side of the communication and coordination phases, being a proper application field for the articulation schema.

Considering communication, there is still room for the unstructured conversations that occur in the workplace. The same reasoning is applicable to coordination given that workflow only captures the structured side of coordination.

Although not explicitly discussed, the articulation schema does not disregard the role of awareness in articulation. Actually, awareness occupies a central position in the 3C model [10]. Every event taking place during communication, coordination and cooperation generates awareness information, which by its turn mediates all the aspects of the collaboration process. Awareness has a direct effect on the pre- and post-articulation phases, which are more dynamic, in the sense of reflecting the world's changes.

Acknowledgements

The authors are financed by individual grants awarded by the Brazilian National Research Council (CNPq): Hugo Fuks nº 303055/02-2, Alberto Raposo nº 305015/02-8 and Marco Gerosa nº 140103/02-3. Thanks to Prof. Carlos Lucena, Head of LES / PUC-Rio, and Prof. Marcelo Gattass, Head of Tecgraf / PUC-Rio, which is a group mainly funded by Petrobras, Brazilian Oil & Gas Company.

References

1. Allen, J.F., Towards a General Theory of Action and Time. Artificial Intelligence, 23, ACM, (1984) 123-154
2. Bond, A.H., A Computational Model for Organizations of Cooperating Intelligent Agents. Proceedings of Conference on Office Information Systems, SIGOIS Bulletin, 11(2- 3), ACM, (1990) 21-30
3. Carstensen, P.H., and Nielsen, M., Characterizing Modes of Coordination: A comparison between oral and artifact based coordination. Proceedings of GROUP, ACM, (2001) 81-90
4. Cohen, R.P., and Levesque, J.H., Persistence, Intention and Commitment. In Cohen, R.P., and Perrault, C.R. (eds.), Formal Theories of Communication, 171-203, Report nº CSLI-87-69, (1987)
5. Deiters, W., Goesmann, T., and Löffeler, T., Flexibility in workflow management dimensions and solutions. International Journal of Computer Systems Science & Engineering 15(5), CRL Publishing, (2000) 303-313
6. Ellis, C.A., Gibbs, S.J., and Rein, G.L., Groupware - Some Issues and Experiences. Comm. of ACM, 34(1), ACM, (1991) 38-58
7. Ellis, C.A., and Wainer, J., A Conceptual Model of Groupware. Proceedings of CSCW (1994), ACM Press, 79-88
8. Fuks, H., Ryan M., and Sadler, M., Outline of a Commitment Logic for Legal Reasoning. Proceedings of 3rd International Conference on Logics, Informatics and Law, V2, (1989) 391-405

9. Fuks, H., Negotiation using Commitment and Dialogue. PhD. Thesis, Department of Computing, Imperial College, University of London, (1991)
10. Fuks, H., Gerosa, M.A., and Lucena, C.J.P., The Development and Application of Distance Learning on the Internet. Open Learning - The Journal of Open and Distance Learning, 17(1), Carfax Publishing, (2002) 23-38
11. Koo, C, and Wiederhold, G., A commitment-based communication model for distributed office environments. Proceedings of Office Information Systems, SIGOIS Bulletin, 9(2-3), ACM, (1988) 291-298
12. Laufer, C.C., and Fuks. H., ACCORD: Conversation Clichés for Cooperation. Proceedings of COOP'95 – First International Workshop on the Design of Cooperative Systems, INRIA Press, (1995), 351-369
13. Malone, T.W., and Crowston, K., What is coordination theory and how can it help design cooperative work systems? Proceedings of CSCW, ACM, (1990) 357-370
14. Malone, T.W., and Crowston, K., The Interdisciplinary Study of Coordination. ACM Computing Surveys, 26 (1), ACM, (1994) 87-119
15. Raposo, A.B., and Fuks, H., Defining Task Interdependencies and Coordination Mechanisms for Collaborative Systems. In Blay- Fornarino, M. et al. (eds.), Cooperative Systems Design. Frontiers in Artificial Intelligence and Applications, 74, IOS Press, (2002) 88-103
16. Raposo, A.B., Magalhães, L.P., and Ricarte, I.L.M., Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. International Journal of Computer Systems Science & Engineering 15(5), CRL Publishing, (2000) 315-326
17. Schmidt, K., Riding A Tiger, Or Computer Supported Cooperative Work. Proceedings of The 2nd European Conference on Computer-Supported Cooperative Work, Kluwer, (1991) 1-16
18. Schmidt, K., and Bannon, L.J., Taking CSCW seriously - Supporting articulation work. Computer Supported Cooperative Work, 1(2), Kluwer, (1992) 7-40
19. Schmidt, K., and Simone, C., Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. Computer Supported Cooperative Work, 5(2-3), Kluwer, (1996) 155-200
20. Schmidt, K., and Simone, C., Mind the gap! Towards a unified view of CSCW. Proceedings of COOP 2000 - 4th International Conference on the Design of Cooperative Systems, IOS Press, (2000) 205-221
21. Simone, C., Mark, G., and Giubbilei, D., Interoperability as a Means of Articulation Work. Proceedings of WACC: Int. Conf. on Work Activities Coordination and Collaboration, ACM, (1999) 39-48
22. Suchman, L.A., Do Categories Have Politics? Computer Supported Cooperative Work, 2(3), Kluwer, (1994) 177-190
23. Tapscoot, D., Ticoll, D., and Lowy, A., Digital Capital: Harnessing the Power of Business Webs. Harvard Business School Press, USA, (2000)
24. van der Aalst, W.M.P., The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1), World Scientific, (1998) 21-66
25. Winograd, T., and Flores, F., Understanding Computers And Cognition. Addison-Wesley, USA, (1987)
26. Winograd, T., Categories, Disciplines, and Social Coordination. Computer Supported Cooperative Work, 2(3), Kluwer, (1994) 191-197
27. Woo, C.C., Sact - A Tool for Automating Semi-Structured Organizational Communication. Proc. of Conference on Office Information Systems, ACM, (1990) 89-98

ThinkLets as Building Blocks for Collaboration Processes: A Further Conceptualization

Gwendolyn L. Kolfschoten[1], Robert O. Briggs[1,2], Jaco H. Appelman[1], and Gert-Jan de Vreede[1,3]

[1] Faculty of Technology, Policy and Management
Delft University of Technology
P.O. Box 5015 – 2600 GA Delft, The Netherlands
{g.l.kolfschoten,j.h.appelman}@tbm.tudelft.nl
[2] Center for the Management of Information, University of Arizona
rbbriggs@cmi.arizona.edu
[3] University of Nebraska at Omaha
gdevreede@mail.unomaha.edu

Abstract. In the past decade, there has been a steady increase in the importance of collaboration to value creation in organizations, which has given rise to a new research field. Collaboration Engineering aims to model, design, and deploy repeatable collaboration processes to be executed by practitioners themsleves of high-value recurring collaborative tasks. Thus the aim of collaboration engineering is to create ready made designs for group processes. A key concept in Collaboration Engineering is a thinkLet - a codified facilitation intervention in a group process to create a desired pattern of collaboration. This paper presents an analysis of the thinkLet concept and possible thinkLet classification schemes to support collaboration engineers in effectively designing collaboration processes.

1 Introduction

People frequently join forces to accomplish goals through collaboration that they could not achieve as individuals. By collaboration we mean joint intellectual effort toward a goal. While team efforts can be productive and successful, group work is fraught with challenges that can lead to unproductive processes and failed efforts [21]. Many teams therefore rely on professional facilitators to design and conduct high-value or high-risk tasks [13, 19].

The need for facilitation increases when teams seek to use Group Support Systems (GSS) technology. Under certain circumstances, GSS can lead to order-of-magnitude increases in team productivity (see [11,12] for a comprehensive overview of GSS research). However, the success of a GSS session is by no means assured, see e.g. [3,27]. As with many tools, GSS must be wielded with intelligence guided by experience in order for its potential to be realized. Novice users find the GSS tools easy to operate, but they typically do not know the variety of useful patterns of collaboration that one could create with a given tool, nor do they typically know which patterns of collaboration would be most useful for the task at hand. Most GSS users must therefore rely on professional facilitators in order to derive the benefits offered by GSS [6].

Skilled facilitators, however, tend to be expensive. They either have to be trained in-house, or hired as external consultants. Therefore many teams who could benefit

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 137–152, 2004.

from facilitation interventions and from GSS must often manage without them. One solution to this challenge would be to reduce the need for skilled facilitation expertise; to find a way that a team could wield the GSS and manage its collaboration process for itself, without the ongoing intervention of a professional facilitator but with predictable results. Addressing this challenge is the domain of the emerging field of Collaboration Engineering.

Collaboration Engineering researchers seek to codify and package key facilitation interventions in forms that can be re-used readily and successfully by team members themselves, without the ongoing intervention of a professional facilitator. Collaboration Engineering is *a design approach that models and deploys repeatable collaboration processes for recurring high-value collaborative tasks for execution by the practitioners themselves.* There are two new roles defined within collaboration engineering next to the facilitators' role.

- A *facilitator* creates a dynamic process that involves managing relationships between people, tasks, and technology, as well as structuring tasks and contributing to the effective accomplishment of the meeting's outcome [6].
- A *collaboration engineer* designs a collaboration process in a way that it is transferable to a practitioner. This means that the practitioner can execute the process without any further support from the collaboration engineer, not from a professional facilitator.
- A *practitioner* is a domain expert who can facilitate a single team process as a team leader in this particular domain. A practitioner does *not* design such a process. Rather (s)he executes a team process designed by a collaboration engineering.

A practitioner moderates the team process. A practitioner is *not* a professional facilitator; a practitioner executes a specific collaboration process on a recurring basis as a part of a standard way of working [6]. Table 1 describes the collaboration engineering roles, their tasks in terms of collaboration process design and execution, and their required expertise. Textbox 1 provides an example of a collaboration engineer designing and transferring a risk management process in a large financial services firm.

Table 1. Collaboration Engineering roles

Role	Process Design	Process Execution	Expertise
Collaboration Engineer	Repeatable, transferable processes	No execution, just process transfer	Both process and application domain
Facilitator	Ad-hoc, context specific processes	Execution and ad-hoc modification	Process
Practitioner	No design	Execution	Application domain

One of the current foci of Collaboration Engineering research is to identify and document reusable elementary building blocks for group process design. Toward that end, researchers have begun to codify a collection of such building blocks, called thinkLets, see e.g. [10, 14, 15, 16, 22]. A thinkLet is a named, packaged thinking activity that creates a predictable, repeatable pattern of collaboration among people working toward a goal [7]. ThinkLets can be used as conceptual building blocks in the design of collaboration processes [15]. A thinkLet is meant to be the smallest unit of intellectual capital required to be able to reproduce a pattern of collaboration among people working toward a goal.

A large international financial services organization was faced with the challenge to perform hundreds of operational risk management (ORM) workshops. They requested a repeatable collaborative ORM process to be developed that operational risk managers could execute themselves. Based on the experiences and the requirements from the ORM domain experts, collaboration engineers developed a first prototype of a repeatable collaborative ORM process. This process was evaluated in a pilot project within a particular business unit, leading to a number of modifications to the definition of the overall process in terms of collaborative activities, their interdependencies, and the facilitation techniques used. The resulting collaborative ORM process was shown to a group of 12 ORM experts. During a half day discussion, the wording and order of activities was modified and the proposed collaborative activities where tested with a number of chosen facilitation techniques. In the period that followed, about 200 ORM practitioners were trained to execute this process. To date, these ORM practitioners have moderated hundreds of workshops where business participants identify, assess, and mitigate operational risks.

Textbox 1: Collaboration Engineering example

A few examples of thinkLets are presented in Table 2 (see [26] for a more elaborate description of the mechanics of these thinkLets). Each thinkLet provides a concrete group-dynamics intervention, complete with instructions for implementation as part of some group process. Collaboration process designers who have a set of specific thinkLets available can therefore shift part of their attention from inventing and testing solutions to choosing known solutions. This may reduce both the effort and the risk of developing group processes.

Table 2. Examples of thinkLets [26]

Name	Purpose
LeafHopper	To have a group brainstorm ideas regarding a number of topics simultaneously.
Pin-the-Tail-on-the-Donkey	To have a group identify important concepts that warrant further deliberation.
RichRelations	To have a group uncover possible categories in which a number of existing concepts can be organized.
StrawPoll	To have a group evaluate a number of concepts with respect to a single criterion.
MoodRing	To continuously track the level of consensus within the group regarding a certain issue.

To date, Collaboration Engineering researchers have formally documented approximately 70 thinkLets. Field experiences suggest that these thinkLets fill in perhaps 80% of a given collaboration process design; the other 20% of actions a group must perform needs to be customized for the task at hand. In this sense, thinkLets have become a powerful pattern language for collaboration engineers, who use thinkLet names to describe and communicate sophisticated, complex process designs in a compact form [6].

While only a few thinkLets have been formally documented to date, it appears that the number of possible thinkLets may be infinite. The original conceptualization of thinkLets frames them as having three components: A tool, a configuration of that tool, and the facilitation script [7]. Each adaptation variation of any of these components on a known thinkLet can become a new thinkLet in its own right. Unfortunately, an unbridled explosion of thinkLets could lead to much redundancy and overlap among thinkLets, and could lead to thinkLet "dialects" where collaboration process designers in different communities use different names for the same concepts, and so are not able to communicate and transfer their knowledge across the boundaries of their local community of practice. It is therefore an important goal for the Collaboration Engineering research community to minimize the need for an explosion of thinkLets by identifying a stable core of conceptual thinkLets, and to assist facilitators and collaboration engineers to select among thinkLets and to design new ones when necessary. To this end, we present several approaches to classifying thinkLets and propose a new conceptualization of the thinkLet concept as a first step towards enabling collaboration engineers to:

- More easily identify the optimal thinkLets for a collaboration process design,
- More easily distinguish the relevant differences among similar thinkLets,
- More easily identify specific areas of collaborative endeavor for which useful thinkLets have not yet been devised, and
- Consolidate similar thinkLets into a uniform, non-redundant base set.

The remainder of this paper is structured as follows. In the next section we first present a more detailed discussion of the role of thinkLets in the design of collaboration processes. Then, we summarize current classification schemes for thinkLets, and discuss the merits and limitations of each. Based on this analysis, we present a reconceptualization of the thinkLet. We conclude this paper by discussing the implications and limitations of our research and proposing directions for future research.

2 ThinkLets and Collaboration Process Design

A fundamental assumption in the design of repeatable collaboration processes is that each process consists of a particular sequence of thinkLets that create various patterns of collaboration among the team members. To further define the role of the thinkLet in the design of collaboration processes, we developed a model that depicts the constituent elements of a collaboration process, their relationships, and the relevant details that need to be documented (figure 1). The name of each element is bold. Below the element's name, the documented aspects are listed. A line links related elements. Sub elements are attached below the master element.

The top level element in the model represents a **collaboration process,** a series of steps a team performs to accomplish a goal. The process embodies what a group must do, without conveying how each step will be accomplished. Each step in the design of a collaboration process consists of one or more **thinkLets** and **transitions.** As presented above, a thinkLet is a named, packaged, scripted collaboration activity that produces a predictable, repeatable pattern of collaboration among people working toward a goal. The initial conceptualization of thinkLet comprised three components: A **tool,** a **configuration,** and a **script** [7].

The tool concerns the specific technology used to create the pattern of collaboration – anything from yellow stickies and pencils to highly sophisticated collaboration technologies such as GSS.

The configuration defines how the tool is prepared (e.g. projected on a public screen), set up (e.g. configured so all contributions are anonymous), and loaded with data before the start of the thinkLet (e.g. a set of questions to which people must respond).

The script concerns everything a facilitator would have to do and say to a group to move them through the thinkLet [6]. Each differentiation in the components of a thinkLet influences the way in which people collaborate and is by definition a new thinkLet.

Knowledge of the three components of a thinkLet, it was argued, would be sufficient for a practitioner to recreate the pattern of collaboration [6,7]. Field trials with more than 50 novice trained practitioners bore out the proposition that non-facilitators who knew the tool, configuration, and script for a thinkLet could, in fact, predictably and repeatable engender the pattern of collaboration a given thinkLet was meant to produce.

If thinkLets are building blocks that support collaboration process, then transitions are the mortar that connects them. The transition defines all the changes, events and actions that must take place to move people from the end of one thinkLet to the beginning of the next. In the model we posit that a transition might include changes of data and changes of orientation. Changes of data might include transforming the content or presentation of existing data, removing data, or acquiring new data.

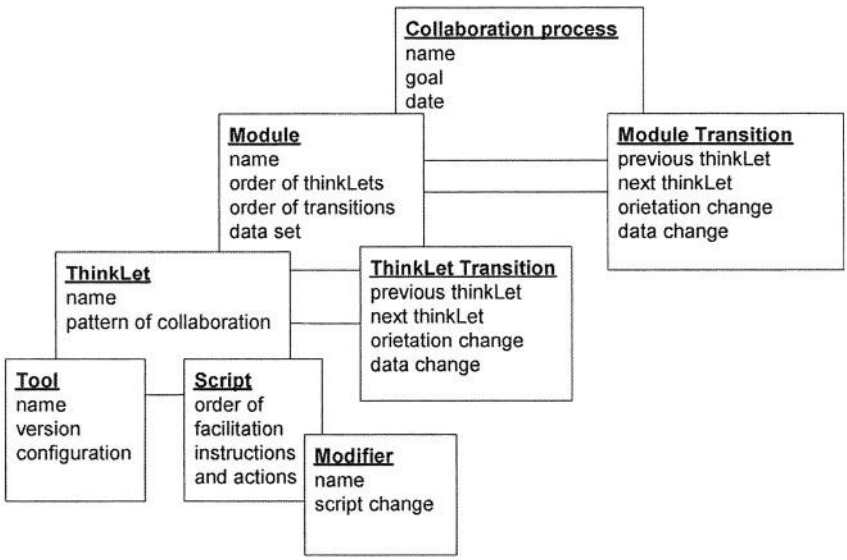

Fig. 1. Definition of the components of a thinkLet-supported collaboration process

Changes of orientation means changing peoples' understanding about what they were doing and what is expected of them. For example, a facilitator changes the participants' orientation by saying "You've now finished your brainstorming activity. In

order to reach our goal, next its time to pick out the best ideas and get them onto a list this will help us to focus our best ideas".

Sometimes a specific combination of several thinkLets and transitions are reused frequently in different collaboration processes. Such a sequence of thinkLets and transitions can be amalgamated into a named, reusable compound thinkLet called a **module.** The model includes a separate class for transitions between modules. A **module transition** is similar to a thinkLet transition. Finally, we noted that certain repeatable variations could be applied to a variety of thinkLets to create a predictable change in the dynamics the thinkLet could produce. We called these variations **modifiers,** and gave them names so they could be reused. For example, there are a variety of thinkLets for getting people to generate new, creative ideas. These thinkLets all allow people to contribute any idea that comes to mind. A OneUp modifier would change the ground rules for any ideation thinkLet such that people may only contribute new ideas that are arguably better than the existing ideas along some dimension. Most modifiers represent changes to the original thinkLet script. It is important to be aware of such changes as research shows that small changes to thinkLet scripts can create significant differences in group interactions, see e.g. [23].

This model helps one to understand the elements and attributes of a collaboration process design, and provides a topology of elements that must be modeled or documented at design time. However, as currently conceived, this model has a number of drawbacks. First, it ties a thinkLet closely to a specific technology in a specific configuration. This implies that it would not be possible to run the same thinkLet on a computer system and on paper. Yet, feedback from practitioners in the field clearly indicated that it was possible to do exactly that. Under the current conception of thinkLet, it would be necessary to record a new thinkLet for each change-of-technology, even if the change-of-technology did not change the patterns of collaboration among team members. Thus, the tool and configuration constructs in this model may be only an instance of something more fundamental. Second, the model also ties a thinkLet to a particular script. The purpose of the script is to convey certain concepts to the people executing a thinkLet. The current conception of thinkLet requires that a new thinkLet be recorded for any change of script. Thus, the script may only instantiate certain underlying concepts. Later in this paper we suggest a more fundamental framing of thinkLets that addresses these drawbacks.

3 ThinkLet Classification

As the number of thinkLets in the known pool grows, it will become increasingly difficult for a collaboration process designer to find, understand, and choose the most useful thinkLet for a given step in a given design. It would therefore be helpful for collaboration engineers that use thinkLets if there were a comprehensive classification of available thinkLets. The classification should support the collaboration engineer in choosing a specific thinkLet. In this section we present a number of possible classifications. Inspiration for these classifications was drawn from the model of a group process presented by Nunamaker et al. [21]. This model (figure 2) suggests at least two possible classification schemes for thinkLets: a process-based schema and an outcomes-based one. These became starting point for several thinkLet classification schemes.

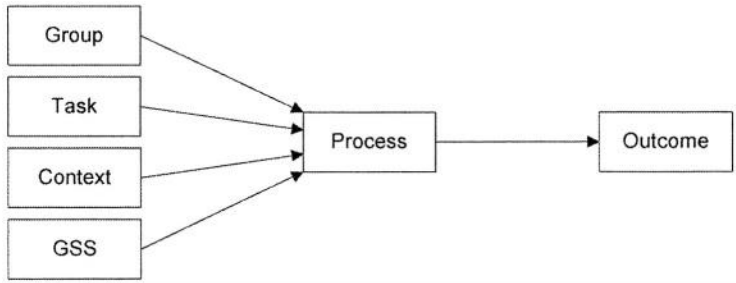

Fig. 2. A descriptive model of teamwork [21]

3.1 Classification of ThinkLets by Group Process Phase

The first classification of thinkLets focuses on the phase of the group process that the team is going through. This classification takes a general goal attainment process as a starting point, variations of which have been described by many researchers in many disciplines. Authors have variously identified the process as creativity [9], problem solving [1, 18], decision-making [24], and systems analysis and design [8], among others. Different authors divide or aggregate the phases differently, but the variations readily map to the summary of phases below:

- Understand the problem
- Develop Alternatives
- Evaluate Alternatives
- Choose an alternative
- Make a plan
- Take action
- Monitor the outcome

We reasoned that, because thinkLets are building blocks for group processes, and because group processes tend to fall into the phases above, it might be possible to classify thinkLets in terms of the phases in which they are most often used. ThinkLets that focus on the pros and cons of an option, for example, might fit comfortably in the Evaluate Alternative phase. Brainstorming thinkLets might fit comfortably in the Develop Alternatives phase, and so on.

On first trial, this scheme appears to be useful because every thinkLet fits into a phase, and every phase has thinkLets. Yet, this scheme comes short for several reasons. First, it turns out that many thinkLets fit comfortably into all phases. For example, depending on the task, a particular brainstorming thinkLet may be useful during all phases of the goal attainment process – to develop lists of symptoms in the Understand phase; to develop alternatives, to develop evaluation criteria, and so on. Second, experience in the field showed that the goal attainment process could, in fact, be fractal in nature: each of the phases could be a smaller iteration of the entire process. Understanding the problem, for example, might require that a team develop alternative explanations for what the problem might be, evaluate the merits of each alternative, and choose the most plausible explanation. Likewise, evaluating alternatives

could require developing alternative criteria, evaluating the merits of the possible criteria, and choosing which criteria to use.

Thus we conclude that this classification scheme is not a useful way to help decide which thinkLets to use. Yet this effort gave rise to a new understanding that produced a more useful scheme based on patterns of collaboration, rather than phases of a process.

3.2 Classification of ThinkLets by Collaboration Pattern

Briggs et al. [6] argue that, regardless of which stage of the goal-attainment process phase a group is in, they accomplish the phase by moving through some combination of patterns of collaboration. Briggs et al defined five such patterns in terms of moving a team from some state to some other state. The patterns they identify are:

- **Diverge**: Move from having fewer to having more concepts.
- **Converge**: Move from having many concepts to a focus on and understanding of a few deemed worthy of further attention.
- **Organize**: Move from less to more understanding of the relationships among concepts.
- **Evaluate**: Move from less to more understanding of the benefit of concepts toward attaining a goal relative to one or more criteria.
- **Build Consensus**: Move from less to more agreement among stakeholders on courses of action.

To test this classification scheme, we examined the archives of Delft University of Technology in the Netherlands (See [15] for more details). At Delft, researchers organized and facilitated well over a 1000 GSS meetings for industry, government, and educational groups over the past 11 years. The session transcripts of about 100 recent meetings were retrieved from the archives. For many sessions, also other preparation materials were retrieved, such as preparation notes, the meeting agenda, and introductory slide shows. We determined whether a) there were repeated thinkLets that recurred across many sessions; and b) the pattern-based classification scheme was useful to categorize all such thinkLets.

Our analysis revealed 16 identifiable distinct thinkLets (Table 3). All thinkLets could be classified as fitting into at least one of the patterns of collaboration. A few seemed to fit in more than one category. For instance Leafhopper, described in table 1, describes the activity of brainstorming in several categories, making the group diverge and organize at the same time.

Table 3. ThinkLets categorized by collaboration pattern [15]

Pattern	Different thinkLets	Number of occurrences
Diverge	5	356
Converge	3	49
Organize	3	27
Evaluate	4	146
Build consensus	1	1

In practice, this classification scheme was more useful than the phase-based scheme. Further, this categorization seem to fit well into a collaboration process design effort that proceeded from "What phase will the team be in?" to "Which patterns must a team follow and in what order to accomplish the phase?" to "Which thinkLet(s) shall we use to create the pattern(s)?"

While this scheme continues to be useful, we discovered a few challenges. First, the categories themselves are somewhat interdependent. Convergence, for example, always includes an element of evaluation. Consensus Building always includes an element of evaluation and divergence. This means that upon classifying a thinkLet careful attention must be paid to determine what the resulting group collaboration pattern is. Second, a given thinkLet may be used to create more than one pattern as in the example above. This means that some thinkLets may need to be categorized in more than one pattern. Both of these limitations suggest that there may be units of collaboration that are smaller than the thinkLet.

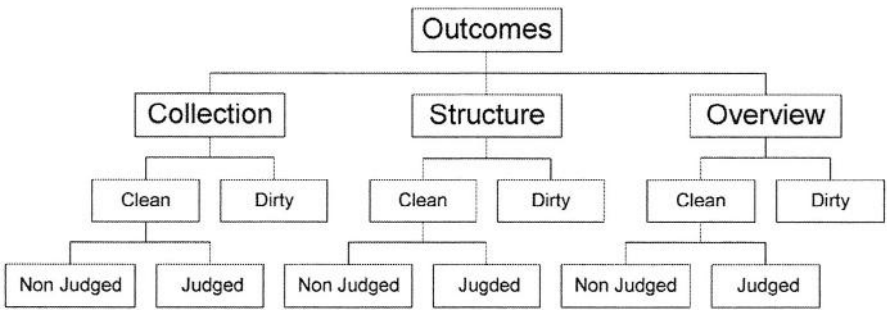

Fig. 3. Outcome-based classification scheme for thinkLets

3.3 Classification by Outcome

Finally, one can classify thinkLets based on the attributes of their outputs (see figure 3). We made this attempt because some thinkLets let participants produce an unstructured collection of concepts, while other thinkLets let them produce overviews of their given inputs, and still other thinkLets let them produce structures, such as lists, trees, and directed graphs. This classification can be extended with other output attributes – judged vs. non-judged concepts, referring to the use or neglect of specific consideration of each data element in a data set, and clean vs. dirty collections. A dirty collection is a set of concepts with redundancy, unresolved ambiguity, irrelevant content, and disagreements about meaning. A clean collection of concepts has no redundancy, no unresolved ambiguity, no irrelevant content, and no disagreements about meaning.

The benefit of a classification of thinkLets by outcome is that it also aligns well with a facilitator's mental model during the preparation of a collaboration effort, where the problem owner (or client) mostly refers to deliverables or results he wants to accomplish. Although it is fairly straightforward to create an output-based classification scheme, it turns out that this classification has some major limitations as well. For example, when we reclassified the thinkLets from the Delft analysis described above into this new scheme (see table 4), the distinction between the categories poses

a problem. The same thinkLets that can be used to clean a structure can also be used to clean a collection and vice versa. Also, the distinction between judged and non-judged results is relatively fuzzy.

Table 4. ThinkLets categorized by result [15]

Result	Distinct thinkLets	Number of occurrences
Collection	6	167
Structure	6	326
Overview	4	92

Thus, while the outputs from thinkLets may seem important, they are, on their own, still insufficient as a basis for identifying candidate thinkLets for a given process design. Notwithstanding these limitations, this classification does offer useful insights about constraints on the order in which thinkLets could be sequenced: Kolfschoten et al. [15] notes, for example, that a thinkLet, which produces a dirty collection of concepts as an output, cannot precede a thinkLet that requires a clean tree structure as an input.

3.4 A Best Classification?

Based on our analysis, we conclude that no classification is without its limitations. Overall, the classification based on patterns of collaboration appears to be the most useful in terms of supporting a collaboration engineer in selecting among available thinkLets. This classification led to a fairly unambiguous fit between each thinkLet and a pattern of collaboration – only a few thinkLets had to be classified under two different patterns simultaneously. The result is that the number of thinkLet options that a collaboration engineer has to consider is greatly reduced as long as (s)he has determined which pattern of collaboration has to be established.

Our definition and analysis of thinkLet classifications also revealed that there may be limitations regarding the current conceptualization of thinkLets in terms of tool, configuration, and script. Since it turns out that the same thinkLet can be classified for example under different patterns of collaboration, it may be that a more finely grained conceptualization of a thinkLet is required. This would also address the limitations regarding the current conceptualization as noted in section 2.

4 Towards a New Conceptualization of ThinkLets

The above descriptions and findings suggest that the thinkLet concept has not yet been decomposed into its most elementary forms. Although a given thinkLet can be implemented on many different technological platforms, the original formal definition of a thinkLet still includes a specific technology in a specific configuration [6]. Although a script describes the exact instructions that a facilitator or practitioner needs to moderate the group process, it appears that professional facilitators or experienced practitioners at times deviate from it. This became apparent in the archival research at Delft, which showed that different facilitators deviated in a variety of ways from the formal script of a thinkLet [15]. At the same time empirical research showed that

slight script deviations may produce dramatically different results [e.g. 22]. It appears that some aspects of a script result in more dramatic changes of individual thought and behavior than others. For a script to be reusable it should be rather generic, and be adaptable to a specific situation. Thus, there is a flaw in the definition of the components of a thinkLet.

Originally thinkLets were defined by the specification of the tool, configuration, and script that was used. This required that every variation of tool, script, and configuration constituted a new thinkLet. The documentation of every possible combination of changes is impractical because the definition leads to an exponential explosion of thinkLets. However, when we examined the current pool of thinkLets we noted certain regularities of behavior, thought, and technology that cut across thinkLets. When we analyze these regularities we might identify the elements that lead to patterns of collaboration. In this section we abandon the concept of thinkLets comprising of tools, configurations, and scripts. We focus instead on the supporting capabilities tools offer participants, the concomitant action people evidence, the rules by which we hope to constrain and enable their actions to allow for a viable and simple design approach for collaboration, and the situation dependent information described by parameters that have to be included to give appropriate facilitation instructions. Finally, we briefly address the different roles that actors in a group process might be required to perform.

Actions. A thinkLet prescribes how to create a pattern of collaboration in a group by effectuating specific individual actions on the part of the individual group members. The combined actions of the individuals, produce a particular pattern of collaboration at the group level, such as divergence or convergence. A preliminary analysis of the existing thinkLets yielded the following set of fundamental actions:

- **Add:** a participant may add concepts to the group's work space.
- **Delete:** a participant may remove one or more concepts from the group's work space.
- **Edit:** a participant may change the representations of existing concepts on the group's work space.
- **Relate:** a participant may establish the relationships between concepts.
- **Judge:** a participant may consider the value of concepts or their relationships with respect to goal attainment.

A thinkLet describes how to effectuate the individual participant's actions for each specified role in three manners:

1. It describes the technological **capabilities** that are needed for the participants to perform the required actions.
2. It describes the **rules** that are used to constrain particular actions.
3. It describes the **parameters** that are required to instantiate the effectuation of the participants' actions.

Each of these three components of a thinkLet is discussed in more detail below.

Capabilities. To execute a thinkLet, certain requirements and privileges regarding the group's physical work space must be defined. These requirements and privileges are to be afforded to the group by a particular technology it uses. In other words, the capabilities represent the means that can be offered to the participants in order to skill-

fully perform an activity [20]. To illustrate capabilities we draw on the metaphor of people contributing concepts to **pages** for which they have certain **privileges.** For example, to conduct an organize thinkLet, the group requires a page with viewing privileges for each category, and one or more viewable pages containing unclassified concepts and the ability to move concepts from any page to any other page. These capabilities could be realized by using computer-based GSS, or by using yellow stickies on a wall, or by using felt pens and a whiteboard. As long as the technology allows for the required capability, the thinkLet's intended pattern of collaboration can be reproduced. Thus, capabilities must be an elementary component of a thinkLet.

Rules. To execute a given thinkLet, the facilitator must emphasize rules that enable each individual participant to perform the desired actions. If each participant uses the capabilities provided and respects the rules the facilitator puts forward, (s)he should produce the desired actions. For example, the capabilities afforded by the FreeBrainstorm thinkLet require at least one page for each participant. Participants are enabled to view one page at a time, and to add concepts to any page. Their 'add' action is (a) guided by a rule stating that their contributions must relate to the brainstorming question at hand, and (b) constrained by a rule that requires them to swap pages after each contribution. A GSS can automatically enforce the page-swapping rule; a group working on paper must rely on social protocols and voluntary compliance.

The rules that guide actions can give rise to very different patterns of collaboration and therewith outcomes. For example, an 'add' action guided by a 'summarize' rule will give rise to abstraction, synthesis, and generalization, while an 'add' action guided by an 'elaborate' rule will give rise to increasingly detailed exposition of present concepts. Thus, rules must be an elementary component of thinkLets.

Parameters. The final elementary component of a thinkLet concerns parameters. For many thinkLets specific information has to be included so that it can be meaningfully executed. This information includes for example facilitation prompts or pre-defined brainstorming topics. These inputs are called **parameters.** They vary from situation to situation. For example, a brainstorming thinkLet requires a brainstorming question, and a polling thinkLet requires polling criteria. Some thinkLets have extra rules that are instantiated by parameters, for instance an added criterion or a specified aspect for detail. Other thinkLets have data parameters. For example, a polling thinkLet has ballot items as a parameter, and an organization thinkLet has categories and unsorted concepts as parameters. Thus, *parameters* must be an elementary component of thinkLets.

Roles. During a collaboration process, different actors may perform different roles. For example, someone may be the chauffeur, operating the GSS, while someone else may be a subject matter expert. For each role, the capabilities, rules and even parameters in the thinkLet can change. For instance, during a PopcornSort the participants should have the ability to move ideas in the prepared categories while in a ChauffeurSort only the chauffeur has this ability and participants are only asked to discuss the ideas. Thus, a thinkLet must specify to which *roles* the capabilities, rules, and parameters apply.

4.1 ThinkLets and Collaboration Process Design Revisited

The above re-conceptualization of the thinkLet leads us to a new model to describe the key elements in a collaboration process design (Figure 4). This model is based on the model presented earlier in Figure 1. The tool, configuration, and script elements have all been removed. These turned out to be mechanisms that allowed for the instantiation of the more elementary constructs: capability, rules, and parameters to effectuate specific actions for each role in the collaboration process. The thinkLet aims to create a specific pattern in the collaboration of the group. This approach makes the thinkLets more suitable for the design of collaboration processes.

The transition element is now expressed in terms of change-of-data and change-of-orientation. The transition does not influence the thinkLet and thus does not influence its components, they are separate building blocks. The modifier element is now labeled a named, standardized package of changes-of-the capabilities, rules, and parameters that change the actions evoked by a thinkLet. The modifiers can be applied to many thinkLets to change their dynamics in predictable ways. The collaboration process, module and module-transition elements remain unchanged.

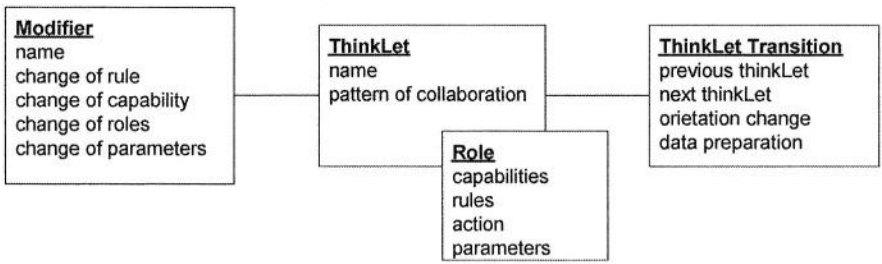

Fig. 4. New definition: elements of a thinkLet-supported collaboration process

The new conceptualization of thinkLets now closely resembles Bostrom et al.'s [5] description of the structure that a facilitator should provide to a collaboration process:

1. Meeting outcomes
2. Role specialization, participants are assigned specific roles (for example devils advocate, facilitator, decision maker and so on) for an activity phase or entire meeting.
3. Rules to follow during an activity phase or entire meeting
4. Procedures to accomplish an entire meeting, a specific meeting phase or a specific activity.
5. Techniques/technology to carry out procedures

5 Conclusions

A thinkLet is a named, packaged activity that produces a predictable, repeatable pattern of collaboration among people working toward a goal. The purpose of a thinkLet is to capture the smallest-possible unit of intellectual capital required to recreate a particular pattern of collaboration. This paper offers a re-conceptualization of the

thinkLet concept in terms of elementary participant actions, technological capabilities, rules, and parameters, rather than tool, configuration, and script. As a result, the new thinkLet conceptualization describes the requirements to create a certain pattern of collaboration independent from a technology and its configuration. This enables a collaboration engineer to choose appropriate thinkLets and subsequently select and adapt available technology and facilities to afford the required capabilities. This reframing has clarified some ambiguities and eliminated some apparent paradoxes surrounding the old framing of the thinkLets concept. Although it is impossible to predict everything about a dynamic collaboration process, the new thinkLet concept is expected to offer more support to the collaboration engineer. In particular, it has the following advantages:

1. The thinkLet concept is now technology independent. As technologies change, a concept that is independent of these changes will be more consistent. Some of the descriptions of thinkLets already indicated that several tools could be used to reach the same result. This ambiguity has been removed.
2. The cognitive load of the thinkLet concept is reduced. Since the ambiguity is resolved, it will be easier to transfer specific thinkLets to novice collaboration engineers and the instantiations of these thinkLets to practitioners.
3. Based on the new thinkLet concept, a very large, if not infinite set of thinkLet instantiations can be defined. If each variation of capabilities, rules, parameters, and roles is named as a separate thinkLet, the resulting set would rapidly become unmanageable and unusable for collaboration engineers. Yet, the new conceptualization may allow us to think in terms of thinkLet 'families' – a group of variations in terms of capabilities, rules, and parameters on a core thinkLet that all intend to create the same pattern of collaboration. This would be useful for designing collaboration processes as the choice for a thinkLet family which is subsequently customized using a modifier is likely to be easier than the direct choice for a specific thinkLet.

There are a number of limitations with respect to the research presented in this paper. Each of these limitations gives rise to exciting avenues for future research. *First,* none of the presented thinkLet classification schemes enabled an unambiguous classification of existing thinkLets. To this end, we advocate research into other classifications in disciplines that are both close and distant from collaboration engineering, such as small group research (e.g. McGrath's [17] task circumplex), GSS research (e.g. Bostrom et al. [4]'s electronic meeting tasks and Zigurs and Buckland's [29] task-technology fit model), workflow systems research (e.g. the Workflow Management Coalition's [28] specification of workflow processes), telematics (e.g. BETADE's [25] telematics services building blocks), and architecture (Alexander [2]'s pattern language). A taxonomic thinkLets classification is critical to facilitate the choice of a thinkLet. Given the new, more elementary, conceptualization of thinkLets, we expect the creation of a pure taxonomic classification to be easier. In addition, we anticipate it be more straightforward to define a set of thinkLet choice criteria. A sound classification will also help to identify redundancy in the current thinkLet set. Many of the current thinkLets might be identified as modifications of other thinkLets, resulting in a more limited yet more sharply defined final set of basic thinkLets that clearly differ from each other. If we can classify these we can also identify the area's where the current thinkLets do not provide solutions and new thinkLets or modifiers are required. This research is one step further on the path towards this goal. *Second,* the

current framing of the thinkLets still leaves open some questions. For example, Shepherd et al. [23] demonstrated that slight variations in facilitator instructions that had no impact on rules, but rather touched on motivation, produced significant differences in group-productivity. The proposed new framing of thinkLets does not address that effect. *Third,* new thinkLets classifications may be explored based on the new components of thinkLets. It appears that especially the rules and capabilities represent a promising starting point. For example, the rule 'generalize' could classify all thinkLets that use a given data set and produce the essence of that data set. *Finally,* although the thinkLet concept seems to have evolved into a more useful one, we need to confirm that indeed it is an improvement both in practice and in theory by testing its use in collaboration process design and execution.

Acknowledgments

The authors gratefully acknowledge Daniel Mittleman, Peter Jacobs, and Alexander Verbraeck for their contributions to the new conceptualization of thinkLets. Furthermore we thank Johanna Bragge, Mariëlle den Hengst-Bruggeling and the anonymous reviewers for their constructive feedback on an earlier version of this paper.

References

1. Ackoff, R.L.: The Art Of Problem Solving, John Wiley & Sons (1978)
2. Alexander, C.: The Timeless Way Of Building, New York: Oxford University Press (1979)
3. Appelman, J., Rouwette, E., Qureshi, S.: The Dynamics of Negotiation in a Global InterOrganizational Network: Findings from the Air Transport and Travel Industry, Group Decision and Negotiation, volume 11, issue 1, (2002) 145-163
4. Bostrom, R.P., Anson, R., The Face-To-Face Electronic Meeting: A Tutorial, In: Bostrom, R.P., R.T. Watson And S.T. Kinney (eds.): Computer Augmented Teamwork, A Guided Tour, Van Nostrand Reinhold, New York (1992)
5. Bostrom, R., Anson, R., Clawson, V.: Group Facilitation And Group Support Systems, Jessup And Valacich (eds.): Group Support Systems: New Perspectives, Macmillan, (1993)
6. Briggs, R.O., Vreede, GJ. 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
7. Briggs, R.O., Vreede, G.J. de, Nunamaker, J.F.: David T.H., Thinklets: Achieving Predictable, Repeatable Patterns Of Group Interaction With Group Support Systems Proceedings Of The 34th Hawaii International Conference On System Sciences (2001)
8. Checkland, P.B.: Systems Thinking, Systems Practice, John Wiley & Sons, Chichester (1981)
9. Couger, J. D.: Creative Problem Solving And Opportunity Finding. Danvers, Mass: Boyd And Fraser (1995)
10. Enserink, B.: Creating A Scenariologic – Design And Application Of A Repeatable Methodology, Proceedings Of The 36th Hawaiian Internal Conference On System Sciences, Los Alamitos, IEEE Computer Society Press (2003)
11. Fjermestad, J., Hiltz, S.R.: An Assessment Of Group Support Systems Experimental Research: Methodology And Results, Journal Of Management Information Systems, 15 (3) (1999) 7-149
12. Fjermestad, J., Hiltz, S.R.: A Descriptive Evaluation Of Group Support Systems Case And Field Studies, Journal Of Management Information Systems, 17 (3) (2001)

13. Griffith, T., Fuller M.: Northcraft G., Facilitator Influence In Group Support Systems, Information Systems Research, 9 (1) (1998) 20-36
14. Harder, R.J., Higley, H.: Application Of Thinklets To Team Cognitive Task Analysis, Proceedings Of The 37th Hawaiian Internal Conference On System Sciences, Los Alamitos: IEEE Computer Society Press (2004)
15. Kolfschoten, G.L., Appelman, J.H., Briggs, R.O., Vreede, G.J. de: Recurring Patterns Of Facilitation Interventions In GSS Sessions, Proceedings Of The 37th Hawaiian Internal Conference On System Sciences, Los Alamitos, IEEE Computer Society Press (2004)
16. Lowry, P.B., Albrecht, C.C., Nunamaker, J.F. Jr., Lee, J.D.: Evolutionary Development And Research On Internet-Based Collaborative Writing Tools And Processes To Enhance Ewriting In An Egovernment Setting, Decision Support Systems, 34 (2002) 229-252.
17. Mcgrath, J.E.: Groups: Interaction And Performance, Prentice Hall, Inc., Englewood Cliffs, NJ. (1984)
18. Mitroff, I.I., Betz, F., Pondly, L.R., Sagasty F.: On Managing Science In The Systems Age: Two Schemas For The Study Of Science As A Whole Systems Phenomenon, TIMS Interfaces, Vol. 4, No. 3 (1974) 46-58
19. Niederman, F., Beise, C.M., Beranek, P.M.: Issues And Concerns About Computer-Supported Meetings: The Facilitator's Perspective, MISQ 20 (1) (1996) 1-22
20. Nooteboom. B.: Learning And Innovation In Organizations And Economies, Oxford, Oxford University Press (2000)
21. Nunamaker, J.F., Dennis, A.R., Valacich, J.S., Vogel, D.R., George, J.F.: Electronic Meeting Systems To Support Group Work, Communications Of The ACM, 34(7) (1991) 40-61
22. Santanen, E.L., Vreede, G.J., de,: Creative Approaches To Measuring Creativity: Comparing The Effectiveness Of Four Divergence Thinklets, Proceedings Of The 37th Hawaiian Internal Conference On System Sciences, Los Alamitos, IEEE Computer Society Press (2004)
23. Shepherd, M.M., Briggs, R.O., Reinig, B.A., Yen, J., Nunamaker, J.F., Jr.: Social Comparison To Improve Electronic Brainstorming: Beyond Anonymity, Journal Of Management Information Systems, 12 (3) (1996) 155-170.
24. Simon, H.A. The Structure Of Ill Structured Problems, Artificial Intelligence, Vol. 4 (1973) 181-201
25. Verbraeck, A., Dahanayake, A. (eds.): Building Blocks For Effective Telematics Application Development And Evaluation, Delft University Of Technology, Delft (2002)
26. Vreede, G.J., de, Briggs, R.O.: Thinklets: Five Examples Of Creating Patterns Of Group Interaction, In: Ackermann, F., Vreede, G.J. de (eds): Proceedings Of Group Decision & Negotiation 2001, La Rochelle France June 4-7 (2001) 199-208.
27. Vreede, G.J. de, Davison, R., Briggs, R.O.: How A Silver Bullet May Lose Its Shine – Learning From Failures With Group Support Systems, Communications Of The ACM, 46 (8) (2003) 96-101
28. Workflow Management Coalition Workflow Process Definition Interface – XML Process Definition Language,
(http://Www.Wfmc.Org/Standards/Docs/TC-1025_10_Xpdl_102502. Pdf, Last Retrieved 14 April 2004) (2002)
29. Zigurs, I., Buckland, B.: A Theory Of Task/Technology Fit And Group Support Systems Effectiveness, MIS Quarterly, 22(3) (1998) 313-334

Bridging the Gap Between Decisions and Their Implementations

Marcos R.S. Borges[1], José A. Pino[2], and Renata M. Araujo[1,3]

[1] Post-Graduate Program in Informatics, NCE&IM, Universidade Federal do Rio de Janeiro
PO Box 2324, CEP 20001-970, Rio de Janeiro, Brazil
{mborges,renata}@nce.ufrj.br
[2] Department of Computer Science, Universidad de Chile
PO Box 2777, Santiago, Chile
jpino@dcc.uchile.cl
[3] School of Applied Informatics, Universidade Federal do Estado do Rio de Janeiro
Av. Pasteur, 458, Rio de Janeiro, Brazil
renata.araujo@uniriotec.br

Abstract. Decisions are frequently sent to implementers without much detail. It should not be a surprise, then, that results are not as expected. The lack of accompanying information and a common context produces wrongly implemented or lost decisions. This paper proposes a solution to this problem based on computer technology. In particular, a combination of tools including shared workspaces, process modeling with workflow and a discussion tool, is proposed. A case is used to illustrate the problem and its solution.

1 Introduction

There has been much emphasis on improving the decision making process but little attention has been paid to the implementation phase following a decision. The gap between the end of a decision making process and its implementation activities may, in fact, turn the decision inconsequent, due to lack of interaction and negotiation between decision makers and those who will implement the decision. Often, decisions that are implemented without the necessary clarification and negotiation may generate outcomes, which are different from those planned at the time of the decision.

This paper addresses the gap that exists between decision makers and implementers around the complete understanding of a decision context and the form of implementation. It discusses why linking decision implementation activities to the corresponding decision meeting is essential to make the meeting cycle fully successful. We claim supporting such link with a computer system is both efficient and effective.

Meetings rarely die. That is how Oppenheim [1] summarizes the cycle in which decision meetings occur: preparation of a meeting (pre-meeting), meeting itself and implementation (post-meeting). The post-meeting results and follow-up in turn may motivate the preparation of the next meeting, thus closing the cycle [2].

The complete meeting cycle can be computer supported. Pre-meetings can be advantageously supported to encourage careful pondering of arguments in favor or against decision options before the meeting [3]. Brainstorming, voting and other meeting activities can be supported both in distributed or face-to-face situations [4, 5]. Follow-up activities can be tracked with a workflow system [6].

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 153–165, 2004.

Although the need for relating decision meetings and the activities following them may seem obvious, cultural barriers and lack of appropriate tools induce just informal and very incomplete links. As a result, important decisions are not properly or timely implemented. It appears, then, that all efforts to make good decisions with Information Systems and/or Operations Research models and techniques are threatened by deficient implementation. Therefore, there is a missing link between decisions and their implementations, which needs to be well understood and supported.

We identify three problems in the connection between decision and its implementation. The first and possibly the most visible problem is when decision makers detect implemented results differ from what they expected. A second problem is the typical insufficient information attached to the decision. The third problem concerns the different contexts decision makers and implementers have. Each of these three problems is studied below.

The approach we chose for the proposed solution is the use of a combination of tools. These tools are: a shared workspace, a discussion supporting tool and a process-modeling tool. The later will be used to represent the decision as viewed by the implementation team. The implementation process will enable decision makers and implementers to discuss over a common basis. The discussion supporting tool will enable the clarification of issues which may arise during the implementation process. The shared workspace will enable both teams to bring their specific contexts to a common understanding.

The paper is composed of six sections. Section 2 presents a case, which will help us to understand the problem and to illustrate the proposed solution. Section 3 analyzes the requirements for the link between the decision meeting and its corresponding implementation for the case. Next we present the functionality of the proposed solution and how the solution can be implemented. Section 5 has a general discussion on the suitability of the approach. Section 6 concludes the paper.

2 The Problem

The Trucco Company needs to install a Call Center to support one of its marketing campaigns to be launched next month. The project requires 10-12 work desks and the Project Manager decides to request 14 workstations to support the operation. She justifies the two additional pieces of equipment as back-ups in case of failure. She sends the request to the company Board. The request includes the equipment specification, an estimate of the cost and a justification for the number of workstations. She also informs the Board the Call Center operation will start operation within 35 days from that date.

The Board discusses the request on its weekly meeting. It decides to approve the request and establishes the amount of US$14.000 (the estimated amount) as the project budget, after a brief discussion on the need for the two additional workstations and the cost estimate. The decision is passed to the Purchasing Department together with the original documentation prepared by the Project Manager.

The Purchasing Department sends a RFP to its traditional suppliers. In view of the project deadline, a Purchase Officer defines that the equipment must be delivered within 15 days. The RFP contains the specification, deliverable conditions and the cut-off date, i.e., when the company expects to receive the proposals. Two of the three

conditions have been established: the specification and the delivery date. The third condition - the budget - will be verified when the proposals are known.

After 3 days (at the cut-off date) the Purchasing Department received four proposals. Proposal A agrees to deliver the equipment as specified within 15 days, but presents a price exceeding the budget in 20%, i.e., US$16.800. Proposal B also agrees with the deadline but offers an alternative supply more powerful than specified, with a price exceeding the budget in 50% (US$21.000). The third proposal (C) presents the exact specification at a price within budget, but it requires 30 days to deliver the 14 workstations. Finally, Proposal D requires 20 days to deliver the equipment, it offers a very attractive price (US$11.200) but it does not comply with the specification (it offers a 20 MBytes hard disk, while the specification asks for 40 MBytes).

What happens next? There are several possible outcome scenarios. We describe three of them below. In scenario I, someone from the Purchasing Department believes the budget and the specification are the most important constraints and decides in favor of Proposal C. Projects are frequently late. The Purchasing Officer also reasons the Purchasing Dept. arbitrarily established the 15-days, anyway. After two weeks, the Call Center Project Manager asks the Purchasing Department about the order and finds out the project will be late. Knowing she will not be able to accomplish the project goals, she angrily complains to the Board. She also tells them the campaign will probably be a fiasco.

In scenario II, the Purchasing Dept. concludes no proposals match the project requirements and decides to check with other suppliers, strengthening the deadline and the specification constraints. After three days it receives proposals very similar to the first round. As a result, this scenario turns to one of the others, three days late. Another variation of this scenario is as follows: suppose one of the new suppliers has a proposal satisfying all requirements. The purchasing order is awarded to this supplier. It is not clear the equipment will arrive within 15 days (it is an unknown supplier). Let us assume the equipment arrives within that period. However, there still may be problems: the Technical Support Dept. may find the quickly estimated time for installation to be too tight (the 15 days period was an arbitrary decision by the Purchasing Officer).

Scenario III is complex. The Purchasing Dept. believes Proposal D – low price, but with insufficient features – is a good opportunity, but before making a decision, it puts together all proposals and asks the Technical Support Dept. and the Call Center Project Manager whether the specification can be relaxed. Technical Support is not aware of the purchase and is not able to respond, but it sees Proposal B as a good opportunity to deal with a request from another project for upgrading its equipment. The Technical Dept. envisages a plan in which Proposal B equipment could be acquired for this other project and its equipment, in turn, could be transferred to the Call Center Project. By coincidence, the specification matches the requirements. The Call Center Project Manager, on the other hand, prefers the simplest option, i.e., Proposal A. She thinks she can get a quick approval to increase 20% the project budget. A higher authority should then decide; this takes time (e.g. the Board meets once a week) and thus, some of the Proposals may be unfeasible by then.

In this example, extracted from a real case, we can observe the gap problem between a decision and its implementation. If people implementing the decision were the same who initially made the request, then probably no misunderstanding would have occurred. But of course, a small group of people cannot do everything within a company. Next section will examine this problem in further detail.

3 Analyzing the Problem

The problem we described in the previous section can be generalized to many situations occurring in Organizations. In these cases, there is a gap between the decision made by a group of persons and the implementation of that decision, probably carried out by other people. Lack of good communication is the main culprit for this gap.

The typical way of realizing something is wrong with the implementation of a decision appears when the results are different from those expected by decision makers. In our example, suppose the outcome scenario is No. I. There is not much to be done: the campaign will be late and probably it will be a failure.

Let us further explore this assumed outcome. The Head of the Purchasing Dept. will probably punish the person from his Dept. who chose option C. Is that right? Perhaps not: it was not totally his fault to be unaware about the importance of launching the campaign on time. Moreover, punishing him will probably hurt his own confidence at work and thus, his initiative to make decisions will be reduced. Perhaps no one was guilty; what everyone seems to ignore is that the environment is not facilitating people to make the right choices.

The symptom – results different than expected – is not all the sickness within the company. There may be lack of detailed information in what was required from the Purchasing Dept. If the request had specified the deadline was extremely important to be met, then obviously the outcome would have been different. Sometimes, then, detailed information from decision makers to implementers may help to fill the gap.

In some cases, however, even detailed information is not enough. Decision makers cannot imagine all the implementation choices there may occur and thus, they are unable to produce all possible information that may eventually become relevant. What is actually happening is people have different contexts and therefore, they do not work coherently.

What is context? Context may be defined as a complex description of shared knowledge about physical, social, historical, or other circumstances within which an action or event occurs [7]. For a step of a task, Brézillon and Pomerol [8] distinguish the part of the context being relevant for the current performer's focus of attention from the irrelevant part. The latter part is called external knowledge. The former part is called contextual knowledge because it has strong relation to the current focus although it is not directly considered in it. Always at a given focus, part of the contextual knowledge is proceduralized. This proceduralized context is a part of the contextual knowledge, which is invoked, organized, structured, and situated according to the focus and used while performing the task at this focus.

Context evolves with focus. This dynamics of context can be observed by the movement between the contextual knowledge and the proceduralized context. Thus, a part of the context is static, e.g. the context at a step of the focus of attention is defined by a fixed number of contextual elements and a fixed proceduralized context, but the overall focus of attention is associated with a dynamic context through this movement between the contextual knowledge and the proceduralized context. Static and dynamic parts of the context are intertwined and must be considered jointly.

Brézillon [9] points out it is possible to organize various types of context in a two-dimensional representation: in depth first, from the more general to the more specific, and in width first as a heterogeneous set of contexts at each level. In "depth first", contexts differ by their granularity. For example, a company context (with its tradi-

tion, habits, rules, etc.) is more general (at a higher level) than the context of an employee. In this case, context has strong relationships with the enterprise organization in terms of roles [10]. According to its depth, a context contains more general information than contexts at a lower level. However, context at one level is not a simple instantiation of the context at the upper level [11]. A context is like a system of rules (constraints) for identifying triggering events and for guiding behaviors in lower contexts. A context at one level contains contextual knowledge when the application of rules at the lower levels develops proceduralized contexts. A context (the contextual knowledge part) is like a frame of reference for the contexts below it. For instance, a person visiting Costa Rica knows the language spoken there is Spanish (contextual knowledge in the context of the country), and he pays attention to speak this language there, assuming he knows it (proceduralized context in his individual context).

In "width first", each actor has its own context. An actor's context contains information on the reasons for his actions, the results of his activities, etc. The context of the software agent possesses information on the available means for the accomplishment of the task, the access restriction to the databases, a user model, etc. For a given granularity of the context, there is thus a set of contexts rather heterogeneous, and the horizontal movement from one individual context to another one goes through either the upper context (e.g. the group context) or a lower context (e.g. the project context). Note that at the group level, a group is, recursively, like an actor with his individual context and interacting with other groups in other contexts.

Pomerol and Brézillon [12] discuss the transformation of contextual knowledge into some functional knowledge or causal and consequential reasoning in order to anticipate the result of actions. Data are facts, which have not been analyzed or summarized yet (e.g., see Watson [13]); information is data processed into a meaningful form, and knowledge is explained as the ability to integrate the information in his body of knowledge.

We can easily explain the behaviors of some of the Trucco Company people in the case we are analyzing considering the context. In Scenario I, the Purchasing Officer has a context including mainly previous purchases (impact, results, typical flexibility in delivery times), knowledge about the current suppliers (reliability), knowledge on the Technical Support Dept. (time to install computer-related equipment). With this context, he chose option C, thinking that would be the best for the project and the company. In Scenario III, note personal contexts influence choices. The Technical Support person looks for optimizing equipment for all his clients; his context includes the equipment features required by all projects he is currently supporting, previous experiences, etc.; it is not clear he is taking into consideration the urgency for the current project. The Call Center Project manager's context has knowledge about campaign effectiveness, competitors' actions, campaign messages, customers' requirements, etc.; for her, the equipment purchase should be trivial, an almost automatic activity, and differences in equipment costs are secondary. For any of the scenarios, it is not clear the Board context: how important is the project for the company? How does it relate to other company efforts? How relevant are these procurement budgets?

4 A Solution to the Problem

A solution to the described problem necessarily includes people to be aware of it. It is useless to provide technology if people do not believe there is such a problem: they

simply will not use the systems. A second requirement is a collaborative attitude from workers, in particular, some appreciation concerning what others do for the company success. We can then consider technology. Our proposal calls for a combination of supporting tools, as described below.

4.1 The Use of Workflow Technology

Workflow has been used to represent processes and to provide a control of its execution. By modeling the process, it provides both users and process players with a general view, yet abstract in some cases, of the activities involved to achieve job completion. It provides managers with the evolution of the work by controlling process execution, so they can take action in the presence of an unexpected behavior. WfMSs (Workflow Management Systems) have been used mainly in production processes. A production process is one repeatedly occurring with little variation from its expected flow. When the variation can be predicted, the process designer represents it in the form of optional paths generated by decisions during the process.

In our solution we make use of the process model and the workflow technology in order to represent the implementation steps, which materialize a decision. As discussed in [6], a proposal can be submitted together with the process description that supports its implementation. Decision makers can also generate the process model to represent how the job should be done. Alternatively, the implementation team can provide the process model under request from the decision makers. The decision implementation model is a way to achieve a common context and to support interaction, through a shared workspace among the people involved in the decision [14]. The different contexts in this situation are the one that comes from the decision makers and on the other coming from those who implement the decisions.

Figure 1 shows an example of an initial process model representing the steps and the people in charge to purchase a product in response to a request from the Project Manager. We can note that in this model, the Technical Dept. is not involved because the Board assumed it has been done by the Project Manager before requesting the purchase of the equipments. Also, there is almost no interaction among the three groups involved.

The advantages of this approach are two fold. First, the implementation plan is a starting point for the negotiation, in case one does not agree with the plan. The plan can be annotated and modified in response to the issues raised by people involved. Second, all divisions involved in the decision have a clear view of the entire implementation plan and their responsibility in it. Besides, decision makers can learn from mistakes in the past and include the necessary adjustment in future decisions. In some companies, the plan could also serve as the input for process enactment, which will provide implementation follow-up [6].

We claim the implementation process model is a better way to bridge the gap between decision makers and implementers than the traditional forms: textual messages or informal communication, e.g. by the phone. Besides, it provides persistent memory and awareness to people not directly involved in the interaction. A process model is intuitive and its adoption does not require extensive training. A modeling tool associated with a discussion support such as that implemented by Mendes in Lotus Notes [15], is a good example of a shared workspace aimed at providing common context.

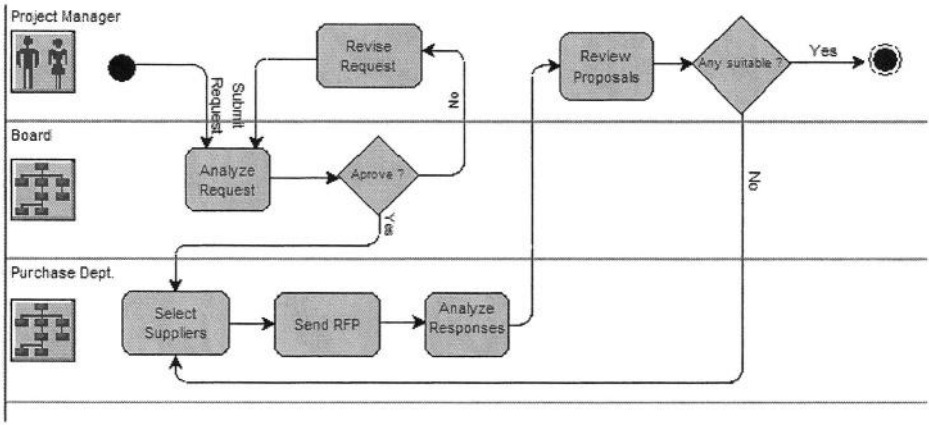

Fig. 1. Initial Process for Purchasing Equipment

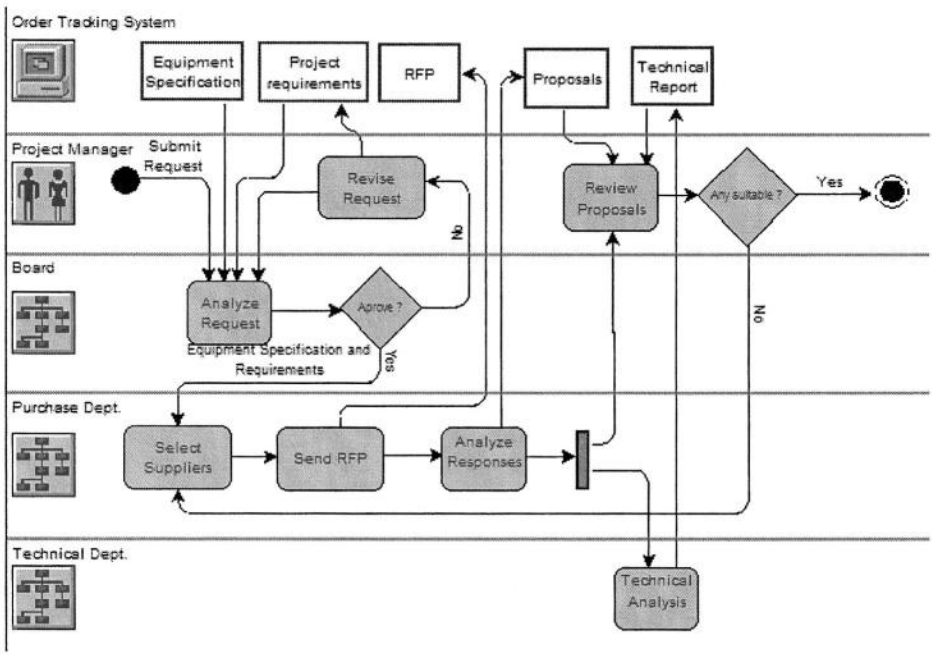

Fig. 2. The Equipment purchase implementation plan revisited

In the Trucco company case, an implementation plan such as the one depicted in Figure 2 could have been designed and have its execution controlled by a WfMS. Instead of passing the decision implementation only to Purchasing Dept., the Board would activate this implementation plan. Based on previous experiences, the plan predicts at least some of the scenarios, preventing some undesirable outcomes, such as Scenario I. The same would happen with Scenario II, unless agreed by Project Manager. As for Scenario III, the plan would support the discussion around the four proposals.

4.2 The Use of Discussion Supporting Tools

The case shows four initial options are possible. With additional information we can perhaps discard some or all of them, at the same time we create new ones, such as the variant generated by the Technical Support Dept. How do involved people argument around these options? How do they document that?

Again, a shared workspace is our preferred solution. The IBIS-based discussion forum [16] used in several implementations [3, 17] was the underlying communication model. Although the IBIS model does not support the decision itself, it documents the discussion in such a way the decision process becomes a straightforward activity.

There are several IBIS-based tools. The one we used to illustrate our example was the QuestMap tool [18]. We used this tool to illustrate our example due to its convenient graphical representation of the discussion, but in our implementation we used Lotus Notes [19]. Figure 3 reproduces the discussion diagram generated by QuestMap. Initially, there are four options represented by the four proposals. The fifth option was suggested by the Purchasing Dept. in view of the unsatisfactory results obtained in the first round. Finally there is the sixth option raised by the Technical Dept. All options have advantages and disadvantages, as shown by the QuestMap diagram in Figure 3. With this information at hand, the Board can decide faster and wisely, based on organizational policy.

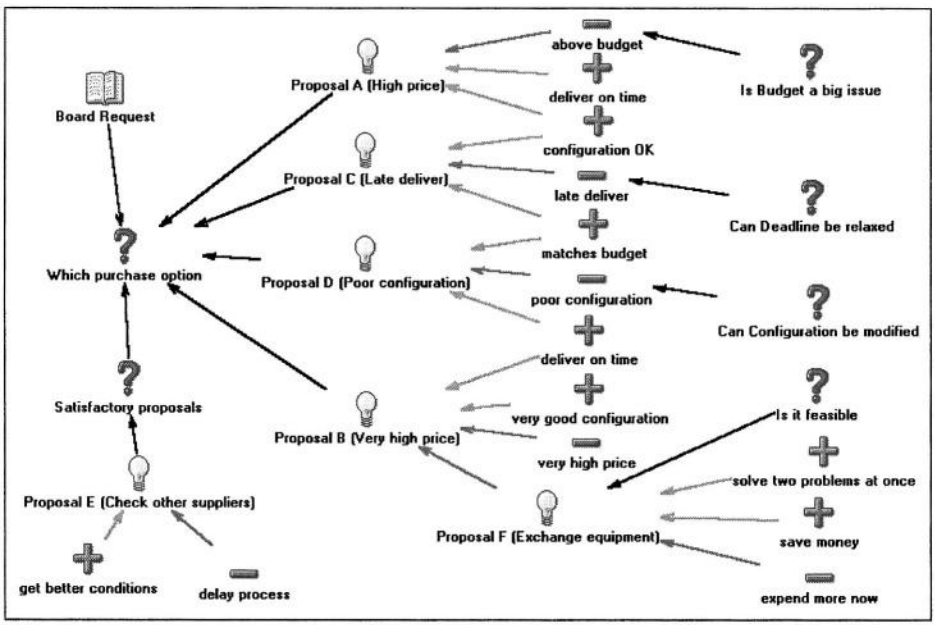

Fig. 3. The discussion around the purchase options

4.3 The SUPRE Implementation

The SUPRE (Post-Meeting Support) System has been developed using Lotus Notes technology. The system consists of two environments. The first supports the interac-

tion necessary to model the implementation plan. The second, which is out of the scope of this paper, supports the implementation follow-up, which is the actual execution of the plan. The implementation plan is designed using the Lotus Workflow Architect [20] and the discussion around it is supported by Lotus Notes.

The initial implementation plan is described using the Lotus Workflow Architect with the aid of pre-defined process - the process beans. Those pre-defined processes are drawn based on the analysis of typical decision plans within the domain of the organization. In our case, the purchase process could have been designed as a result of previous purchase operations. In any case, the environment allows the use of the entire process or part of it. It also allows, of course, the design of a completely new process. An example of the Lotus Architect environment augmented by a set of process beans is reproduced in Figure 4.

Associated with each process, there is a discussion forum, implemented using Lotus Notes. The advantage of having both the process design and the discussion forum under the same environment is that associations between elements of these tools can be easily made. Besides the organization structure used to represent the roles in the process design is the same used for the definition of members of the discussion group.

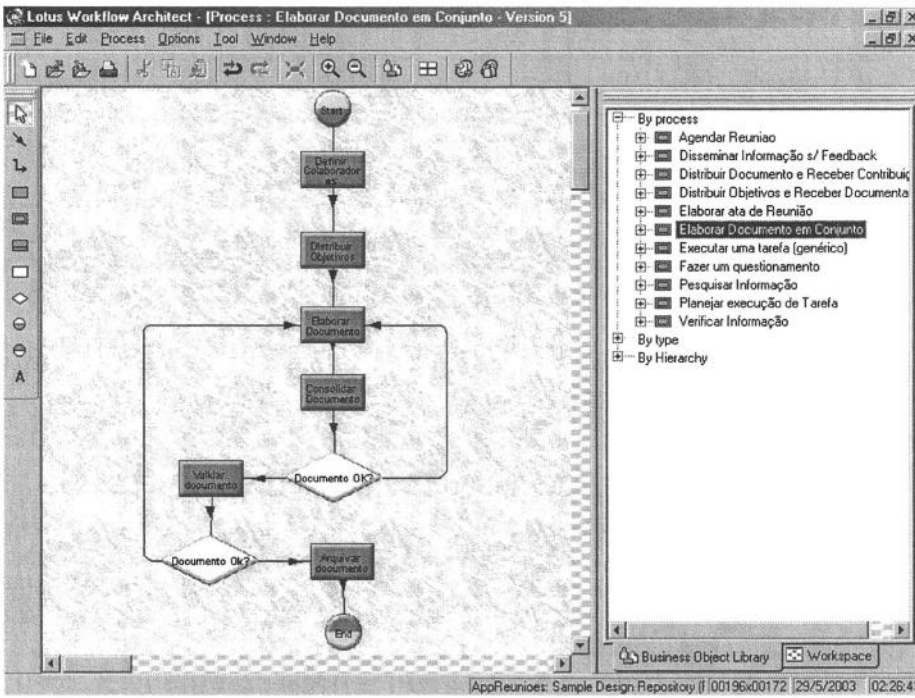

Fig. 4. The Lotus Workflow Architect augmented by Process Beans

5 Discussion

The previous sections have described some situations where there is a clear gap between the environment where the decision process takes place and the environment

where the decision is enforced and implemented. Lack of good communication is thought to be the main cause for this gap, but a number of other reasons can be also associated with this chasm [21]:

- The decision makers weren't really addressing the right problem;
- The decision makers had no real interest in or commitment to solving the problem, anyway;
- Whatever the decision makers thought they decided satisfied their needs for finishing the meeting, but was not detailed or clear enough to be called a real solution;
- The decision makers have the false belief that only policy is important and that the details are someone else's problem;
- The decision makers realize that execution is important, but reserve all perks and privileges for the decision makers themselves.

The reasons listed above show that not all decision problems in organizations could be solved with good communication. In some, communication might play an important role to putting these problems in evidence, but their solution should come from management procedures. Although we recognize the veracity of these situations, our work focus on the problems that people are willing to cooperate to their solution, but do not have the means or the adequate support to address them.

The approach presented in Section 4, based on process modeling and discussion support, is a possible solution to fill this gap. The first question to be answered is what impacts these two technologies would produce in the decision life cycle. We try to demonstrate the advantages of our approach, but some disadvantages can be foreseen. This section presents a discussion on the trade-offs of our approach.

Process modeling of an implementation plan is not an easy task. It is unrealistic to assume that members of a Board of Directors would be able to generate such model from scratch. Although intuitive, the process model requires some expertise both on the modeling techniques and on the domain, i.e., on decision implementation issues. Two solutions are suggested to overcome this problem. First, all proposals should carry an implementation plan, prepared in advance jointly by the proponent and the implementation teams. Second, a library of typical implementation processes could be built based on previous experiences from similar decisions. In the latter case, the process stored in the library may not fit exactly the decision, but can be used as a starting point towards the actual process. An interesting approach, based on a library of process beans has been proposed by Borges et al. [6]. Besides storing complete processes, this library also stores sub-processes, called process beans, which can be assembled together to form a new process.

A library of typical implementation processes may also work as an organizational memory. Like many libraries it is difficult to start, but, given time, it may become an important source of knowledge, where much (many) informal knowledge (procedural contexts) are captured and saved. This long-term advantage counterbalances the initial efforts and possible delays caused by the introduction of a new procedure.

Cultural barriers also raise an issue; people naturally prefer the easy way. Support from managers and pilot projects are part of the suggested prescription. Based on ethnographic observations, we identified another obstacle. Decisions are not always as rational as one would wish. In many cases, the rationale behind a decision is not explicit. In our case, e.g., the Call Center Manager may have the power to persuade the Board to approve Proposal A, because she does not want additional problems

(uncertainties, complications): she has had enough already. Or, on the contrary, she may insidiously favor Proposal C because the project is late anyway and she may use the delay on the equipment purchase as an excuse for the project failure.

The discussion support technology also raises some interesting issues. The argumentation model reproduced in Figure 3 gives us a clear view of the options and their pros and contras. On the other hand, would not it be easier and quicker to discuss the options over the phone? Again, the advantage of having a retrievable memory is a clear-cut. In the Call Center case, the impasse would be sent back to the Board. The discussion map provides a good way of presenting the current situation without additional effort. It also serves as a documentation to justify decisions and to avoid discussion over the same topics, as it has never occurred in the past.

Several IBIS-based systems have been built and used to support discussion [3, 17, 18]. Although the discussion map represents the essence of the discussion it presents at least two drawbacks. First, it is not easy to impose the required discipline to discussion participants. People often mix the three elements (question, position and arguments) in the same statement. This indicates that an intensive training on the model is suggested before starting a real discussion. In some cases, they want only to comment on one element and not add a new element. In our example, the decision made by the Purchasing Dept. about the delivery deadline was not explicitly represented. In order to give room for these requirements, some systems have modified the original IBIS model [3].

The second problem relates to the dynamics of the discussion. The resulting discussion map does not show how and why some elements have been added. For example, Proposal E resulted from the unexpected outcome generated by Proposals A-D, while Proposal F resulted from the later involvement of Technical Dept. By the way, the attempt of the Technical Dept. to find a common solution to two problems is also implicit. This suggests that the discussion stages have to be associated with the activities of the implementation process, providing context to its evolution.

From the contextual point of view, notice our solutions try to build a large shared context to all involved actors. Both at the group level and inter-group level, a shared context allows coherent decisions. Instead of attempting to diminish the number of decisions the implementers make, our approach tries to enable implementers to make the right decisions.

6 Conclusions

A solution to the gap between decisions and their implementations has been presented. It represents a cooperative approach involving technology, which supports people to achieve the common goal of obtaining rightly implemented decisions for the Organization. As such, it needs people willing to collaborate with other employees.

The solution involves using process modeling, shared workspaces and discussion tools. These tools may be used in conjunction with other tools intended to support other parts of the decision meetings cycle, such as the meeting preparation and the decision making support itself.

A number of issues remain open. While the discussion map eases documentation, there is also a danger of information overload for those who later have to relate to this material. Besides, there is also the issue of who should submit the implementation

plan in the first place. Sometimes, new options arise when the implementation is in progress. An important functionality of the WfMS would be the support for changes emerging during running processes.

The approach needs to be tested in real settings. We must find an appropriate environment to start a pilot project and observe benefits and drawbacks of the proposed solution. What type of evaluation will take place? We can qualitatively measure results as a first step. As we mentioned above, many of the benefits will be obtained only after a period of intensive use.

Acknowledgements

This work has been partially supported by a grant from CNPq (Brazil) Prosul Program No. 490187/2003-0 and a grant from Fondecyt (Chile) No. 1040952. The Workflow Architect software was used under the IBM Scholars Program. We would like to thank the three anonymous reviewers for their insightful comments on this paper.

References

1. Oppenheim, L.: Making meetings matter: A report to the 3M Corporation. 3M Meeting Management Institute, Austin, TX, USA (1987)
2. Bostrom, R., Anson, R., Clawson, V.: Group Facilitation and Group Support Systems. In L. Jessup and J. Valacich (eds.): Group Support Systems – New Perspectives. Macmillan Pub. Co., New York (1993) 146-168
3. Borges, M.R.S., Pino, J.A., Fuller, D., Salgado, A.C.: Key issues in the design of an asynchronous system to support meeting preparation", Decision Support Systems, Vol. 27 N. 3, (1999) 271-289
4. Nunamaker, J., Dennis, A., George, J., Valacich, J., Vogel, D.: Electronic meeting systems to support group work. Communications of the ACM Vol. 34 N.7 (1991) 40-61
5. Romano, N. & Nunamaker, J.: Meeting analysis: Findings from research and practice. Proceedings of the 34th Hawaii International Conference on Systems Science, Hawaii, USA (2001) In CD-ROM
6. Borges, M.R.S., Pino, J.A., Valle, C: Support for Decision Implementation and Follow-up. European Journal of Operational Research (2004) (in press)
7. Brezillon, P., Borges, M.R.S., Pino, J.A., Pomerol, J-Ch.: Context-Based Awareness in Group Work, Proc. of the International FLAIRS Conference, Miami Beach, FL (2004)
8. Brézillon P. and Pomerol J-Ch.: Contextual knowledge sharing and cooperation in intelligent assistant systems, *Le Travail Humain Vol.* 62 N. 3, PUF, Paris (1999) 223-246
9. Brézillon, P.: Individual and team contexts in a design process. Proceedings of the 36th Hawaii International Conference on Systems Science, R.H. Sprague (Ed.), Los Alamitos, USA (2003) In CD-ROM
10. Brézillon, P. and Marquois, E.: Une approche centrée contexte de l'activité. Symposium "Tâche, Activité et Contexte" (TAC). In: J.M.C. Bastien (Ed.), Actes des Deuxièmes Journées d'Etude en Psychologie Ergonomique – EPIQUE, INRIA, France (2003) 263-268
11. Brézillon, P.: Using context for Supporting Users Efficiently. Proceedings of the 36th Hawaii International Conference on Systems Sciences, "Emerging Technologies" Track, R.H. Sprague (Ed.), Los Alamitos, USA (2003) In CD-ROM
12. Pomerol J-Ch. and Brézillon P.: Dynamics between contextual knowledge and proceduralized context. In *Modeling and Using Context* (CONTEXT-99), Lecture Notes in Artificial Intelligence Vol. 1688, Berlin Heidelberg New York (1999) 284-295

13. Watson, R.T.: Data Management. *Databases and Organizations.* John Wiley and Sons, Inc., Second edition, New York
14. Gutwin, C., Roseman, M. and Greenberg, S.: A Usability Study of Awareness Widgets in a Shared Workspace Groupware System. Proceedings of ACM Conference on Computer Supported Cooperative Work, Boston, Mass., ACM Press, New York (1996) 258-267
15. Mendes, C.D.L.: Post-meeting Support using Workflow Technology . M.Sc. Dissertation, Informatics Program, Federal University of Rio de Janeiro, (2003) (In Portuguese)
16. Kunz, W., Rittel, H.: Issues as Elements of Information Systems, Working paper #131, Institute of Urban and Regional Development, U. of California at Berkeley (1970)
17. Guerrero, L.A., Pino, J.A.: Preparing Decision Meetings at a Large Organization, Proc. of the Decision Making and Decision Support in the Internet Age, F. Adam, P Brézillon, Patrick Humphreys and J-Ch. Pomerol (editors), Cork, Ireland (2002) 86-95
18. Compendium Institute: Questmap, Available at http://www.compendiuminstitute.org/tools/questmap.htm Acessed on 13 April, 2004
19. IBM Lotus Notes: Available at http://www.lotus.com. Accessed on 30 March 2003
20. IBM Lotus Workflow: Available at http://www.lotus.com. Accessed on 30 March 2003
21. Unknown author: Referee's comments on the submitted version of this paper

CreEx: A Framework for Creativity in Cooperative Problem Solving

Adriana S. Vivacqua[1] and Jano M. de Souza[1,2]

[1] Computer Science Department, Graduate School of Engineering (COPPE)
{avivacqua,jano}@cos.ufrj. br
[2] Computer Science Department, Institute of Mathematics (DCC/IM)
UFRJ - Federal University of Rio de Janeiro
Rio de Janeiro, Brazil

Abstract. Creativity has become an important factor in recent years, as companies need to be able to quickly adapt to take advantage of new opportunities and handle fast paced changes in their environment. Creativity theorists have proposed models to explain creative thought that go beyond the individual to encompass social aspects of creativity. We are interested in the interactions between team members that lead to innovative solutions for problems and new ideas, and in computer support for collaborative creativity in problem solving. In this paper, we present CreEx, a framework to foster and support group creativity. By creating appropriate environments for the exploration of problems and discussion of ideas, we hope to enable users not only to generate novel solutions and capture decisions made, but also to learn about each other's domains and think differently.

1 Introduction

In recent years, fast paced changes have created a volatile environment, in which companies must function. Technological, political and economical changes have generated a need for flexibility, and companies must be able to adapt to survive. The problems companies must face have also become more complex, and can often only be handled by groups of individuals. These groups are frequently interdisciplinary in nature, which often means communication problems, since individuals will have different views of the problem at hand and different opinions on how to go about solving it.

We are interested in the cooperative aspects of creativity: group creativity and the in-group interactions that lead to creative results [1]. Nowadays, much significant work is no longer undertaken by a lone creator, but by groups of individuals cooperating on a problem. The result of their interactions is a creative piece of work, and several members contribute to the construction of this shared artifact. While it may be the case that one member had a "creative breakthrough" and practically solved the problem by him or herself, this breakthrough occurred in a group context, possibly as a result of previous interactions between group members.

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 166–173, 2004.

These interactions, the timing and knowledge involved, group dynamics and characteristics of individual members and how to support them with computational tools are the focus of our research. In this paper, we present CreEx, a framework for creativity in cooperative problem solving. In the next section, we introduce some background work, followed by our framework in section 3 and in we wrap up with a discussion.

2 Background

In this section, we present some theories upon which we have based our work, a selection of the work most relevant to ours. We are most concerned with creativity in problem solving, for it exhibits some constraining goals and objectives and more often involves groups of people than artistic expression (which also interests us, but isn't the focus of our research).

2.1 Creativity Theories

Although several different definitions exist, many authors define creativity as the process that leads to the production of an artifact that is both innovative and useful [2,3]. This definition involves a product (the artifact) and an assessment (of its usefulness), two important elements in problem solving. Several theories that attempt to explain the creative process exist, but we are especially interested in the ones that deal with the social aspects of creativity.

According to Csikszentmihalyi [4], creativity is produced by a system of three interacting elements: the individual, the domain and the field. The *domain* establishes shared symbols and rules; the *individuals* work within a domain to create something new and the *field* is a set of experts that judges these contributions to determine whether or not they are creative and deserve to be incorporated in the domain (effectively changing it). This is a cyclic process, through which knowledge is continually built on previous knowledge, with the best ideas being absorbed into the domain and the bad ones being discarded, in a process similar to the evolutionary.

Inspired by Csikszentmihalyi's system model of creativity, Shneiderman [5] proposed GENEX, a framework for the generation of excellence. According to him, the creative process is a four-stage cycle: collect (gathering information from diverse sources); relate (consulting with individuals who may be able to furnish useful insights); create (experimenting with possibilities, trying to generate new solutions); donate (disseminating results to the community). A person can move from one stage to another as needed. This process should be supported by creativity support systems.

2.2 Problem Solving

Most problems faced by companies nowadays aren't easily tractable and demand unique, innovative solutions. These problems are often poorly defined and open-

ended, having more that one possible solution and no clear stopping point. Therefore, the problem solving process stops when the group runs out of resources. Solving these so-called wicked problems is an iterative and exploratory process, where designers experiment with different paths [6], learning about the problem while devising a solution. The process is one of defining the problem as well as solving it.

Some problems are naturally interdisciplinary and require teams of experts from different domains to address it. The introduction of other individuals increases the potential for confusion and miscommunication, especially when these come from different backgrounds and disciplines. The number and diversity of individuals involved in the problem solving process is an additional complication, since it can make communication harder and fragment the group [7].

However, many complex problems can only be handled by pooling together resources from a number of different disciplines. Nissani [8] argues that forming interdisciplinary groups to handle complex problems leads to more creative solutions, as outsiders bring fresh insight and methodology to the problem at hand. Along similar lines, Fischer [9] suggests that, in interdisciplinary teams, the ignorance of one person in relation to another's field of expertise stimulated creativity. This "symmetry of ignorance" leads to discussions and explanations of concepts and points of view, which in turn leads to the generation of new ideas.

This is in general agreement with creativity research that points towards external stimuli as a factor for creativity. Santanen and colleagues [10] have proposed a model to explain how individuals reach creative solutions: knowledge is mapped in a person's mind as a network of concept frames, and creativity stems from the combination of unrelated (or distant) frames. Their model maps causal relations between certain factors such as cognitive load or external stimuli and the production of creative solutions to problems. Their experiments have ascertained that external stimuli contribute to more creative solutions: bringing up different concepts led to the generation of novel ideas and links between frames.

3 CreEx Framework

Creative work involves a certain amount of pre-existing domain knowledge and its transformation into new knowledge [12]. The combination of concepts from different domains allows problem-solvers to think in non-conventional ways.

There are two levels of interaction that should be addressed by a system attempting to support cooperative creative problem solving: extra-group and intra-group. Much research concerns individual creativity and the relations of the individual with the external world (knowledge and experts). When dealing with a group of individuals, their in-group interactions (with shared knowledge and each other) are as important as those with the external world [1].

3.1 Handling the Problem

Many researchers recognize that, in problem solving, creativity stems from the way one looks at the problem, and that defining the problem is as important as finding its solution [4]. This is especially true of wicked problems, which are ill defined and open to interpretation.

When studying their methods, Cross [11] noted that designers not only work with domain rules, keeping track of basic principles that govern the domain and the application (such as physical laws and functionality), but also tend to introduce new constraints or characteristics they believe the solution should have, effectively reducing their search space while looking for a solution. This, in fact is a common way of dealing with poorly defined problems: one makes assumptions or introduces new characteristics or constraints, in an attempt to make the problem tractable. Through this process, the problem evolves into a solution. On the other hand, there is also a danger to lose oneself on speculation, never reaching a solution, or to lose track of the new characteristics introduced and the consequences of their introduction.

With this in mind, we are changing our working paradigm from *solving* a problem to *exploring* a problem, redefining and shaping its solution in the process. We believe it is important for a team to work with a shared representation to help guide the process and make sure all participants are on level ground regarding assumptions about the problem space and characteristics the solution should/must have.

Our problem model was created so that a team of individuals can explore and define the problem space as they work, keeping track of assumptions, constraints and features. It is based primarily on reported problem-solving research. This problem model has a set of basic interrelated elements:

- Problem: initial problem description, serves as root for graphs that represent the problem space. Includes a description of the problem and goals.
- Sub-problems: breakdowns of the bigger problem, are also problems and include goals, characteristics, etc.
- Characteristics: features the solution should have. May be determined by the client (a requirement), group (a feature the group has agreed upon), individual (a suggestion an individual has made) or domain (a constraint inherent to the domain, for instance the laws of gravity). They may be mandatory (must be addressed for the problem to be solved) or optional.
- Assumptions: assumptions individuals base their work on when designing a solution. These may lead to certain characteristics or constraints that will have to be addressed.
- Questions: questions may be raised by an individual when working alone or by the group, and they may be resolved with the client or by the group. These sometimes lead to decisions or new characteristics or constraints.
- Decisions: decisions made regarding the solution, usually generated after an evaluation process (formal or informal). These are defining elements of the solution statement, and may have implications that translate into new characteristics the solution will need to have.

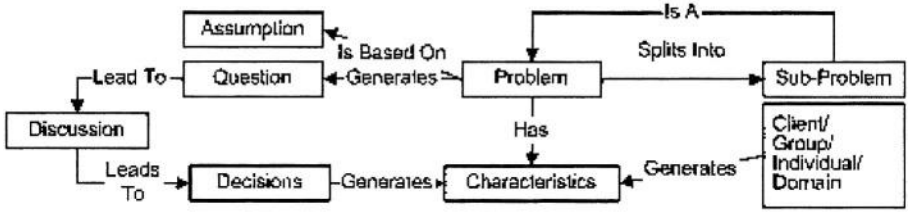

Fig. 1. Problem Model

Problem solvers work on this model (shown in Figure 1), adding new constraints or features and expanding it as they go. They may attach additional resources to the model, such as explanation nodes, files or contacts that may help the understanding of the problem and its evolution. This model can also serve as a basis for discussion with other stakeholders (such as clients or external consultants), who can use it as a basis for understanding what is being discussed and more quickly establish common ground with team members. It is important to note that a user can also work alone. Each individual can work privately on the representation, exploring alternatives and them present them to the group for discussion.

Additionally, problem solvers should be able to map their discussion and decision making process, generating a history of interaction. Dialog mapping [6] is an interesting approach, for it allows individuals to interact by mapping their thoughts, questions and responses, generating a graph that helps them visualize the argumentative process leading to decisions. We are inclined to use such an approach as well: any of the elements described above could serve as a root for a new discussion thread, which would explain and capture the decision making process, possibly leading to the introduction of new elements in the problem model.

In our framework, the team works together refining and exploring the problem as they move towards a solution. Discussion elements originate in the problem model, and may also cause the generation of new characteristics for the problem. This shared representation guides the group in their solution attempts, and the mapped dialogs serve as basis for reflection and explanation of previous decisions.

3.2 Managing Knowledge

Plenty of knowledge is involved in creative work. From shared domain knowledge to individual, extraneous knowledge that can generate insights, it should be managed and saved as it becomes relevant to the problem-solving context. Thus, part of our work involves capturing and organizing information and knowledge generated during the problem solving process and capitalizing on individual differences to generate more creative ideas. This means selecting individuals with distinct backgrounds and having them utilize this knowledge when working on the problem. We created a simple user model that maps a user to his or her interests and resources (which may be files or contacts). The following elements are involved: the user, resources (may be files or contacts) and knowledge areas related to each user and resources.

User models are initially built through text mining of individual's documents, emails and contact lists and clustering these into specific knowledge areas. Based on this information, it becomes possible to select appropriate individuals to serve as external consultants in the project and breaking an individual's frame of mind through the presentation of external information (which could potentially lead to new insights). The selection of these individuals is based not only on the domain knowledge they possess, but also on their range of external knowledge.

As mentioned before, creative sparks come from the introduction of external knowledge into the problem domain [10]. This could be achieved by displaying documents from different domains, to have the user switch context for a brief period of time and look at the problem under a different light. This is what we call *context bridging:* searching for relations in other contexts and bringing them into focus, thus forcing a different outlook on the problem. Initially, this will be accomplished through the use of WordNet, looking for words applicable in more than one domain the user knows and following links between words to get to external information that is somehow linked to the domain.

To manage this knowledge we have envisioned shared and private knowledge bases, containing the group's and each individual's knowledge. Agents are used for information retrieval, looking for relevant information as members work on the problem. So as not to breach privacy, these agents work only within the limits of the knowledge bases: agents mining private knowledge bases can only present their suggestions to the owner of the knowledge base, never to the group. It is then up to the user to decide whether this information should be shared with the group or not. It is important to note that it is the users, who have an understanding of the knowledge and its relation the problem, who introduce new knowledge into the process. It is also likely that group members will need to engage in an explanatory process, so all understand the relevance of a piece of knowledge to the problem. This externalization process helps others understand a different point of view and each individual better structure his or her thoughts.

During the creative process, especially when it is constrained by having to produce a solution in a timely manner, there are two very distinct moments: there is a moment of *divergence,* opening up new alleys and experimenting new solutions; and there is a moment of *convergence,* closing down on the best solutions, making decisions, criticizing and choosing options. Initially, it is important not to discard every idea, but to keep an open mind, allowing the exploration of new ideas. However, as time goes on (and deadlines approach), it becomes important to focus and decide on one particular path, generating a solution to the problem. This means that available resources (such as time, money or individual availability) must be managed carefully. These resources should be determined at the beginning of a project and the system watches them as the process goes on. The system should warn the group of approaching deadlines so that members can focus on evaluating options and finding a solution. Additionally, the system could suggest a line of thought (based, for instance, on how much work has gone into it) for the problem solvers to work with. It can also remind the group which constraints of the problem have been met and which haven't, so that problem solvers can focus on open ends.

4 Discussion and Further Work

We have envisioned a framework and environment to support the following activities: idea generation and management, branching from one idea to the next or generating new ideas and saving alternatives; exploration, discussion, extension and study of each idea; evaluation, analysis and critique of ideas and proposed solutions; decision-making on the final solution.

This environment should not be too restrictive, but should keep tabs on deadlines and prerequisites, to help ensure that the project will come to an end. In such a system, it is important to keep knowledge bases of the problem domain and of external domains (these belong to team members, mapping their knowledge and interests), plus a shared problem representation. Our system framework is composed of individual and shared knowledge bases, plus sets of agents (assistants) to retrieve relevant information as the problem solving process evolves. Users should also be able to work synchronously or asynchronously.

We believe supporting creative work to be an important issue. Within our framework, we expect not only to support and foster creative work, but also to study creative dynamics and interactions. We are trying to determine which interactions lead to creative results and what requirements a collective creativity support system should have. The system is undergoing initial implementation of user profiles for tests with rules for the introduction of external consultants and knowledge.

There is much work to be done, models need to be perfected and validated; and issues such as motivation (a key issue for creativity), conflict management and negotiation (to ensure convergence) still haven't been addressed. Privacy and authorship will also need to be addressed. Situations such as a person not being able to objectively evaluate results and ideas should also be addressed. Determining the timing of the system, so that it will foster creativity is important and we have been studying group behaviors to address these different situations.

A few systems have been proposed to support creativity: [12] and [13] present systems that introduce external knowledge into the process, both working at an individual level. Brainstorm [14] enables anonymous submission and criticism of ideas by a group. EDC [9] is an enhanced physical environment for group problem solving, targeted at urban design.

In an inspiring parallel with Jazz music, Kao [15] highlights the interaction between individuals and the need for an appropriate environment for creativity. He points out that there is a balance between cooperation and competition, as musicians try to outdo each other without losing the agreed-upon theme. There are rules that must be followed and a common language they all speak in order to be able to produce something coherent. The interplay between individual and group efforts creates a new experience each time, with each person contributing to the whole according to his or her own preferences and style. These interactions, where individuals have the freedom to create and add to the group creation, are at the heart of improvisational jazz music and are a source of some great cooperative endeavors. We believe this sort of creative cooperation to be important not only in jazz, but also in business and educational environments.

Acknowledgements

This work was partially supported by CAPES and CNPq.

References

1. Paulus, P.B.: Groups, Teams and Creativity: The Creative Potential of Idea-Generating Groups. Applied Psychology: An International Review, 49 (2) (2000) 237-262
2. Bonnardel, N.: Creativity in Design Activities: the Role of Analogies in a Constrained Cognitive Environment. Proceedings of Creativity & Cognition (1999)
3. Burleson, W., Selker, T.; Creativity and Interface. Communications of the ACM, vol 45, n. 10 (2002)
4. Csikszentmihalyi, M.: Implications of a Systems Perspective for the Study of Creativity. In: Handbook of Creativity. Cambridge University Press, Cambridge, UK (1999)
5. Shneiderman, B.: Creating Creativity: User Interfaces for Supporting Innovation. ACM Transactions on Computer-Human Interaction, vol 7 n.1 (2000)
6. Buckingham Shum, S., MacLean, A., Bellotti, V., Hammond, N.: Graphical Argumentation and Design Cognition. KMI-TR-25, Knowledge Media Institute, UK (1997)
7. Conklin, J.: Wicked Problems and Social Complexity. In: Dialogue Mapping: Defragmenting Projects through Shared Understanding. Forthcoming. CogNexus Institute (2003)
8. Nissani, M.: Ten Cheers for Interdisciplinarity: A Case for Interdisciplinary Knowledge and Research. Social Science Journal, 34 (2) (1997) 201-216
9. Fischer, G.: Symmetry of Ignorance, Social Creativity, and Meta-Design. Proceedings of Creativity and Cognition (1999)
10. Santanen, E.L., Briggs, R.O., de Vreede, G.: The Impact of Stimulus Diversity on Creative Solution Generation: An Evaluation of the Cognitive Network Model of Creativity. Proceedings of 36th Hawaii International Conf. Systems Sciences (HICSS'03). IEEE Computing (2003)
11. Cross, N.: Creative Cognition in Design: Processes of Exceptional Designers. Proceedings of Creativity and Cognition 4, Loughborough, UK (2002)
12. Nakakoji, K., Yamamoto, Y., Ohira, M.: A Framework that Supports Collective Creativity in Design using Visual Images. Proceedings of the 3rd Conference on Creativity and Cognition (C&C'99), Loughborough, UK (1999)
13. Shibata, H., Hori, K.: A System to Support Long-term Creative Thinking in Daily Life and its Evaluation. Proceedings of the 4th Conference on Creativity and Cognition (C&C'02), Loughborough, UK (2002)
14. Stenmark, D. Klang, M., Olsson, S.: A Critical Look at Knowledge Creation. Proceedings of IRIS22, Jyvaskyla, Finaland (1999)
15. Kao, J.: Jamming: the Art and Discipline of Business Creativity. Harper Collins Publishers, New York (1997)

This page intentionally left blank

SaGISC: A Geo-Collaborative System

Paula André[1,2] and Pedro Antunes[1]

[1] LaSIGE / Department of Informatics, Faculty of Sciences of the University of Lisboa
Bloco C6, Piso 3, Campo Grande, 1700 Lisboa, Portugal
paa@di.fc.ul.pt
[2] INETI - Instituto Nacional de Engenharia, Tecnologia e Inovação
Estrada do Paço do Lumiar, 22, 1649-038 Lisboa, Portugal
Paula.andre@ineti.pt

Abstract. The production of geological mapping by conventional processes is a complex work of data gathering and integration, along with expert and team analysis. This process is very time consuming, since it implies several expeditions to the study location. In the organization studied by this paper, this process can take several years to be completed. The objective of this project is to build a remote collaborative system that supports information sharing by the teams that participate in geological data gathering. The developed system integrates several tools for information sharing and geological/topographical data referencing, as well as support to group discussion and decision. The integration of these tools makes up a geo-collaborative system. The development of this system was done in the context of the Portuguese Geological and Mining Institute (IGM). The evaluation of the prototype by 30 experts from IGM revealed that the proposed goals were accomplished: the system was considered better than the conventional approach.

1 Introduction

The process of gathering geological data has two components of extreme importance: office and fieldwork. The first stage of this process takes place in the office, searching for preliminary information about the target area. This preliminary information includes bibliography, notes from previous field works, charts, geochemical results, geophysical results, etc.

A second stage occurs in the field, with geological data gathering. Field work varies significantly according to the intended goals. For instance, one may have to carry out a detailed analysis, using a 1/5.000 scale, for a geo-technical and hydrological study, or a more regional study, using 1/50.000 or higher scales.

After this stage and back in the office, the field technician, together with a geological coordinator and/or experts from other areas, review all the gathered data and search for the best method to integrate the information in a meaningful way. This is similar to setting together the pieces of a jigsaw puzzle.

Finally, the geological coordinator goes to the field to validate and/or review the results and, in case of doubt about any geological element, requests for help from

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 175–191, 2004.

specialists in specific areas such as structural geology, paleontology, petrology, sedimentology, etc. This collaboration usually implies one or more visits to the study area, sometimes in remote places with difficult access for any of the experts, including the fact that all this process can be expensive and time consuming (for instance, in the case of IGM, studies in the Azores islands are usually very expensive and time consuming, being scarcely populated and located in the middle of Atlantic).

With this action-research project we tried to design, develop and experiment a prototype of a geo-collaborative system that supports gathering geological data and, at the same time, allows field technicians to exchange information with other specialists on the office. The fundamental idea behind the geo-collaborative system is to economize on the number of visits to the field necessary to analyze dubious situations, or to get second opinions on specific geological elements.

The developed prototype interacts with Geographic Information System (GIS) tools and supports obtaining geo-referenced data such as notes, messages, photos, sketches and sound. The prototype generates an e-book with all the information associated to the points gathered on the field, which represents an electronic substitute of the artifacts traditionally used by the specialists. The prototype centers the collaboration between all experts in this e-book, giving a new dimension to information sharing supported by instant messaging tools like MSN® Messenger.

The development of this prototype followed a user-centered approach, based on the Contextual Design Methodology proposed by [Beyer 98].

This paper is structured in the following way: In the next section we will describe the traditional information gathering process, without any computer support; Next, we will present the design of a geo-collaborative system that will transform this traditional process, identifying the requirements and the new information gathering process; Then, we will give some additional details about the prototype; Finally, we will describe the experiments done with the prototype and present the obtained results from the prototype evaluation carried out by 30 experts from IGM.

2 Traditional Data Gathering in Field Work

The detailed analysis of geo-referencing work on the field, as well as information gathering and decision making tasks necessary to carry out geological mapping, was made with the support from experts of IGM.

Field work begins with the technician trying to locate himself on the chart and on the field, using a compass or GPS and any conspicuous points that may be referenced on the chart. Next, the technician checks the geomorphology of the zone to see if any spot heights could give her some hints of the geology of the zone. For instance, spot heights can give an indication of hard formations, e.g. Quartzites. Conversely, a water line could hint a zone of geological weakness or fault. From this stage, the field technician annotates the chart (not exactly the chart, as will be described later) and georeferences data in a field book with everything relevant that she observes.

In Figure 1 we present an example of a page of the field book, which is filled up with several notes, descriptions, doubts and sketches.

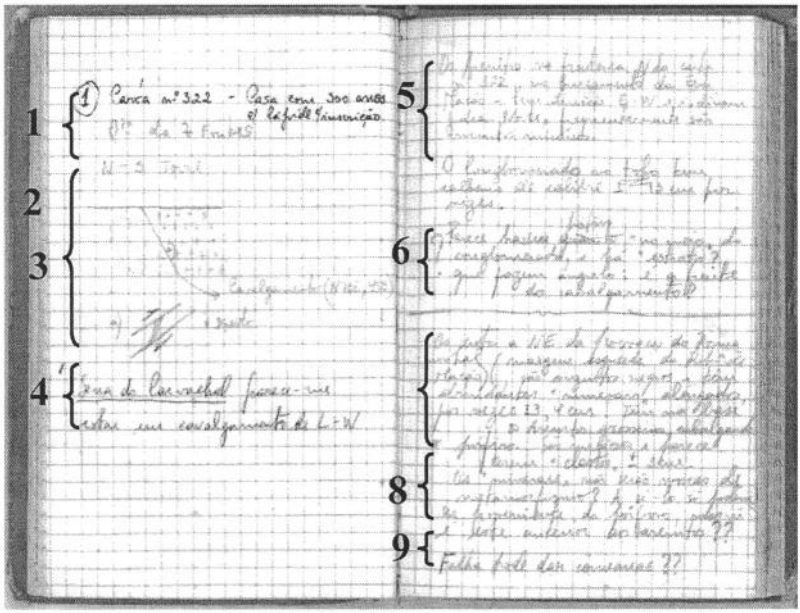

Fig. 1. The field book

As can be seen on the transcription below, the localization, geological descriptions, sketches and doubts that arise on the technician's head are important to characterize field work:

1) Importance of localization – "Map nr. 322 – 300 year old house"
2) Sketch of trust fault and orientation
3) Doubt – "Carvalhal hill seams to be in trust fault from East-West"
4) Description with localization – "The sandstone in the border of map nr. 322, in the intersection of road Mação [located approximately 150Km North of Lisbon] have East-West direction and dip towards North, are frequently Micaceous"
5) Interrogated Description – "Porphyry seem to exist in the middle of conglomerate. Are there any strata? Which have an angle: is it Pyrite in the trust fault?"
6) Doubts with geo-referencing
7) Doubts – "Are the minerals metamorphism derived? Being so, they have their origins in Porphyry, but are not they previous to sandstones?"
8) Doubt – "Fault could give Hornfels?"

In the field notes transcribed above there are several doubts that might disappear with a second visit to the study area. This is the case of point 3). The clarification of this doubt could give a whole new geological interpretation of the area. Another doubt is point 7) where the field technician is not sure if the minerals there are metamorphism-derived. And in that case, there is another doubt with a structural context: "but is not Porphyry previous to sandstones?" This kind of a doubt could be raised or answered afterwards by the coordinator geologist or by the mineralogist/petrologist. If the

doubt persists, the technician may have to consult another expert, e.g., a structural geologist, who may have to visit the study area or alternatively request further measurements from the field technician.

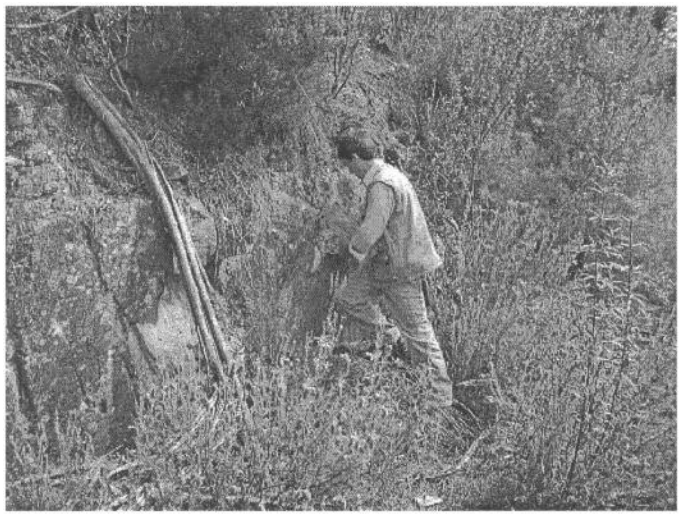

Fig. 2. The technician taking measures with the compass

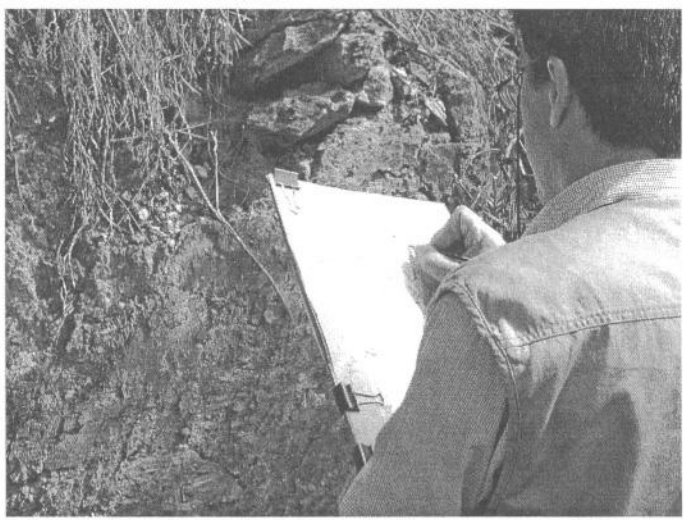

Fig. 3. The technician tracing on the transparent paper overlaying the chart

While the technician goes through the territory, she measures with the compass the attitude and inclination of the geological formations (Figure 2). This action can help reviewing the geology of the field, as the same values for inclination can indicate the same formation.

Fig. 4. Example of the notes taken on the transparent overlay

The technician preserves this information by tracing on a transparent paper which is overlaying the 1/25.000 chart (Figures 3 and 4). In order to help identifying the geology of the territory, for example Silurian or Ordovician, the technician also uses the geologist's hammer. It is important to check not only rock fragments but the sound produced by the hammer striking the rock. For instance, in the concrete field work illustrated in Figures 1-4, the Quartzites produced an acute sound, almost metallic.

2.1 Collaboration in Field Work

Usually, considering a traditional field work, there is a team on the field, which may be composed by one or more elements, who observe, analyze and interpret the geology and data in situ, gather information and reference that information on the overlay paper and field book. The field team reports to the IGM team (the members of the field team may also be members of the IGM team), while promoting discussion, reviewing and consolidating field information and, whenever it is necessary, obtaining clarifications and explanations from the members of the IGM team.

The IGM team is usually composed by a geological coordinator, responsible for a particular geological mapping project, and experts in geology and other related fields, like hydrogeology, geophysics, etc. This team, having overall responsible for the project, observes, analyses, and interprets the geological data obtained on the field and at the IGM itself. The team is also responsible for resolving doubts, discussing results and guiding future activities from the field team. Whenever it is necessary, members of the IGM team go to the field in order to revise, clarify and consolidate geological information. The collaboration between these two teams is illustrated in Figure 5.

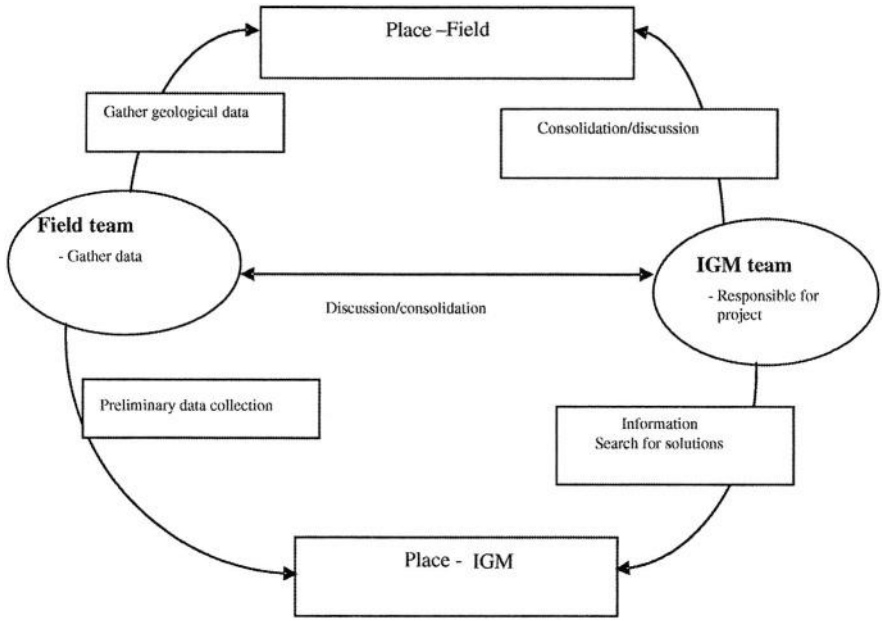

Fig. 5. Collaboration between field and IGM teams

3 Collaborative System Design

As it can be seen in the previous description, the gathering of geological data and execution of geological mapping have to be made both in the field and the office, requiring collaboration between teams and a high degree of expertise in geology. The proposed system should allow high mobility to the people in the field and also preserve as much as possible the freedom to use their hands for tasks such as analyzing the geology of the territory, using the hammer, compass, GPS, etc.

The collaboration support to field and IGM teams should afford some flexibility getting in contact with the IGM teams, since the fundamental purpose of the system is to avoid the need to get back to the field. On the other hand, these contacts should not be very disruptive to the work of the IGM teams, since the contacts are varied and occasional. For these two reasons, we adopted an instant messaging mechanism like MSN® Messenger. Such a mechanism is well-aligned with the current workplace at IGM, and affords the immediacy of contacts from field teams. In that way, the teams will not hesitate to discuss questions, doubts or different opinions [4]. Furthermore, the approach affords two alternative work arrangements for IGM teams: either scattered among their usual workplaces, or grouped together on a team room.

To simplify the establishment of contacts from the IGM to the field teams an audible warning is implemented.

Instant communication between the field and IGM teams is related to georeferenced information available in geologic and topographic charts. Fortunately, this

shared context does not require exchanging charts between the two teams while working in the field (which would impose significant requirements to the communications infrastructure). This shared context is developed during the planning stage, where relevant information is identified and disseminated to both teams. The system must however grant the coherence of the references to these charts.

We also noted on our observations of field work that the field book assumes a very significant role, being the place for inserting notes related with any information gathered in the field, including doubts, sketches and drawings. Based on this observation we decided that the support to collaboration should be centered on a digital artifact which could reproduce the actual field book. We designate this artifact as a field e-book. The field e-book shares information between the field and IGM teams, although on demand only, in order to lower the requirements to the communications infrastructure. The information introduced in the field e-book is geo-referenced, letting the users search for available information about a specific point, at any moment and either locally or remotely. This feature represents an extension to the traditional field book.

The types of data that can be introduced in the e-book are: (1) location coordinates; (2) notes, where geologists write doubts, descriptions and comments; (3) drawings of geological elements; (4) photos; (5) sounds of the geological hammer striking rocks; and (5) messages exchanged between teams related with the specific coordinates.

We previously mentioned that technicians gather information in the field using three main physical artifacts: the field book, the topographic/geological chart and the transparent paper overlaying the topographic map and attached to a backing board. We already mentioned that we substituted the field book with an e-book. Concerning the chart, our solution uses ArcPad® [1], a GIS tool belonging to ArcGIS Mobile Software from ESRI® [3], which is commonly used by IGM. Concerning the transparent overlay, we came to the conclusion that it would not be feasible to implement a computational substitute for this physical artifact (lack of time, money and resources). Thus, such functionality is not developed and the user must rely on a workaround, using a drawing tool not integrated with the topographic/geological chart.

Finally, to connect all the components referred above, we developed a tool designated SAGISc. This tool allows the technician on the field to manipulate the e-book, establishing communications with the IGM team (using MSN® Messenger) and interacting with ArcPad® and the drawing tool. The remaining hardware and software used was: GPS device, digital camera, microphone and sound recorder.

Because there is a low level of control exerted by the system on the collaborative activities, the system usage will work best with technicians that already have good work relationships. An informal but disciplined social environment corresponds to the ideal situation for working with the system.

3.1 Redesigned Collaboration Using SAGISc

In the diagram presented in Figure 6 we illustrate how the field and IGM teams collaborate using SAGISc.

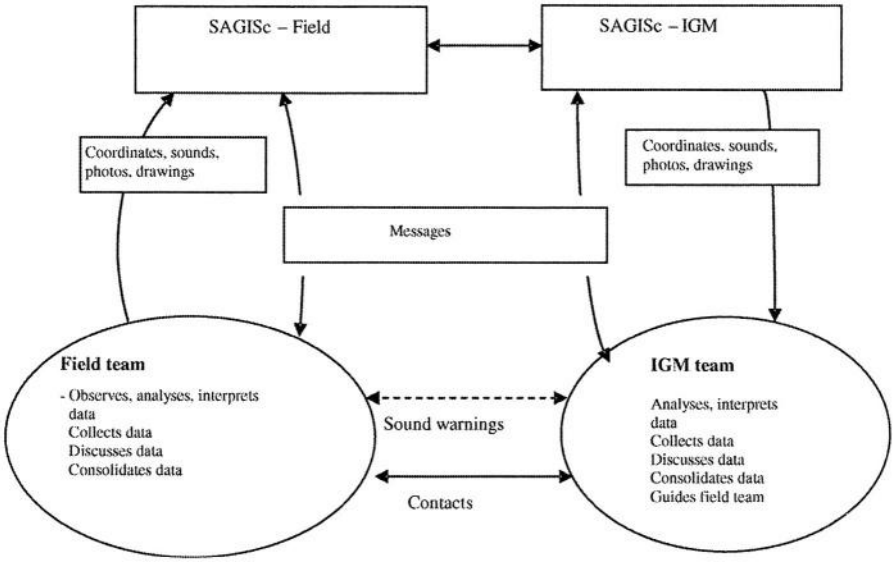

Fig. 6. Collaboration between field and IGM teams using SAGISc

4 Implementation

The SAGISc prototype was developed with the VB.NET® language, basically because it would be adequate for prototyping and easy to integrate with ArcPad®.

Since this a prototype aiming at experts in geo-sciences working on the field, we had particular care with the structural navigation aspect of its development, focusing on simplicity and consistency of use. In Figure 7 we describe the structural navigation structure of SAGISc.

5 Evaluation

After the implementation of the SAGISc prototype, we had it evaluated by users. The evaluation process was done in two successive steps. First, we had a preliminary evaluation with users working in the field with the system. The experiment was done with technicians from IGM, where the field team was composed by a Geologist and the IGM team was composed by three specialists of different areas in Geology. After this preliminary evaluation, we set up a broader but "static" evaluation. Overall, 30 IGM specialists in Geo-sciences participated in the evaluation process.

The preliminary evaluation was carried out in circumstances very close to reality, during which a field team was sent to Oeiras (located approximately 15 Km West of Lisbon) with the goal to verify and discuss the geology of the area with the IGM team that was in the IGM main office. The field team was composed by one geologist with low experience in this kind of work (collecting data in the field), because her spe-

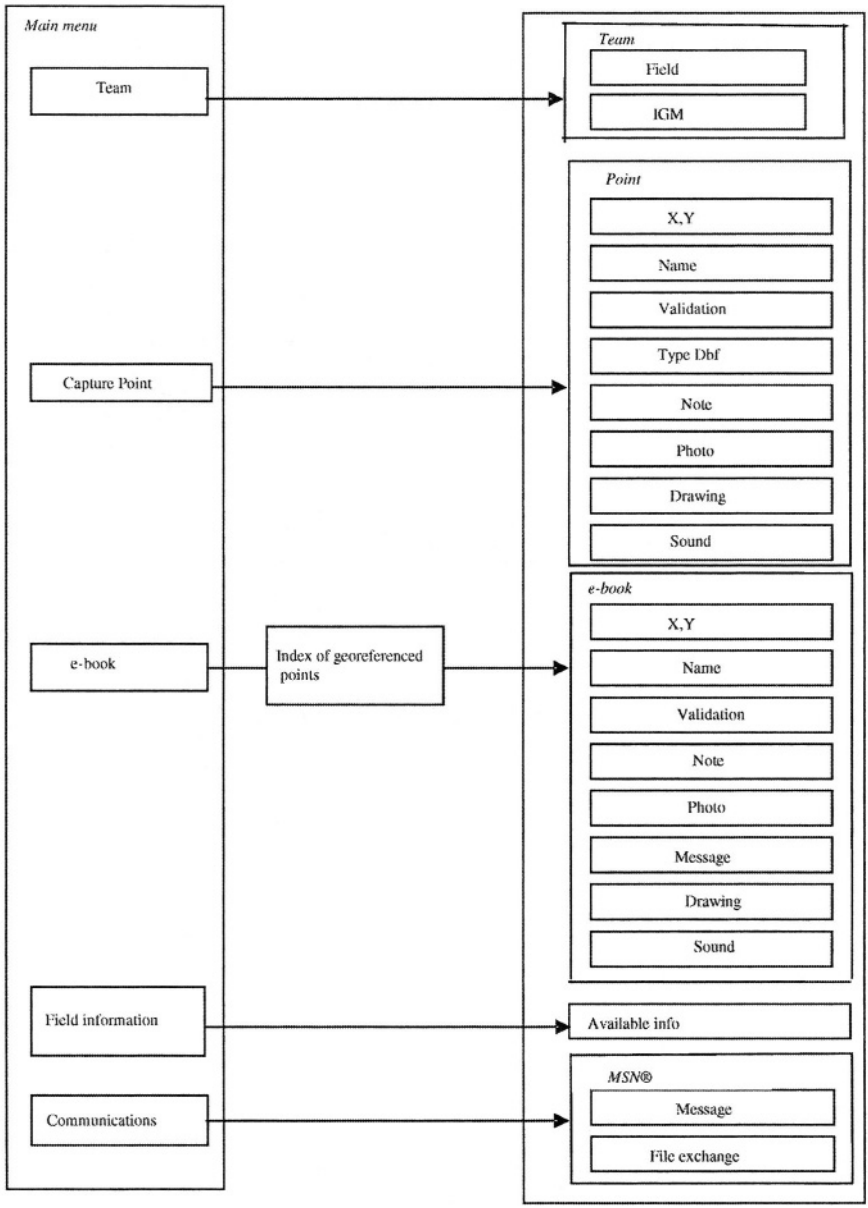

Fig. 7. Structural navigation of SAGISc

cialization area is Micropaleontology. The IGM team was composed by three specialists in the Geo-sciences fields (a geologist, a hydro-geologist and a geological engineer). Both teams were given the necessary hardware and software, and were briefly instructed on the system usage.

The geologist on the field carried the following equipment: a laptop with GPRS card, GPS, compass, geologist's hammer, microphone and digital camera. The software included in the system was: SAGISc, Olympus Camedia®, Notepad®, Freehand®, Sound Recorder® and MSN® Messenger. See Figure 8.

The IGM team had the following artifacts: computer with Internet connection, sounds cards, ArcPad®, MSN® Messenger, SAGISc, scanner and telephone. See Figure 9 for details.

Fig. 8. Element of field team using artifacts (laptop, hammer, compass, GPS, etc.)

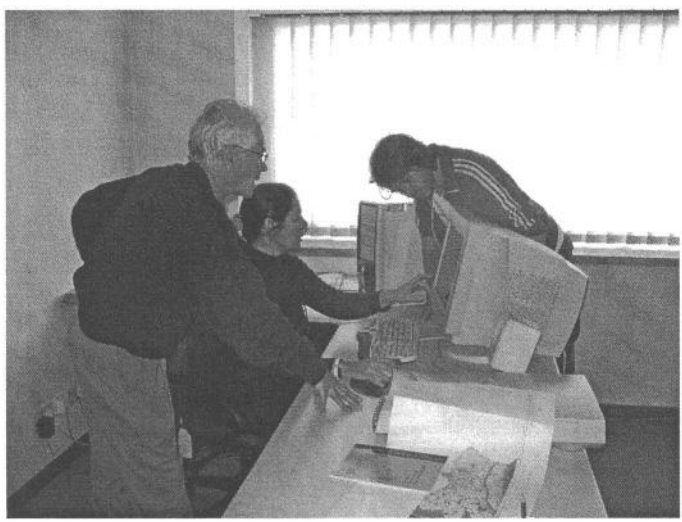

Fig. 9. The IGM team working around the e-book at the IGM main office

5.1 Preliminary Evaluation Results

The participants in the preliminary experiment were all individually interviewed in order to identify the system strengths and weaknesses. The obtained results indicate that the system was easy to use and helpful, especially because of the component supporting communication between teams. The expeditious way to locate points and insert information related with these points was also positively considered.

Generally, the system has worked well and met the users' expectations. A few problems arose in communications, because one of the points was on a cliff close to the shore and originated losses in communications. The participants also mentioned that the user validation and file exchange with MSN® Messenger suffered from slow communications.

The portability and, in some situations, the usability of the equipment were also considered negative factors.

In what concerns collaboration, the participants had a very positive experience, especially at the second point where, through the exchange of messages the users were able to identify the type of geology of the area.

5.2 Results from the Broader Evaluation

This evaluation process was carried out by questioning a panel with over 30 specialists in Geo-sciences from IGM. The evaluation involved a detailed demonstration of ArcPad® and SAGISc, and a reproduction of the preliminary experiment, showing photographs, messages exchanged, the produced e-book and how both teams cooperated. After these explanations the participants were requested to fill up an individual questionnaire.

5.2.1 Results

The evaluation focused on the work activities of the panel, experience with IT and experience with field work. We requested an evaluation of the "new system," consisting of three new components: ArcPad®, SAGISc and the combination of both (ArcPad®+SAGISc). We also requested the panel to compare the "new system" with the traditional method of doing field work. We used a Likert [7] classification scale, varying between 1 and 5 (Bad, Low, Average, Good and Very Good).

After an analysis of the questionnaires, the following points were highlighted:

a) About the experience with IT and field work

As can be seen in Figure 10, the distribution of the level of experience with computers is median, biased to the right, with this distribution: 50% of the respondents considered themselves as Average and 30% Good. Statistical values: Mean = 3.3; Standard Deviation = 0.9.

Concerning the experience with field work, it can be seen that the panel is more at ease, since 63% of the participants consider themselves Good and 27% Very Good. Statistical values: Mean = 3.8; Standard Deviation =1.1.

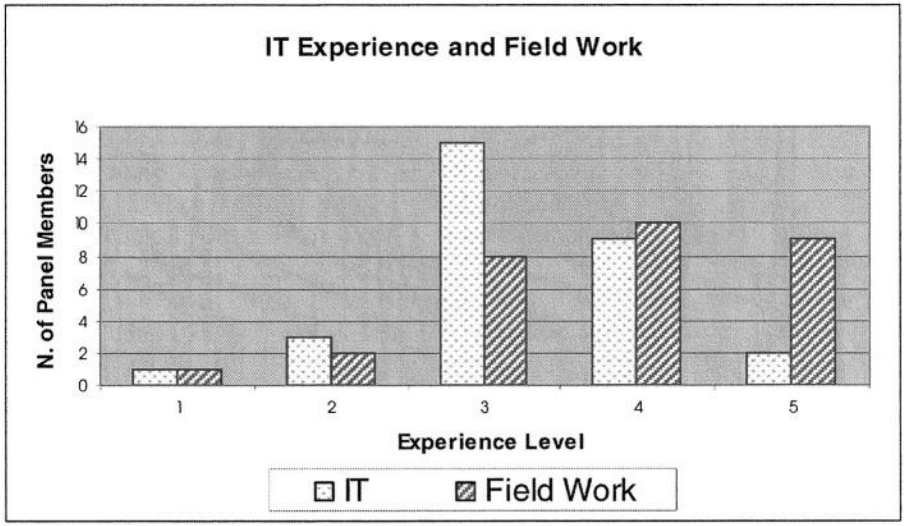

Fig. 10. Experience with IT and field work

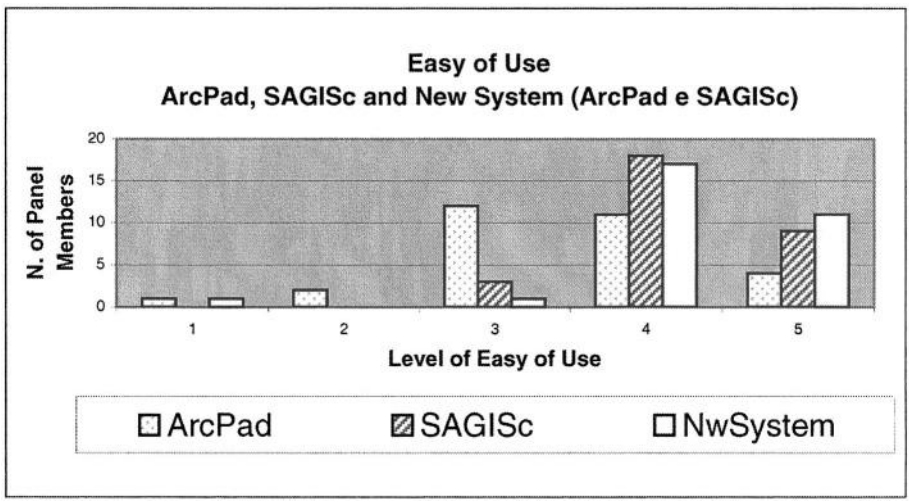

Fig. 11. Ease of use

b) Evaluation of the "new system"– ease of use

Generally the tools under evaluation were all considered to be easy to use (Figure 11), with a large number of answers in level 4 – 60% panel members rated SAGISc as easy to use, while 57% had the same appreciation about ArcPad®+SAGISc (NwSystem). Statistical values: ArcPad®: Mean = 3.5; Standard Deviation = 0.9. SAGISc: Mean = 4.2; Standard Deviation = 0.6. ArcPad®+SAGISc: Mean = 4.2; Standard Deviation = 0.8.

c) Comparison of methods – Traditional field work vs. "new system"

We used the following questions to compare both approaches:

In relation to a traditional geological data gathering, this system:
a) Will make your work easier?
b) Will make the work developed by IGM easier?
c) Will make the exchange of opinion easier?
d) Is it quicker to get a second opinion?
e) Is there and increase in observation/problem analysis abilities?

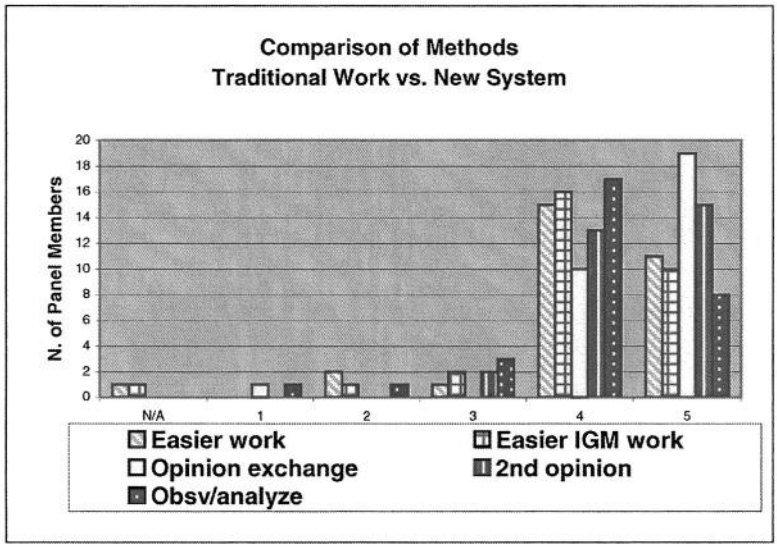

Fig. 12. Traditional field work vs. "new system"

In general, the panel considered the "new system" Good or Very Good (Figure 12). However, 63% and 50% panel members evaluated the "new system" as, respectively, Very Good for easing the exchange of opinions and speed at obtaining second opinions, respectively. As a side note, one respondent did not answer (N/A) the first and second questions. Statistical values: Easier work: Mean = 4.2; Standard Deviation = 0.8. Easier IGM work: Average = 4.2; Standard deviation = 0.7. Opinions exchange: Average = 4.5; Standard Deviation = 0.8. 2nd opinion: Mean = 4.4; Standard Deviation = 0.6. Obsv/analyze: Mean = 4; Standard Deviation = 0.9.

d) Evaluation of components of the "new system"

In this question the panel was asked to rank the "new system" components from the worst to the best. 57% panel members pointed out that the ergonomic aspects of the equipment were the worst component of the system (Figure 13). We should note that this answer was based only on the equipment presented to the panel: a laptop plus

several peripherals. 43% respondents preferred the communications between teams and gave the maximum classification to that component. The "new system" had an evaluation of Average and Good from 33% and 30% respondents, respectively. The SAGISc was evaluated as Average and Good by 30% and 27% of the respondents, respectively. 37% panel members considered ArcPad® below SAGISc and the "new system."

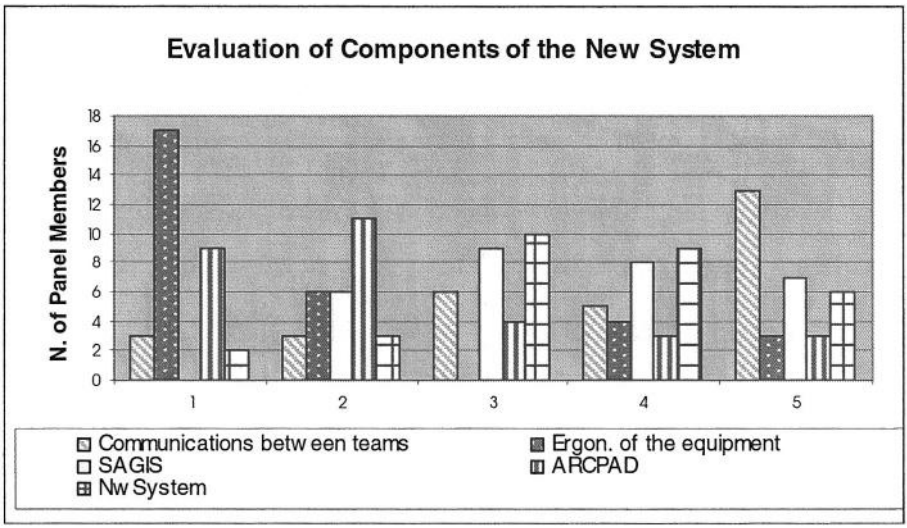

Fig. 13. Evaluation of components of the "new system"

5.2.2 Analysis and Conclusions from Evaluation

The results obtained from the panel indicate an easier usage of the components that integrate the collaborative system, particularly SAGISc, which was considered the easiest to use by 60% of the panel members.

Comparing with the traditional method, the "new system" was considered to be a great development; facilitating data exchange (63% of the respondents rated Very Good) and access to a second opinion (50% rated Very Good).

7% of the respondents considered that the "new system" is negative (Low) in facilitating field work. Nevertheless, about 50% of the respondents considered that field and office work will be facilitated by this system (Good). The Mean value for making the work easier is 4.2 and Standard Deviation is 0.8; concerning the work done in IGM being improved, the Mean and Standard Deviation are respectively 4.2 and 0.7.

The increased capacity of observation/analysis is considered positive by 57% of the respondents (the Mean is 4 and the Standard Deviation is 0.9). 43% of the respondents pointed out that the communications component is the best aspect of the "new system." The evaluation of SAGISc and "new system" revealed very similar results, since the respondents seem to had some difficulties in distinguishing one from the other.

The worst component of the "new system" was ergonomics (57% of the answers), which is coincident with the results obtained in the preliminary evaluation.

This evaluation also revealed that ArcPad® did not achieve a good acceptance. It was considered the worst component of the system by 30% of the respondents (Bad), while 37% rated it Low.

In general, the "new system" had a good acceptance and was considered to be very useful to the work developed by Geologists; although more ergonomic equipment is required to be properly used in the field. The communications also need to be improved both in performance and reliability.

6 Related Work

The research in geo-collaboration is very recent and there are few papers published on this subject in the scientific literature. Essentially, work in this area can be separated in two different categories: one centered in technology and another in human geo-collaboration.

In the first category we should account for systems like Open GIS [5] and COPA [13] that study different ways to integrate scattered information with geo-referenced information. These systems do not directly support geo-collaboration but explore structured solutions that afford such functionality.

Still in the first category, we should highlight the efforts in the development of synthetic environments for geographic visualization (geo-visualization) [8; 9]. In this perspective, geo-collaboration refers to the support of information exploration by various users inside synthetic environments. In this case, fieldwork is not considered.

In the second category, related with human geo-collaboration, we find two projects that are currently being developed [11; 12], both concerned with the problem of interacting with geo-referenced information, in digital format and during field work. These projects study the integration of contextual information (photos, etc.) with geo-referenced information, and mention that they are studying several ways to navigate through information gathered on the field. Concerning these projects, only preliminary information is currently provided, without any experimental results.

Still in the second category, [6] studies the impact of geo-collaboration in work practices. Contrary to the work reported in this paper, the goal was centered in the access to remote databases. [10] supports the coordination between geo-collaborators, developing an integration system between GIS, workflow systems and other unstructured tools (meeting support, argumentation and discussion forums). Once again, fieldwork is not discussed.

In summary, we consider that the project described in this paper is about a subject that is not well studied yet: the human aspects of geo-collaboration, in a decision-making environment supporting fieldwork. In this context, we identified the main design issues that should be considered, leading to the development of geo-collaborative artifacts. And we also presented the results on usage of such artifacts, as well as their perceived utility for future users.

7 Conclusions and Future Work

In this action-research project we analyzed the process actually being used by IGM to gather geological data and produce geological cartography. We identified a problem in this process, basically that it takes a long period of time to do it. We studied the possibility to develop a collaborative system with the objective of reducing process time and costs.

The followed approach used several concepts from the Contextual Design methodology [2], which fundamentally focus on the need to understand work processes through contextual inquiry and, from there, derive system design.

With this objective we accompanied IGM field technicians doing geological data gathering. After these observations, we proceed to the construction of descriptive models of work processes and specification of a collaborative system for geological data gathering. The obtained system integrates several tools, from which we would like to highlight two: SAGISc and ArcPad®. The first tool was developed by this project and the second is a commercial GIS tool.

The functionality of SAGISc is centered in the support to a digital artifact that emerged in the contextual inquiry phase: the field book, an artifact where the technician gathers several types of data, related to geological work. The developed prototype allows sharing information on this field book and collaboration between remote and local work teams.

The proposed system was evaluated by specialists in Geo-sciences. According to the obtained results, the collaboration component is the most positive one, the reason being the possibility of decreasing the duration of geological data gathering. On the opposite, the most negative factor concerns communication problems (performance and coverage of GPRS), as well as ergonomic problems (because there are a lot of cables, weight and volume of laptop). The SAGISc tool was considered to be easily understood and usable.

Concerning future work, the system architecture will be modified for Internet usage, where the teams will be able to work over the Web. This development will facilitate the work of Geo-science specialists and will support the simultaneous collaboration of several field teams (currently only one is supported).

The participants in the evaluation process made several suggestions, which are appropriate for future work:

- Allow a geo-referenced point to have more than one associated photo and drawing;
- Include an ortophotomap field in the e-book;
- Include a date field in the e-book;
- Create a library of symbols to use over photos and/or ortophotomap;
- Remove the sound field, since some specialists have the opinion that the sound is too distorted and does not allow to precisely understand the type of geology being studied.

References

1. ARCPAD Http://www.esri.com/software/arcpad/index.html.
2. Beyer and Holtzblatt K.: Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann. (1998)
3. Esri Http://www.esri.com/software/index.html.
4. Fagrell, K. Forsberg, and J. Sanneblad: FieldWise: A Mobile Knowledge Management Architecture. In Proceeding of the ACM 2000 Conference on Computer Supported Cooperative Work, ACM Press, Ed. Philadelphia, pp. 211-220. (2000)
5. Gardels: Open GIS and on-Line Environmental Libraries, SIGMOD Record, vol. 26, no. 1. (1997)
6. Hope, T. Chrisp, and N. Linge: Improving Co-Operative Working in the Utility Industry Through Mobile Context Aware Geographic Information Systems. In Proceedings of the Eighth ACM International Symposium on Advances in Geographic Information Systems. Washington, D.C. (2000)
7. Likert Http://trochim.human.cornell.edu/kb/scallik.htm.
8. MacEachren, R. Edsall, D. Haug, R. Baxter, G. Otto, R. Masters, S. Fuhrmann, and L. Qian: Virtual Environments for Geographic Visualization: Potential and Challenges. In Proceedings of the 1999 Workshop on New Paradigms in Information Visualization and Manipulation in Conjunction with the Eighth ACM Internation Conference on Information and Knowledge Management. Kansas City, Missouri. (1999)
9. Manoharan and M. Manoharan: A Collaborative Analysis Tool for Visualisation and Interaction with Spatial Data. In Proceedings of the 19th Annual Conference on Computer Science. San Antonio, Texas. (2002)
10. Medeiros, J. Souza, J. Strauch, and Pinto G.: Coordination Aspects in A Spatial Group Decision Support Collaborative System. SAC 2001. Las Vegas (2001).
11. Nusser, L. Miller, K. Clarke, and Goodchild M.:Digital Government: Geospatial IT for Mobile Field Data Collection, Communications of the ACM, vol. 46, no. 1. (2003)
12. Pinto, S., Souza J., Strauch J., and Marques C.: Spatial Data Integration in a Collaborative Design Framework, Communications of the ACM, vol. 46, no. 3. (2003)
13. Touriño J., Rivera F., Alvarez C., Dans C., Parapar J., Doallo R., Boullón M., Bruguera J., Crecente R., and González X.: COPA: a GE-based Tool for Land Consolidation Projects. In Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems. Atlanta, Georgia. (2001)

Blind to Sighted Children Interaction Through Collaborative Environments

Jaime Sánchez, Nelson Baloian, and Tiago Hassler

University of Chile, Department of Computer Science
Blanco Encalada 2120, Santiago, Chile
{jsanchez,nbaloian,thassler}@dcc.uchile.cl

Abstract. In this paper we present a software system implementing a remote synchronous collaborative "Battleship" game that can be played by two persons. The system provides two different interfaces, one to be used by sighted people and the other to be used by blind people based on spatialized sound, thus allowing sighted and blind people play against each other, without knowing if the adversary is a sighted or blind person. The paper also presents a framework which supports the synchronisation of heterogeneous applications sharing only some common objects. This is the key for developing collaborative applications with very different interfaces as it is shown in this work.

1 Introduction

In the recent times, many efforts have been done in order to integrate people with disabilities into the digital world [6]. Deaf people have not so many problems using the standard software interfaces as most of them rely principally on visual information. For blind people the situation is much more disadvantageous. In order to create software for them it is necessary to design other ways to interact with the computer. Applications conceived for the blind user have been developed using auditory information as the main output channel and haptic devices for input [1, 3, 4, 7, 8]. These systems have been principally developed to help blind people to overcome their difficulties with standard interfaces such as Web-pages "readers" e.g. Jaws. Others focus on the development of 3D audio interfaces used to develop the user's skills to recognize spatial environments through sound. However, we have no record of research work on collaborative environments including people with disabilities performing blind-blind and blind-sighted interactions.

The WISIWYS paradigm has been very important in the development of synchronous collaborative applications but we cannot apply it in this context. We need to implement different types of interactions since people with different ways of perceiving the environment need specific strategies to establish a common ground.

In this research work we tried to make a first attempt to study the challenges and issues that arise when trying to develop and use collaborative systems for people with different perceptual abilities. To do this we implemented a computer-based version of the famous board game Battleship. This game is a good metaphor for implementing such a collaborative system. On the one hand, when playing the game blind users will train their abilities of spatial memory. On the other hand, it allows the interaction between blind and sighted people without knowing each of them the condition of the other, thus allowing an interaction without prejudgments.

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 192–205, 2004.

Developing collaborative systems for integrating people with different capabilities poses a higher challenge because of the synchronization of different interfaces for each type of user. Interfaces for the sighted user normally consist of computer graphical user displays and some sound for the output. The keyboard and mouse are the most common devices for input. Interfaces for blind users rely principally on auditory information for output and keyboards, and other haptic devices for input [9]. Sometimes joysticks are used for both input (movement) and output, but they pose some resistance in order to simulate the contour objects.

This research study presents the design, development, and usability testing of AudioBattleShip, a sound-based interactive and collaborative environment for blind children. This system is a similar version of the traditional battleship game for sighted people but including both a graphical interface for sighted users and an audio-based interface for blind people. AudioBattleShip provides a game environment to enhance group interaction, abstract memory, spatial abstraction, and haptic perception in blind learners. This study also introduces a platform which supports programming of distributed applications, especially for the case of synchronizing applications having different interfaces.

To design and develop information-equivalent interfaces (see [12]) for sighted and blind people we followed a similar process such as the one described in [2] for developing interfaces for people with disabilities.

Finally, a full usability study has been implemented to evaluate the cognitive impact of interacting with AudioBattleShip, showing that blind learner collaboration and cognition skills can be enhanced through the interaction with spatialized sound.

2 The AudioBattleShip Design

2.1 Playing Modes and Phases

As the aim of this game is to engage blind people in a collaborative environment to help them develop cognitive skills we decided to provide various playing modes. AudioBattleShip can be played blind to blind and blind to sighted. Blind to blind mode presents the same interfaces to both players. Blind to sighted mode provides a variety of tools to the blind learner to minimize the disadvantages in comparison with sighted learners who can have in any moment snapshots of the state of actions.

AudioBattleShip has three phases (see Figure 1):

1. *Ship positioning phase:* the player chooses the position of ships taken from a predefined set on the battlefield. A matrix represents a battlefield where ships can be placed over a column and a row by covering different numbers of spaces according to the type of ship.
2. *Creating and joining session phase:* After placing the ships, a player can choose between creating a new session of the game, joining an existing one, and playing against the computer. If a new session is created, another player can join it. To do this the player has to know the session's name and the host address.
3. *Shooting phase:* By taking turns both players can manipulate a matrix with the board's state by identifying a cell of the matrix representing the contender's battlefield for dropping a bomb on that place. The system responds whether a contender's ship was hit or not.

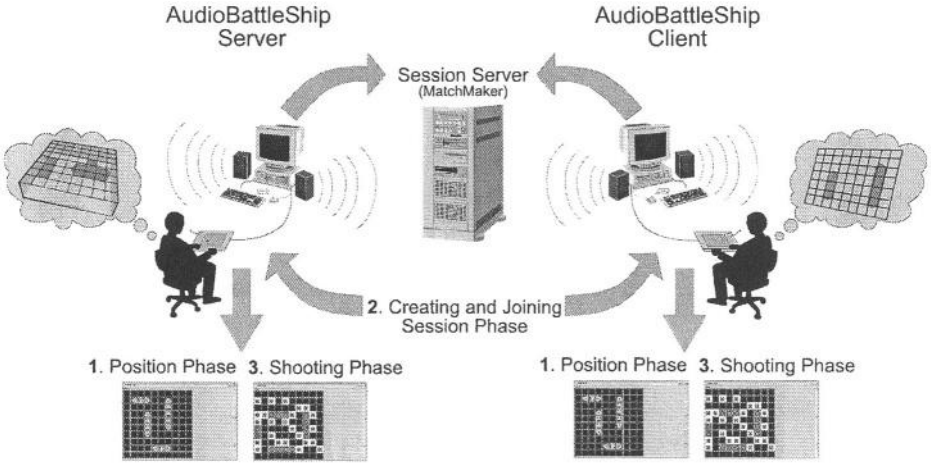

Fig. 1. Phases of the AudioBattleShip game.

2.2 Interface Design for a Prototype

In [2] authors propose a model for developing learning software for people with different disabilities by using diverse interfaces. This model can be briefly summarized in the following steps: 1. Development of the computational model of the real world; 2. Modeling the input/output needs for the system (including feedback); and 3. Development of interfaces by "projecting" the model over a range of user's interaction capabilities.

In this case, we modeled AudioBattleShip by mapping the two players, each with a matrix representing the space where their own ships are positioned. A player's matrix will also register the information about where the contender has dropped a bomb. A token represents whose player has the turn for dropping a bomb in the contender's field.

We created two different applications, one for blind and one for sighted users. Each application has two interfaces, one for the ship positioning phase and the other for the shooting phase. The ship positioning interface for sighted people consists of a window containing a grid representing the battlefield. Ship positioning takes place by defining a position and direction (up, down, left, right) with the mouse. The shooting phase interface consists of a window displaying two matrices: one for the battlefield where the player's own ships are positioned, and the other representing the contender's field, showing the places where the player has already drooped bombs and the outcome of this operation. A text area was provided for informing about the course of the game.

The application for blind users uses a tablet as input device (see Figure 2a). The tablet can map the entire screen and by using a pen-based pointing device diverse mouse events can be triggered. A grille was built over the tablet in order to represent the matrix of the battlefield and some additional "help buttons" for triggering actions. During the ship positioning phase the player has to point to a certain place on the grid and indicate the direction by moving the pointing stick to a cell up, down, left, or right. During the shooting phase the player has just to double-click over a certain cell

of the grid (see Figure 2b). Sound feedback is provided to inform about a specific spatial location on the board and the occurrence of certain actions, for example, the outcome of dropping a bomb in a cell of the contender's battlefield. Help buttons assist the player in remembering the places where an enemy's ship was hit or not.

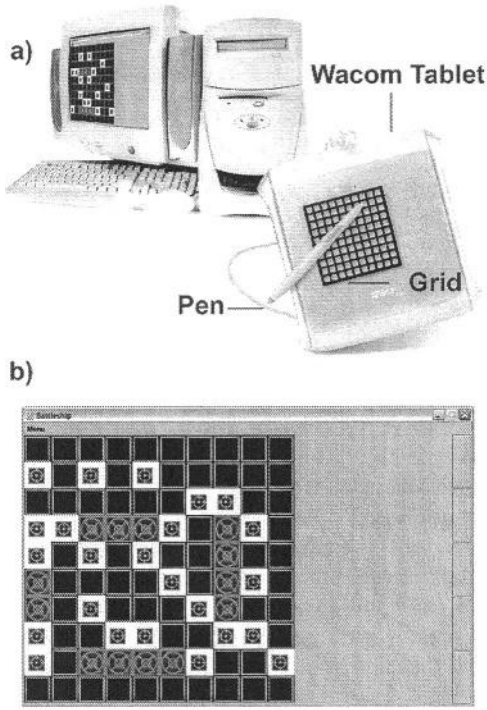

Fig. 2. Input devices (a) and shooting phase (b) for blind players.

3 A Platform for Replicated Collaborative Applications

Building collaborative software, a developer has to decide a mechanism for making a distributed application. Basically, the developer has to decide between two paradigms. One of them is to select a framework with a centralized server who is capable of serving several clients. In this case, the information needed by all clients is first sent to the server and then somehow broadcasted to the clients. In these scenarios, e.g. NetMeeting [13], it is not possible to work with the clients any longer if the server is for some reason not accessible anymore. The second paradigm is a replicated architecture. Here all the necessary data for clients is replicated in the client applications. The advantage of this kind of framework is that clients are still operable even if they loose their network connection.

The MatchMaker TNG [11] framework, which is based on RMI, combines both paradigms for making collaborative software. First of all it has a centralized server,

which manages to send the needed data to the clients, and also replicates the whole internal data structure of the application. By replicating the data structure it is possible for each application to have a comparative synchronized status.

Within this server application the data structure is arranged as a tree called the "synchronization tree". It is not necessary for every client to listen to the whole tree but clients may also listen to certain sub-trees. Most likely, the synchronization tree will reflect the internal data structure of the application as shown in Figure 3.

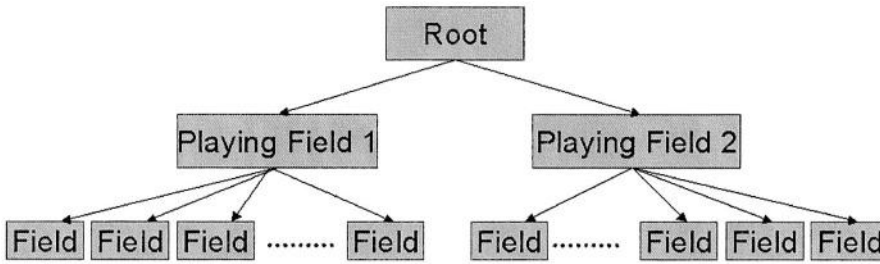

Fig. 3. The synchronization tree for AudioBattleShip.

A great deal of frameworks for synchronization use event-based technologies [5], but MatchMaker TNG replicates according to the Model-View-Controller paradigm, by implementing a so-called MatchMaker TNG model for every object to be replicated. This model consists of the object model but may also have information about the view.

By using these models the possibility of interpreting them differently in diverse applications is achieved. For example a model may be fully interpreted by an application that is running on a PC with high performance, but some pieces of information may not be considered if the same model is interpreted by an application running on a PDA.

Another example is the use of the model in AudioBattleShip. Here it is possible to integrate information needed for people with visual disabilities in the model. This could be ignored by an application that is running for sighted people.

The models are propagated to the clients using the common event architecture of Java. The server has four basic operations to inform the clients about changes in the synchronization tree when a new node is created, deleted, changed, or activated. Every MatchMaker TNG client has to implement a special method for each of these four operations.

3.1 Application Programming with MatchMaker TNG

MatchMaker TNG is a framework that is developed to easily implement synchronization in collaborative applications with heterogeneous interfaces. The developer has to be sure that the application is capable of creating, joining, and leaving MatchMaker TNG sessions as well as updating the status of the application.

The first thing a developer has to do is to define models for synchronizing objects. The model has to implement java.io.Serializable, otherwise it cannot be sent to them over a network.

The next thing to do is decide which classes should be responsible for keeping the application synchronized. Those classes have to implement the SyncListener interface that traces the classes as a listener for the MatchMaker TNG server. By implementing the interface, the developer has to develop four methods:

```
public void objectChanged(SyncEvent event)
public void objectDeleted(SyncEvent event)
public void objectChanged(SyncEvent event)
public void actionExecuted(SyncActionEvent event)
```

Once writing the listeners, the developer has to make sure that they are added to the right part of the synchronization tree. A listener will be informed about changes that happened in a part of the registered synchronization tree. It is not necessary that all listeners be added to the root element of the synchronization tree. Most likely this will not be the case; usually the listener will only be added to a certain part of the synchronization tree.

After writing and adding the listeners, the developer has to guarantee that the application updates the server status if necessary. Therefore it is possible to read the whole tree or even partial trees from the server, perform changes on them, and write them back. It is also possible to create, change, delete, or execute an action on a specific part of the tree.

3.2 The AudioBattleShip Implementation

Since the interfaces of both applications are very different the most convenient way to synchronize them is at the data model level. AudioBattleShip has objects that store the information of players, the state of the game (turns), and the coordinates of a given point with the corresponding state (ship, water, destroyed ship, shooting to water). By using these objects a synchronization tree is created with nodes that store information about the players containing objects with the corresponding size of the board.

According to this, the MatchMaker synchronizing tree has the turn-taking token at the root of the tree. The information about the two players is stored as two child nodes of the root. Each cell representing the battlefield of a player is stored as a separate child node from the player's node (see Figure 3). This has the advantage that only the information of one cell will be transmitted to the coupled application if the information changes, but not the whole matrix. Synchronizing applications using sound have also an additional problem: the sound playing is usually implemented as a separate thread in most programming languages (and Java is not the exception). In many cases, we must be sure that some auditory information has been completely heard by the user before continuing with the program execution (which may mean to play another sound clip). For this, Java Media Framework was used to synchronize multimedia events. AudioBattleShip contains a directory of sounds that can be easily modified to change language and sounds.

4 Testing Collaboration

AudioBattleShip was usability tested in both during and after software implementation. Three usability experiences were designed. Two of them were centered on the

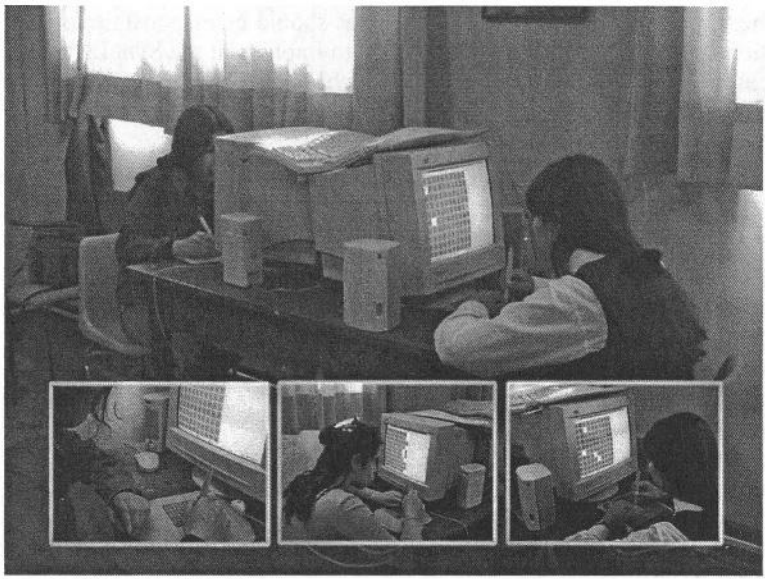

Fig. 4. On site interaction with AudioBattleShip between two blind learners.

usability of the interface provided, and the last testing was concerned the cognitive impact of using AudioBattleShip.

4.1 Interface Usability

The interface of AudioBattleShip was evaluated by legally blind learners (blind and low/residual vision). The first experience consisted of testing different software modules with four blind learners that regularly attend the "Santa Lucia" School in Santiago, Chile. After finishing each software module we tested it with learners. They interacted with the software, make comments, and answered questions. Thinking aloud methods were also implemented. All of these procedures helped designers to improve the software's mapping of the blind learner's behavior.

The first prototypes were fully tested. The interface design and functionality were analyzed. Four blind learners from the same school participated in the study. They played the game and answered questions during four sessions. Two people from the research team observed them, asked questions, and registered their comments and behaviors in all sessions. Most questions were related to the audio interface and the mapping and use of input devices such as the tablet. We tested how well the concrete matrix of the traditional board game Battleship was mapped into the interactive environment. The mapping of the screen matrix by a tablet with a rough grille on it using a synthetic material was also evaluated. We also tested the usability of some interface elements such as sounds (intensity and quality), volume, and feedback to the user actions.

The tasks followed by learners were to explore AudioBattleShip, understand the structure, rules, and adaptations made, as well as the differences between the concrete

Battleship board and the digital version of the game. Users also learned how to play the game by using both the keyboard and tablet. They explore fully the pros and cons of using a tablet, its functions and analogies with the AudioBattleShip software interface. We looked for audio stimuli association to game actions and the understanding of cause-effect action relationships when interacting with the virtual environment (see Figure 5).

4.2 Blind to Blind Interaction

As a result of this preliminary testing, learners enjoyed interacting with AudioBattleShip. They helped us to redesign the software by making insightful and meaningful contributions about sound synchronicity, sound overlapping, sound help, color contrast, size of the cursor, position identification, and tablet mapping.

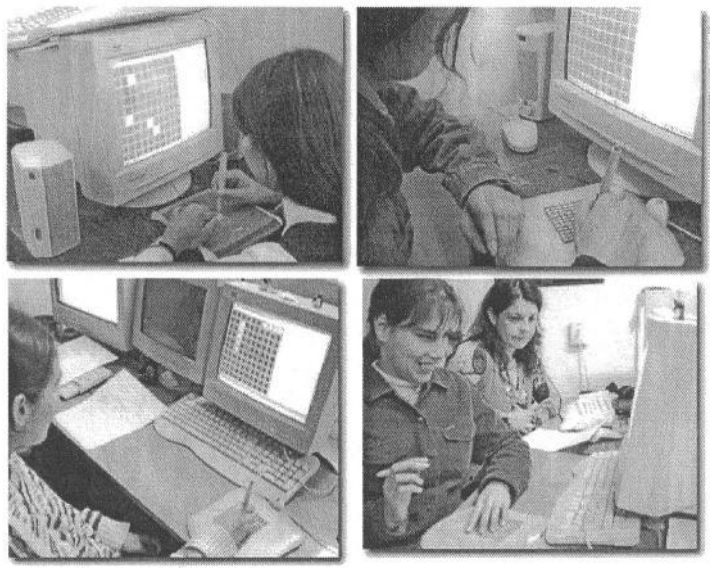

Fig. 5. Interface usability testing by blind learners.

We tested the cognitive impact of AudioBattleShip with five learners during six sessions of ninety minutes each. All sessions were captured and registered using a video digital camera. The comments made and the answers to specific questions were also registered. Two special education teachers observed blind learners doing tasks with AudioBattleShip using the tablets and filled observation forms specifically designed for this purpose. We developed a form for each cognitive task containing several statements with a four answers scale and open observations criteria.

The sequence of actions followed by learners included to navigate AudioBattleShip, to identify different audio stimuli, to explore both the concrete board of the traditional game Battleship and the Wacon tablet, to play AudioBattleShip, and finally to make concrete representation models of the navigating space and position moves made by using clay and Lego (see Figure 7). Children interacted with their partners

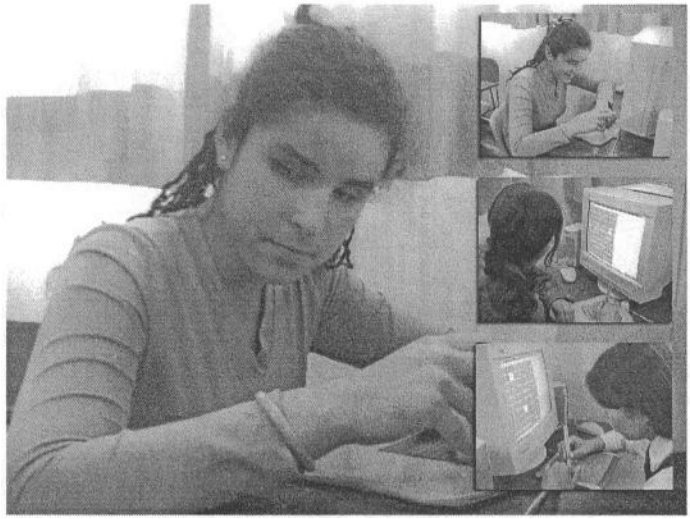

Fig. 6. Evaluating the cognitive impact of using AudioBattleShip.

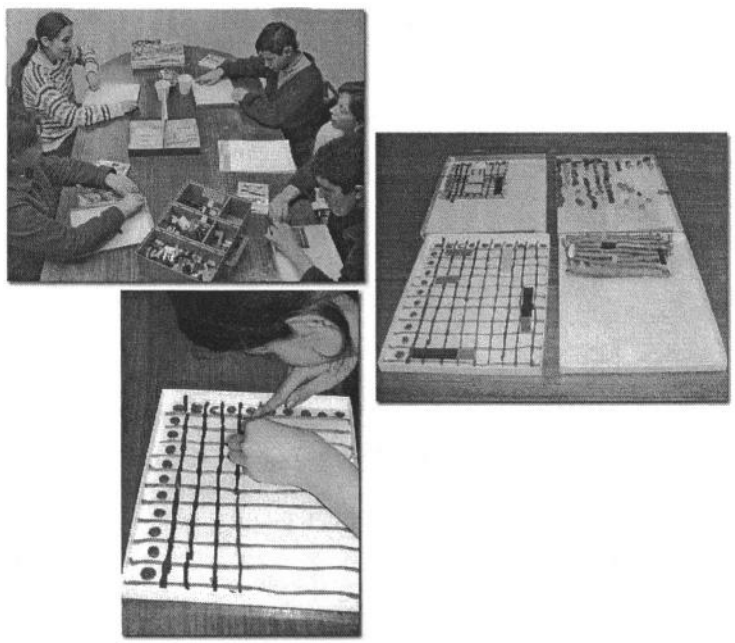

Fig. 7. Concrete representation models of the navigating space and position moves.

and then represented the moves of the adversary in a concrete board. One player described the coordinates and other localized the pieces and marked them on the board. They also searched the ships of the adversary by using a strategy previously designed (see Figures 4 and 6).

We looked for the impact of using AudioBattleShip on abstract memory through spatial references with sound, spatial abstraction through the concrete representation of the virtual environment, and haptic perception (tactile and kinesthetic) by generating mental images of the navigated space. We also observed how they integrated spatial references through sound and haptic references through the tactile manipulation to construct mental images of the navigated virtual environment.

Task one was concerned to integrating spatial references through sound and haptic references through the tactile manipulation to construct mental images of the navigated virtual environment. We also wanted to check if by using AudioBattleship learners can perceive the virtual space through haptic perception by generating mental representations. Most learners could clearly differentiate each audio stimulus. Four out of five learners could make an excellent analogy between AudioBattleShip and the concrete board of the traditional Battleship game. The same number could make an analogy between the Wacon tablet and the virtual environment of the adversary. All learners did not need a concrete board to orientate their movements and navigation throughout AudioBattleShip (see Figure 8). Two learners performed excellent against an adversary and three did a good performance. The same number was obtained when played against the computer.

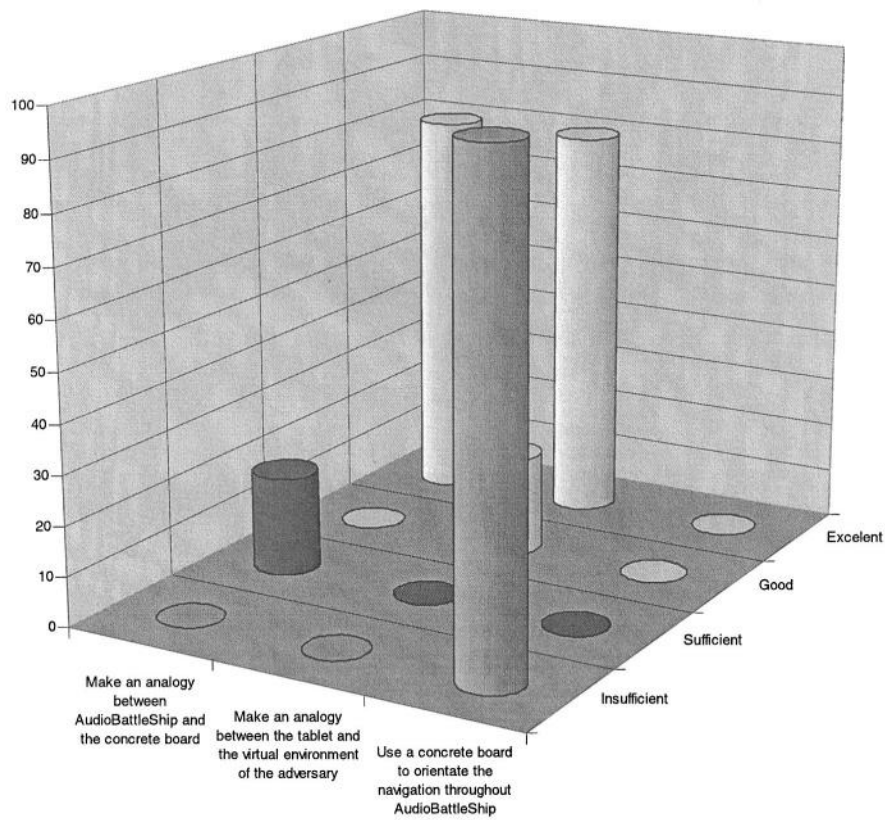

Fig. 8. Mental images and haptic perception through using AudioBattleShip.

Task two measured abstract memory through spatial references with sound and spatial abstraction through the concrete representation of the virtual environment. Four out of five learners could completely represent in a concrete board the moves of the adversary. All of them did not need to use a concrete board to orientate the navigation throughout AudioBattleShip (see Figure 9).

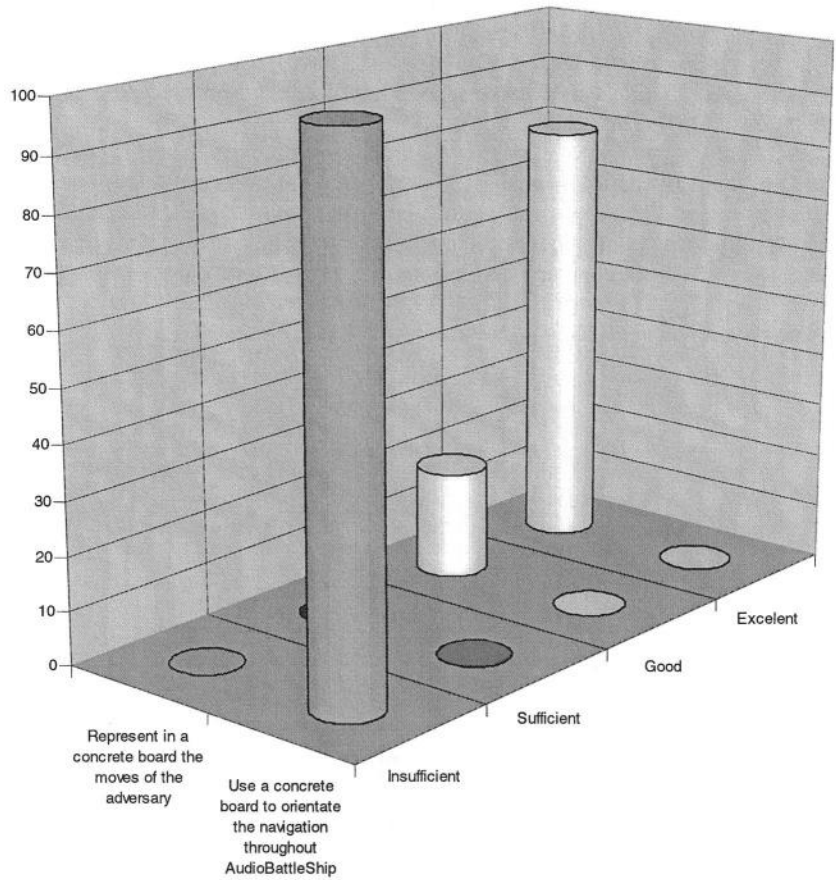

Fig. 9. Abstract memory and spatial abstraction through AudioBattleShip.

Task two also included a specific activity related to detect how well learners could develop a strategy to overcome the adversary. Four out of five could design and implement a strategy to win the game. All of them could orientate easily within the matrix implemented through a rough grille on the Wacon tablet.

4.3 Blind to Sighted Interaction

We also designed a setting to observe the interactive behavior displayed by users with different sensory capacities when playing AudioBattleShip. The setting included

playing blind to low vision children, blind to sighted children, and low vision to sighted children. All of them had previous experience with the board game Battleship by doing specific tasks during the interface usability testing.

We observed the behavior of each of the five legally blind players (totally blind and low/residual vision) and interviewed them after they ended the game. Then we analyzed the data obtained. Most players used the tablet adequately but the totally blind children had some difficulties to orient themselves within the tablet. Actually, they initially blamed the tablet for loosing the game arguing that it was not big enough to recognize the cells.

Children were able to perceive through audio stimuli if a ship had been shrunk by the opponent and could discriminate when it happened. Totally blind and low vision children could mentally mapped the game. All children were motivated when interacted and collaborated each other. Blind and low vision children were motivated to play again the game even when they lost. The motivation of blind children was related to winning or loosing, mentioning that it was hard for them to play. Interesting was to observe that for both the sighted and the blind children, the experience was entertaining and challenging. Although sighted children performed better it was not a boring experience for them and they were motivated to play AudioBattleShip again.

Sighted and low vision children used some strategies to win the game. Totally blind children used trial and error type of behavior when playing. Sighted and low vision children could decode and describe the strategies used by the opponent to win the game. Children with low vision made suppositions that were not always true. Blind children could not decode the strategies of the opponent.

From interviews we got some interesting data. When we asked them about the reasons why they won or lost the game there were no difference in the arguments given by blind and sighted children, such as the strategies implemented and the capacity to orient and not loosing the positions. At the beginning blind children argued that their opponents won because they can see. After we told them that they were not looking at the screen to play, their answers were very similar to the sighted users. All of them thought that the opponents were good players by highlighting the strategies used.

5 Final Discussion

This research work aimed at analyzing collaborative interaction of children with diverse visual capabilities in a virtual game. We showed that it is possible to design computer-based collaborative games where blind and sighted children can interact in a motivating and challenging way.

In this paper we have presented the design, development, and usability testing of a sound-based interactive environment for blind children. AudioBattleShip is an interactive version of the board game Battleship, providing different interfaces for both sighted and blind people. The interface for blind people is based on spatialized sound as a way of navigating and exploring through the environment. Sighted people can use either an interface based on graphical output or an interface based on sound, as it was the case of our tests. The application was programmed using a framework which supports the development of distributed heterogeneous applications by synchronizing only some common objects, thus allowing the easy development of interactive applications with very different interfaces.

AudioBattleShip was tested for collaborative tasks between children with different visual capabilities. We wanted to know if users with different visual abilities can interact to each other by playing the same game with different interfaces. We found out that the use of AudioBattleShip can help to develop and rehearse abstract memory through spatial reference, spatial abstraction through concrete representations, haptic perception through constructing mental images of the virtual space, and cognitive integration of both spatial and haptic references.

There were diverse forms of interaction among blind learners. Some collaborative abilities can be enhanced by using interactive sound-based applications such as AudioBattleShip. We need to know in more detail what type of skills can be stimulated and ways of improving them through the interaction with sound. This is especially important for blind learners because they are accustomed to do individual work with little social interaction when using digital devices. We envision constructing new ways and scenarios for collaboration between blinds as well as between blinds and sighted learners by using digital audio interfaces such as the one introduced here.

Learners enjoyed the interaction with AudioBattleShip. Even the sighted children enjoyed playing against children with low and without vision. This can be explained because this interaction allowed them to compete and measure their competences and abilities. This experience also had an affective side. They did improve their self-esteem that helped them to compete and win to other learners with and without vision.

Our data indicate that blind learners using AudioBattleShip mediated with cognitive tasks and concrete materials can develop and exercise mental images of the space, haptic perception, abstract memory, and spatial abstraction.

Blind learners showed more interest to play with a sighted partner. They were motivated to play and interact with sighted learners because it was a way to show that they can make the same performance as sighted if minimal conditions are met at the beginning.

A future retest study will give us more detailed insights and knowledge about how cognition and collaboration can be enhanced by interacting with sound. We will contrast this quantitative data to evaluate more fully the real contribution of audio-based interfaces in cognition and collaboration.

The use of Wacon tablets as interaction devices for AudioBattleShip was a critical strength. It allowed blind learners to interact with different interface haptic devices and demonstrated to be highly appropriate for this type of software.

Our research work indicates that blind children can benefit when interacting with sighted children. In the same way, sighted children can see this experience as challenging. Initially blind children thought that due to the fact that sighted children can see they can have better strategies to play the game. As the game went on they figured out that sighted children did not have available a graphical interface and this motivated them to improve their strategies and challenging themselves.

The observations made by learners, their comments, and meaningful ideas were critical to improve AudioBattleShip. This validates one of our premises in a sense that interactive software environments based on sound should be designed with and for blind learners in order to map their needs, interests and characteristics.

Finally, we have validated our hypothesis that it is possible to design and develop collaborative software for blind children to interact between them and with sighted users. This interaction is a motivating, challenging, and constructive experience which helps them to pursue higher goals and expectancies. Moreover the interaction with

sighted people increases their self-teem thus helping blind children to actively integrate them to the society.

Acknowledgment

This report was funded by the Chilean National Fund of Science and Technology, Fondecyt, Project 1030158.

References

1. Baldis, J. Effects of spatial audio on memory, comprehension, and preference during desktop conferences. Proceeding of the ACM CHI '01 Vol 3, 1, (2001), 166-173.
2. Baloian, N.; Luther, W.; & Sánchez, J.: Modeling Educational Software for People with Disabilities: Theory and Practice. Proceedings of the ASSETS 2002 Conference, 8.-10. July 2002, Edinburgh.
3. McCrindle, R. & Symons, D. Audio space invaders. Proceedings of the Third International Conference on Disability, Virtual Reality and Associated Technologies, (2000), 59-65
4. Mereu, S. & R. Kazman. Audio enhanced 3D interfaces for visually impaired users. Proceedings of CHI '96, ACM Press., (1996).
5. Jansen, M., Pinkwart, N. & Tewissen, F. (2001). MatchMaker - Flexible Synchronisation von Java-Anwendungen. In Klinkenberg, R., Rüping, S., Fick, A., Henze, N., Herzog, C., Molitor, R. & Schröder, O. (eds.): LLWA 01 - Tagungsband der GI-Workshopwoche "Lernen-Lehren-Wissen-Adaptivität" October 2001. Forschungsbericht 763. Universität Dortmund, Germany.
6. Newell A. F. Assets: where do we go from here?. Proceedings of The Fifth International ACM SIGCAPH Conference on Assistive Technologies ASSETS 2002, pp.1-3, Edinburgh, July 8-10
7. Sánchez, J. Interactive 3D sound hyperstories for blind children. Proceedings of ACM-CHI 99, (1999), 318-325.
8. Sánchez, J. Interactive virtual acoustic environments for blind children. Proceedings of ACM CHI '2001, pp. 23-25. Seattle, Washington, April, (2001), 2-5
9. Sjostrom, C. Using haptics in computer interfaces for blind people. Proceeding of the ACM CHI '01, Vol 3, 1, (2001), 245-246.
10. Tan, H. Haptic interfaces. Communications of the ACM, 43(3), (2000), 40-41.
11. Tewissen, F., Baloian, N., Hoppe, H.U. & Reimberg, E. (2000) "MatchMaker" – Synchronising objects in replicated software architectures. In Proceedings of CRIWG 2000 (Sixth International Workshop on Groupware), Madeira, Portugal, October 2000, pp. 60-67. Los Alamitos: IEEE Press.
12. Veijalainen, J. & Gross, T.: Mobile Wireless Interfaces; in Search for the Limits. In: Developing an Infrastructure for Mobile and Wireless Systems; König-Ries, B.,Makki, K., Makki, S.A.M., Pissinou, N., Scheuermann, P., (Eds.) NSF Workshop IMWS 2001, Scottsdale, AZ, October 15, 2001, Revised Papers. Springer Verlag LNCS Vol. 2538, Dec. 2002, pp. 153-163.
13. http://www.microsoft.com/windows/netmeeting/

Implementing Stick-Ons for Spreadsheets

Shermann S.M. Chan and José A. Pino

Depto. de Ciencias de la Computación, Universidad de Chile
Av. Blanco Encalada 2120, Tercer Piso, Santiago, Chile
{schan,jpino}@dcc.uchile.cl

Abstract. Spreadsheet systems were initially intended for individual use. Collaborative use implies there should be a local versioning mechanism. As such, it is proposed to use a modified Stick-On. This means to define them to cope with the wide variety of data types, which can be associated to spreadsheet cells. The proposed design includes a hierarchy of these new Stick-Ons, a Field Dependency Graph and a Peer Referencing Mechanism.

1 Introduction

Spreadsheets were initially intended for individual use. Later improvements allowed some group facilities. However, current systems do not allow a group of synchronous users to have local versioning control. Suppose, e.g., a small group of business analysts trying to adjust a company budget for next year. One analyst can make a proposal. Another analyst may be looking at the same shared view and wishes to change one of the assumptions, i.e., the value on one of the cells. If he changes it, perhaps many other numbers (depending on it through formulas) will change as well, including the final financial result. A third analyst may then wish to compare results and also see all the intermediate figures for both proposals. The spreadsheet should be able in this case to keep both versions and allow users to easily see them. Notice it is not trivial to define what a local version is and how to manage it.

The focus of this paper is to characterize local versions [1] for spreadsheets and study their implementation. We base our work on a concept proposed by Pino [9] for handling local versions for text, called Stick-On. From a user's point of view, a Stick-On is a virtual piece of paper which can be written; removing the piece of paper lets see what is written under it. Technically, Stick-On holds an alternative version for the text that is covering. The rest of the paper is organized as follows. Section 2 presents background information on the subject. Section 3 presents the System Architecture we are considering. Section 4 contains the analysis and design of the Stick-Ons and their internal structure and manipulation. Finally, section 5 presents the conclusions.

2 Background Information

Spreadsheets may be considered general-purpose programming languages [10]. However, some problems may become awkward to solve with spreadsheets, such as those

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 206–214, 2004.

involving much data. On the contrary, spreadsheets seem adequate to solve problems related to computations specified in formulas. Spreadsheet systems also include an interesting visual interface: values are presented on a bi-dimensional matrix. It may be drawn to resemble the balance sheets used by accountants. By careful design of the user interface, people believe to be just "filling data" when they are actually programming the spreadsheet system. Thus, many final users apply them to make computations on business data, results from lab experiments, simulation data, etc.

Nevertheless, spreadsheets have problems [7]. One of the problems is users lose context of related cells when working on a separate area of the matrix; Morrison et al. [4] have proposed a code inspection approach to reduce error due to this difficulty. A second problem of spreadsheets is the insufficient feedback to the user on cell dependencies: a cell may affect another cell because it is used in the formula defining the contents of the latter one. The user indirectly sees this dependency only when he explicitly asks for the display of that formula. This problem will motivate us to propose a field dependency graph. Another spreadsheet shortcoming concerns local versioning [1], as mentioned in the introduction. Our proposal will be based on Stick-Ons. Stick-Ons were originally proposed as alternative local versioning mechanisms for text to version columns [5], flexible diffs [6], and active diffs [3]. The original local versioning mechanisms do not seem appropriate to be used with spreadsheets. Stick-Ons could be re-defined for spreadsheets, but they should be very different: There are cells, they are dynamic and the contents may depend on other cells.

3 System Architecture

The architecture of our proposed collaborative authoring tool for spreadsheet applications is multiple-client/single-server, as shown in Fig. 1. Collaborative clients can form logical and dynamic groups for specific tasks. Each client can start his spreadsheet application, in which a plug-in of our proposed system is embedded. Three main client-components in the system are: a Peer Referencing Mechanism (PRM), a search engine, and an update engine. PRM has an intelligent agent, which links the Stick-On Hierarchy and the Field Dependency Graph (FDG) Hierarchy together (ref. section 4). The search engine retrieves data from the server's Spreadsheet DataBase (SSDB) and stores the data to the client-side as a temporary copy; whereas the update engine synchronizes the data from client-side to server-side.

When a user starts the spreadsheet application and the embedded system, he should be authenticated and authorized by the User Authentication and Synchronization (UAS) component, which is located at the server side. UAS consists of three parts: Authentication (Login), Concurrency Control (CC), and Access Level Permission (ALP). The Authentication component checks the user profiles, which are stored in the User DataBase (UserDB). Once a user is allowed to enter the system, he will be provided with a user-level. Any action he does (e.g. search or update) is monitored by the CC and the ALP components. Users may have various roles within a group (e.g. clerk, manager). Thus, user-levels are necessary to control their activities and behaviors (e.g. whether the user is able to invite other users to join a group for a specific

task, and whether the user is able to access some confidential documents). The CC component is to ensure data consistency and user concurrency among collaborative clients. The ALP component is used to determine whether the data requested by the client is permissible between the server's SSDB and the client's local copy.

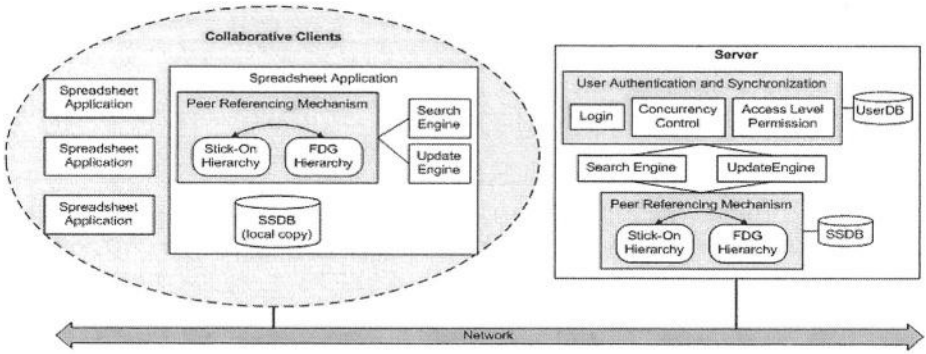

Fig. 1. System architecture.

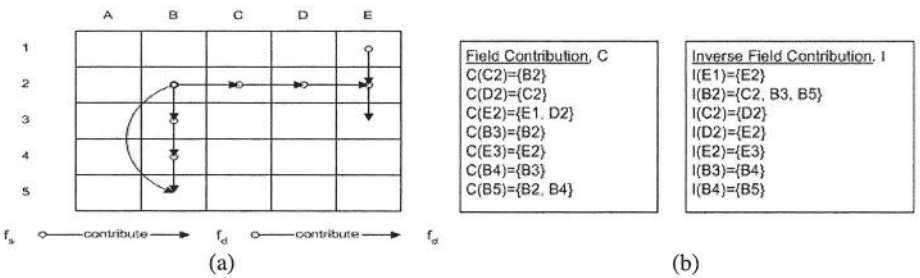

Fig. 2. A sample scenario of the Field Dependency Graph.

4 Internal Structure

4.1 Field Dependency Graph (FDG)

A Complete FDG. Fig. 2 shows a sample scenario of the FDG of a spreadsheet. According to dependency relationships, fields can be classified into two types: Dynamic Field or Static Field. A static field, f_s stores a static value and it may contribute its value to the dynamic fields; a dynamic field, f_d may in turn, contribute its resultant value to other dynamic fields. In general, dynamic field values are defined by mathematical formulas. In the case shown in Fig. 2(a), the definitions used to construct the FDG are as follows: Dynamic Fields, $F_D=\{C2, D2, E2, B3, E3, B4, B5\}$, where $f_d \in F_D$, Static Fields, $F_S=\Phi-F_D$, where Φ is the set of fields in the spreadsheet. Of course, $F_D \cap F_S = \emptyset$. In Fig. 2(b), the left-hand box shows the Field Contribution function, C (strictly speaking, C is a relation); while the right-hand box shows the Inverse

Field Contribution function, I. The C function accepts a field f, as the parameter and outputs the fields making a value contribution to f. In contrast, the I function accepts a field f as the argument and outputs the fields for which f makes a value contribution. These contribution/dependency relationships are stored in the FDG.

A Partial FDG. Sometimes, users may want to have a look on the FDG of a particular field. Fig. 3 shows a partial FDG generated by a user request. The highlighted box, E2 is the central field and the partial FDG is extracted from the original FDG (ref. Fig. 2(b)) as shown in Fig. 3(b). In addition to the structure of the FDG, we also concentrate on the design of a good graphical user-interface that will be provided to the users in the sense of HCI.

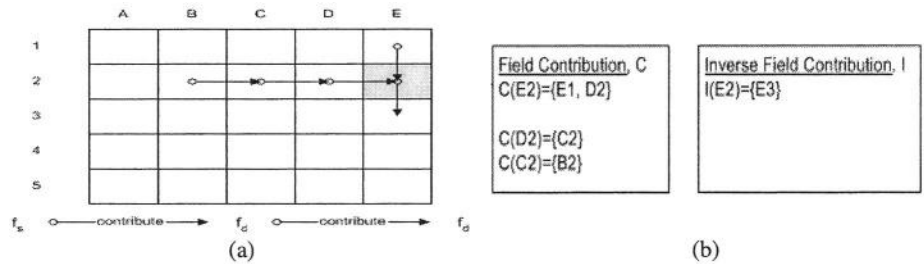

Fig. 3. A sample scenario of a partial Field Dependency Graph.

4.2 Stick-On Hierarchy

Stick-Ons in word processing are intended to hide words and sentences behind other text pieces. Words and sentences are replaced in fixed areas, i.e., the shapes of Stick-Ons are extensions of the rectangle concept [8, 9]. However, if the Stick-On is implemented on a spreadsheet, it will replace words, sentences, numbers, formulas, graphs, and pictures, i.e. everything that can be attached to a field. Hence, Stick-On Spreadsheet is more complicated than Stick-On Word. Besides group editing of a *spreadsheet document,* Stick-On Spreadsheet can also be used for data versioning by groups of people, e.g. various parameters obtained from the same type of experiment done by many persons. Thus, our problem is *how to glue a Stick-On on a spreadsheet*?

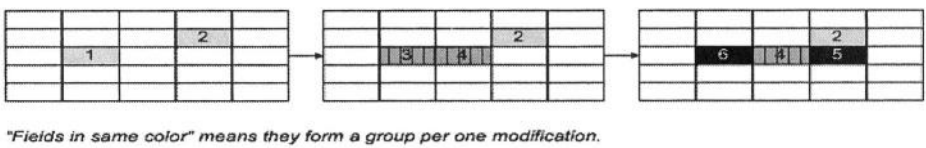

Fig. 4. How to glue Stick-Ons on a spreadsheet.

How to Glue Stick-On on a Spreadsheet. We have two basic requirements for the design of a Stick-On: (1) efficient to change from one version to another, and (2) no

unnecessary duplication of content. Thus, we choose the following approach: to allow "random" fields to form a group for one change. This would be more complicated than the approach proposed by Fuller et al. [2] but it will not cause unnecessary data duplication. However, extra data structure and storage are needed to be designed and modeled. Fig. 4 shows the look and feel of our approach to users. Fields grouped in light gray color belong to the first Stick-On; fields grouped in dark gray belong to the second one; and fields grouped in black belong to the third one. Using this approach, at the last state, users can see the editing work more clearly. Of course, a history of all versions of Stick-On should be available when a cursor moves over the field.

How to Manage the Stick-On Hierarchy. When a user changes a value of a field, his decision is based on the current outlook of the spreadsheet. That means the content of the fields are related to each other. For example, the change of the content of {3, 4} shown in Fig. 4 depends on its latest view of the data, i.e. the content of {1, 2}. From the concept of Stick-On, users can detach any Stick-Ons for the cleaning process[1]. So, what happen if {1, 2} are detached from the spreadsheet? In addition, if the field is a static field, which has contributed its value to a dynamic field (e.g. a formula), is it necessary to group these related fields together to the Stick-On?

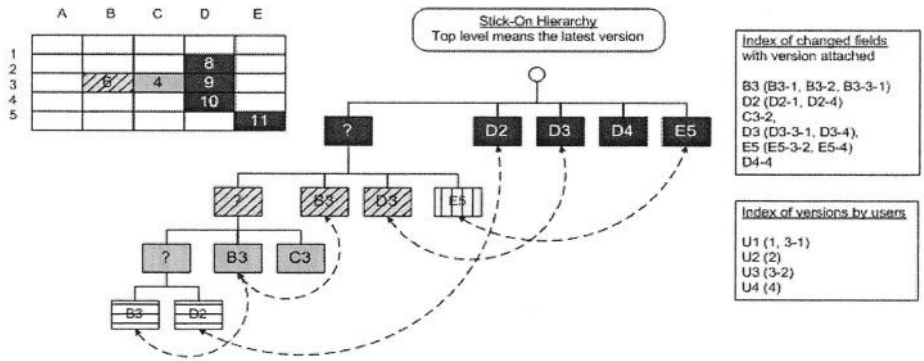

Fig. 5. Attaching five Stick-Ons to the Stick-On Hierarchy.

Attaching Stick-Ons. Concerned with these problems, we propose a Stick-On Hierarchy, along with the help of the FDG, to form a Peer Referencing Mechanism. In order to classify which fields belong to which groups, a simple approach is to provide a "version" number. The design of the version number can represent a simple group, and with an extra hierarchy to indicate the super-sub relationships (i.e. the Stick-On Hierarchy). With these relationships, Stick-Ons can be attached one on top of the others and it will simplify users to change from one version to another.

Fig. 5 demonstrates how to attach Stick-Ons to the hierarchy. The left-hand sided diagram shows the layout of the spreadsheet and the middle one shows the Stick-On Hierarchy, together with two indexing structures at the right-hand side. The first Stick-On is created with two fields involved (i.e. D2 and B3) and it is attached to the

[1] "Cleaning" is the operation of removing all the visual cues of Stick-Ons [9].

hierarchy (the bottom level). In order to improve the retrieval time, the fields and the user information are indexed. Totally, there are five Stick-Ons attached in Fig. 5. The top level is the last attached Stick-On. As one of the fields of the second Stick-On (i.e. B3) overlaps one of the fields of the first Stick-On, there is a link crossing level one and level two in the hierarchy. A special case is the attachment of the fourth Stick-On, this Stick-On is attached to level three of the hierarchy (rather than the fourth level). There are two restricted requirements to do so: (1) the fields of this Stick-On should not overlap with the fields of the previous Stick-On, (2) there are no dynamic fields involved in this editing work (i.e. this Stick-On shares the same FDG with its previous Stick-On). Even though these two Stick-Ons seem to have no relationships, but according to our assumption, the last Stick-On does somehow depend on its instant "ancestor". Consequently, the content of this Stick-On still exists (i.e. the content will be replicated) even after the detaching or cleaning process. Therefore, the sequence of these two Stick-Ons in this level should be maintained explicitly.

```
DetachStickOn
 Input:
  version: integer   # version of the Stick-On
  level: integer     # level of the Stick-On
  max_level: integer # max level of the Stick-On Hierarchy

 begin
  if level<>max_level
   foreach stickon s in level do
    if s.version=version
     foreach field f in s do
      if f with links to its instant upper-level
         do nothing
      else
         copy f to its instant upper-level
      end-if
     end-foreach
    end-if
   end-foreach
  end-if
  restore all links
  regenerate index tables
 end.
```

```
CleanStickOn
 Input:
  version: integer   # version of the Stick-On
  level: integer     # level of the Stick-On
  max_level: integer # max level of the Stick-On Hierarchy

 begin
  if level<>max_level
   foreach stickon s in level do
    if s.version=version
     foreach field f in s do
      if f with links to its instant upper-level
         do nothing
      else
         copy f to its instant upper-level
      end-if
     end-foreach
    end-if
   end-foreach
  end-if
  bridge the levels {level-1, level+1}
  delete the stickon by version
  restore all links
  regenerate index tables
 end.
```

Fig. 6. Algorithms for *DetachStickOn* and *CleanStickOn.*

Detaching Stick-Ons from the Hierarchy. As detaching is a temporary process, it allows users to redo/undo any process before the permanent cleaning process. Thus, the system will keep a record of all editing work before the cleaning process. Fig. 6 shows the DetachStickOn algorithm. During this process, the Stick-On involved will be shaded only; so that this Stick-On can be "re-attached" to the hierarchy later. In order to improve the retrieval process, the links between the fields and the index tables will be re-generated after the detaching process. Note that some fields will be copied to their instant upper-level because the content from their upper levels may depend on these fields. The copied fields will be marked for future reference.

Cleaning Stick-Ons. Fig. 6 presents the algorithm for the cleaning process. During the cleaning process, Stick-On involved will be removed permanently. As there will be a gap between two levels in the hierarchy, the links between the fields and the index tables will be re-generated after this process. Note that some fields will be copied to

their instant upper-level because the content from their upper levels may depend on these fields. However, the copied fields will not be marked for future reference.

4.3 The Peer Referencing Mechanism

In previous sections, the management issues of Stick-On such as attaching, detaching, and cleaning processes, are simplified by replicating the dependent fields to their instant upper levels. In this section, we deal with how to identify such dependent fields. Before start using Stick-On, user has to design the usage of spreadsheet. Thus, he may enter some values, formulas, and/or some editable content to the fields. After his preliminary design of his spreadsheet, a FDG will be generated by his content if there is any field with dependency. Thus, an agent process can utilize the FDG to determine which (mandatory) fields are needed to be replicated onto the Stick-On Hierarchy, if applicable. Of course, some semantics linking different fields cannot be reasoned out by the agent, and so, a functionality must be provided for users to make their preferences on field dependency by themselves. In one case, static value of a field (e.g. static field) may be changed to a formula (i.e. dynamic field). Hence, FDG is also needed to maintain its versions with the Stick-On Hierarchy simultaneously.

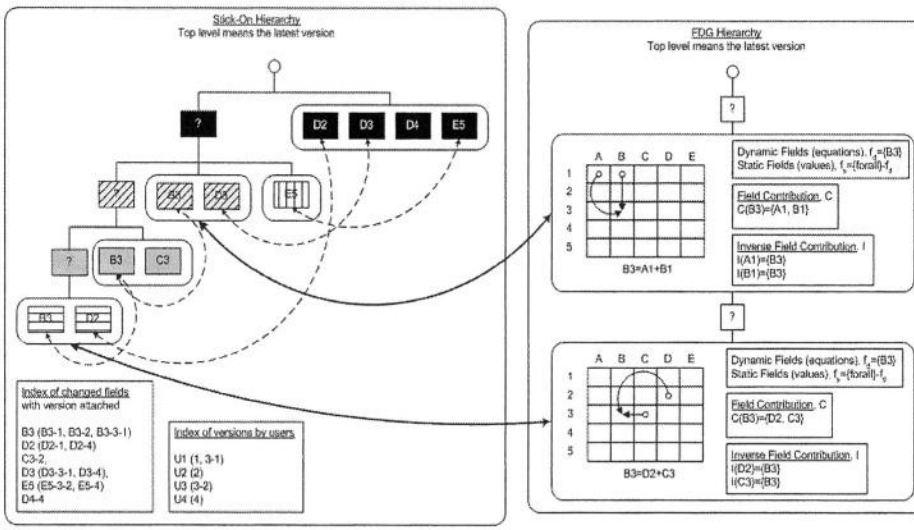

Fig. 7. The Peer Referencing Mechanism.

Fig. 7 shows the relationships between the Stick-On Hierarchy and the FDG Hierarchy by the Peer Referencing Mechanism (PRM). When a user starts a spread-sheet application, if the spreadsheet involves the use of dynamic fields (e.g. formulas), a FDG will be generated and attached to the FDG Hierarchy. Therefore, when the user changes the value of the field D2, the agent of the PRM will reason out there is a change to the field B3 (i.e. change value of a dynamic field). As a result, a Stick-On will be created and attached to the Stick-On Hierarchy, with a reference linking to the

FDG Hierarchy. Then, he changes the value of field C3, so the agent would include C3 and B3 to the second Stick-On. As there is no change to the field dependency, Stick-On of this level remains referencing to the bottom-level FDG. However, when the user changes the content of the dynamic field B3 (e.g. change of a formula), a new FDG will be re-generated and attached to the third level of the FDG Hierarchy. Thus, the agent should consider whether there is a value-change (i.e. spreadsheet calculation) or content-change (user's modification) of a dynamic field. In this case, the static field D3 attaching together with the dynamic field B3 in the same Stick-On may be due to the user's preference. If the user explicitly states there are dependencies between two or more fields (e.g. static vs. static), the FDG should cater for this situation.

5 Conclusion

Local versioning for spreadsheets has been proposed. The concept of Stick-On was extended to do it. In the process, we had to define the FDG, which may be useful by itself, as it makes clear cell dependencies. Perhaps, making this graph available to users will be appreciated by them. However, the versioning mechanism will allow collaborative work. Users may discuss figures, make proposals, state results, etc. without deleting contributions from other users. The Stick-On Hierarchy allows going back and forth between various versions. Versions may also be detached or cleaned. We can expect a group of users to have a richer discussion when using a versioning mechanism than without it, because their use will not "destroy" the data previously contributed by other group members. This was exactly what Sharples et al. asked for in the case of text co-authoring: "mechanisms to allow alternative documents to be developed (and thus for conflicting viewpoints to be expressed)" [11]. A forthcoming paper will deal with the collaboration issues: roles, scenarios, usability and evaluation.

Acknowledgments

This work was supported by Fondecyt (Chile) grant No. 1040952 and MECESUP (Chile) project No. UCH0109.

References

1. Vitali, F.: Versioning Hypermedia. ACM Computing Surveys, 31(4):24, (1999)
2. Fuller, D., Mujica, S., Pino, J.A.: The Design of an Object-oriented Collaborative Spreadsheet with Version Control and History Management. Proc. ACM Symposium on Applied Computing, Indianapolis, IN USA (1993) 416-423
3. Minor, S., Magnusson, B.: A Model for Semi (A)synchronous Collaborative Editing. Proc. ECSCW'93, Milano Italy (1993) 219-231

4. Morrison, C., Morrison, J., Melrose, J., Wilson, E.: A Graphical Approach for Reducing Spreadsheet Linking Errors. Advanced Topics in End-User Computing. Idea Group Publishing, Hershey, PA USA (2003) 173-189
5. Neuwirth, C., Kaufer, D., Chandhok, R., Morris, J.: Issues in the Design of Computer Support fro Co-authoring and Commenting. Proc. CSCW'90, USA (1990) 183-195
6. Neuwirth, C., Chandhok, R., Kaufer, D., Erion, P., Morris, J., Miller, D.: Flexible Diff-ing in a Collaborative Writing System. Proc. CSCW'92, Toronto Canada (1992) 147-154
7. Panko, R.: What We Know About Spreadsheet Errors. Journal of End-User Computing 10(2), (1998) 15-21
8. Pineda, E., Pino, J.A.: Stick-Ons Revisited. Proc. CRIWG'0l. Germany, 6-8 Sept. (2001) 26-35
9. Pino, J.A.: A Visual Approach to Versioning for Text Co-authoring. Interacting with Computers 8(4), (1996) 299-310
10. Rommert, J.C.: Real Programmers Don't Use Spreadsheets. ACM SIGPLAN Notices 27(6), (1992) 10-16
11. Sharples, M., Goodlet, J., Beck, E., Wood, C., Easterbrook, S., Plowman, L.: Research Issues in the Study of Computer Supported Collaborative Writing. In Sharples, M. (ed.): Computer Supported Collaborative Writing, Springer-Verlag, London (1993) 9-28

Empirical Evaluation of Collaborative Support for Distributed Pair Programming

Jesus Favela[1], Hiroshi Natsu[1], Cynthia Pérez[1], Omar Robles[1], Alberto L. Morán[2], Raul Romero[1], Ana M. Martínez-Enríquez[3], and Dominique Decouchant[2]

[1] Departamento de Ciencias de la Computacion, CICESE, Ensenada, Mexico
{favela,hnatsu,orobles,cbperez,romero}@cicese.mx
[2] Laboratorire LSR, Grenoble, France
{Alberto.Moran,Dominique.Decouchant}@imag.fr
[3] Depto. de Ing. Electrica, CINVESTAV-IPN, D.F., México
ammartin@mail.cinvestav.mx

Abstract. Pair programming is an Extreme Programming (XP) practice where two programmers work on a single computer to produce an artifact. Empirical evaluations have provided evidence that this technique results in higher quality code in half the time it would take an individual programmer. Distributed pair programming could facilitate opportunistic pair programming sessions with colleagues working in remote sites. In this paper we present the preliminary results of the empirical evaluation of the COPPER collaborative editor, developed explicitly to support pair programming. The evaluation was performed on three different conditions: pairs working collocated on a single computer; distributed pairs working in application sharing mode; and distributed pairs using collaboration aware facilities. In all three cases the subjects used the COPPER collaborative editor. The results support our hypothesis that distributed pairs could find the same amount of errors as their collocated counterparts. However, no evidence was found that the pairs that used collaborative awareness services had better code comprehension, as we had also hypothesized.

1 Introduction

Pair Programming is a software development technique that is part of the Extreme Programming methodology. In pair programming two developers work side by side, on a single computer, to jointly produce an artifact (design, algorithm, code, etc.). It has been reported that this technique can be accounted for the development of higher quality software in half the time it required a single programmer [1,8].

The two programmers work as a unit, as a single mind responsible for all aspects of the artifact. One programmer, the driver, controls the pen, mouse, or keyboard to write the code. His colleague actively observes the work produced by the driver, looking for defects, alternatives and considering the implications of the strategy being followed. The pair changes roles periodically. Both participants are active throughout the process and share the responsibility for the work being produced [7].

The successful application of this technique requires the use of an appropriate workplace: "The programmers should be able to sit side by side and program, looking simultaneously at the computer screen, sharing the keyboard and the mouse" [7]. While they work together, the couple should be able to share the keyboard without

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 215–222, 2004.

changing seats. Extreme programmers need to be constantly in touch with their team and for this, their workspace has to be open and facilitate communication among peers as well as casual and informal encounters.

An important trend in software development has been the globalization of the software industry [2]. With increased frequency, software developers are required to work in groups that are geographically distributed. Under these circumstances the requirement for pair programmers to be in the same location seems an important limitation of this approach. For this reason we have developed the COPPER synchronous collaborative writing tool, designed specifically to support pair programming among distributed collaborators [5]. In this paper we report the preliminary results of an empirical evaluation performed to determine the feasibility of distributed pair programming and the services offered by COPPER in this regard.

2 The COPPER Pair Programming Editor

COOPER is a synchronous collaborative editor designed to support pair programming [5]. It is a based on a client-server architecture and composed of two main subsystems: the Collaborative Editor, and the User and Document Presence module (UD&P). On the client, these subsystems form an application used to write programs, access document services and communicate with peers.

The editor can be used disconnected from each other (in individual mode) or connected in pairs (in synchronous collaborative mode). Users can be either co-located or distributed on the Internet. The editor implements a turn-taking synchronous editor (see Figure 1). The user holding the floor (or editing right) uses the editing window (Figure 1F) to work on the currently loaded document.

Common document management and editing functionality is provided by means of the application‘s Menu bar and the toolbar (Figure 1A). Actions performed in the editor are propagated to the collaborator’s client. As an example of their use, consider the Open button (third button left to right) of the toolbar. When pressed, an Open dialog (Figure 1G) appears, providing access to documents stored in the Document server. This dialog presents an integrated file hierarchy with documents from the local machine and from other distributed WebDAV servers [6]. This allows for seamless navigation and document retrieval from the individual or collaborative work environment. Several documents can be edited at the same time, even if these documents come from different WebDAV servers.

The Document server offers a centralized information repository, which provides document storage, editing access control, user authentication and permissions to avoid unauthorized accesses, and document presence extensions through an instant messaging client, which “listens” and informs the UD&P system the results of operations performed on the documents [4].

Each editor client “owns” the documents opened by its user, and it is the only one allowed to perform operations, such as *Save, Save As,* and *Close,* on these documents. When the connection is broken, each client keeps their “own” documents, and the user can continue working on them in individual mode. While in this mode, clients are ready to send or receive invitations to begin collaborating.

Floor management (or editing access control) is represented using a “traffic light” metaphor (Figure 1B), which is activated when working in collaborative mode. This component includes an action button to request and grant floor control.

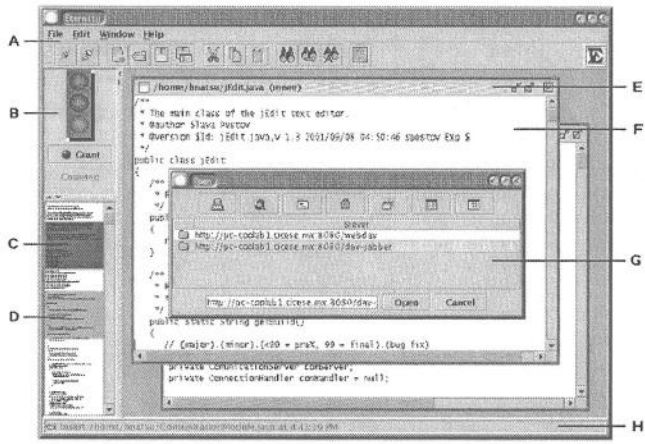

Fig. 1. The COPPER Pair Programming System.

Collaboration awareness is provided by means of a radar view (Figures 1C and 1D), editing window titles (Figure 1E), the status bar (Figure 1H), and the floor or editing control access component (described earlier). The radar view provides a general overview of the document being edited; it shows document changes in real time, and serves as a document navigation tool; selecting a particular line of the document in the radar view causes the current editing window to display the segment of the document where the selected line is present.

Editing window titles (Figure 1E) display the name and location of the document, as well as the identities of the owner of the document and of the collaborator (if present).

The status bar (Figure 1H) provides information on the last operation performed by any of the collaborators, as well as on the date and time it was performed.

The client-side module of the UD&P subsystem is used to send and receive messages to/from collaborators, manage the user's own presence and provide this presence information to other collaborators. Additionally, it extends this functionality to interact with the documents in a similar way as one would interact with users [4]: adding or deleting documents from the document (presence) list, and sending "group" messages to subscribed users of a document. Furthermore, subscribed users receive messages from the UD&P service whenever the document's availability and status information changes. Finally, this module implements the functionality offered by WebDAV, to perform basic editing operations on subscribed documents (e.g. locking a document to avoid users from concurrently modifying the same document).

3 Experimental Design and Procedure

We conducted an experiment to explore the potential and limitations of distributed pair programming. To guide our research we established two working hypothesis:

H1: Distributed pair programmers working remotely will find the same number of defects as collocated programmers in the same amount of time.

H2: Distributed pair programmers using COPPER will have better code comprehension as distributed pair programmers using application sharing in NetMeeting.

The first hypothesis aims at establishing that pair programming can be as successful when the group is distributed as when they are collocated, if appropriate technical support is provided. The second hypothesis is established from our believe that tools like NetMeeting, which offer limited collaboration awareness services are not adequate to support the intense level of interaction required for synchronous tasks such as software programming or design.

The subjects of the study were 12 graduate students in computer science and proficient in the Java programming language. The students were randomly grouped in 6 pairs. All subjects attended a one hour session that included an introductory lecture on pair programming, an explanation of the COPPER editor and time for hands-on experience with the system. In addition, the subjects completed a questionnaire which focused on their previous programming experience.

We used a within-subjects design; with all six groups asked two perform three different programming tasks in three different setups.

3.1 Experimental Setup

The modality in which the subjects performed the task was our independent variable. The experiment required both subjects to collaborate in performing three different programming tasks, each task was performed under a different condition.

Collocated Condition. In this condition, the two programmers were in the same office and shared a single computer running the COPPER editor in single-user mode. The programmers shared the display and keyboard as in traditional pair programming tasks.

NetMeeting Condition. In this condition, the programmers were in different offices, each of them with a workstation. The subjects had a voice connection through a telephone for the duration of the task. The telephone was left in speakerphone mode to free them from having to carry the handset. The subjects used the COPPER editor in single-user mode and shared the application through MS NetMeeting.

COOPER Condition. The physical setup of this condition was similar to the NetMeeting condition. The only difference was that rather than using NetMeeting to share a single application, they used the COPPER editor in collaborative mode. In this way, they had access to the collaboration awareness features of COPPER, such as the radar view, and the use of the semaphore for floor control.

The programmers were videotaped in all three conditions. In addition, there was a researcher in each office who was present at the time of the experiment. This researcher kept control of the time and recorded the number of times the programmers exchanged control of the floor as well as the errors identified, corrected and introduced in the code. To track the progress of the programming team without looking over their shoulders, the researchers had a monitor in a separate desk connected to the programmer's workstation.

3.2 Procedure

The pairs had approximately 55min. to complete each task, which was divided in five phases. The tasks were performed one after the other with 5 to 10 minute rest periods between tasks for an approximate total duration of the experiment of 3 hours per couple. The tasks were executed as follows:

Phase 1. During 10 minutes the programmers were introduced to the programming task they had to perform and the condition in which they had to work. Each person was given an initial version of the program assigned for that task. Each program was injected with 10 errors of different types (syntax, assignment, logic, and interface). The subjects were told that the code included errors but not their type or number.

Phase 2. For the next 15 minutes the couple introduced the code into the editor. To keep this time constant for all programming tasks part of the code was already in the editing window, they were asked to type the remaining 60 lines of code. The pair was able to choose who will be the driver and who the observer. They could also interchange roles during the process. Although the objective of this phase was to introduce the code into the editor, we expected them to detect and correct some errors and become familiar with the code.

Phase 3. Once the code was introduced the subjects were given a maximum of 15 minutes to correct the program, that is, to find and correct errors. This included errors that were introduced by the researchers, and at times, errors introduced by the programmers during Phase 2.

Phase 4. In this phase the programmers were given a text describing a simple modification to be made to the code. They were given a maximum of 15 minutes to complete this phase.

Phase 5. Finally, the pair was asked to complete a survey with 12 Likert-scale assertions, which included topics such as their satisfaction with the modality in which they worked, the perception of the participation of their colleague, and their understanding of the programming task.

3.3 Experimental Tasks

The programming tasks were designed to be simple to assure that they could be done in the short amount of time that was provided and that the students would concentrate on correcting errors and delivering a high quality code. To avoid learning effects the pairs worked on the tasks in different order. Each group was asked to perform the following three programming tasks on a different condition.

LOC. The purpose of this program is to count the number of lines of source code in a file. The program takes a source file as input and returns the total number of lines contained in it. In the last phase the programmers were asked to modify the program to eliminate the number of lines with comments from the final count and also report the number of methods in the source code.

SORT. This program reads a file containing a list of numbers in random order and sorts them in ascending order using the selection sort algorithm. The modification required the students to use insertion sort and present the results in descending order.

N Figures. This is an object-oriented program that reads a file with data of geometric figures of different types and calculates their area, using the method appropriate to each type of figure. The extension requested was to add a new class with a different type of figure.

3.4 Measures

Since we kept constant the amount of time dedicated to the task, we concentrated on measuring the quality of the code since this is an area in which one would expect pair programming to be useful. Additionally we measured code comprehension, since we hypothesize that the help of the colleague would help a programmer understand the code more easily and this might be negatively affected by distance and positively affected by awareness tools.

To estimate code quality we measured the number of errors that were detected and corrected during the task. A researcher observed the programmers as they performed the task and recorded when and by whom errors were detected and corrected. She also indicated when new errors were introduced.

We measured code comprehension by whether or not the pair was able to successfully modify the code and through the questionnaire at the end of the task where we explicitly asked them if they understood the program and whether the help of his colleague helped in this regard.

4 Results and Discussion

4.1 Program Quality

Table 1 shows the number of errors detected by each team for each of the three programming tasks. The results are grouped by condition, with two teams per condition (A and B), per task. The total number of errors detected in the *Collocated* mode was 24, for an average number of errors found in each task of 4 out of the ten that were injected. The *NetMeeting* condition was the most productive in finding errors, with a total of 27. In the *COPPER* mode 22 errors were identified. Of the 73 errors detected, only 3 were not solved. Additionally, one error was solved without the programmers explicitly detecting it. Five groups detected between 11 and 16 errors in the three programming task, with one pair (No. 6), detecting only a total of 4 errors, clearly the team with the poorest performance.

Table 1. Errors detected by each programming team.

Condition		N Figures		LOC		SORT		Total
		Errors detected	Team	Errors detected	Team	Errors detected	Team	
Collocated	A	2	(5)	5	(1)	6	(3)	24
	B	5	(2)	4	(4)	2	(6)	
NetMeeting	A	4	(3)	5	(5)	7	(4)	27
	B	2	(6)	3	(2)	4	(1)	
COPPER	A	5	(1)	0	(6)	3	(2)	22
	B	5	(4)	4	(3)	5	(5)	

These results provide some evidence in support of our first hypothesis. That is, that the number of errors detected is not reduced when the pairs are working in remote locations. In fact, the results were slightly better for the groups that were distributed and used NetMeeting to share the code while the tasks performed with COPPER in distributed mode resulted in a slightly lower number of errors found.

4.2 Code Comprehension

In table 2 we present the results to five of the questions, related to code comprehension, from the survey completed after each task. The survey used a seven-point Likert scale with anchors ranging from strongly disagree (1) to strongly agree (7). Although the programmers were in general able to understand the code, the responses to questions 1, 4, and 5, the ones more directly related to the understanding of the code, indicate that code comprehension was slightly better when the authors were collocated. When comparing the two distributed conditions NetMeeting seemed to work slightly better than COPPER.

Table 2. Perception of code comprehension.

Question	Collocated	NetMeeting	COPPER
1. I understood the program	6	5.58	5.5
2. I found the program to be complex	2.75	2.92	2.75
3. Working with someone helped me find more defects	6.42	6.17	6.25
4. Working with someone helped me to modify the program	6.17	6	5.67
5. Working with someone helped me understand the program	6.33	6.08	5.83

With respect to the actual completion of the tasks assigned to the programmers, we have that 4 groups completed the modification when using NetMeeting, while 5 groups successfully modified the program using COPPER. Results that give a slight edge to the COPPER mode.

The results from the questionnaires and the completion of the task do not provide evidence that the awareness features incorporated in COPPER helped pairs understand the code more than NetMeeting did. Although one more group completed the task on time, the subjects had the perception of having better tackled the task when working with NetMeeting.

5 Conclusions

Extreme programming techniques, and in particular pair programming, are gaining considerable attention given their advantage at handling software development projects with vague or changing requirements. As the software industry continues to grow and their practice becomes global, distributed teams will require appropriate tools to support their software development practices. The development of such tools needs to be supported by empirical research aimed at establishing the necessary services required to support the intensive nature of the collaboration required during this practice.

Towards this end we have presented results of an evaluation conducted with 6 groups that were asked to complete three programming task in different conditions: collocated, distributed and with limited collaboration awareness, and distributed and several collaboration services. Our results seem to support our hypothesis that distributed groups could be as effective in finding programming errors as collocated ones. On the other hand, the additional awareness provided by the COPPER tool didn't seem to facilitate the programmer's understanding of the code over what simple application sharing can accomplish. Analysis of the videos will be conducted to better understand the advantages and disadvantages of each condition.

References

1. Cockburn A. and Williams L., The Cost and Benefits of Pair Programming. Addison Wesley. (2001)
2. Herbsleb J.D. and D. Moitra., Global Software Development. IEEE Software. 18(2), (2001), 16-20
3. Kircher M., Jain P., Corsaro A., and Levine D., Distributed Extreme Programming. Extreme Programming and Flexible Processes in Software Engineering, Italy, May, (2001)
4. Morán, L., Favela, J., Martínez, A., y Decouchant, D., Document Presence Notification Services for Collaborative Writing. In Proc. of CRIWG'2001. IEEE Computer Press. Darmstadt, Germany, Sept. 6-8, (2001), 125-133
5. Natsu, H., Favela, J., Moran, A.L., Decouchant, D., and Martinez, A.M., Distributed Pair Programming in the Web. In Proc. ENC'03, IEEE Comp Society, Mexico, 2003, 81-88.
6. Whitehead E. J., Collaborative Authoring on the Web: Introducing WebDAV. Bulletin of the American Society for Information Science, 25(1), (1998), 25-29
7. Williams L. and Kessler R., All I Really Need to Know about Pair Programming I Learned in Kindergarten. CACM, 43(5), (2000), 109-114
8. Williams L., Kessler R., Cunningham W., Jeffries R., Strengthening the Case for Pair Programming. IEEE Software, 17(4), (2000), 19-25

Communicating Design Knowledge with Groupware Technology Patterns

The Case of Shared Object Management

Stephan Lukosch and Till Schümmer

University of Hagen
Department for Computer Science
58084 Hagen, Germany
{stephan.lukosch,till.schuemmer}@fernuni-hagen.de

Abstract. Many groupware frameworks offer programming abstractions to relieve developers from recurring issues during groupware development. However, some properties of the frameworks complicate their usage. We identify these properties and argue that the framework approach alone is not sufficient for supporting groupware developers. Groupware development support should instead educate developers on how to design and implement groupware applications and foster the reuse of proven solutions. We propose pattern languages as an educational and communicative vehicle for reaching this goal. To assist developers in the development process of groupware applications, we provide a pattern language that offers proven solutions for recurring issues in the area of shared object management and allow developers to reuse them.

1 Introduction

Developing groupware applications is a difficult and time-consuming task. Apart from the actual task of the application, e.g. editing texts or spreadsheets, developers have to consider various aspects on a technical level. Among others,

- network connections between the collaborating users have to be established,
- parallel input from many users has to be handled,
- specific group functions have to be included, and
- shared data have to be managed.

These issues are often not part of the professional training of software engineers. Instead, software engineers learn the basic principles that empower them to create any kind of software. These include modelling techniques like the Unified Modelling Language or programming techniques like object-oriented programming and programming languages like Java or C#. These principles and techniques theoretically empower developers to create groupware applications, too. However, developers first have to learn how to deal with the groupware specific issues discussed above before they can implement a groupware application for the first time.

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 223–237, 2004.

Groupware development support should

- educate developers on how to design and implement groupware applications,
- introduce novice developers in the complex interdisciplinary field of groupware design, and
- foster reuse of proven solutions.

During several projects at our faculty we noticed that the main obstacles are concerned with data sharing issues, e.g. to distribute the data, to keep the shared data consistent, or to permanently store the shared data. In the remainder of this paper we first discuss related work before we present our approach for groupware development support concerning shared data management, provide an example of a pattern language, and discuss first experiences.

2 Related Work

One approach to support developers is the use of groupware frameworks, e.g. Rendezvous [1], Suite [2], NSTP [3], GroupKit [4], COAST [5], DreamObjects [6], Sync [7], Habanero [8], DyCE [9], DOORS [10], GINA [11], or DistView [12]. They help during the development process by providing components that hide most of the *dirty and difficult* work that addresses for instance network connection management or process scheduling. They also impose a specific way of shaping the group process by, e.g., providing means for starting a collaborative session. If the framework perfectly matches the requirements of the project, it will simplify the development. Unfortunately, this is often not the case. We thus identified several properties that complicate the use of frameworks:

Programming Languages: Framework developers often chose the programming language that is currently en-vogue or that has been used in other projects before. But when it comes to the point that the framework should be used by other groups, the programming language becomes a crucial issue. Reuse is thus limited by the chosen programming language.
Re-implementations of frameworks in other programming languages are one way to widen the framework's application [13]. An example is the testing framework XUnit[1], which is available in most programming languages. Anyhow, translating a framework is a difficult and error-prone process and there are almost no groupware frameworks available that cover a wide range of programming languages. Exceptions are first directions to .NET groupware frameworks that are more or less language independent, e.g. CoCoWare [14]. Two examples of frameworks where the programming languages are a critical issue are COAST and GroupKit. Both used programming languages that provide a very high level of abstraction and make the process of implementation very easy. But the languages, i.e. Smalltalk for COAST and Tcl/Tk in the case of GroupKit, do not have a large commercial impact. Especially, they are not part of the standard curricula for software developers, which implies that a user of the framework would have to learn the programming language before he can use the framework.

[1] http://www.xprogramming.com/software.htm

Distribution Architecture: As with the programming languages, framework developers often limit the applicability of the framework by choosing a distribution architecture that fits the first intended applications best. This may mean that the framework for instance requires a client-server architecture or that it uses peer-to-peer mechanisms. Although some frameworks, e.g. DreamObjects [6], support different distribution architectures, it remains difficult for the designers to determine which architecture should be used for a specific application.

Isolation: Framework developers often assume that the application can be created on top of the framework. This gets complicated when the application under development could benefit from more than one framework [15]. Since the components of the framework define an intended context, components from different frameworks usually make different assumptions regarding their context, which makes it hard to use them together [13].

Consider for example the e-mail framework James[2] and the web application framework TomCat[3]: Both frameworks include mechanisms for deployment and installation that require for instance specific directory structures and persistence infrastructures. Both frameworks also assume that they are under control of the Java virtual machine (if they are deployed in the standard way). Although it is possible to adapt the frameworks so that they run in the same Java virtual machine and use the same domain model, it is still a hard task. Especially, it requires from the developers that they translate between the different framework architectures and map them to their domain model.

Black-Box Components: Frameworks often only provide black-box components (especially Java frameworks that are shipped without source code). These components are designed for a specific context like a specific structure of the underlying data model or specific mechanisms for processing user input. When the framework is instantiated, it can be customized by exchanging the components of the framework with application specific components that implement the same interface. Unfortunately, the access points for exchanging these components are limited and depend on the configuration needs foreseen by the framework developers.

Documentation: Since most groupware frameworks have emerged from research projets, the main focus has been on the functional aspects of the framework rather than the documentation of the framework for novices. Johnson et al. [13] noted that frameworks describe the application domain by means of a programming language, which makes it hard to "learn the collaborative patterns of a framework by reading it". A didactically sound description of the framework's dynamic aspects thus is crucial for training developers in using the framework.

[2] http://james.apache.org/
[3] http://jakarta.apache.org/tomcat/

3 Approach

From the observations made in the previous chapter, we conclude that the framework approach is not sufficient for supporting the groupware developers. As other authors required for general software development (e.g. [13], [16], or [17]), we argue that groupware reuse should focus on design reuse rather than code reuse. The developers should be trained in a way so that they can reproduce the framework's developer's design decisions and act as experts.

Johnson et al. [13] and Brugali [18] pointed out that pattern languages can be an educational and communicative vehicle for reaching this goal. Pattern languages consist of patterns and relations between these patterns. By means of design patterns, one can describe expert knowledge in the form of rules of thumb. These rules include a problem description, which highlights a set of conflicting forces and a proven solution, which helps to resolve the forces. Initially, patterns were used in the area of construction by the architect Christopher Alexander [19].

Pattern languages combine patterns of a specific application domain and unfold a set of different design decisions. Beck and Johnson [20] argued, "that existing design notations focus on communicating the 'what' of designs, but almost completely ignore the 'why'. However, the 'why' of a design is crucial for customizing it to a particular problem.". Compared to traditional design notations (like UML diagrams), patterns do not only focus on the result but also discuss the design rationale. They can be seen as a complementary approach to documentation of frameworks.

We consider patterns on two levels, each with a different target group.

- High-level patterns describe issues and solutions typically targeted at end users.
- Low-level patterns describe issues and solutions typically targeted at software developers on a more technical level.

High-level patterns focus on the system behavior, as it is perceived by the end user. They empower the end users to shape their groupware application to meet their demands. Thus, high level patterns need to be more descriptive and prosaic than low level patterns that focus on how to implement that behavior. Low level patterns focus on the implementation of the system and thus include more technical details.

Both, low-level and high-level patterns can be positioned on a continuous abstraction scale. The more a pattern discusses technical issues, the lower is its level. Groupware technology patterns that deal with class structures, control flow, or network communication form the lower bound on the abstraction scale of patterns. Groupware usability patterns that focus on human interaction and address computer systems just as tools to support the human interaction would be placed near the upper bound on the abstraction scale. In the extreme, high-level patterns would describe how the end user can compose off-the-shelf components and embed them in his work process. This would then mean that the software developer would no longer need to assist the end user in implementing the pattern.

First pattern collections have evolved in the area of groupware design. On our scale some of them can be regarded as low-level patterns for distributed systems, e.g. POSA2 [21]. Several collections have focussed on usability issues in groupware applications, e.g. GAMA [22], or on special sub-domains like knowledge management [23]. As Erickson [24] pointed out, pattern languages can serve as a *Lingua Franca* for human computer interaction design. This motivated the collection of human computer interaction patterns, e.g. Borchers [25]. We are aware of only one collection that focusses on technical infrastructures for groupware applications [26]. Within this paper, we will present additional low-level patterns that provide a deeper insight in the problems and solutions related to the management of shared objects.

4 A Pattern Language for Managing Shared Objects

As we noticed during the development of several groupware applications, data sharing is among the main obstacles for novice developers. Within this paper, we describe parts of a pattern language for managing shared objects in groupware systems. The pattern language concentrates on how to share data objects and maintain their consistency. We used the pattern language to communicate design knowledge to students in a software lab in which the students had to implement collaborative games (see section 5).

Our patterns follow the pattern structure outlined in the Oregon Software Development Process [27]. The *pattern name* is followed by the *intent,* and the *context* of the pattern. All these sections help the reader to decide, whether or not the following pattern may fit into his current situation.

Then follows the core of the pattern composed of the *problem* and the *solution* statement separated by a *scenario* and a *symptoms* section. The *scenario* is a concrete description of a situation where the pattern could be used, which makes the tension of the *problem* statement tangible. The *symptoms* section helps to identify the need for the pattern by describing aspects of the situation more abstract again.

After the *solution* section, the solution is explained in more detail and indications for further improvement after applying the pattern are provided. The *participants* section explains the main components or actors that interact in the pattern and explains how they relate to each other. The *rationale* section explains, why the forces are resolved by the pattern. Unfortunately, the application of a pattern can in some cases raise new unbalanced forces. These counter forces are described in the section labelled *danger spots.*

The pattern map in fig. 1 shows the patterns of this language and the relations between the patterns. If a pattern A points to another pattern B, pattern A is important as context for the referred pattern B. One can start exploring the pattern language by reading CENTRALIZED OBJECTS. One reasonable sequence through the pattern language can then for instance be CENTRALIZED OBJECTS→REPLICATE FOR SPEED→UPDATE YOUR FRIENDS→BELIEVE IN YOUR GROUP. We only discuss the REPLICATE FOR SPEED and UPDATE

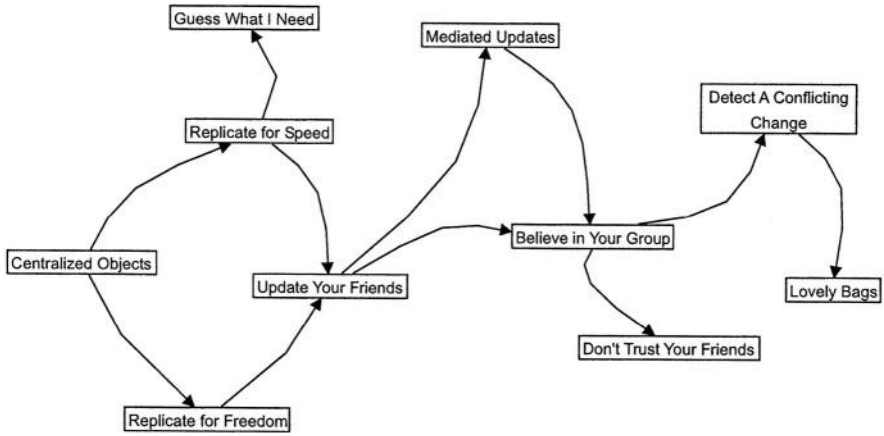

Fig. 1. Pattern map

YOUR FRIENDS patterns in detail as these are used later in this paper to illustrate our experiences during the software lab. To understand the relations between all patterns, the following list gives a short description of the other patterns that are not fully included in this paper (in alphabetical order):

BELIEVE IN YOUR GROUP: You want to ensure consistency but you do not have much time, until you can perform these changes. *Therefore,* perform the change immediately. If another client performed a conflicting change earlier (cf. DETECT A CONFLICTING CHANGE), undo or rearrange your change.

CENTRALIZED OBJECTS: To enable collaboration users must be able to share the data. *Therefore,* manage the data necessary for collaboration on a server that is known to all users. Allow these users to access the data on the server.

DETECT A CONFLICTING CHANGE: If two or more users change the same data at the same time, their changes interfere. This can lead to inconsistent data or contradict the users' intentions. If the users are not aware of this conflict, they will no longer have a common base for collaboration. *Therefore,* whenever a change is received from another client check it against those changes that have not yet been replayed by the other client and affect the same shared object. If the performed operations will produce a conflict then undo one of these changes.

DON'T TRUST YOUR FRIENDS: You want to ensure that nobody messes up your changes on replicated data objects, but many users are working with the same objects at the same time. Each site performs its changes locally before informing the other sites. This can lead to different execution orders of the changes. If the changes are not commutative, i.e. changing their execution order does not lead to the same shared state, the shared state becomes inconsistent. *Therefore,* let a site request and receive a distributed lock before it can change the shared state. After performing the change let the site

release the lock so that other sites can request and receive it for changing the shared state.

GUESS WHAT I NEED: The response time of interactive applications has to be short. The network latency and delay wastes time in distributed systems. Full replication, i.e. every user maintains a replica, demands high communication costs for initialization and consistency. If network communication between the users is slow, the response time of interactive applications increases and makes collaboration hard. *Therefore,* let a user only hold a replica for a shared data object which she currently accesses or is supposed to access in near future.

LOVELY BAGS: Access operations to shared container objects change the content of the container by adding or removing elements. Most of these operations are very bad regarding concurrency. Thus, synchronous collaboration on container objects seems often impossible. *Therefore,* wherever a high level of concurrency is needed model your container objects by means of a bag. If the container's entries need to be ordered equip the data entries with an order criterium that can be uniquely assigned by each client (e.g. the current time stamp together witch a unique client number) but still store the entries in the bag.

MEDIATED UPDATES: Clients want to propagate update messages to other clients who keep replicas of the same data. If they contact the other clients directly, they have to maintain information who those clients are and have to establish communication with these clients. This is complicated and error-prone.Especially if some clients may disconnect and reconnect in an unpredictable way (if the set of clients changes over time). *Therefore,* after changing a replicated object inform a mediator which will distribute an update message to all interested clients.

REPLICATE FOR FREEDOM: Users may not have a permanent connection to the system, where relevant data is kept. Without a permanent connection to the data users will not be able to finish their work, if the data cannot be accessed. *Therefore,* replicate the data to the user's device. Update the replicas whenever two systems, which hold copies of the data, connect.

4.1 Replicate for Speed

Intent: Allow users to access data objects without network delay.

Context: You have considered to manage the shared data using CENTRALIZED OBJECTS. Now you are concerned about speed.

Problem: The response time of interactive applications has to be short. The network latency and delay wastes time in distributed systems. Thus interactive applications are inappropriate if the response time depends on client-server communication.

Scenario: Imagine a group that works with a collaborative diagram editor. Each diagram element is represented by a shared data object. Now, imagine

that a user wants to drag a diagram element and drop it somewhere else on the drawing areas. If the diagram elements are managed on a server (compare CENTRALIZED OBJECTS), a user has to access the server for each movement step while dragging the diagram element. As this causes network communication, the movement of the diagram element is not smooth or may even freeze. This makes interaction with the editor hard.

Symptoms: *It is obvious that the pattern is missing, when ...*

- users with a permanent network connection collaborate in a highly interactive application.
- users perform many incremental changes on large objects.
- the response of the application is too slow.

Solution: Replicate the shared data to the users' sites. Let a user change its local replicas and ensure consistency by using the UPDATE YOUR FRIENDS pattern.

Participants: When a user joins an already collaborating group, its site has to request all shared data objects from a site which already participates in the collaboration. Upon request the site transfers the shared objects to the requesting site.

Rationale: Since users can access the data locally, they can perform local changes or display refreshes without network costs.

Danger Spots:

- Replication ensures that the delay for accessing the shared objects is low, but it may cause a large initial delay, when the common objects are accessed for the first time. Thus, you should schedule the initial transmission in times, where the user does not need fast response times (using the GUESS WHAT I NEED pattern).
- Replication causes high communication costs [28]. If this is an issue and you want to reduce the communication costs, use the GUESS WHAT I NEED pattern.
- As every user can locally access and modify the shared data, the consistency of the shared data is an issue. To ensure consistency, use the UPDATE YOUR FRIENDS pattern.
- Requesting and transferring the state of a shared object is complicated, as you have to ensure that the transferred object is consistent to all other replicas. There exist different approaches to solve this problem, compare [29] for a starting point.

Known Uses:

GroupKit [4] organizes shared objects in so-called environments [30]. An environment is a hierarchical data structure, where a node either can hold a value or have other nodes as children. The runtime system transparently replicates an environment. A developer can bind callbacks to an environment and receive a notification when a node is added, changed, or removed.

COAST [5] allows clients to keep replicas of shared objects. To maintain a replica, the clients register at a central server, who provides a primary copy of the objects. Clients can directly change their replicas and visualize the new state of the replica in their user interface. This ensures a high level of interaction in the application. Whenever a client changes the state of a replica, this change is propagated by means of the MEDIATED UPDATES pattern. Changes can be state changes or the creation of new replicated objects.

DreamObjects [6] supports replicated objects for highly interactive applications. All participating sites maintain a replica and can perform reading accesses locally. Changes to the replica are handled transparently by a local substitute for the shared object that is returned to the developer when creating a new shared object.

HTTP/1.1 [31] supports caching at the client and the server site. Caching at the client site is the most common used practice. To ensure the consistency of the cached document HTTP/1.1 uses an expiration mechanism. Instead of again requesting a cached document the client asks the server if the document is still valid or is already expired. In connection with REPLICATE FOR SPEED a cached document can be compared with a replicated shared object.

Related Patterns:

CENTRALIZED OBJECTS can be used to reduce the communication costs and to simplify the management of the shared data.

GUESS WHAT I NEED reduces the communication costs by distributing a replica to only those users who access the shared data.

REPLICATE FOR FREEDOM supports collaboration between users that are not permanently connected to the network.

UPDATE YOUR FRIENDS can be used to propagate changes to all other sites that maintain replicas of the shared data.

4.2 Update Your Friends

Intent: Distribute local state changes to the other users to achieve consistency.

Context: You have developed support that lets users access replicated objects (c.f. REPLICATE FOR SPEED and REPLICATE FOR FREEDOM). Now you are thinking about effects of changes to these objects.

Problem: Users change their local copies of the replicated artifacts and the other users cannot notice these local changes. This makes collaboration impossible.

Scenario: Consider, e.g., a collaborative diagram editor where all diagram elements are represented by a replicated object. Users collaborate in the diagram editor to specify the class diagram of a new project they are working on. Each user locally modifies the class diagram by adding new classes or changing the interface definition. As these changes are not propagated, each user has a different class diagram at the end of the collaborative session. After implementing their part of the class diagram, the users try to integrate their work and wonder why this is not possible without a lot of errors.

Symptoms: *It is obvious that the pattern is missing, when ...*

- users collaborate by sharing and changing replicated artifacts.
- the state of the replicated object diverges, as users locally change their replicas.

Solution: After changing a replicated object locally

- send an update message for this object to all clients that also maintain a replica,
- ensure that all clients receive this update message, and
- let these clients change their replica according to the information in the update message.

Participants: A client modifies its replica and distributes an update message to all clients who also hold the replica. This update message may contain the new object state or the transformation that led from the old to the new state. Both cases are equivalent regarding update distribution but different regarding consistency management. After distributing the update message collect acknowledgements from the receiving clients to ensure that each one received the update message. To know who has to receive the update each client has to maintain a list of these clients. The time to which the update message is sent depends on the connectivity of the clients.

Rationale: Since all clients sooner or later receive an update message, they can change the state of their replica to the new state.

Danger Spots:

- You might not know all your friends. In this case [32] proposes to use a flooding technique where the informed friends inform their friends. The latter is, e.g., done in Gnutella.
- Update messages might get lost and thus not all your friends get to know the update.
- If users change the state at the same time, there can be a conflict.
- If network bandwidth is an issue, distribute the actions describing the state change instead of the new object state to reduce the network load. Be careful as this might not be true in every case, e.g. actions may need arguments larger than the whole object state.

- If the execution of a state-changing action is more time-consuming than transmitting the whole object state to the other clients, distribute the new object state as update message.
- If clients are disconnected, use push- and pull-phases [32].

Known Uses:

GroupKit [4] transparently replicates an a programming abstraction called environment. Whenever users change the value of a node in the shared environment, the runtime system automatically distributes the new content of the node to the other participating sites. Additionally, GroupKit supports a multicast RPC like described in [33]. By using the multicast RPC developers can distribute local changes to all other participating sites.

DreamObjects [6] supports a more sophisticated multicast RPC as described in [33] to achieve consistency. In DreamObjects each site maintains for each shared object a list of sites that hold a replica of the shared object. Based on these lists the runtime system of DreamObjects distributes the necessary update messages. These update messages describe the change and allow each receiving site to re-execute the change.

USENET uses a flooding mechanism to exchange messages between the different machines [34]. A message is posted on one machine to a list of newsgroups. This machine accepts it locally, as if applying a local change, and then forwards it to all its neighbors. The neighbors check if they are really interested in the new message and then apply the change. Furthermore, the neighbors forward the new message to all their neighbors. By using this technique, a site can apply a change without knowing all sites that are interested in the change. It is only necessary to now some sites. However, there are two further danger spots. First, to reduce network load loops must be avoided. Second, it has to be ensured that still all sites can be reached.

Related Patterns:

REPLICATE FOR SPEED: As users can locally change their replicas, it is necessary to inform the other participating sites to keep the shared data consistent. The UPDATE YOUR FRIENDS pattern can be used for this purpose.

REPLICATE FOR FREEDOM: If users are allowed to modify their local replicas, the UPDATE YOUR FRIENDS pattern can be used to distribute the modifications and keep the shared data consistent.

MEDIATED UPDATES: In MEDIATED UPDATES all changes are distributed by one site. Compared to this, the UPDATE YOUR FRIENDS pattern does not have a single-point of failure, is not a bottleneck, and the communication costs are lower, which reduces the response time of the application.

BELIEVE IN YOUR GROUP provides a way to optimistically ensure consistency when different users modify same parts of the shared data at the same time.

5 First Experiences

We used the pattern language during a software lab course on groupware development held at our faculty. Six groups were asked to create a collaborative game. The groups consisted of up to six students, which either study in part-time or full-time. Apart from the lab course the students also attended other lectures. The lab course lasted half a year. All groups were introduced orally to the patterns of our pattern language at the beginning of the lab course.

Three of the groups used the COAST groupware framework and thus had to learn the Smalltalk language and class library in addition to learn how to use the COAST framework. One of these groups, e.g., implemented a collaborative labyrinth. The center of the labyrinth contains a treasure room. When the game starts, all players are positioned outside of the labyrinth. They have the task to reach and enter the treasure room as a group in the fastest possible way. As the walls of the labyrinth sometimes change their position, the path to the treasure room can be blocked for some players. These position changes can only be undone from one side of the wall. Thus, sometimes players need help to unblock their path.

The other groups were implementing the collaborative game from scratch. One of these groups, e.g., decided to implement a game called *Pacmen.* Pacmen is a multi-player adoption of the well-known arcade game Pacman, in which one user had to move around a Pacman through labyrinth walks. On its way through the labyrinth the user had to eat pills and avoid collisions with ghosts. In the multi-player version there are several Pacman, each controlled by one user. The users have to coordinate their movement to eat all pills in the fastest possible way.

The groups using the COAST framework were complaining about the introductory overhead concerning the framework. All students reported that they experienced this learning as a difficult and very time consuming task. At the time when they felt familiar in the environment, the course was already in the final third. In the final third of the course, they still could not focus on issues concerning the collaborative game. This emphasizes that the usage of frameworks is often complicated because of the lack of documentation and the chosen programming language. As already noted by Fayad and Schmidt [15], this confirms that due to the large learning efforts the use of a framework only makes sense if it is used in long-term projects.

The members of the Labyrinth group took a long time until they finally agreed to use the framework. This was basically because one group member resisted to start learning a framework. His argument was that collaborative applications were not difficult to implement and the group would reach better results, if they chose a plain Java solution. The other group members were less confident and overruled the sceptic one. The sceptic decided to leave the group. After this discussion, the group started learning the framework, but had problems in using the framework. Finally, we directed their attention to some patterns, e.g. the LOVELY BAGS pattern. These patterns improved their understanding of the framework and they finally finished their project.

The pattern groups chose a programming language, which fit best for their group, and the patterns which fit best for their problems. After choosing the patterns, they started implementing their collaborative game without restrictions given by a framework. With the help of the patterns they soon had first design decisions, while the framework groups were overloaded by the available functionality. Thus, the pattern groups could concentrate on the issues concerning their collaborative game, the applicability of selected patterns, and their implementation. Especially, they focussed on one pattern at a time which helped them to make piecemeal changes to the software and not lose any team members because of a too steep learning curve.

The Pacmen group, e.g., chose to implement a highly interactive game. After we presented the patterns to the group, the members immediately decided to use the REPLICATE FOR SPEED and UPDATE YOUR FRIENDS pattern. Both pattern descriptions note that the application of the pattern reduces the response time of the resulting application. This was the main concern in the development of Pacmen since the group wanted to focus on a fast and fluid game experience. The group members were able to transfer the symptoms of the presented patterns to their problem domain.

Later, there were discussion about using other patterns like MEDIATED UPDATES or even CENTRALIZED OBJECTS. These discussions mainly stem from the fact that these patterns promise less implementation effort. However, the group did not change their mind.

At the end of the software lab all groups had to present their results. Each group tested the collaborative games of the other groups. Finally, the students were asked to vote for the best collaborative game. The first three places were occupied by the games of the groups that were developing their game from scratch, using our pattern language.

These are just first experiences. However, they indicate that developing an application from scratch can lead to competitive results, if developers are instructed on how to design and implement groupware applications.

6 Conclusions

Groupware developers have to consider various issues on a technical level. These issues are often not part of the professional training of software engineers. To relieve developers from recurring issues groupware platforms offer programming abstractions. Anyhow, frameworks have properties that complicate their usage, e.g. the chosen programming languages, the distribution architecture and protocols, their isolated design, the black-box approach, or the lack of documentation.

In our opinion, frameworks alone do not sufficiently support groupware developers. We argue that a groupware development support should help educating developers on how to design and implement groupware applications and foster the reuse of proven solutions. Pattern languages are an educational and communicative vehicle for reaching this goal.

This paper presented a pattern language that focusses on shared object management, which is among the main obstacles during groupware development. It

allows developers to use these solutions in their intended context. First experiences during a software lab at our faculty have shown that the patterns are a sufficient means to instruct novice developers on how to implement groupware applications.

The pattern language described in this paper concentrates on data sharing and keeping the shared data consistent. It covers only a small part of all low-level issues concerning groupware development. Future work is needed to develop a more complete pattern language. We thus invite the community to share their expertise by means of patterns and to evaluate the groupware patterns collection in diverse projects.

References

1. Hill, R.D., Brinck, T., Rohall, S.L., Patterson, J.F., Wilne, W.: The Rendezvous architecture and language for constructing multiuser applications. ACM Transactions on Computer-Human Interaction **1** (1994) 81–125
2. Dewan, P., Choudhary, R.: A high-level and flexible framework for implementing multiuser interfaces. ACM Transactions on Information Systems **10** (1992) 345–380
3. Patterson, J.F., Day, M., Kucan, J.: Notification servers for synchronous groupware. In: Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work, Boston, Massachusetts, USA (1996) 122–129
4. Roseman, M., Greenberg, S.: Building real-time groupware with groupkit, a groupware toolkit. ACM Transactions on Computer-Human Interaction **3** (1996) 66–106
5. Schuckmann, C., Kirchner, L., Schümmer, J., Haake, J.M.: Designing object-oriented synchronous groupware with coast. In: Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work, Boston, Massachusetts, USA (1996) 30–38
6. Lukosch, S.: Transparent and Flexible Data Sharing for Synchronous Groupware. Schriften zu Kooperations- und Mediensystemen - Band 2. JOSEF EUL VERLAG GmbH, Lohmar - Köln (2003)
7. Munson, J.P., Dewan, P.: Sync: A java framework for mobile collaborative applications. IEEE Computer **30** (1997) 59–66
8. Chabert, A., Grossman, E., Jackson, L., Pietrovicz, S., Seguin, C.: Java object-sharing in Habanero. Communications of the ACM **41** (1998) 69–76
9. Tietze, D.A.: A Framework for Developing Component-based Co-operative Applications. PhD thesis, Technische Universität Darmstadt (2001)
10. Preguica, N., Martins, J.L., Domingos, H., Duarte, S.: Data management support for asynchronous groupware. In: Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM Press (2000) 69–78
11. Berlage, T., Genau, A.: A framework for shared applications with a replicated architecture. In: Proceedings of the 6th annual ACM symposium on User interface software and technology, ACM Press (1993) 249–257
12. Prakash, A., Shim, H.S.: DistView: Support for building efficient collaborative applications using replicated objects. In: Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA (1994) 153–164
13. Johnson, R.E.: Frameworks = (components + patterns). Communications of the ACM **40** (1997) 39–42
14. Slagter, R., ter Hofte, H., Kruse, H.C.: The CoCoWare .NET architecture. Technical Report TI/RS/2002/123, Telematica Instituut (2002)
15. Fayad, M.E., Schmidt, D.C.: Object-oriented application frameworks. Communications of the ACM **40** (1997) 32–38

16. Brugali, D., Sycara, K.: Frameworks and pattern languages: an intriguing relationship. ACM Computing Surveys **32** (2000) 2
17. Biggerstaff, T., Richter, C.: Reusability framework, assessment, and directions. IEEE Software (1987) 41–49
18. Brugali, D., Menga, G., Aarsten, A.: The framework life span. Communications of the ACM **40** (1997) 65–68
19. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A pattern language. Oxford University Press, New York (1977)
20. Beck, K., Johnson, R.: Patterns generate architectures. In: Proceedings of the ECOOP'94. Volume 821., Springer (1994) 139–149
21. Schmidt, D.C., Stal, M., Rohnert, H., Buschmann, F.: Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects. John Wiley & Sons Ltd (2000)
22. Schümmer, T.: GAMA – a pattern language for computer supported dynamic collaboration. In Henney, K., Schütz, D., eds.: EuroPLoP 2003, Proceedings of the 8th European Conference on Pattern Languages of Programs, 2003, Konstanz, Germany, UVK (2004)
23. Herrmann, T., Hoffmann, M., Jahnke, I., Kienle, A., Kunau, G., Loser, K.U., Menold, N.: Concepts for usable patterns of groupware applications. In: Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work, ACM Press (2003) 349–358
24. 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
25. Borchers, J.O.: A pattern approach to interaction design. John Wiley and Sons (2000)
26. Guerrero, L., Fuller, D.: Design patterns for collaborative systems. In: Proceedings of the Fifth International Workshop on Groupware (CRIWG). (1999)
27. Schümmer, T., Slagter, R.: The Oregon software development process. In Fowler, M., Eckstein, J., Burmeister, H., eds.: XP2004 – Proceedings of the 5th International Conference on Extreme Programming and Agile Processes in Software Engineering. LNCS 3092, Heidelberg, Germany, Springer (2004)
28. Garcia-Molina, H.: The future of data replication. In: Proceedings of the IEEE Symposium on Reliability in Distributed Software and Database Systems, Los Angeles, CA, USA (1986) 13–19
29. Lauwers, J.C., Lantz, K.A.: Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems. In: CHI '90 Conference on Human Factors in Computing Systems, Special Issue of the SIGCHI Bulletin, Seattle, Washington, USA (1990) 303–311
30. Roseman, M.: When is an object not an object? In: Proceedings of the Third Annual Tcl/Tk Workshop, Toronto, Canada, Usenix Press (1995) 197–204
31. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol – HTTP/1.1. Request for Comments 2616, IETF (1999)
32. Datta, A., Hauswirth, M., Aberer, K.: Updates in highly unreliable, replicated peer-to-peer systems. In: Proceedings of the 23rd International Conference on Distributed Computing Systems, ICDCS2003. (2003)
33. Cooper, E.C.: Replicated distributed programs. In: Proceedings of the 10th ACM Symposium on Operating Systems Principles, Orcas Island, Washington, USA, ACM (1985) 63–78
34. Horton, M.R., Adams, R.: Standard for interchange of USENET messages. Request for Comments 1036, IETF (1987)

Adaptable Shared Workspace to Support Multiple Collaboration Paradigms

Jang Ho Lee

Department of Computer Engineering, Hongik University
72-1 Sangsu, Mapo, Seoul 121-791, Korea
janghol@cs.hongik.ac.kr

Abstract. Several recent collaborative systems provide a room-based metaphor to represent shared workspaces that require use of multiple collaborative tools. These systems provide users with a fairly static usage paradigm of room-centered collaboration, requiring users to mold their collaborative activities to the paradigm rather than molding the paradigm to fit the requirements of their collaborative activities. In this paper, we propose a powerful and yet simple event-action based model, augmented with multi-user features, for room-based systems for providing a high degree of adaptability so that these systems can be adapted to provide support for a variety of collaborative facilities, such as call centers, transient rooms, paper reviewing rooms, and role-based collaboration, including facilities that are not necessarily anticipated by system designers. The model can be used by both system developers as well as by system administrators to customize a system to meet the requirements of their group tasks.

1 Introduction

Shared workspace is a key element of collaborative systems. Several systems, including MUDs [3], Worlds [4], CBE [5], and TeamRooms [7], use the *room* metaphor to represent shared workspace, especially when participants use multiple tools for collaboration in a shared workspace.

Existing room-based systems provide rooms with standard behavior. In fact, the room-based metaphor on these systems evolved from the observation that real-world collaboration often takes place in physical rooms around shared artifacts; thus it is logical to emulate the behavior of physical rooms in synchronous and asynchronous collaboration. Systems vary in the extent to which physical rooms are emulated; MUDs are often text-based, TeamRooms provides a shared window manager in which shared objects can be placed, and CBE provides access-controlled rooms in which users share objects but not the interface.

A key question is whether the room-based collaboration paradigm, as exemplified by these systems, adequately supports the needs of different types of collaborative situations. Can the metaphor support collaboration activities that do not ordinarily occur in physical rooms? If not, what support is needed for the paradigm to be adapted for a wide variety of uses, some perhaps not even

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 238–245, 2004.

anticipated by system designers? In this paper, we explore answers to these questions. We show that there are many examples of collaborative situations where the standard room-based metaphor provides inadequate support. On the other hand, we show that, with suitable extensions, the paradigm can support a range of collaborative activities, such as standard room-based collaboration, call centers, shared repositories, paper reviewing, voting, email, etc.

The key goal is to provide a high degree of adaptability to the room-based metaphor so that it fits the requirements of the application. The adaptability can range from simple (e.g., access control) to behavioral (e.g., establishing a private session with an available agent in a call center room when a customer enters the call center).

The importance of flexibility or tailorability in CSCW systems has been discussed by several researchers. Bentley and Dourish suggest an approach that views a CSCW system as a medium through which collaboration occurs, rather than an embodiment of mechanisms representing perceived regularities in collaborative activity [1]. Roseman and Greenberg argue that personalizable groupware must adapt to a wide variety of different group behaviors, including behavior not originally expected by the groupware developer [6]. Our work is focused on adapting the environment to the characteristics of the collaboration in the context of room-based systems. We propose concrete design recommendations to achieve a high degree of adaptability, while maintaining simplicity of use.

The paper is organized as follows. First, we show various types of shared workspace. Then, we discuss our model of software architecture to provide flexible shared workspace. Next, we discuss how the model can be used to support adaptability of workspace to fit various collaboration paradigms. Finally, we present conclusions and future work.

2 Types of Shared Workspaces

We consider the examples of customization of shared workspaces required for various collaborative tasks: (1) call centers; (2) transient rooms that automatically disappear after a certain time or when no users are active; and (3) verbs that can be applied to a room object.

We briefly describe the characteristics of each of the above. In the later section, we show how the examples above can be supported in a place-based system, after suitable, fairly general, extensions.

Call Center

In a call center, customers wait for service until an agent becomes available. For example, customers may go to an insurance company's Web site and, if they want to talk to an agent, enter a call center and click a chat object. This action results in matching the customer with an available agent. When an agent becomes available, a communication channel is established between the client and the agent so that they can conduct their business.

Support for this is provided by creating a special type of workspace called a call center. A call center behaves differently from a shared workspace of the

standard room type where any number of participants simply enter and share artifacts in the room. For a customer, entering a call center and clicking a chat object implies waiting for an available agent in order to establish a communication channel between the customer and the agent.

Transient Room

A transient room expires after a certain period. It requires a lifetime attribute to determine how long the room can exist in the system. Upon the time-out after the lifetime period, deletion of a room should occur. The lifetime of an object in a room is subject to the lifetime of the transient room.

Verbs for an Object

An object in a room can have a set of verbs that can be applied to it (such as an object in MUDs environment). For example, a verb *pick up* implies moving an object to a user's *bag,* a buffer for keeping an object for temporary use. Similarly, a verb *put down* can mean moving an object from the user's *bag* to a room.

3 Adaptable Workspace Model

We now present a workspace model on which the customization of behavior of rooms and objects is based. The model proposed in this paper is an extension of the model called Collaboratory Builder's Environment, a software architecture for creating extensible collaborative environment. The key elements of the model are entities that include users, rooms, and objects in the room(See figure 1). Users, rooms, and room objects have a set of attributes. The entities can be associated with (event, action) pairs.

Next, we describe elements of the proposed model. Then, we wlll show mechanism to create highly adaptable rooms.

3.1 Entities

Followings are the key entities in our shared workspace model: users, rooms and room objects(URL, regular object, group-aware object, and applet).

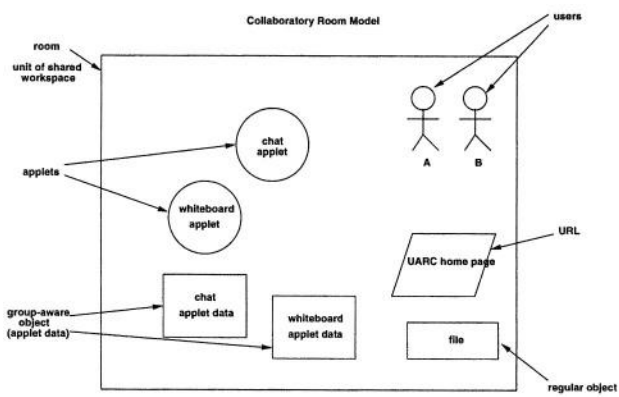

Fig. 1. A room model

Users. Users perform operations on the shared workspace, such as entering rooms, deleting rooms, generating events on objects (e.g., clicking on an object), etc. For every user in the system, the system maintains certain attributes, such as their unique identifier and current role for the purpose of access control.

Rooms. A room contains objects that are shared among users, subject to access control and object-specific attributes. A room has the following system-defined attributes: room name, owner, access permissions, user list, object list, creation date, etc. Among them, user list and object list provide awareness[2] to users. Users can define additional attributes for a room, subject to access control rules, for modifying the room behavior, such as termination time for implementing transient rooms, etc. A user can be inside multiple rooms at the same time.

Room Objects. Rooms contain named objects. Types of objects include URLs, regular objects, group-aware objects and applets. These objects can be added or deleted dynamically from a room. They can also be copied or moved from one room to another. Common system-defined attributes of room objects include object owner, creation date, access control list, current room, and object type. A URL object provides access to Uniform Resource Locator(URL) on the Web. Users can put URLs in a room and make them accessible to other users in the room. Users can use this URLs to share references to objects on the Web. A URL object has an attribute called URL value referring to executable content, such as applets in Java. A regular object is a named sequence of bytes that can be read or written to a room. They allow a room to be used as a repository of shared data with access control. A regular object is associated with attributes such as data type and applet name. A group-aware object is an object created by group-aware applets–tools such as chat and whiteboard are implemented as group-aware applets in CBE. When a group-aware object is opened, the associated group-aware applet is up and initialized with the state of the group-aware object.

3.2 Attributes, Events and Actions

Entities of the CBE model have system-defined attributes as well as user-defined ones. User-defined attributes, coupled with event-action rules, provide means to customize rooms to meet a range of collaborative needs.

Rooms and room objects can be associated with event-action pairs. Events are typically generated in response to a user action (e.g., selecting a menu item after selecting an object or clicking on the object). When an event occurs on the selected entity, an action bound to the event is executed. Events can also be generated by timers. Actions provide the core mechanism to alter the behavior of rooms. Actions can read attribute values, update them, modify access control lists, generate new events, subject to access-control restrictions, etc.

4 Applying the Model to Provide the Adaptability

First we will describe the framework to support the adaptability and then we'll explain how the model can be used to support different types of collaboration.

4.1 Framework to Support the Adaptability

Figure 2 shows the framework for applying the model with the following three key elements: `RoomMgr` object, `RoomModelListener` object, and `RoomEvent` object.

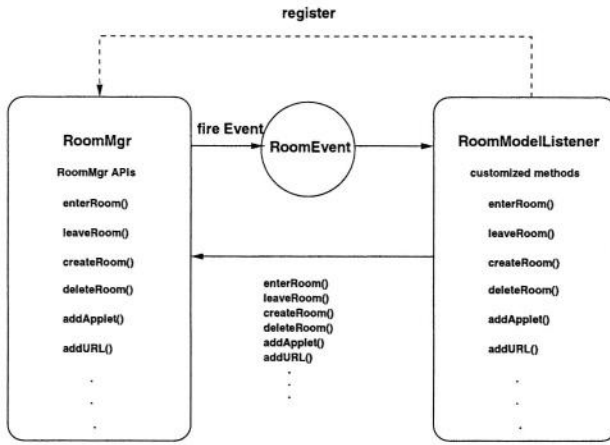

Fig. 2. Room Event Model

A class for a new type of room is derived from the abstract class `RoomModelListener`. `RoomModelListener` registers itself to the `RoomMgr` – a class representing the Room Manager server – so that `RoomEvent` fired by `RoomMgr` is propagated to itself. An object instantiated from the customized class of the new room type shows the intended unique behavior.

Figure 3 shows the `RoomModelListener` class that defines abstract methods – `enterRoom`, `timerExpired`, and `createRoom` – for the customization of rooms. Upon receiving a `RoomEvent`, the `RoomModelListener` invokes a corresponding method. The abstract methods of `RoomModelListener` are overridden by the class which defines a new type of room. For example, a class that defines a new type of room can override the `enterRoom` method with the actions that should occur when a user enters a room, thereby defining the behavior of the room.

The `RoomEvent` generated by the `RoomMgr` has the following attributes: event source, type of room, type of event, arguments for the event of the type, and `RoomMgr` object. For example, when a user enters a call center, an `ENTER_ROOM` event is generated with the following arguments: (1) the event source is `RoomMgr`; (2) the type of room is `CALL_CENTER`; (3) the type of event is `ENTER_ROOM`; (4) the arguments are `roomName` and `userName`; and (5) `RoomMgr` object. Some events

```
// Parent class from which new room model classes inherit
public abstract class RoomModelListener implements
 java.util.EventListener {
  void stateChanged(RoomEvent e) {
    String eventType = e.getEventType();
    if (e.getRoomType().equals(roomType)) {
      roomMgr = e.getRoomMgr();
      args = e.getArgs();
        if (eventType.equals(''ENTER_ROOM'')) {
          String room = (String) args.elementAt(0);
          String user = (String) args.elementAt(1);
          enterRoom(room, user);
        } else if (eventType.equals(''TIMER_EXPIRED'')) {
          timerExpired(args);
        } else if (eventType.equals(''CREATE_ROOM'')) {
          ...
        }
    }
  }
  public void enterRoom(String roomName, String userName) {    }
  public void timerExpired(Vector args) {    }
  public void createRoom(String roomName, String userName) {    }
  ...
}
```

Fig. 3. `RoomModelListener` Class

(such as `ENTER_ROOM`) are caused by a user's action while some events (such as `TIMER_EXPIRED`) are not.

The `RoomMgr` object is a Java application that runs as a server called the Room Manager. All the objects instantiated from the classes representing different types of rooms, register themselves to the `RoomMgr` to catch the events generated in `RoomMgr`. When a user enters a room, a request of `enterRoom` from a client is sent to the Room Manager, which, in turn, fires an event `ENTER_ROOM` with arguments such as room type and event type. As a result, a `RoomModelListener`, which is registered to the `RoomMgr`, catches the event and handles it by invoking various methods on `RoomMgr` for customization.

The basic set of operations supported by `RoomMgr` are as follows: (1) room operations such as createRoom and getRoomAttr; (2) room object operations such as readObject and getObjectAttr; (3) user operations such as loginUser and listUsers; and (4) access control list operations such as getRoomACL and createRole.

We now show how to use these operations to adapt the behavior of rooms to various needs.

4.2 Call Center

Figure 4 shows how the `CALL_CENTER` model works.

When a user creates a room of the type, CALL_CENTER, CREATE_ROOM event generated by RoomMgr is caught by CallCenter which, in turn, invokes createRoom method in CallCenter. Then, createRoom method adds the following attributes required for the customization of CallCenter: (1) customer Queue, a queue of customers waiting for service; (2) agentQueue, a queue of available agents who are waiting for customers; and (3) agentList, a list of persons who have an agent role. A URL object for chat is also created in CallCenter.

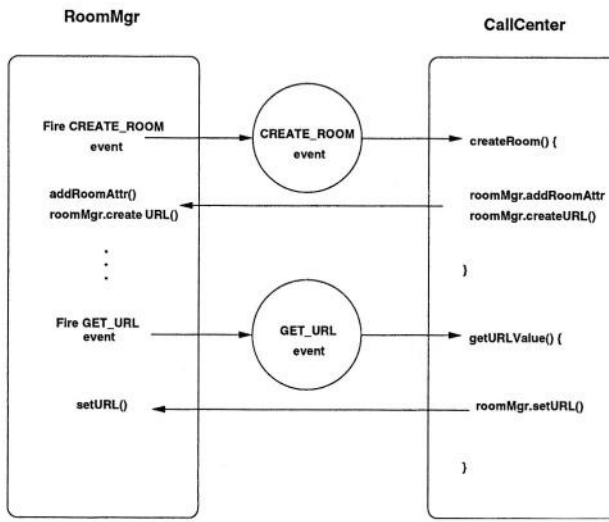

Fig. 4. Call Center Model

When a user enters CallCenter, a chat object is shown to the user. When a user clicks on the chat object, GET_URL event is generated, and sent to CallCenter class. Then, getURLValue is invoked to generate a URL to a chat tool with arguments (such as the name of chat group) for return, based on the user's role (such as agent), status of the queues (customerQueue, agentQueue) etc. When the chat tool receives the name of chat group, it joins the chat group enabling the user to communicate with members of the chat group.

The policy of getURLValue checks if a user is an agent or a customer by checking if the user has an agent role or not. If the user is an agent, it checks if customers are waiting in customerQueue. If there is no customer waiting, the agent brings up a chat creating/joining its own agent chat group. Then, the agent is put into agentQueue. If customers are waiting in customerQueue, the agents joins the first customer's chat group. An appropriate policy can also be made for the case where the user is not a customer.

Besides call center, the proposed model can support other collaboration paradigms such as transient room and verbs for an object. A transient room exists for a certain period. The implementation is based on the TIMER_EXPIRED

event and `deleteRoom` method(action). A shared workspace can also be configured to support verbs(actions) associated with an object. By supporting verbs such as *add a comment* with operations mentioned in the section 4.1, a paper reviewing room can be created that allows people to put papers in a shared workspace and lets reviewers to read the papers and comment on them.

5 Conclusions

We proposed ways of supporting adaptability to room-based collaborative systems, for a variety of collaborative applications that do not conform to a standard room metaphor used in existing systems. Examples of such applications are call center, transient room, paper reviewing room, or a new collaboration paradigm that may not even be anticipated by system designers. To provide adaptability, we proposed a model of room-based systems with the following mechanisms: objects and rooms with user-defined attributes; operations to manipulate objects and rooms; events generated by user operations on objects and rooms; and actions that get or set attributes, or invoke operations on rooms or objects.

We are currently providing support for the model in the Collaboratory Builder's Environment. Currently, the model can be used by system developers by programming in Java to build customized rooms. Future work includes exposing the customization facility to end-users via a simple scripting language.

References

1. R. Bentley and P. Dourish. "Medium versus mechanism: Supporting collaboration through customization". In *Proc. 4th European Conf. on Computer Supported Cooperative Work,* pages 133–148, September 1995.
2. M. Boyle, C. Edwards, and S. Greenberg. "The effects of filtered video on awareness and privacy". In *Proc. 8th Conf. on Computer Supported Cooperative Work,* pages 1–10, Philadelphia, PA, December 2000.
3. P. Curtis and D. Nichols. "Muds grow up: social virtual reality in the real world". In *Proc. 3rd International Conf. on Cyberspace,* 1993.
4. G. Fitzpatrick, W. Tolone, and S. Kaplan. "Work, locales and distributed social worlds". In *Proc. 4th European Conf. on CSCW,* pages 1–16, September 1995.
5. J.H. Lee, A. Prakash, T. Jaeger, and G. Wu. "Supporting multi-user, multi-applet workspaces in CBE". In *Proc. 6th Conf. on Computer Supported Cooperative Work,* pages 344–353, Boston, MA, November 1996.
6. M. Roseman and S. Greenberg. "Building flexible groupware through open protocols". In *Proc. ACM Conf. on Organizaional Computing Systems,* pages 55–65, August 1993.
7. M. Roseman and S. Greenberg. "TeamRooms: Network places for collaboration". In *Proc. 6th Conf. on CSCW,* pages 325–333, Boston, MA, November 1996.

A Decoupled Architecture for Action-Oriented Coordination and Awareness Management in CSCL/W Frameworks

Pablo Orozco[1], Juan I. Asensio[1], Pedro García[2], Yannis A. Dimitriadis[1], and Carles Pairot[2]

[1] School of Telecommunications Engineering, University of Valladolid
Camino Viejo del Cementerio s/n, 47011 Valladolid, Spain
paboro@ulises.tel.uva.es, {juaase,yannis}@tel.uva.es
http://ulises.tel.uva.es
[2] Dep. of Computer Engineering and Mathematics, Rovira i Virgili University
Avinguda dels Països Catalans 26, 43007 Tarragona, Spain
{pgarcia,cpairot}@etse.urv.es
http://www.etse.urv.es

Abstract. This paper introduces AORTA, a software architecture that provides object-level coordination and shared workspace awareness support to synchronous and distributed collaborative applications. AORTA is motivated by the need to enhance current coordination and awareness capabilities of existing software component frameworks for the domains of CSCL *(Computer-Supported Collaborative Learning)* and CSCW *(Computer-Supported Cooperative Work).* AORTA is characterized by the use of *actions* as its key abstraction instead of low-level events, the support for mutual influence between coordination and awareness, the use of coordination and awareness policies for supporting complex and dynamic collaboration scenarios, and the use of software design patterns in order to decouple coordination and awareness from the development of other aspects of CSCL/W applications. The paper motivates, justifies, and describes the main functional features of AORTA as well as its proposed software architecture. The paper also introduces a prototype of AORTA that adds coordination and awareness support to an existing groupware framework called ANTS. Finally it describes a CSCL application developed on top of both AORTA and ANTS that has been used to validate some of the presented contributions: application development is decoupled from coordination/awareness aspects, application development is facilitated by the use of action-orientation, and application coordination/awareness behavior can be configured and changed without modifying the application itself.

1 Introduction

Computer-Supported Collaborative Learning (CSCL) is a research paradigm within the field of educational software that underlines the key role that social interactions play in the process of learning [13]. CSCL applications promote and give support to *collaborative* ways of learning. From a technological point of view, CSCL has its roots into the field of *Computer-Supported Cooperative Work* (CSCW) [5].

The success of CSCL applications largely depends on their capability to be *reused* and *adapted* to different and dynamic collaborative learning scenarios. A change in

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 246–261, 2004.

the learning objectives, the involved participants, the group structure, etc. of a learning scenario might make a previously successful CSCL application unsuitable for the new situation. For example, consider a CSCL application for the collaborative edition of an electronic magazine in which each student is in charge of a particular section. If there is just a change in the way participants interact (e.g. the teacher desires that the students review the whole magazine and make modifications by turns), and the collaborative editor cannot *adapt* to that change, it could not be used anymore. On the other hand, the effort (and therefore the cost) devoted to the development of the collaborative editor might not be justified if it can only be used in that particular learning scenario.

The potential solution to this *reuse* and *adaptation* problem is a traditionally claimed benefit of the Component-Based Software Engineering (CBSE) [20]: first of all, it is easier to reuse a software component that supports a common functionality of several applications than reusing complete applications across different scenarios; and secondly, adaptation can be achieved by replacing one or several application components in order to change just that part of the application that did not fit the characteristics of a new scenario.

CBSE concepts and principles have been successfully applied to the development of non-CSCL educational software (see e.g. [18]). These experiences (and other related to the use of CBSE in other domains) stress the importance of a proper identification and dimensioning of software components in order to guarantee reuse and adaptability. Those potentially reusable components (and the set of design patterns that guide their use) could be grouped into a so-called *component framework* [12]. Such frameworks constitute a starting point for the development of new applications. Even more, the assembly of a set of existing framework components could eventually generate fully functional new applications.

Now, the construction of a CSCL framework is not an easy task: the identification and dimensioning of components implies that software engineers have a proper understanding of the main collaborative learning concepts [2]. Obviously, some of these concepts, namely those related to the support for collaboration, also belong to the CSCW domain.

Although with different motivations [21], CBSE principles have also been applied to the CSCW domain and, consequently, several proposals for CSCW component frameworks can be found in the literature (e.g. [1],[8],[9]). These component frameworks for the CSCW domain should be considered as a starting point for the achievement of those for the CSCL field. Framework components related to functionality such as group management, collaborative sessions management, shared workspaces management, coordination support, awareness management, etc. are also useful for CSCL applications.

This paper is particularly focused on two common functional aspects of CSCL/W applications suitable for being included into the corresponding component frameworks: coordination and awareness. Although coordination and awareness are broad concepts, this paper copes with: *object-level coordination* and *shared workspace awareness.*

Object-level coordination "...deals with multiple participants' sequential or simultaneous access to the same set of objects..." [6]. For example, in a collaborative editor where participants modify a document by turns, object-level coordination decides who has the following turn. If a participant does not own the turn and tries to modify

the document, the object-level coordination system should forbid that operation. *Object-level coordination* ".. .deals with the organization of activities to be performed by the users, and not the organization of processes to be performed by the system..." [6]. *Object-level coordination* should not be confused with *activity-level coordination* that manages the flow of collaborative tasks that participants perform when using a collaborative tool. *Object-level coordination* is a very important aspect in collaborative tools as "...it can enhance close inter-working of groups and the synergy that makes groups productive and energized..." [6].

Shared workspace awareness comprises "...the up-to-the-minute knowledge a participant needs about other participants' interactions with the shared workspace..." [11]. In the previous example regarding a collaborative editor, *shared workspace awareness* would be in charge of informing all participants about who is modifying what part of the document at every moment. There are other types of awareness such as social awareness, task awareness, and concept awareness but they are outside the scope of this paper. Shared workspace awareness is of crucial importance in collaborative tools as it "...allows groups to manage the process of collaborative working" [3]. In this paper, the term awareness refers, by default, to shared workspace awareness.

The analysis of existing CSCW frameworks shows that coordination and awareness support is: very limited or non-existent in some cases; and very biased to particular applications in others (what implies that it can hardly be reused). There are also some proposals of isolated coordination support systems [14] but they do not take into account relationships between coordination and other aspects of collaboration support (mainly awareness).

In this context, this paper introduces AORTA (Action-oriented decOupled aRchitecture for coordinaTion and Awareness) a software architecture for a set of components potentially integrable into existing CSCL/W frameworks in order to support object-level coordination and shared workspace awareness. AORTA is focused on synchronous and distributed collaborative applications according to the well-known taxonomy proposed in [5].

The main goal of AORTA is hiding, to the developers of CSCL/W applications, the complexity associated to the management of coordination and awareness tasks. Those developers should simply include AORTA's components into their applications and perform a limited set of configuration steps. In this scenario, the development of CSCL/W applications would be greatly simplified although it also poses important challenges to AORTA design.

From a functional point of view, AORTA supports a broad range of potential behaviors regarding coordination and awareness. That generality is achieved by the use of coordination and awareness configurable policies. Such policies specify a particular coordination or awareness behavior to be enforced by AORTA. A CSCL/W application developer should only select the most appropriate existing policy or even specify new ones. AORTA coordination and awareness policies are based on the information that characterizes collaborative interactions, and can support dynamic collaboration scenarios in which one particular policy might not be valid for the whole duration of the collaboration.

Another functional feature of AORTA is that it provides joint support to coordination and awareness, enabling the relationships between both aspects.

From a software engineering point of view, AORTA is designed in order to decouple coordination and awareness support as much as possible from the development of collaborative applications. That decoupling is achieved by means of software design patterns that reduce the learning curve for developers that use AORTA and provide specific cutpoints for collecting information required for coordination and awareness purposes.

The paper is structured as follows. Section 2 motivates and describes the above key functional features of AORTA: joint coordination/awareness support whose behavior depends on configurable policies based on collaborative interactions. Section 3 introduces and discusses the software architecture of AORTA and its implications from a software engineering perspective. Section 4 describes our current prototype of AORTA that complements coordination and awareness aspects of the ANTS component collaborative framework [8]. Some features of that prototype have been tested using a CSCL application, also described in section 4, for the collaborative resolution of puzzles. More concretely, section 4 shows how, by using AORTA, application development is decoupled from coordination/awareness aspects, application development is facilitated by the use of action-orientation, and application coordination/awareness behavior can be configured and changed without modifying the application itself. Section 5 compares AORTA with other related proposals that can be found in the literature. Finally, section 6 concludes the paper and introduces some potential future research lines.

2 AORTA: Functional Characteristics

The previous section has introduced the main functional features of AORTA. Why are those features important for CSCL/W applications? This section motivates, justifies and describes two of the most important ones, namely joint coordination and awareness support, and the use of action-based coordination and awareness policies. These functional features have obviously influenced AORTA design decisions (that will be explained in section 3).

2.1 Joint Coordination/Awareness Support

Several authors recognize that coordination and awareness are mutually influencing factors in collaborations. For instance, Dourish and Belloti state that "...awareness information is always required to coordinate group activities, whatever the task domain..." [3]. Similarly, Gutwin and Greenberg stress the usefulness of awareness information "...for many of the activities of collaboration – for coordination action, managing coupling ..." [10]. Coming back to the collaborative editor example, awareness information (e.g. a participant makes more that four illegal modifications during its turn) could be used by the coordination support to take or change a decision (e.g. that participant loses the turn). On the other hand, coordination decisions (e.g. a participant loses its turn) could be used as awareness information to all or part of the participants (e.g. the name of the punished participant is coloured in red in the others participants' list of collaborators to indicate the turn lose).

All these factors have influenced the decision to support, in AORTA, information exchange between coordination and awareness tasks.

2.2 Action-Based Coordination and Awareness Policies

What does AORTA coordinate? What does AORTA make participants aware of? These are not obvious questions as the literature contains several and different approaches to coordination and awareness that reflect each author' own conceptualization on these topics which is not always made explicit.

AORTA coordinates and makes participants aware of *indirect collaborative interactions.* This is a term that comes from the CSCL domain although it could be also consequently translated to the CSCW field. A *collaborative interaction* is "...an action that affects, o may affect, the collaboration process. That action, or its effects, must be perceived by at least a member of the collaborative group or community, different from the emitter" [15]. In this context, an action is an application event (observable from the exterior of the application) expressed in terms of application usage. For instance, an event (a change in the text shown by an editing application) could be an action (a user has modified a document) or not. This action becomes an interaction if another collaborative user becomes directly or indirectly aware of it. There are several types of *collaborative interactions.* AORTA deals with *indirect collaborative interactions* that are mediated by an object (a CSCL application in this case). They usually take place in shared workspaces [16].

The above distinction has two effects on AORTA: first of all, AORTA policies are triggered by *actions* that can potentially become *indirect collaborative interactions.* A coordination policy should evaluate whether an action requested by a participant is allowed or not according to the current *coordination state* (e.g. whether this participant owns the turn or not). If the action is finally executed, it becomes an *indirect collaborative interaction.* The awareness policy decides whether the result of a coordination decision should be communicated to other participants; secondly, *actions* constitute the point of junction between AORTA and the collaborative applications that use it. Therefore, the properties of an action determine what information an application should communicate to AORTA. According to [16], an *indirect collaborative interaction* (and therefore the action that generates it) is characterized by a role (that the participant that generates the action plays), a shared object (over which the action is performed), an operation (over that object), and a time stamp (that indicates when the action is performed).

3 AORTA: The Proposed Software Architecture

AORTA is a layered and replicated architecture that provides *object-level* and *action-based coordination* and *shared workspace awareness* services to synchronous CSCL/W applications.

AORTA is designed so as to offer its services to collaborative applications that follow the replicated or hybrid variants of the MVC (Model-View-Controller) architectural pattern [19]. The integration between the applications and AORTA is achieved by means of the *Controller* that must be totally or partially replicated in all the applications of the participants in the collaboration (see Fig. 1).

In this context AORTA takes decisions about coordination and awareness tasks that are resolved locally at each participant's application (using information previously received from other applications) thus avoiding performance degradation in the synchronous collaboration.

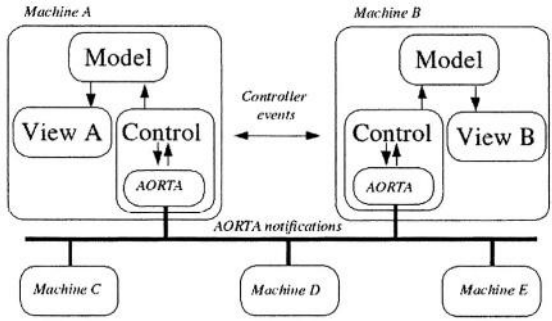

Fig. 1. AORTA within a CSCL/W application with a replicated MVC architecture.

Coordination and shared workspace awareness services provided by AORTA are guided by *policies.* The definition of new *policies* implies the adaptation and extension of AORTA's behaviour without modifying AORTA's architecture.

AORTA is action-oriented. As it was explained in section 2.2, actions are the basic unit of information exchanged among applications and AORTA. Applications request the execution of actions to AORTA. Then AORTA processes them, and decides (coordination) whether an action must be executed (thus becoming an *indirect interaction,* see section 2.2) or not. AORTA afterwards notifies (awareness) the result of this decision to all or part of the other collaborating applications.

The replicated architecture of AORTA is composed of four functional blocks that are located in three software layers that constitute the AORTA layered architecture: application layer, collaboration layer, and communication layer (see Fig. 2).

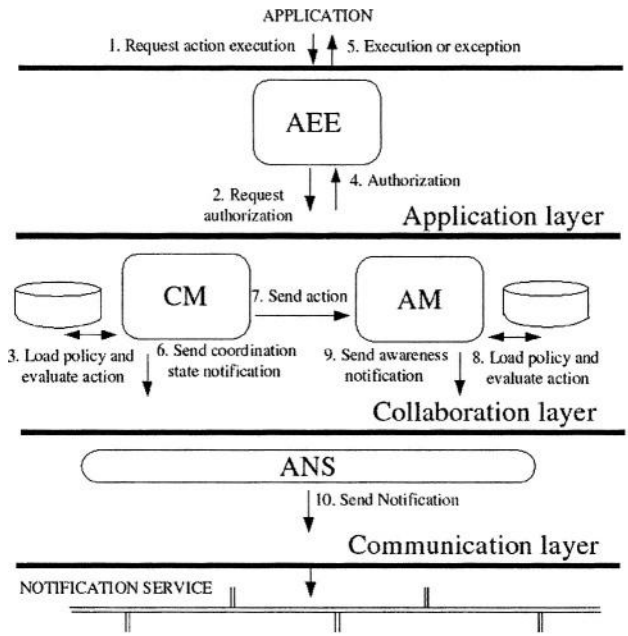

Fig. 2. AORTA layered architecture.

3.1 Application Layer

The *application layer* constitutes the point of contact among applications and the *ActionExecutionEngine* (AEE) functional block of AORTA. The AEE is responsible for the execution of actions. It receives requests from the application for the execution of specific actions and responds with their actual execution or with an exception (if the action cannot be executed).

The relationship among applications and the AEE is mediated by means of the *Command* software design pattern [7] that encapsulates the request for the execution of an action within an object. By means of this object, the AEE knows the action whose execution is being requested by the application user. The execution of that action depends on the fulfilment of a set of rules (contained in a *policy*) that are associated to the request.

The use of the *Command* pattern enables the decoupling of the action execution request from the logic that determines whether it can be executed or not. The advantage of this decoupling is threefold: that logic can be separated from the application itself thus eventually becoming transparent to the developer of the application (e.g. it can be provided by AORTA); action execution can be made dependant on collaboration services (e.g. coordination); and it facilitates the maintenance and reuse of collaboration services.

If the developer of a CSCL/W application wants to use AORTA services, he only has to represent application-level actions according to the prescriptions of AORTA and to request their execution to the AEE (Fig. 2/Task 1). Those prescriptions simply indicate that an action type is a class that implements a predefined interface and that the instances of that class must contain information regarding the identity of the user that requests the action execution, the identity of the shared object affected by the action, and the operation that the user wants to perform on that object. Fig. 3 contains a UML class diagram in which the relationships among all those elements are detailed.

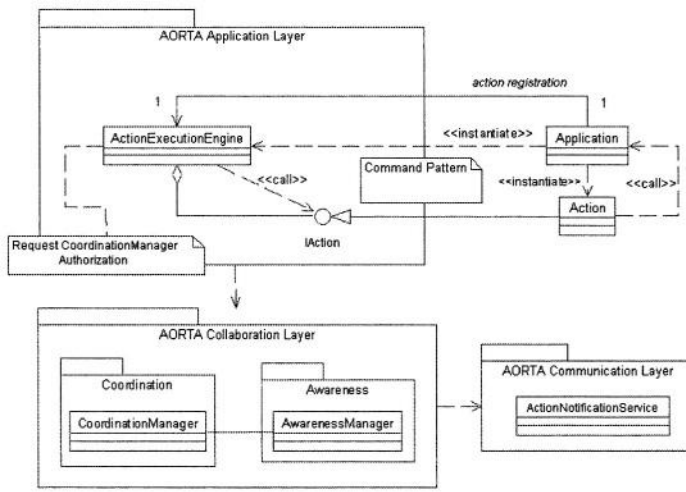

Fig. 3. UML class diagram of the AORTA application layer and its relationship with the supported CSCL/W application.

When the AEE receives an action execution request, it also adds to that action complementary information such as a timestamp (also needed for characterizing a collaborative interaction, see section 2.2). After that, the AEE sends the resulting action object to the *collaboration layer* for its evaluation (Fig. 2/Task 2). Fig. 4 shows a UML sequence diagram detailing these interactions.

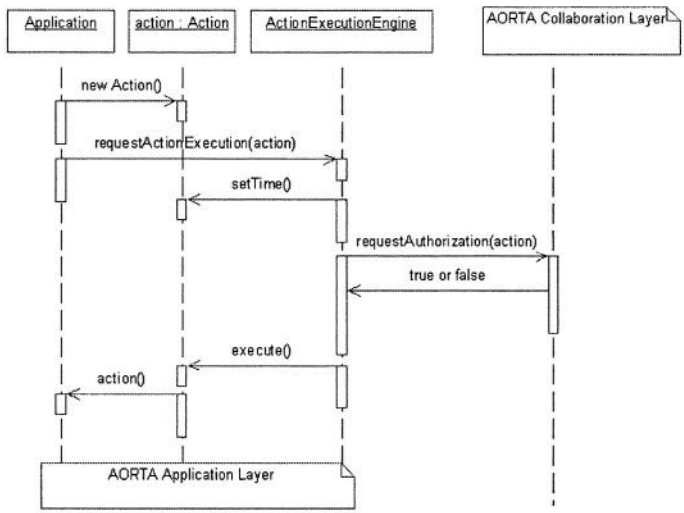

Fig. 4. UML sequence diagram describing the behaviour of the AORTA application layer and its relationship with the AORTA collaboration layer.

3.2 Collaboration Layer

This layer offers coordination and shared workspace services by means of the *CoordinationManager* (CM) and the *AwarenessManager* (AM) functional blocks.

The *CoordinationManager* (CM) is responsible for coordination-related decision making. It evaluates whether requests for action execution can be performed or not. If the evaluation is positive (the action can be executed), the CM also informs other CMs (thus maintaining a consistent *coordination state*).

The *AwarenessManager* (AM) is responsible for shared workspace awareness. It receives decisions about the execution of actions from the CM and decides whether they have to be notified to other participants' applications or not. The AM is also responsible for performing those notifications.

The AEE requests authorization to the CM for the execution of an action. The CM receives an action object and evaluates its contents. That evaluation is based on the contents of a coordination policy that has been previously loaded from a *policies repository* (Fig. 2/Task 3). The decision on which policy should be loaded is dictated by an administrative interface (although a loaded policy may dictate the loading of a new one). Coordination policies may dictate rules for turn management, concurrency conflicts, resource access, etc. Furthermore, these policies may access information regarding shared workspace awareness and adapt their state consequently in a dynamic fashion.

Once a coordination policy has been selected and loaded, the CM uses it for evaluating an action. The policy defines rules that can be used to determine whether an action can be executed or not. Those rules are based on information provided by the action (operation, user, object, timestamp), and in the actual *coordination state* associated to that policy. That *coordination state* is determined by the sequence of previously made coordination decisions. The *coordination state* is replicated in every involved CM thus enabling the local making of further coordination decisions. *Coordination state* is only updated when actions are executed (Fig. 2/Task 6).

The CM uses the result of a coordination-related evaluation to decide whether it allows the AEE to execute an action (Fig. 2/Task 4). If the action execution has been authorized, the AEE performs that execution. If not, the AEE sends an exception (Fig. 2/Task 5).

Finally, the CM updates the action object with the result of the coordination decision and sends it to the AM (Fig. 2/Task 7). The AM evaluates the content of the action object according to a previously loaded awareness policy (Fig. 2/Task 8) from a *policies repository*. Awareness policies determine what actions should be notified and what other AMs should receive those notifications. Once the policy is loaded, the AM performs the corresponding notifications by means of the NS (Fig. 2/Task 9).

3.3 Communication Layer

The communication layer only contains a functional block: *ActionNotificationServices* (ANS). The ANS is an action-based notification service responsible for propagating the occurrence of an action to all AORTA replicas.

The ANS is accessed by the AM and CM for requesting notifications sending. In the AM case, awareness-related notifications are sent to other applications interested in them. In the CM case, the ANS is used to exchange notifications regarding *coordination state* changes. In other words, CMs from the collaborating applications are synchronized through the ANS.

The ANS decouples AORTA from the use of a particular MOM (message-oriented middleware) technology (Fig. 2/Task 10).

3.4 Discussion

As presented before, AORTA bases its overall architecture in a modification of the *Command* pattern that uses actions as its key abstraction. Actions thus represent a higher level abstraction than events, and they create a seamless model for both coordination and awareness services. This is a clear contribution of this paper that can also lead to other two interesting ideas:

- Importance of patterns for CSCL/W development: Applications that use well-known patterns such as *Command* can be more easily adapted to a CSCL/W framework. One key problem in collaborative settings is to convert single-user to multi-user collaborative applications. Design patterns offer clear pointcuts for aspect oriented programming (AOP) that can help to automate such code conversions. As a clear consequence, instructing developers to use patterns help to reduce the learning curve and development cycle of collaborative applications.

- Integrate Actions as a first level service in CSCW/L frameworks: In this paper we propose a decoupled extension to a CSCW component framework. But, we can even go further and propose an action service as a first class member of any CSCL/W framework. The importance of actions justifies such service that can also help to smooth the learning curve of new applications. Furthermore, an Action service is mandatory for creating a fully reflective system. A reflective system must support both introspection and intercession. Intercession is now supported with our decoupled service, but introspection requires action metadata and policy metadata that should be provided by the proposed action service.

In conclusion, we foresee interesting research in the future regarding coordination and awareness models. The joint use of higher level abstractions, designs patterns and CBSD can lead to more advanced models enabling innovative collaborative applications.

4 Proof of Concept and Validation

In order to validate AORTA, we decided to implement a prototype on top of an existing Java-based CSCW component framework. As we mentioned before, CSCW component frameworks represent a good starting point for the achievement of those in the CSCL field.

The ANTS component groupware framework [8] was selected as the basis for the AORTA prototype because it provides interesting collaborative services and because it permits third-party extensions that clearly fit with AORTA's architectural requirements.

Section 4.1 describes the ANTS framework whereas section 4.2 introduces our AORTA prototype and *MagicPuzzle,* a CSCL applications developed on top of ANTS and AORTA that has been used to test and validate some of the AORTA's features.

4.1 ANTS Framework

ANTS framework aims to facilitate development of CSCW components by facading complex distributed services in an easy and comprehensive fashion. It follows a CBSD approach and a layered services model that eases third party extensions. ANTS comprises three main layers (see Fig. 5):

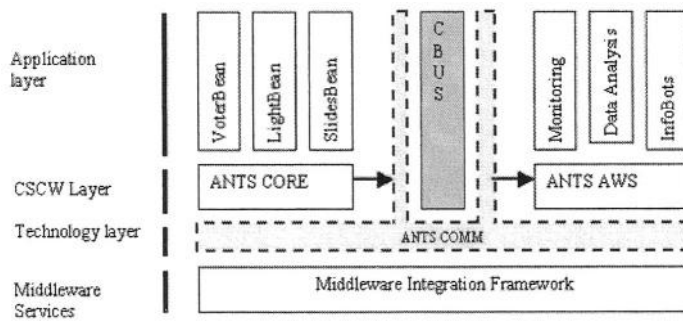

Fig. 5. ANTS framework architecture.

- At the application layer, it provides a client-side container for JavaBeans components, transparently accessing remote properties and a distributed event service.
- At the conceptual or CSCW layer, it provides essential collaborative services: shared sessions, support for synchronous and asynchronous components, security, basic coordination, and a server-side awareness infrastructure.
- At the technological level, the framework is constructed on top of a middleware integration platform (Java 2 Enterprise Edition) and facades, by means of the CBUS *(Collaboration Bus)* different notification services like Sun *Java Message Service* (JMS) or DSTC Elvin.

ANTS heavily relies on the JavaBeans component model as a basic abstraction in order to simplify development of collaborative applications. In this line, ANTS leverages and extends the JavaBeans model to provide distributed properties and events, and to access the underlying collaborative services. Furthermore, the so called collaboration bus (notification service) creates the glue between component's state propagation and awareness and actuator services listening to events in the bus.

Nevertheless, ANTS offers very low-level abstractions to deal with awareness and coordination services. ANTS directly works with events and subscriptions in the awareness service. This approach is very flexible but also more intricate for framework users. Our AORTA action model thus represents a step forward compared to previous low-level abstractions such as events.

Furthermore, ANTS aims to integrate its coordination model with the JavaBeans Vetoable properties and listener approach. This idea is coherent with the overall architecture but it also hinders development of more complex and flexible coordination models. We again outline our coordination model based on actions as an interesting contribution for both CSCW and CSCL environments.

Finally, we outline that ANTS is a suitable platform to work with, mainly because its component based architecture, and its layered services model enabling extensions at all levels of the framework.

In the next subsection, we will present how our current prototype of the AORTA architecture provides an extension to the ANTS framework in terms of coordination and awareness, and a CSCL application built on top of AORTA and ANTS that validates our proposal.

4.2 Proof of Concept: The Magic Puzzle Application

MagicPuzzle is a synchronous application that supports the collaborative resolution of puzzles by groups of primary education students. This type of CSCL applications, that are oriented to the collaborative assembling of small pieces of knowledge, provides important advantages from an educational point of view: they promote the acquisition of social abilities and they also are capable of reflecting the process of learning.

Our current prototype of *MagicPuzzle* is a Java application that uses services provided by the ANTS CORE module of the CSCW layer of ANTS as well as coordination/awareness services provided by AORTA through its application layer (see Fig. 6, which also shows the look and feel of the *MagicPuzzle* GUI). ANTS provides *MagicPuzzle* with collaborative services such as session management, shared objects management, and security. On the other hand, AORTA is responsible for object-level coordination and shared workspace awareness services. As it can also be appreciated in Fig. 6, AORTA uses the services provided by the ANTS COMM module of the

technology layer of ANTS. It is important to point out that this is not a requirement of our current AORTA prototype as it might use a different notification service. If we compare Fig. 5 and Fig. 6, it is possible to appreciate how AORTA can be considered, in this particular case, as an enhancement to ANTS that provides coordination/awareness improvements: left-hand part of Fig. 6 includes those ANTS elements of Fig. 5 used by *MagicPuzzle.*

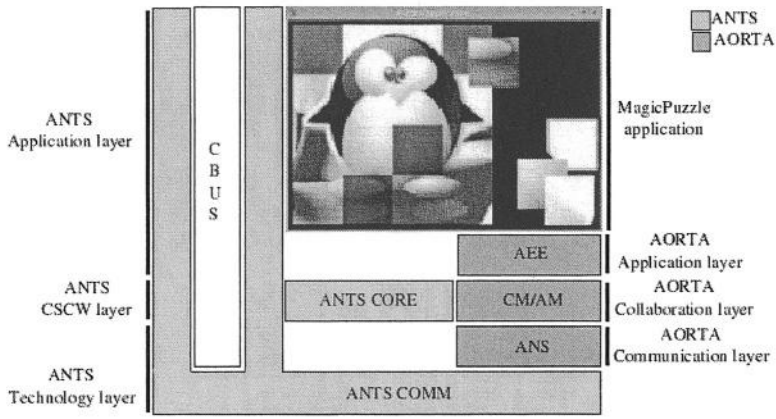

Fig. 6. Joint ANTS-AORTA support for the *MagicPuzzle* application.

The action-orientation of the AORTA architecture had several implications in the development of *MagicPuzzle.* First of all, those actions that characterize the *MagicPuzzle* behaviour were identified and subsequently coded according to the prescription summarized in Fig. 3. Two action types were considered in this first *MagicPuzzle* prototype: *SelectPiece* and *DropPiece.* A second implication consisted of a proper coding of the logic associated to the identified actions so that it could be triggered by requests generated by the AEE module of the application layer of AORTA (see Fig. 2/Task 5).

Once the *MagicPuzzle* prototype was coded (taking into account the above considerations), its behaviour with respect to coordination and awareness could be changed without requiring further modifications. These behavioural modifications were possible by just changing AORTA coordination and awareness policies. In our current AORTA prototype policies are Java classes that can be dynamically loaded by means of an administrative interface.

Several well-known coordination/awareness policies were tested in order to validate the flexibility that AORTA provides. The following paragraphs describe some of the most representative (it is a non-exhaustive list):

1. *Token* coordination policy: it restricts concurrent accesses to shared objects (puzzle pieces in the *MagicPuzzle* case). The policy registers the action that a participant wants to perform over a shared object *(SelectPiece* or *DropPiece)* and prevents accesses to that object until the requested action is performed.
2. *Leader* coordination policy: this policy contains a list of participants and the identity of the so-called *leader.* Only one participant can perform an action on any of

the shared objects at a time. The *leader* decides, once a previous action has been performed, who has the *turn* for performing a new action.

3. *Everybody* awareness policy: this policy dictates that all participants have to be aware of other participants' attempts to perform actions (even when those actions cannot be performed due to coordination decisions). Several variants of this policy (e.g. only action requests generated by a subset of participants are notified, only performed actions are notified, etc.) were also tested.

This flexibility provided by AORTA enabled the handling of complex collaboration scenarios by just changing coordination and awareness policies for *MagicPuzzle.*

Also, the decoupled nature of AORTA facilitated the development and maintenance of *MagicPuzzle.* In fact, after *MagicPuzzle* was finished, several changes that were performed into the AORTA prototype did not affect *MagicPuzzle,* thanks to the decoupling provided by the *Command* pattern.

Finally, and taking into account our previous experience developing collaborative applications (including other versions of *MagicPuzzle* using different underlying technologies), working with actions (which is the key AORTA abstraction) appeared as a more natural and easy way of developing CSCL tools: first of all, it is simpler, for developers, to update the model (or even the view) of an application from information provided by actions than from lower-level events; secondly, actions are abstractions more closely related to the CSCL domain, as it was already explained in section 2.

5 Related Work and Discussion

Many collaboration frameworks address the problem of coordination control over shared structures. Most of them also offer suitable abstractions to develop concurrent multi-user applications. We will however outline at this point that most approaches still focus on low level abstractions such as events, event ordering, locks and shared structures. Furthermore, a joint support for both coordination and awareness services is almost inexistent in many platforms.

In this context, this section analyses five existing collaboration frameworks and compares them with AORTA. Three of them, GroupKit, ANTS, and Groove, are general purpose frameworks whereas COCA and Intermezzo are particularly focused on coordination and awareness support.

GroupKit [9] is a classic toolkit for CSCW development that pioneered many advances in CSCW architectures. It offers seamless coordination support within the so called environments (shared structures) and permit third-party extensions through the "open protocols". More specifically, open protocols have three components: a controlled object (server) that maintains state, a controller object (client), and a protocol describing how they two communicate. GroupKit authors showed three examples like floor control, conference registration, and brainstorming that benefit from open protocols.

In conclusion, GroupKit provides clean and extensible support for coordination but it focuses on data structures, events and user interface widgets. Regarding awareness, they also permit event monitoring and workspace awareness widgets. We however believe that our action model represents a higher level abstraction than events and that

both coordination and awareness services achieve interconnection in a more comprehensible fashion.

ANTS was briefly described in section 4.2. We outlined that ANTS is built on top of the JavaBeans component model and provides additional services like shared data structures (bean properties) and a distributed event system (bean events and listeners). ANTS coordination model is based on *VetoableActionListeners* and distributed locks and also permits third party extensions. State propagation, coordination and the awareness service are seamlessly interconnected by the collaboration bus (notification system). As we mentioned before, ANTS also bases its overall model on events in the Collaboration bus instead of actions.

Groove (http://www.groove.net) is a peer computing framework for developing collaborative applications. It encompasses a broad set of APIs and services that devise the biggest CSCW framework ever developed. Coordination services in Groove are handled with the so-called *deltas.*

Tools in Groove transform a user gesture into a transactional unit of change called a *delta.* A *delta,* which is a container that houses one or more commands created by an engine, indicates that something has changed in a shared space. When the tool has completed writing the engine's commands into the *delta* container, it submits the *delta* to the dynamics manager for execution and eventual dissemination. The dynamics manager is responsible for executing changes to shared space data requested by tools. It is the mediator in Groove's mediated model-view-control architecture.

Despite the nice and huge Groove's architecture, it does not provide hooks for developing coordination policies. Its main framework hotspots are the Tools, and coordination control is restrained to *deltas* and transactional services. Regarding awareness, Groove also provides subscriptions for user presence and activity awareness. We understand that Groove's main goal is to offer rich services to the main hotspot of the framework: the Tools in the Workspace. They do no offer a clear extensible model for coordination and awareness, and they also focus on lower level abstractions such as user gestures. We however think that Groove's model could easily evolve towards a fully integrated action support in *deltas* and create more extensible coordination and awareness services.

Intermezzo [4] is a client-server architecture which offers collaboration services to groupware applications. Intermezzo is related to AORTA in the sense that both of them specifically provide coordination and awareness support. However, Intermezzo coordination support is based only on user access control rights on shared objects. Action information is never considered as a part of the semantics that is evaluated by coordination rules. In the case of awareness, although Intermezzo does provide workspace information, coordination services never take advantage of it.

COCA [14] is a coordination framework for developing collaborative applications. COCA provides a powerful specification language for defining coordination policies that are interpreted at run time. COCA is similar to AORTA because all distributed peers are connected by a collaboration bus. Also, coordination policies can be changed and loaded at run time to handle unexpected collaboration states. Nevertheless, it does not provide awareness services. We truly believe awareness is an important part of synchronous collaboration and that support for workspace awareness can greatly improve coordination [10].

6 Conclusions and Future Work

This paper has presented AORTA, a software architecture for enhancing CSCL/W component frameworks with object-level coordination and shared workspace awareness support. The use of action-oriented policies in AORTA in order to determine coordination and awareness behaviour enables synchronous and distributed collaborative applications to achieve a better adaptation to dynamic collaborative scenarios. Also, the use of a modification of the *Command* pattern provides a very convenient decoupling between AORTA and the applications developed on top of it. Finally, the use of action orientation provides the developers of CSCL/W with abstractions more closely related to those of the collaboration domain.

The paper has described a prototype of AORTA developed to enhance some of the capabilities of the ANTS groupware framework. Also, it has shown how a prototype of a CSCL application for the collaborative resolution of puzzles has been used to test the main features of AORTA and its software engineering implications.

Of course, there are still a lot of open research issues. In terms of validation, more CSCL application types should be developed on top of AORTA to have a better idea of its applicability. Furthermore, our current AORTA prototype has only been tested in conjunction with the ANTS framework. One of our future goals is checking its potential use in other collaborative component frameworks.

We also recognise that the availability of a policy specification language, similar to that proposed by COCA, would greatly improve AORTA flexibility. Also, it would be very useful to provide educators and even learners with tools for edition and management of their own policies.

Other interesting research lines include: the use of design patterns for providing pointcuts for aspect oriented programming so as to facilitate the conversion of single-user to multi-user collaborative applications; and the inclusion of actions as first class members of any CSCL/W framework.

Finally, we foresee promising research work in new middleware services for collaborative work. More concretely, new decentralized peer to peer abstractions like DERMI's multicalls, anycalls and manycalls [17] can help to devise more flexible and autonomous collaborative scenarios. We are also studying how to leverage existing work in collaborative systems in order to permit a smooth transition to such decentralized scenarios. Furthermore, AORTA's decoupled and replicated model considerably help us to transition to the aforementioned p2p setting.

Note that ANTS, AORTA and *MagicPuzzle* are freely available, including source code, in the ANTS web site: http://ants.etse.urv.es.

References

1. Beca, L., Fox, G. C., Podgorny, M.: Component Architecture for Building Web-Based Synchronous Collaboration Systems. Proceedings of the 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '99) (1999)
2. Dimitriadis, Y. A., Asensio, J. I., Martínez, A., Osuna, C. A.: Component-Based Software Engineering and CSCL in the Field of E-Learning. Upgrade (digital journal of European Professional Informatics Societies), special issue on e-learning - boarderless education. 4 (5) (2003) 21-28

3. Dourish, P., Belloti, V.: Awareness and Coordination in Shared Workspaces. Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work (CSCW'02) (1992)
4. Edwards, W. K.: Policies and Roles in Collaborative Applications. Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work (CSCW'96) (1996)
5. Ellis, C. A., Gibbs, S. J., Rein, G. L.: Groupware: Some Issues and Expeciences. Communications of the ACM. 43 (1) (1991) 39-58
6. Ellis, C. A., Wainer, J.: A Conceptual Model of Groupware. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94) (1994)
7. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
8. García, P., Gómez-Skarmeta, A.: ANTS Framework for Cooperative Work Environment. IEEE Computer. (2003) 56-62
9. Grundy, J., Hosking, J.: Engineering Plug-in Software Components to Support Collaborative Work. Software Practice and Experience. 32 (2002) 983-1013
10. Gutwin, C., Greenberg, S.: The Effects of Workspace Awareness Support on the Usability of Real-Time Distributed Groupware. ACM Transactions on Computer-Human Interaction. 6 (3) (1999) 243-281
11. Gutwin, C., Stark, G., Greenberg, S.: Support for Workspace Awareness in Educational Groupware. Proceedings of the 1st International Conference on Computer Support for Collaborative Learning (CSCL'95) (1995)
12. Jonsson, T., Crnkovic, I., Hnich, B., Kiziltan, Z.: Specification, Implementation and Deployment of Components. Communications of the ACM. 45 (10) (2002) 34-40
13. Koschmann, T.: CSCL: Theory and Practice of an Emerging Paradigm. Lawrence Erlbaum, Malwah, NJ, USA (1996)
14. Li, D., Muntz, R.: COCA: Collaborative Objects Coordination Architecture. Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW'98) (1998)
15. Martínez, A., Dimitriadis, Y. A., de la Fuente, P.: Contributions to Analysis of Interactions for Formative Evaluation in CSCL. In: Llamas, M., Fernández, M. J., Anido, L. E. (eds.): Computers and Education. Towards a Lifelong Learning Society. Kluwer Academic (2003) 227-238
16. Mühlenbrock, M.: Action-Based Collaboration Analysis for Group Learning. IOS Press, Amsterdam, The Netherlands (2001)
17. Pairot, C., García, P., Gómez, A. F.: Dermi: a Distributed Hash Table-Based Middleware Framework. IEEE Internet Computing (to appear). (2004)
18. Roschelle, J., Kaput, J., Stroup, W., Kahn, T. M.: Scalable Integration of Educational Software: Exploring the Promise of Component Architectures. Journal of Interactive Media in Education. (1998)
19. Suthers, D.: Architectures for Computer Supported Collaborative Learning. Proceedings of the IEEE International Conference on Advanced Learning Technologies, Madison, Wisconsin, USA (2001)
20. Szyperski, C.: Component Technology - What, Where and How? Proceedings of the 25th International Conference on Software Engineering (ICSE'03) (2003)
21. Teege, G.: Users As Composers: Parts and Features As a Basis for Tailorability in CSCW Systems. Computer Supported Cooperative Work. Kluwer Academic Publishers (2000) 101-122

Reusing Groupware Applications

Sergio F. Ochoa[1], Luis A. Guerrero[1], José A. Pino[1], and César A. Collazos[2]

[1] Department of Computer Science
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
{nbaloian,pgaldame,luguerre}@dcc.uchile.cl
[2] Department of Systems
Universidad del Cauca
FIET-Sector Tulcan, Popayán, Colombia
{ccollazo}@unicauca.edu.co

Abstract. Many groupware applications have been developed and continue being developed over white-box groupware platforms. These platforms have brought important contributions to the development of groupware systems. However, the lack of compatibility among these platforms is limiting the portability of such solutions. This paper presents a middleware, which allows to improve the portability of new and legacy groupware applications supported by white-box platforms. The middleware translates a set of functionalities provided by the groupware platforms to a set of common groupware services used by the applications. These services provide groupware support and allow to improve the portability of groupware systems. A prototype of the proposed middleware has been tested and the interim results are encouraging.

1 Introduction

Concepts like portability and reuse of components and designs have already been accepted and promoted by the software industry [1, 6]. The main advantage is the reduction in the development time and cost. In addition, an improvement in quality of the final product can be expected because of the reuse of mature designs and software components. These results have motivated the software industry to move towards the reuse of adaptable solutions that use standard interfaces, since they are easy to scale, update, and replace. Although reuse of solutions brings important advantages for the software industry, it also requires a high level of product standardization. Portability is poor in the case of groupware systems, specifically, applications supported by white-box groupware platforms. White-box groupware platforms, such as Coast [12], Groupkit [7] and TWiki [5], provide through the server, a set of classes or services implementing typical concepts of the groupware area, such as floor control, shared repository and session management. These services are not compatible among different platforms, because of the lack of standards in the groupware area. Therefore, applications using these services are not easy to reuse.

Many organizations have spent important efforts on developing and inserting groupware applications to support vital parts of their functionality (e.g. workflows, discussion forums or messaging systems). Lack of portability of their groupware applications forces organizations to avoid or delay the change of such supporting

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 262–270, 2004.

platforms. This would mean the re-development of the current applications, the re-training of the employees and the migration of the information.

This paper presents a middleware component, called GPC (Groupware Platform Compatibility), which acts as an intermediary between groupware applications and white-box groupware server platforms. GPC homogenizes the main groupware services required by applications and hides the differences among white-box groupware platforms.

Next section describes the groupware concepts that could be used by groupware applications in order to provide collaborative functionalities. Section 3 presents related work about portability of groupware solutions. Section 4 presents the main components of the GPC middleware. This section also shows how to work GPC and the restrictions required for using it. Sections 5 and 6 present the interim results, the conclusions and future work.

2 Groupware Services

Guerrero et al. have defined nine basic concepts (or patterns) which are eventually present in a groupware system to be supported by server platforms [8]. These concepts are: *sessions, users, roles, messages, objects, repositories, views, environments* and *floor control.* The *sessions* (also called work sessions, groups, or conferences) maintain information about the users who interact in an asynchronous or synchronous way through a groupware application. The *users* (or collaborators) are the members of the work group. Every user can have specific access rights according to his/her *role.* Users send *Messages* (notifications or events) to communicate. Application modules also send messages for the same purpose. *Objects* (meta-objects or information objects) are information about object instances produced by the users during group work. Most of the time, it is needed to store object attributes, which is information about the work object (or meta-information), such as the owner of the object, the creation date and time, and previous versions. The objects are stored in *repositories* and it is possible to see a portion of a repository by using various views. The *environments* organize and coordinate multiple working sessions, or multiple user groups working on the same groupware applications. They also allow sharing groupware applications among many user groups. The *floor control* policies define the strategies used to manage the shared resources.

Table 1. Common features in white-box groupware platforms.

Features	Coast	GroupKit	Habanero	JSDT	MetaWeb	TOP	TWiki
Sessions	X	X	X	X	X	X	X
Users	X	X	X	X	X	X	X
Roles		X				X	
Messages	X	X	X	X	X	X	X
Objects	X		X	X	X	X	X
Repositories	X			X	X	X	X
Views	X				X	X	
Floor Control	X	X	X	X	X	X	X
Environments	X		X	X		X	X

White-box groupware platforms, such as: *Coast* [12], *GroupKit* [7], *Habanero* [4], *JSDT* [3], *MetaWeb* [14], *TOP* [8], and *TWiki* [5], provide services to support most of these concepts (see Table 1). However, the services are not compatible among the different platforms. This generates the main problems of portability for groupware applications.

3 Related Work

Although there are no specific solutions to address the problem of portability in applications supported by white-box groupware platforms, some options designed with other goals could be used to address it. Software components [13, 6], such as: EJB, DCOM and .Net, could be used as intermediary to increase the portability of such groupware applications. These technologies have demonstrated being useful to reduce the dependencies among client and server applications. However, they involve the use of heavy-weight server platforms, and some of them are tied to a specific family of operating systems. In contrast, groupware applications and white-box platforms tend to be light-weight and independent of the operating system; therefore, the use of these component technologies could be a high cost to pay in order to improve the portability of such groupware solutions.

An alternative would be to use Web services [2], which propose an architecture based on XML messages allowing applications to talk to each other. Although the Web services paradigm is the ideal for developing portable applications, it uses connection-less interactions between client and server applications. Awareness, replication of operations, and users and sessions management are some of the groupware services that require connection-oriented interactions. In other scenarios, such as databases, operating systems and communication protocols, interesting solutions have been created to solve similar problems. The most related works are the ODBC and JDBC frameworks [10]. They reduce the dependency that an application has respect to the database it is using. This solution has demonstrated to be useful to reduce the incompatibilities among the services that databases provide to client applications. In addition, it is light-weight and independent of the operating system of both, the client and the server application. By reusing this idea, GPC middleware was created to help overcome the limitations of portability of such groupware solutions. Because most of white-box server platforms use event-based client-server architectures [9], GPC acts as an intermediary homogenizing the interaction between groupware applications and white-box server platforms. This homogenization gives portability to the applications.

4 Groupware Platform Compatibility (GPC)

The GPC middleware was implemented using the J2EE platform. The access to the functionality of the middleware was implemented through a DLL (Dynamic Link Library) that runs on the client side, similarly to an ODBC driver. Even though the rest of components of the GPC also run on the client side, they are not available to be used by external applications.

The main components of the GPC middleware are three processes and three configuration files (see Fig. 1). The processes are *Service Deliverer, Service Manager*

and *Auxiliary Service Provider*; and the configuration files are *Communication Status File* and a *Groupware Platform Specification File* and *Current Client-Server Settings.* The integrated work of these elements allows GPC to provide groupware services to the server and client applications.

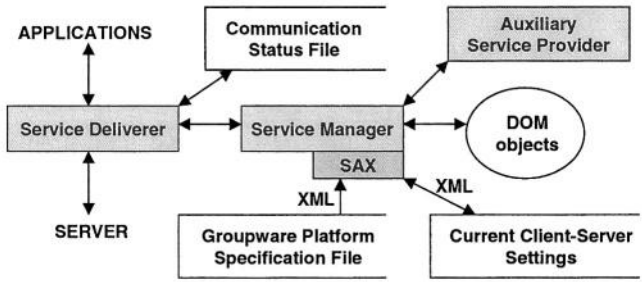

Fig. 1. Architecture of GPC.

Service Deliverer is a software agent which acts as intermediary among clients and servers, managing the communications and maintaining their status. Typically it uses the *Communication Status File* to store and retrieve information about the current communication which is being managed by GPC. This information concerns IP numbers and identifiers of client and server applications, communication protocols and ports, status of request between a server and a client, and related data. On the other hand, *Service Manager* is also a software agent, which translates the requests and responses from the original format to one understandable by the receiver. Also, it is responsible to emulate the message interchange protocol that is available in each supported white-box groupware platform. This emulation capability is needed when the granularity of the service requested by a client is different to the offered by the server. In this case, the Service Manager also acts as temporal buffer emulating the server or client behavior, and keeping track of the interactions among them.

The Service Manager carries out these tasks supported by the Communication Status File (CSF), Groupware Platform Specification File (GPSF), Auxiliary Service Provider (ASP) and Current Client-Server Settings (CCSS). CSF provides updated information to generate the communication instances (commands) between the Service Deliverer and the receivers, in order to carry out the interactions required by collaboration processes. GPSF provides general information to understand such communication instances (request or results) and assigns them a right meaning for each white-box platform. GPSF is an XML document, which also stores data about how to translate native requests or responses from one supported platform to one or more groupware services provided by GPC middleware. In addition, this configuration file provides the inverse function; this means to translate each GPC groupware service to one or more native services provided by a specific white-box server platform.

Because most of white-box server platforms do not provide the nine basic concepts which could be required by groupware applications, an *Auxiliary Service Provider* (ASP) is included in the middleware. This helps Service Manager to carry out the proposed service translation. ASP is a small server process based on the TOP white-box platform [8]. This process provides some TOP services not offered by the server platform but possibly requested from a groupware application.

Finally, the CCSS is an XML file - similar to the GPSF - storing information about the relationship between each client application and its corresponding server platform. This file is modified every time a client application changes from a server platform to another one. Using this file, the GPC middleware can manage multiple interactions among clients and servers. However, the GPC middleware may become itself a bottleneck, reducing the performance of groupware applications.

4.1 Functionality of GPC

Typically, when a groupware application sends a request (in a native format) to a server through GPC, the middleware Service Deliverer receives the message. Then, it sends it to the Service Manager, which checks if client and server share the same groupware platform. If they do, the Service Manager returns the original request to the Service Deliverer, which sends it to the server. Otherwise, the Service Manager translates the original request into a set of equivalent GPC service requests. The Service Manager uses the CCSS file to identify the specific server platform that provides support to each client application. Moreover, Service Manager uses GPSF to identify which requests will be sent to the server through the Service Deliverer and which ones will be sent to the Auxiliary Service Provider (ASP). Then, the response results provided by the server will be received by the middleware Service Deliverer and processed by the Service Manager. The results provided by the ASP are directly received and processed by the Service Manager. This software agent also translates the results into a native format for the client (understandable by the client) when it is needed. Thus, the processed results are delivered to the clients in the same way the native server would do it. The Service Manager uses SAX [11] to parse the GPSF and CCSS files and then it generates a DOM tree. This tree stays in memory in order to reduce the overhead produced by the translation process.

4.2 Implementing the Groupware Platform Specification File

One important challenge when implementing GPC middleware was to define an appropriate structure for the Groupware Platform Specification File (GPSF). This structure must be general enough in order to exclude limitations to the translation processes. A map of the resulting GPSF structure is presented in Fig. 2. It is currently implemented as an XML file.

GPSF has two entries, one through the native server platform and the other through the standard service (common groupware service). By using some of these identifiers, it is possible to retrieve the information about how to translate a service from the native server platform to GS (Groupware Services) and vice versa. For example, when the Service Manager needs to understand a service request from a client application, it should specify the platform of the client and the requested service. Being a request of service, the *request* section of GPSF provides information to validate the format of the received command and to assign it the right meaning. Thereafter, such request is translated to one or more GS provided by GPC middleware. Later, these GS are translated to a set of one or more services provided by the current server platform and/or the Auxiliary Service Provider. Such translation is done using the second entry of GPSF, which uses a GS as primary key. Finally, the translated services are delivered to the corresponding server platform through the Service Deliverer.

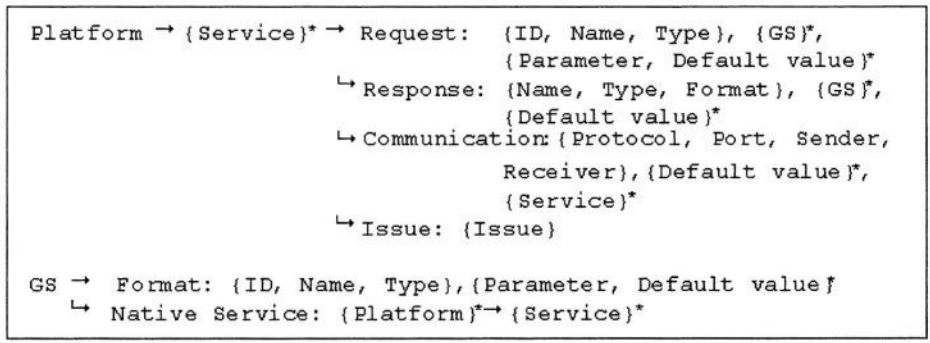

Fig. 2. Structural Map of the Groupware Platform Specification File.

Similarly, Service Manager uses the information specified in the *response* section of GPSF to translate the results from the server format to the format understood by the client. It should also emulate the interaction protocol between the client and server, by using the information specified in the *communication* section of GPSF, Communication Status File and Current Client-Server Settings.

4.3 Using the GPC Middleware

The first time GPC is included in a groupware solution, the connection to the server should be changed in the source code of the client application. This change seeks to address GPC as its new server, to indicate the white-box server platform the client application believes to be addressing, and to assign a unique identifier for such application. This allows GPC to identify each client, to assign the right semantics to receive requests and to translate the data to be delivered as result of such requests. In other words, the client application will continue believing the original white-box server platform is providing such services, and GPC is responsible to induce such beliefs. The application must change its connection function call, including its own identifier as an additional parameter in each server request. These modifications to the source code are required the first time such application is configured to work with the server using GPC as intermediary. Then, every time the server platform needs to be changed, only a CCSS file is modified.

On the other hand, server platforms should recognize GPC as a client application which is using the groupware services provided by them. It involves updating of the same configuration file included in GPC, which allow client applications change the server platform without the need to update their source code.

Initially, GPC works as a listener of the interactions between client server, and it assumes more important roles when the server platform is changed. During these interactions the middleware is in charge of encapsulating a set of typical groupware services by providing interfaces allowing client applications to talk with a server platform different than the one it was designed. Client applications can reach portability by using only these groupware services. Otherwise, if a client application uses extra groupware functions provided by a specific server platform, it must modify its source code to fit this restriction.

The groupware services provide support for collaborative work to groupware applications. These services are never accessed directly by client or server applications, but by GPC components. Usually, each request from a client application or server platform is translated to one o more groupware services. These services seek to establish a common intermediate language to translate the functionality provided by each white-box server platform.

5 Results

The current implementation of the GPC middleware supports TOP and TOP II white-box platforms. These platforms are partially compatible because many functions available in TOP were redefined in TOP II.

Three groupware applications supported by TOP were used to test the GPC middleware. The applications were: an asynchronous discussion forum, a shared drawing environment, and a notification system. The migrated applications were selected keeping in mind the incompatibilities among the services provided by these platforms.

The migration of the first application, the asynchronous discussion forum, was slow and problematic because of incomplete implementations of the Groupware Platform Specification File (GPSF) and the Service Manager. The implementation and appropriate use of these two components were challenging tasks. Once the first application was translated, the rest of the applications were easy to migrate. The effort spent to adjust the source code of the client application was within the estimated limits. However, small adjustments were required to the GPC middleware in order to better fit the migration process. The authors' feeling is that most of the problems were because it was the first test for the GPC middleware; therefore, many problems were made clear in such instance. In addition, the lack of expertise in the setting of the GPC middleware was another cause for problems encountered during the first migration process. Currently, most of the services provided by the JSDT platform [3] have been included in the current version of GPC middleware. They have been partially tested and successful results have been obtained. The process to include support for JSDT in the middleware was, at the moment, faster and easier than for the previous platforms.

The authors believe an important effort will be required to include support for every white-box groupware platform. However, once the support is well implemented, the groupware applications migration will be fast and with low cost.

6 Conclusions and Further Work

Many groupware applications developed on top of platforms supporting collaborative work are limited in their portability due to the lack of compatibility among the services they provide. Currently there is not a specific solution to help improve the portability of groupware applications supported by white-box server platforms. This paper presents the GPC (Groupware Platform Compatibility) middleware, which offers an option to overcome some of these limitations using a set of common groupware services. GPC works as a bridge between the groupware applications (clients) and the white-box groupware platforms (servers). This hides incompatibilities among the

requested and the given services. GPC can be seen like an ODBC/JDBC driver, which translates requests and responses from clients and servers to a compliant format and meaning. This strategy has strengths and weaknesses. An important strength is the small changes in the source code of groupware applications, and a small set up effort required by the new solution. In addition, this solution is light-weight and independent of the operating systems used by groupware application and server platforms. On the other hand, a limitation of GPC is that it does not take maximum advantage of the server platform full functionality, since the collaboration should be based on the groupware services provided by the middleware. In addition, there would be a performance reduction for groupware applications using this solution. Despite the limitations, GPC provides a way to improve the portability of groupware applications. The current supported white-box platforms are TOP and TOP II Three applications have been ported from TOP to TOP II platform. The preliminary results have shown a high portability of these applications. Although it is early to obtain conclusions, important advantages on reuse of groupware applications could be obtained if the initial results endure. Currently, the authors are working to include support for JSDT [3] soon and other platforms later, in order to get well supported conclusions on the portability features provided by GPC.

Acknowledgments

This work was partially supported by Fondecyt (Chile) grants No. 1040952 and 1030959 and by MECESUP (Chile) project No. UCH0109.

References

1. Brown, A., Large-scale component-based development. Object and Component Technology Series, Prentice Hall., (2002)
2. Burner, M., The deliberate revolution: creating connectedness with XML Web services. ACM QUEUE. 1 (1), (2003), 28-37
3. Burridge, R., Java Shared Data Toolkit: user guide. Sun Microsystems, Inc., (1998)
4. Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., Seguin, C., Java object-sharing in habanero. Comm. of the ACM 41, 6, (1998), 69-76
5. Fabre,Y., Pitel, G., Soubrevilla, L., Marchand, E., Géraud, T., Demaille, A., Asynchronous architecture to manage communication, display, and user interaction in distributed virtual environments. Proc. of EGVE'2000, J.D. Mulder and R. van Liere (Eds.). Computer Science/Eurographics Series, Springer-Verlag, (2000), 105-113
6. Gokhale, A., Natarajan, B., Schmidt, D., Wang, N., Modeling and synthesis of middleware components. Communications of the ACM, Special Issue on Enterprise Components, Services and Business Rules, edited by Ali Arsanjani, (2002)
7. Greenberg, S., Roseman, M., Groupware toolkits for synchronous work. Beaudouin-Lafon, ed., Computer-Supported Cooperative Work, Chapt. 6, John Wiley & Sons, (1999), 135-168
8. Guerrero, L., Fuller, D., A pattern system for the development of collaborative applications. Information and Software Technology 43, 7, (2001), 457-467
9. Ochoa, S., Guerrero, L., Fuller, D., Herrera, O., Designing the communications infrastructure of groupware systems. In J. Haake, J.A. Pino (eds.): Groupware: Design, Implementation and Use. Lecture Notes in Computer Science 2440, (2002), 114-123

10. Peterson, R., Database development with Jdbc, Odbc and SQL/Sqlj. Sams Publishing, (2001)
11. Devsphere., SAX + DOM Mix = SAXDOMIX. URL: www.devsphere.com/xml/saxdomix/
12. Schuckmann, C., Schümmer, J., Seitz, P., Modeling collaboration using shared objects. In: S. C. Hayne (ed.): Proc. of ACM SIGGROUP Conf. on Supporting Group Work (GROUP'99). Phoenix, Arizona, USA, 189-198, (1999)
13. Szyperski, C., Component software. Addison-Wesley, (2002)
14. Trevor, J., Koch, T., Woetzel, G., MetaWeb: bringing synchronous groupware to the World Wide Web. Proc. of ECSCW'97, Lancaster, (1997)

Distributed Dynamic-Locking in Real-Time Collaborative Editing Systems*

Xianghua Xu, Jiajun Bu, Chun Chen, and Yong Li

College of Computer Science, Zhejiang University
Hangzhou 310027, P.R.China
{xuxh_cs,bjj,cchen,lyzju}@cs.zju.edu.cn

Abstract. In this paper, a Customizable and Dynamic Locking (CDL) scheme is proposed for concurrency control in Internet-based real-time collaborative editors. The idea of dynamic-locking is that: locking mechanism dynamically determines locking set according to locking policies and latest collaborative activities happened in the shared workspace, and pre-locks objects for succeeding editing and preventing from other user's edit. Dynamic locking is optional: user decides whether and when to use locking mechanism. In the proposed scheme, locking policy is separated from locking mechanism. Locking policies can be customized for different collaboration tasks. Locking scope is dynamically determined according to locking policies and collaborative activities among users. Multiple users can select different policies in the same collaborative session, and change locking policies at different phases of collaboration as well. Protocols and algorithms for locking conflict resolution and consistency maintenance are also presented in this paper.

1 Introduction

Internet-based collaborative editing systems are a special class of distributed applications which allow a distributed community of users to simultaneously edit a shared document[9]. A replicated architecture is usually adopted in which the shared documents are replicated at local storage of each collaborating site. With the replicated architecture, multiple users can concurrently edit their local copies of the shared document and get their operations reflected on their local interfaces immediately. One of the most significant challenges in the design and implementation of replicated collaborative editing systems is consistency maintenance of replicated documents in face of concurrent updates [10].

Locking is a widely used concurrency control technique in distributed computing and database systems to ensure data integrity by prohibiting concurrent conflicting updates on shared data objects [8]. Locking has also been used for consistency maintenance of shared documents in various collaborative editing systems. From users' points of view, locking can be classified into two categories: compulsory locking and

* This material is based upon work funded by Zhejiang Provincial Natural Science Foundation of China under Grant No. Z603231.

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 271–279, 2004.

Table 1. Compulsory-optional vs explicit – implicit.

	Explicit	Implicit
Compulsory	√	√
Optional	√	

optional locking. Locking is compulsory in the sense that a lock must be requested before editing an object, whereas a system with optional locking allows users to update any unlocked objects without necessarily requesting a lock on it. Locking can also be classified into explicit lock and implicit lock. In explicit locking, lock and unlock are requested by a user before and after an editing operation, whereas implicit locking is requested by systems when an object is selected and released after deselecting. The relationship between compulsory-optional and explicit-implicit locking is shown in Table 1. The existing compulsory locking can be explicit or implicit either, whereas optional locking can only be explicit. In previous researches, the optional-implicit locking was unmentioned.

In explicit locking scheme, the granularity can be object, region, or whole document. It's determined by users at runtime. In implicit locking scheme [7], the granularity is statically determined by systems in advance. Obviously, explicit locking is more suitable for collaborative environment as it is more flexible than implicit locking. However, explicit locking leaves too much burdens on user since users need to request for lock before updating. Comparatively, implicit locking mechanism releases users from locking and unlocking, while losing flexibility and adaptability.

In optional locking, when to lock and how to lock are all decided by user at runtime. Because multiple users are often editing different regions in many occasions, these will not cause concurrent editing conflict problems. Therefore, optional lock is more appropriate choice for collaborative application than compulsory locking. However, this requires users to entirely determine when to lock and how to lock. Moreover, multi-users' working areas are often interweaving with each other and constantly changing in the evolution of collaboration. Hence, how to determine the locking scope is a burden on user.

Therefore, we propose a dynamic locking approach in this paper. The main idea is: users can define the locking policy to be used in the system and decide when to use dynamic locking; lock mechanism dynamically determined locking scope at runtime according to the locking policies user defined. The key issues in dynamic locking approach is how to dynamically determine the locking scope and how to resolve the locking conflict and maintain the consistency of locking data in a distributed collaborative environment. In this paper, we focus attention on the locking conflict problem. How to determine the appropriate locking scope will be discussed in another paper.

2 Dynamic-Locking Approach

2.1 Basic Definitions

A real-time collaborative editing system can be represented as *(S, U),* where *S* is the *shared workspace, U* is a set of participants. There is a collection of *objects* (*O*) exist-

ing in *S* shared by *U*. *S* is a metaphor for compositing together with *U* to manipulate a collection of objects (*O*) and their attributes. The concepts related to user's activities in *S* are defined as follow:

Operation (Op) is the action issued by user to manipulate the objects. Each operation is stamped by time-vector as defined in [3, 5]. A dependent (causal) relationship or independent (concurrent) relationship between any two operations can be determined according to operation's time-vector [9]. Further more, a real-time collaborative editing system is said to be consistent if it always maintains the three properties: convergence, causality-preservation and intention-preservation [10].

Location (Loc) is an object that the user currently selects or edits.

Lock set (Ls) is a set of objects locked by a user. For each user, a current locking set is maintained at each site, which represents the objects currently locked by that user.

Locking point (Lp) is defined as a locked object that is currently edited by lock owner. A user's Lp is a member of his locking set and also is user's Loc.

Context value (Con) represents U's relevancy degree on the local context affected by Op. *Con* value is used in calculation of locking set and conflict resolution protocol.

Lock table (Lt) is maintained at each site. There is an entry in lock table for each user, which points to the user's lock set.

2.2 The Architecture of Dynamic Locking Model

Locking mechanism is the core component in dynamic locking. When a user needs to utilize the locking mechanism, it will be enabled, and the specific locking policies are selected. The locking procedure is described briefly as follow:

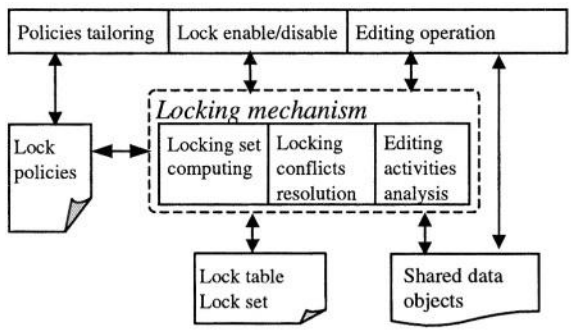

Fig. 1. Architecture of dynamic locking.

When a user selects, edits or creates an object, the *locking mechanism* calculates the user's locking set according to locking policies and user's latest editing activities, then, locks objects belong to the locking set, and then, time-stamp and broadcast the locking operation to other sites. When the user begins to edit next object, new locking set will be recalculated and locked, and the previous locks will be released.

When a remote locking operation is received, the *locking mechanism* performs conflict-check with operations in HB. If conflict occurs, it will be resolved according to the conflict resolution protocol. Otherwise, objects belonging to the locking set will be locked by the remote user.

For each editing operation, the locking mechanism does some works of activity statistics. The statistic result is used by itself for the calculation of locking set.

2.3 Example of User's Activity Analysis

When multiple users participate in an editing or design task (except the task like brainstorm), each user is often responsible for a subtask assigned. Although the subtask may be only a rough division of the collaboration task, the history of editing operation will present out the pattern of task division, and, such pattern can be represented as rough and irregular division of the shared workspace. Hence, we can extract the pattern by the analysis of users' activities. Here we propose a simplified method for a 2D shared workspace.

First, we divide the 2D shared workspace into M x N small rectangle grids, denoted as S[M][N]. For each collaborative session, we maintain an array, defined as $AF^U[M][N]$, for each user U. $AF^U[M][N]$ is used to express the frequency of U's editing activity affect on S[M][N]. Hence, $AF^U[M][N]$ represents a subtask pattern of user U mapped on S[M][N]. $AF^U[m][n]$ expresses the frequency of U's editing activities at sub-workspace S[m][n]. For a group users: u1, u2 and u3, which participate in the same editing session, $AF^{U1}[M][N]$, $AF^{U2}[M][N]$ and $AF^{U3}[M][N]$ represent the pattern of group editing work.

For a given user U and a Op issued by U:

$$AF^U[i][j] = AF^U[i][j] + 1, \text{ for all } S[i][j] \cap Op, \text{ where } I \in [0,M-1], j \in [0,N-1]$$

For an Op issued by user U, U's context relevance (Cr) for Op is defined as:

$$Cr^U = \sum_{\forall S[i][j] \cap Op} AF^U[i][j], i \in [0, M-1], j \in [0, N-1]$$

Given a group of users, $u_1, u_2, \ldots, u_k$, participating in a group editing session. For an operation Op generated by user u_s, the *local context value (Con)* of Op is defined as:

$$Con_{U_s}^{Op} = \frac{Cr^{U_s}}{\sum_{t=1, t \neq s}^{K} Cr^{U_t}}, \quad Con_{U_s}^{Op} \in [0,1]$$

The context value *(Con)* represents U's relevance on the local context affected by Op. *Con* value is used in calculation of locking set and conflict resolution protocol in following section.

2.4 Locking Policy and Locking Set

Locking policy and *Context* value are two key factors in computation of locking set. Users can describe specific locking policy according to characteristics of application and subjective requirement. Locking policy is expressed as:

```
Policy(PolicyName, Type, [ [scale] | [prediction] ])
Type = [ Rect | Circle | Irregular | Bool ]
Scale = [ RegionSize: 1,2,…,n]
Prediction = [Attr(Obj) = value] |[Prediction [ Bool ] Prediction]
             [BoolPolicyName]
Bool = [and | or]
```

There are three types of locking policy: object's geometry constraints (rectangle, circle, irregular); object's attribute constraints (such as object owner, owner's privilege); combination of geometry and attribute constrains. In geometry constraints, a max region size and a scale are defined. The max region is dynamically mapped into actual region at runtime: (max-region, scale) X (*Con*-value) → (actual-region). Here we illustrate the locking policy with following examples:

- Policy(rule1, Rect, [[200,200]: 1, 2, 3]), define a rectangle scope which will be mapped to a actual scope by scale [1,2,3] and *Con.*
- Policy(rule2, Bool, [Owner(Op) = Owner(O)]), describe an attribute constraint: only the owner's objects included.
- Policy(combinedRule, Bool, [rule2 and rule3]), define a combined rule of rule1 and rule2.

Computation of locking set: when a policy was enabled, locking mechanism is responsible for determination of locking set. When a user is selecting or editing an object, a current locking set is calculated out simultaneously, then the objects belonging to the set are locked immediately and locking set is broadcast to other sites. However, the locking set may conflicts with a concurrent editing Op or locking set received from other sites in face of concurrent locking and editing operations.

3 Dynamic Locking Protocol and Conflict Resolution Protocol

3.1 Dynamic Locking Protocol and Algorithm

Dynamic Locking Protocol (DLP):

1. When a Locking set is generated at a local site:
   Locking set will be granted at local site and propagated to all remote sites. A locking subset, which belongs to old Locking set but not to new Locking set, is released.
2. When a Locking set arrives at a remote site:
   a. If it does not conflict with any Op in HB (History Buffer of executed operation), the new Locking set is granted and compared with old Locking set. A locking subset, which belongs to old Locking set but not to new Locking set, is released.
   b. If it conflicts with Ops in HB, the conflicts are resolved by conflict resolution protocol.

   According to DLP, we present a Dynamic Locking algorithm:

```
Function DynamicLock(Op, U)
/* Op: a local/remote operation currently executed. U:
Op's issuer; Ls(U): U's locking set, depicts a group of
object locked by U;  Loc(U): the location of U in S; HB:
history buffer of executed operation. */
//for the local generated operation
If (Op is a local op) Then
 //How to compute context value and locking set is
 //application-oriented problem, hence the algorithms
 //will not given here
 Con = ComputeContext(Op);
 NewLs = ComputeLs(Op);
 //a subset of objects needs to be locked and unlocked
 Ladd = NewLs - Ls(U);
 Lfree = Ls(U) - NewLs;
 Ladd->lock();
 Lfree->unlock();
 // Pack Ls and Con into Op which will be sent to remote
site
 Op->Ls = NewLs;
 Op->Con = Con; Broadcast(Op);
 //setting the new lock set of U
 Ls(U) = NewLs;
 HB->Add(Op);
Else //for the remote operation
 //If Op is a concurrent operation with other operation
 If (IsConcurrent(Op, HB)) Then
  Resolve conflict by LCRP protocol;
 Else
  NewLs = Op->Ls;
  //a subset of objects needs to be locked and unlocked
  Ladd = NewLs - Ls(U);
  Lfree = Ls(U) - NewLs;
  Ladd->lock();
  Lfree->unlock();
 EndIf
 HB->Add(Op)
EndIf
```

3.2 Locking Conflict Resolution Protocol and Algorithm

The concurrent conflicts are resolved according to *Locking Conflict Resolution Protocol (LCRP):*

For each Op in HB conflicting with remote Op,

1. If Op's Lock point is same as remote op's Lock point, then conflict is resolved by negotiation.
2. If Con(Op) = Con(remoteOp) then the conflicting subset of Ls(U(Op)) / Ls(remoteOp) will be released/ungranted.
3. If Con(Op) > Con(remoteOp) then the conflicting subset of Ls(remoteOp) will be ungranted.
4. If Con(Op) < Con(remoteOp) then the conflicting subset of Ls(U(Op)) will be released.

Fig.2 illustrates conflicts may be occurred in dynamic locking, yellow and green areas represent the locking set of op1 and op2 respectively, blue and red dots represent the locking points of op1 and op2 respectively. Suppose Con(op1)=Con(op2). In fig.2a, both locking points are out of conflict part of locking set, the conflicts are resolved according to rule 1 of LCRP. In fig.2b, both locking points are the same object, it is resolved by negotiation (rule 2). In fig.2c, op2's locking point is within the conflict part, it's resolved according to rule 2, but u2's locking point is still included in u2's locking set. In fig.2d, both locking points are in the conflict part, it's resolved according to rule 2, both locking points should be still included in their locking set respectively. Conflicts between locking and editing operation could be resolved similarly.

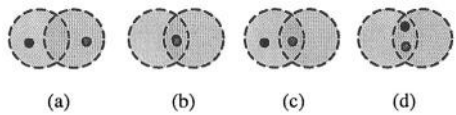

Fig. 2. A scenario of locking conflicts.

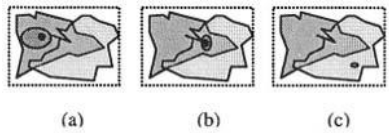

Fig. 3. A scenario of dynamic locking.

Fig.3 illustrates a simplified scenario of dynamic locking. Green and yellow are working areas of u1 and u2 respectively. Blue area represents u1's locking area and the size of lock set. When u1 edits objects in his working area (Fig.3a), his locking area will be max size defined in locking policy. When u1 edits objects in the intersected area of u1 and u2 (Fig.3b), his locking area will be adjusted to medium size. When u1 edits objects in the working area of u2 (Fig.3c), his locking area will be adjusted to minimum size.

4 Comparison to Related Work

A variety of locking schemes have been proposed to maintain consistency in collaborative editing systems. Ensemble[7], GroupKit[4], Suite[6] and REDUCE[8] are closely related to our work.

Ensemble adopts implicit-optimistic locking scheme, whereas GroupKit adopts explicit-optimistic locking scheme. They only support single locking granularity predefined in systems, hence these schemes cannot meet the requirement of collaboration because of lacking of flexibility.

Suite [6] provides multi-granularity locking on a general structured data model. It employs a flexible concurrent control model to associate data semantics. When conflict occurs, conflict resolution rules examine the operation rights to determine who gets the object. Specific lock policies use heuristics to decide whether a lock in use should be taken away and granted to another requester. Suite is an optimistic and compulsory locking scheme with multi-granularity. The granularity can be dynamically decided at runtime according to predefined policy and data semantics. However, Suite is designed for structured data model, hence it not suitable for semi-structured or unstructured data model adopted in many collaborative applications.

In Ensemble, GroupKit and Suite, locking is compulsory, in which users need to passively abide the locking constraints implemented in system and have to use locks before editing. In human-human collaborative interaction, however, when to lock and how to lock mainly depend on the collaborative users and the context they shared. Hence, the locking scheme should be user-centric, and it should be flexible so that user can decide when to lock and how to lock according to dynamic collaborative environment. Moreover, they all adopt a centralized server to resolve the locking conflicts.

The optional locking, presented in REDUCE [8], GRACE [2], is most related to our work. Optional locking is also a distributed, high responsive locking approach used in collaborative environment. However, the locking conflict is resolved by Coordinator-Based Protocol by means of a centralized coordinator and a locking transformation algorithm [8]. In our work, the locking conflicts are resolved in a fully replicated architecture, and no coordinator is needed. Moreover, in optional locking, the locking set is decided by users, but the locking set is dynamically determined by locking mechanism in our approach.

[1] presents an adaptive multi-granularity locking scheme used in object-oriented database. Multiple logical granularity units are chosen with the knowledge of the data model and changed dynamically during the execution of a transaction by means of escalation and de-escalation technique. It aims to improve the OOB' s throughput. However, it's not suitable for real-time collaborative system, because collaborative systems are more sensitive to response-time than to system throughput, and required to support unstructured activities with dynamic and context-specific consistency requirements.

5 Conclusions

In this paper, a Customizable and Dynamic Locking (CDL) scheme is proposed for concurrency control in Internet-based collaborative systems. CDL is fully distributed, highly responsive, dynamic and tailorable locking mechanism. In CDL scheme, locking policy are separated from locking mechanism. Locking policies can be customized to meet the requirements of users and application. Locking scope is dynamically determined according to locking policies and collaborative activities among users. Multiple users can select different policies in the same collaborative session. A user can change locking policies at the different phases in the same session. CDL is also optional: whether and when to use the locking mechanism is determined by users. When the CDL is enabled, it automatically decides the dynamic locking scope according to latest activities among collaborative users and locking policies selected. Protocols and algorithms are devised to resolve locking conflict and maintain consistency of global locking status.

The CDL scheme is implemented in CoDesign [11] to verify the feasibility. We are currently investigating the interface, activities analysis and usability issues related to CDL, including how the locking status is presented to the user, how to tailor the policy for different phases of collaborative works and users' subjective evaluation in

using CDL. The implementing issues, such as approach of activities analyzing, locking set calculation, conflicts resolution and consistency maintenance algorithm, etc, will be discussed in a follow paper.

References

1. C.T.K. Chang, *Adaptive Multi-Granularity Locking Protocol in Objectoriented Databases,* PhD Dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, 2002
2. D. Chen and C. Sun. Optional and Responsive Locking in Distributed Collaborative Object Graphics Editing Systems. *Proceedings of the First International Conference on Web Information Systems Engineering (WISE'00).* p.414-418.
3. C.A. Ellis and S.J. Gibbs. Concurrency Control in Groupware Systems. *In Proceedings of the ACM SIGMOD Conference on Management of Data,* May 1989. Seattle, WA, USA. p.399-407.
4. S. Greenberg and D. Marwood. Real Time Groupware as a Distributed System: Concurrency Control and Its Effect on the Interface. *Proceedings of the ACM CSCW Conference on Computer Supported Cooperative Work,* October 22-26. North Carolina: ACM Press.
5. L. Lamport, Time, Clocks, and the Ordering of Events in a Distributed System. *CACM.* 21,7 (1978): p. 558-565.
6. J. Munson and P. Dewan. A Concurrency Control Framework for Collaborative Systems. *In Proc. of ACM Conference on Computer Supported Cooperative Work,* Nov. 1996. p.278-287.
7. R.E. Newman-Wolfe, et al. Implicit Locking in the Ensemble Concurrent Object-Oriented Graphics Editor. *Proceedings of the ACM conference on Computer supported cooperative work,* November 1992. Toronto, Ontario, Canada. p.265-272.
8. C. Sun, Optional and Responsive Fine-Grain Locking in Internet-Based Collborative Systems. *IEEE Transactions on Parallel and Distributed Systems.* 13,9 (2002): p. 994-1008.
9. C. Sun and D. Chen, Consistency Maintenance in Real-Time Collaborative Graphics Editing Systems. *ACM Transactions on Computer-Human Interaction.* 9,1 (2002): p. 1-41.
10. C. Sun, et al., Achieving Convergence, Causality-Preservation, and Intention-Preservation in Real-Time Cooperative Editing Systems. *ACM Transaction on Computer-Human Interaction.* 5,1 (1998): p. 63-108.
11. 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.

This page intentionally left blank

A Model for a Collaborative Recommender System for Multimedia Learning Material

Nelson Baloian[1], Patricio Galdames[1,2], César A. Collazos[3], and Luis A. Guerrero[1]

[1] Department of Computer Science
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
{nbaloian,pgaldame,luguerre}@dcc.uchile.cl
[2] Department of Electrical Engineering
Universidad de Chile
Tupper 2007, Santiago, Chile
[3] Department of Systems
Universidad del Cauca
FIET-Sector Tulcan, Popayán, Colombia
{ccollazo}@unicauca.edu.co

Abstract. In a cluster of many servers containing heterogeneous multimedia learning material and serving users with different backgrounds (e.g. language, interests, previous knowledge, hardware and connectivity) it may be difficult for the learners to find a piece of material which fit their needs. This is the case of the COLDEX project. Recommender systems have been used to help people sift through all the available information to find that most valuable to them. We propose a recommender system, which suggest multimedia learning material based on the learner's background preferences as well as the available hardware and software that he/she has.

1 Introduction

As the amount of available information in the world increases, recommender systems are becoming more important to help us receive that information which is more important to us. Recomender systems may be based on content analysis or on collaborative filtering. In the first case the content of a document is usually analyzed automatically to extract its relevant characteristics by the application of different metrics. The characteristics are then compared with those desired. In the second case, these systems base their decisions on opinions given by users that previously. Both types of systems use user-modeling methodologies. Some techniques, which have been successfully used to accomplish this task, are Bayesian networks with Markov chains [15] and similarity metrics de [14]. In a computer supported learning environment consisting of many servers located in different parts of the world containing rapidly changing learning material of very heterogeneous characteristics a user will certainly have some problems locating suitable learning material for him/her. Moreover, if the learners have very different backgrounds, learning needs, equipment, and connectivity a recommender system may be a very useful tool. This is exactly the case of the COLDEX [6] project, which aims to develop distributed collaborative learning envi-

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 281–288, 2004.

ronments by remote distributed experimentation for learning communities in Europe and South America. The COLDEX network consists of interconnected networks of server nodes, each one supporting a particular learning community. Experimental equipment however maybe distributed across the entire network. For example, there are telescopes in Chile and Spain, Seismographs in Chile, and a greenhouse in Sweden, all them connected to the network. Appropriate software provides interfaces for collaboratively work and learn with them.

In this paper we present a preliminary work which aims to develop a recommender system for the above-described environment. Since the learning material consists mainly in learning objects of different multimedia nature, an automatic analysis of the content is not practicable. Therefore human experts do a first characterization of the learning objects of the network. It is important to note that this characterization aims to capture the essential learning potentials of a learning object, as opposed to [7], where a methodology for evaluating multimedia is described which takes in account only the presentation and usability of the material. There have been some works aimed to predict the degree of acceptation a user may have of a multimedia file, like the work presented in [13] with images but it does not consider a the importance this material has for the user in the context of his/her work.

This work proposes a methodology for characterizing multimedia learning material based on the use of collaborative techniques in order to define a vector of characteristics for a certain document. This vector will reflect the opinion the people who have seen this document before and will evolve as new people express their opinion about the document. However, not all users will get the same vector as description of one document. In order to construct the vector for a certain user, the opinion given by those with similar interest will have more weight. Current recommender systems mostly do not use implicit ratings, nor is the ability of implicit ratings to predict actual user interest well understood. An adaptive method should be able to learn and "calibrate" the learner's preferences based on her/his behavior. Apart from the preferences about the content of the learning material expressed by the learner the system will also consider the possibilities he/she has to display/perform a certain kind of learning object. This corresponds to the characteristics of the software, hardware and connection the learner has available by the moment a searching requirement is expressed or a certain learning material suggested. These might or might not be taken in account by the learner in order to decide whether to download the material or not. The next section presents some works related to recommender systems. In section 3 we illustrate the model proposed. Section 4 describes how the model works in the real situation of the COLDEX project. Finally section 5 presents some conclusions and further work.

2 Related Work

A variety of collaborative filters or recommender systems have been designed and deployed. The Tapestry system relied on each user to identify like-minded users manually [10] and is one of the earliest implementation of collaborative filtering-based recommender systems. However, because this system depends on each person knowing the others, it is not suitable for large. Later several rating-based automated recommender systems were developed. Grouplens (Resn94) and Ringo (Shar95) developed independently systems, where the first CF algorithms for automatic predic-

tion were used. The work of Breese et al. [5] identifies a general class of Collaborative Filtering algorithms called model-based algorithms. The authors describe and evaluate two probabilistic models, which they term the Bayesian clustering and Bayesian network models. In the first model, like-minded users are clustered together into classes. Given his/her class membership, a user's ratings are assumed to be independent (i.e., the model structure is that of a naive Bayesian network). The second model also employs Bayesian network, but of a different form. Variables in the network are titles and their values are the allowable ratings. Bayesian networks create a model based on training set with a decision tree at each node and edges representing user information. The model can be built off-line over a matter of hours or days. The resulting model is very small, very fast, and essentially as accurate as nearest neighbor methods [5]. Other technology that has been used is Horting, that is a graph-based technique in which nodes are users, and edges between nodes indicate degree of similarity between the users [1]. Walking the graph nearby nodes and combining the opinions of the nearby users produces predictions.

3 The Systems's Model

The recommender system will be used for performing the following tasks:

1. An agent proactively proposes certain learning material to the learner, triggered by a tutor of the learning community.
2. The user makes a search of relevant learning material based on a keyword list.
3. Given a document, the system evaluates it for the user according to her/his profile.

Now we will describe the principal components of the model on which the recommender system will be based and their functionality.

3.1 Learner's Profile or Metadata of the Learner

The learner's profile is used to describe the characteristics of a certain learner in order to make an automatic evaluation of the available learning material in order to filter documents which will be of no use to her/him. The historical registers about the material the user has selected and evaluated in the past, as well as the preferences declared explicitly by her/him will automatically generate the description of the user's interests. In our system we consider two types of properties describing the learner's profile. On one hand we have the *user preference properties,* which describe the learner's preferences for a certain type of material, and on the other hand, we will have those describing the hardware and software which the learner has available to display it. We will call these ones the *user hardware properties.* The preference properties we are going to consider are: interest fields, described by a list of keywords, preferred multimedia format, language, date of profile creation, author, age, expected difficulty, expected time of learning, semantic density and context. The values for these properties are the components of the preference properties vector UPPi. Figure 1 shows the XML description, which is used to characterize this vector.

```
<userProfile
xmlns="http://www.d.cl/~pgaldame/tesis"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.d.cl/~pgaldame/tesis
http://www.d.cl/~pgald/userProfile1.xsd">
    <identifier id="000001">
        <version/>
    </identifier>
    <topic>
        <name>astronomy</name>
        <degreeTopic value="4"/>
    </topic>
    <topic>
        <name>chess</name>
        <degreeTopic value="5"/>
    </topic>
    <format/>
    <downloadTimeExpected tolerance=""/>
    <educationalLevel>
        <source/>
         <value> high school student</value>
    </educationalLevel>
    <expectedifficulty>
        <source> CS Thesis <source/>
        <value> medium </value>
    </expectedifficulty>
  </userProfile>
```

Fig. 1. Example of a user's preference profile.

The characterization of the learner's available hardware profile is done according to the CC/PP recommendation [11] and defines the user hardware properties vector UHP_i. Figure 2 describes the content of this vector and some example values. Therefore a user's profile UP_i is completely defined by the two vectors $\{UPP_i , UHP_i\}$. While UPP_i is defined and maybe normally occasionally changed by the user, UHP_i can be completely automatically defined by the system every time the user logs in or manually defined by the user, since it can vary as often as the learner changes the computer in which he/she is working.

3.2 Metadata of the Multimedia Learning Material

In our system all the learning material will have an associated metadata for description. In order to facilitate their manipulation these will be divided in metadata describing the content itself and those describing its contribution to learning a certain subject. Learning objects will be classified in a certain class according to the principal learning field they are supposed to be used for. A class is composed of a set of items describing more specific topics of the learning field. A document will be evaluated according to the contribution it makes for the learning of each topic of the field of the class it belongs to. This evaluation will be made collaboratively by all learners that used that learning object before and will be adapted to the preferences of the user to which this evaluation will be presented. For describing the learning object from the

Hardware	*Processor:* Computer's processor. Example values = {PPC, Intel-Pentium X; Athlon, Motorola}. *Memory*: Computer's RAM size in MB. *Screen:* Screen Resolution in pixels. *Free_hard_disk_space*: in MB.
Software	*Sound:* values = {ON, OFF} *Images:* Image viewing support, values ={ yes, no} OS : Operative System, values = {pc-dos, ms-windows, ..., other} *Browser*: Installed Web-browser, state={any, netscape, communicator, microsoft_internet_explorer, mosaic, mozilla, opera} *Version*: installed browser's version *Internet connection type*: values = {dial-up, leased link, wireless} *Bandwidth*: Maximal bandwidth provided in KB/s. *Mean download rate*: in KB/s.

Fig. 2. Description of the components of the user's hardware profile.

point of view of its content we take elements of the LOM [12] standard which describe the format, language, semantic density, size, installation requirements, and previous knowledge required.

Formally a document will be described in the following way: Let Doc_{jx} ,be the j-th document of class x. $FE_0(Doc_{jx})$ represents the vector initial evaluation of the document made by an expert. This evaluation is done having "normal" user profile in mind, for which this learning object is aimed in the context of the learning community. The expert evaluates the contribution of the document to all topics of the class giving a value between 0 and 1, in which 0 denotes no contribution and 1 a high contribution. For each user k that has used and evaluated that document before, there is an evaluation vector $FE_k(Doc_{jx})$ similar to $FE_0(Doc_{jx})$ which contains the evaluation the learner has made about the learning contribution of the document for each topic of the class. For the Doc_{jx} document, there is also a vector describing it form the point of view of the characteristics of its content taken from the LOM standard denoted by $CC(Doc_{jx})$. Let $CP_i(Doc_{jx})$ be the estimated learning contribution of the document Doc_{jx} to the user i automatically generated by the system calculated by the following equation:

$$CP_i(Doc_{jx}) = \frac{1}{N} * \sum_k FE_k(Doc_{jx}) * g(UP_i, UP_k) + FE_0(Doc_{jx}) \qquad (1)$$

In this equation N is a normalization factor in order to have values between 0 and 1 for each component of $CP_i(Doc_{jx})$, so N is the number of vectors $FE_k(Doc_{jx})$ including $FE_0(Doc_{jx})$. The function g of the equation determines the degree of similitude between the profiles of the user i and user k. This function will give a value close to 1 if both profile tend to be similar and close to 0 if they tend to defer. As said, the vector $CP_i(Doc_{jx})$ is the estimation of the system about how would the user i evaluate the document. After exploring the document, the user may agree with this evaluation, thus declaring his final evaluation equal to the given by the system thus making $FE_i(Doc_{jx}) = CP_i(Doc_{jx})$ or providing a vector with fully new or partially modified values in order to be used by the system for calculating estimations for other members of the learning community.

3.3 Characteristics of the "g" Function

The g function described in the equation 1.0 should evaluate the degree of leverage an opinion a certain user has about a document (expressed by the vector $FE_k(Doc_{jx})$) to the user for which the vector $CP_i(Doc_{jx})$ is being calculated. We want, of course, that this function gives more importance to opinion expressed by people having similar interests, and backgrounds, which is reflected in the user's profile information stored in the UPi. In our context the system will define two user's profile as similar if the keywords contained and the values for those keywords in both vectors are within a certain defined "distance" or "threshold". In our system we use statistical correlation like those defined [14]. This metric incorporates not only the preferences of the users by also their background knowledge.

3.4 User Adaptive Filtering Techniques

For predicting the acceptance of a certain user will have of a certain document most of the existing recommender systems take in account only the user's requirements declared in the user's profile. Zhang [16] points out that this approach may be incomplete, because it does not take in account the contribution of the document to the knowledge of the user, that is, if the learner will learn something new with the learning object. In order to consider this aspect in our system we have to keep some information about what kind of material the user has already received. We do this by keeping for each user, a set of vectors TLC_{xi} which represents for a user i the total amount of information received for each topic of the class x from all documents she/he has already downloaded. For calculating the estimated increase to this value a certain document Doc_{jx} may cause by downloading and using it our system uses the vector TLC_{xi} and the estimated evaluation of the learning contribution of the document $CP_i(Doc_{jx})$. We can thus describe this estimated increment by the equation $EDTLC_{xi} = f(TLC_{xi}, CP_i(Doc_{jx}))$. The function f will follow the law of the decreasing returns, that is, the increment is smaller when TLC_{xi} is bigger. At the beginning, when the learner has downloaded only a few documents, the contribution of a new document might be higher than after the user has downloaded many documents. After the learner gives the final values for evaluating the learning contribution of a document $FE_i(Doc_{jx})$, the vector TLC_{xi} is updated by applying $TLC_{xi} = TLC_{xi} + f(TLC_{xi}, FE_i(Doc_{jx})$

For this calculations use some mechanisms borrowed from the field of Information Retrieval for modeling the user's behavior like vectorial spaces [14], diffuse logic based models and neuronal networks [13], and bayesian classifiers [2, 15]. In order to introduce a mechanism of automatic learning the model uses the algorithm of Bilmes [3].

4 How Does the System Works

When a new learning material is included in any of the distributed learning community servers, a tutor has classify it in a certain class x and generates the first evaluation

vector $FE_0(Doc_{jx})$ and the content description of it $CC(Doc_{jx})$. According to equation 1.0 at this point, for any user i the evaluation vector for this document will be $FE_0(Doc_{jx})$ since there is no other user who has used this document yet. When a new learner i joins the community, he/she has to define the preferences vector UPP_i. The values for all components of the vectors LC_{xi} for all x classes are set to 0. When the system evaluates a new document for the learner i it first searches for other learners who have an evaluation vector $FE_k(Doc_{jx})$ for that document. Whit this information, plus the initial vector $FE_0(Doc_{jx})$ and the learners preference profile UPP_i it calculates $CP_i(Doc_{jx})$. Then it calculates the increment for the learner's $LC_i(Doc_{jx})$ and compares the document's content characteristics $CC(Doc_{jx})$ with the learner's UHP_i and UPP_i (for determining, for example, if according to the document's size, the current downloading rate of the user and his/her maximum download time tolerance the learner will be willing to download it). At this point the system is able to tell the learner how interesting is the document, which is its learning contribution, and if there are some problems for downloading and/or using it in the hardware/software he/she has currently available. With this information, an HTML page with this information and a link to the respective learning material is generated and presented to the learner. The learner may decide to download the document right now, save this information for deciding later or discard this information. When the learner decides to download the material then the systems requires him/her to provide an evaluation vector $FE_i(Doc_{jx})$ or confirm these evaluation in the near future.

5 Conclusions and Further Work

The system we have presented propose to use the recommender systems approach to support the exchange of information about learning objects available inside a virtual learning community like COLDEX project, in order to help learners find suitable learning material for them. Most existing recommender systems rely upon the exchange of texts between writers and readers in an ongoing discursive activity. The model we have developed tries to be closer to the way human beings operate, including different kinds of recommendations. One of the inconveniences we try to solve in our model is to avoid the information overloading. As Furnas [9] mentions, huge amount of information will generate several interface problems. If we present so much information, it will be difficult for the user to focus on the essential aspects of their work. In order to avoid information overloading we need to show only the relevant required information. In that way, some authors have recommended the use of awareness filters [8] to present only the most relevant information in a similar way as proposed by our model. Considering that not all details are relevant and open to outsiders, an awareness mechanism should somehow filter the information [4]. The system is now in its implementation phase. As future work we hope to carry out several experiments in order to validate our model, comparing our results with other CF models identifying when our proposed schema is more suitable. This work will be ready by the time we might present this work in the workshop.

Acknowledgments

This work was supported by MECESUP (Chile) project No. UCH0109.

References

1. Aggarwal, C., Wolf, J., Wu, K., Yu, P.,: Horting Hatches an Egg: A New Graph-theoretic Approach to Collaborative Filtering. Proc. of the ACM KDD'99, San Diego, CA, (1999) 201-212
2. Ahmad M, Ahmad Wasfy,: Collecting User Patterns for Building User Profiles and Collaborative Filtering. IUI 99, Redondo Beach, CA, USA, (1999)
3. Bilmes, J.A., A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, International Computer Science Institute, Berkeley California (1998)
4. Borges, M., Pino, J., Valle, C.: Support for Decision Implementation and Follow-Up. Accepted for European Journal of Operations Research, (2003)
5. Breese, J., Heckerman, D., Carl, K.: Empirical analysis of predictive algorithms for collaborative filtering. Proc. of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence, (1998) 43-52
6. http://www.coldex.info.
7. Crozat, S., Hu, O., Trigano, P.: A Method for Evaluating Multimedia Learning Software. IEEE Conference on Multimedia Computing and Systems Volume 1, (1999)
8. David, J., Borges, M.: Selectivity of Awareness Components in Asynchronous CSCW Environments. CRIWG'01, Darmstadt, Germany, (2001)
9. Furnas, G., Bederson, B.: Space-Scale Diagrams: Understanding Multiscale Interfaces. Proc. Of Human Factors in Computing Systems, CHI'95, ACM Press, Denver, CO, (1995) 234-241
10. Goldberg, D., Nichols, D., Oki, B., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications. of the ACM, 35(12), (1992) 61-70
11. Kline, G., Reynolds, F., Woodrow, C., Ohto, H., Hielm, J., Butler, M., Tran, L.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation 15, (2004)
12. Computer Society/Learning Technology Standards Committee. IEEE Standard for Learning Object Metadata. Approved Publication of IEEE, (2002)
13. Mitaim, S., Kosko B.: Neural Fuzzy Agents that Learn a User's Preference Map. Proc. of the 4th International IEEE Forum on Research and Technology Advances in Digital Libraries, ADL (1997)
14. Savia, E., Kurki, T., Jokela, S.: Metadata Based Matching of Documents and User Profiles, in Human and Artificial Information Processing, Proc. of the 8th Finnish Artificial Intelligence Conference, August 1998, Jyväskylä, Finland, (1998) 61-70
15. Wong, S., Butz, C.: A Bayesian Approach to User Profiling in Information Retrieval. Technology Letters, 4(1), (2000) 50-56
16. Zhang Y., Xu W., Minka, T.: Novelty and redundancy detection in adaptive filtering. Proceedings of the 25th ACM SIGIR Conference, (2002)

An Integrated Approach for Analysing and Assessing the Performance of Virtual Learning Groups

Thanasis Daradoumis[1], Alejandra Martínez-Monés[2], and Fatos Xhafa[1]

[1] Open University of Catalonia, Department of Information Sciences
Av. Tibidabo, 39-43, 08035 Barcelona, Spain
{adaradoumis,fxhafa}@uoc.edu
[2] University of Valladolid, Department of Computer Science
ETIT, Campus Miguel Delibes, 47011 Valladolid, Spain
amartine@infor.uva.es

Abstract. Collaborative distance learning involves a variety of elements and factors that have to be considered and measured in order to analyse and assess group and individual performance more effectively and objectively. This paper presents an approach that integrates qualitative, social network analysis (SNA) and quantitative techniques for evaluating online collaborative learning interactions. Integration of various different data sources, tools and techniques provides a more complete and robust framework for group modelling and guarantees a more efficient evaluation of group effectiveness and individual competence. Our research relies on the analysis of a real, long-term, complex collaborative experience, which is initially evaluated in terms of principled criteria and a basic qualitative process. At the end of the experience, the coded student interactions are further analysed through the SNA technique to assess participatory aspects, identify the most effective groups and the most prominent actors. Finally, the approach is contrasted and completed through a statistical technique which sheds more light on the results obtained that far. The proposal draws a well-founded line toward the development of a principled framework for the monitoring and analysis of group interaction and group scaffolding which can be considered a major issue towards the actual application of the CSCL proposals to real classrooms.

1 Introduction

The application of innovative educational experiences proposed by research fields such as CSCL is hindered by the obstacles that teachers find in applying them in real classrooms. For example, it is normal that students find difficulties while fulfilling the collaborative assignments and it is nearly impossible for a single teacher to account for all the interaction breakdowns that may happen within a medium-sized classroom. Moreover, if the students' collaborative process is to be taken into account for their assessment, teachers need efficient and effective tools that help them to perform this task. Therefore, there is a need of new approaches and tools that assist teachers in providing formative evaluation and assessment on collaboration in order to be able to apply these techniques to real classrooms.

These questions are closely related to the extensive research on interaction analysis in the CSCW and CSCL fields that has been oriented to identifying and exploring the

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 289–304, 2004.

factors that affect the effectiveness and success of online group work and learning [8]. However, this line of research has proposed rather limited approaches, focusing on a single collaboration channel, such as dialogue [1] or action [13]. Some researchers have recently proposed an integration of different sources of data in the analysis [4, 15]. In spite of this fact, existing approaches have not yet managed to meet these needs satisfactorily, since most of them focus on experimental situations, which do not exactly reflect the issues and problems of a real situation.

Teachers need principled frameworks that help them to carry out all these issues. The evaluation of collaborative learning has to be performed at least at two levels, separating the process (or group functioning) from the product (or task performance) of collaboration [6, 10]. Evaluation of task performance considers the task skills and knowledge acquired by each member as well as the quality of final product of group work. Evaluation of group functioning refers to the analysis and assessment both of the interaction behaviour of group members and the social aspects of group work. According to [14], participation is a further important aspect since, together with acquisition, constitutes one of the two main metaphors of learning.

The complexity of the interaction processes and the need of considering different perspectives claim for *mixed evaluation methods* [9] that integrate different sources of data (i.e. types of interaction, products, student's opinions, etc.) and different analysis techniques in order to tackle with all the points of view that must be considered. To our knowledge, there are still very few mixed evaluation approaches that combine different data sources and methods to provide a more in-depth analysis of collaborative learning interactions.

We have been working in defining a principled, effective and holistic framework for analysing the interaction and assessing the performance of virtual learning groups [7]. This framework can be considered as a first step towards the definition of an evaluation approach that considers both the process and the product of collaboration. However, it still lacked some aspects, mainly related to the participatory metaphor, such as the general structure of the collaborative relationships and the roles that the students play with respect to these structures.

Recently, [12] has proposed a *mixed method* for the formative evaluation of social aspects of CSCL experiences that integrates different data sources and methods of analysis, including SNA [16], in order to evaluate social aspects of group work. SNA seeks to describe patterns of relationships among actors, to analyse the structure of these patterns and discover their effects on people and organizations. Social networks can be visualized as graphs called *sociograms,* which represent the actors as nodes of the graphs and the links among them as lines in the graph. Several studies have demonstrated its value within the CSCL field for the study of structural properties of individuals learning in groups [3, 18]. However, SNA by itself is not enough for achieving a full understanding of the collaborative processes. It needs to be complemented with other methods and perspectives.

The work we present in this paper can be considered as a new step towards the fulfilment of the demands of teachers for the formative evaluation in CSCL real settings, by integrating these proposals in a common evaluation experience. It shows how to use qualitative, quantitative and SNA techniques to analyse and assess a real case study.

The rest of the paper is structured as follows. First, we present the case study on which the analysis was carried out. On the one hand, the proposed approach is justi-

fied, the evaluation criteria (indicators) are set and a basic qualitative evaluation process is defined together with the first level of the mixed evaluation scheme we are proposing. On the other hand, in order to accomplish all evaluation criteria, two further levels of analysis are defined, including a quantitative and SNA evaluation technique (as well as the associated data sources and supporting tools). Section 3 describes the SNA and quantitative techniques in more detail and draws the final conclusions that result from the combination of all the three evaluation techniques. Finally, we present the future lines of our research that aim at building a principled framework for group modelling and scaffolding.

2 Case Study Description and Evaluation Method Proposed

The context of our research is defined by setting virtual learning groups to work on a real, long-term, complex, collaborative problem-solving situation that forms part of a distance learning undergraduate course. As such, it is important to present first a sufficient description of the workings of the case study that was used for the purpose of our analysis. Then we proceed to justify our approach and set the qualitative basis of the methodology proposed to tackle with the complex task of analysing and assessing group interaction successfully. Finally, the rest of the data sources, evaluation techniques and supporting tools that compose our approach are explained.

2.1 Case Study Description

To perform this study we were based on a real collaborative learning experience that was carried out in the scope of a virtual (distance) learning undergraduate interdisciplinary course, called "Application of Information Systems to Business". The experience run over a period of 14 weeks and involved 2 tutors and 122 students distributed into 21 virtual groups of 5 to 6 members. Students had to collaborate and work out a case study that simulated a real project in a business or organisation.

Virtual groups were formed and consolidated during the first 10 days of the course by the students themselves, following a well-structured and guided virtual process, called *group formation process,* based on the work of [5]. The case resolution consists of a set of target goals (phases) that are realised collaboratively (except the first one which aims at studying and understanding the case presented).

The whole project was carried out mostly asynchronously; synchronous interaction occurred in few specific cases of decision-making. All asynchronous collaborative interactions took place on the Basic Support for Cooperative Work (BSCW) system, a groupware tool that enables asynchronous and synchronous collaboration over the web [2]. BSCW offers shared workspaces that groups can use to store, manage, jointly edit and share documents, realise threaded discussions, etc.

To structure the whole collaborative learning process, we set two particularised shared workspaces in the BSCW system. The first one is a general workspace, which can be accessed by all students of the virtual class. The main purpose of this workspace is to let the students interact with each other in order to form the virtual learning groups. In addition, it is used to effectuate specific debates, which form part of the project requirements and involve all students, as well as to share important informa-

tion about the project among tutors and students. The other workspace type is a private space designated to house each virtual group, that is to record and structure the interaction of its members that aims to achieve the project target goals through the resolution of the specific tasks and problems the project consists of.

Table 1. Description of the evaluation criteria defined for each one of the aspects of the collaborative learning analysis. For each aspect we give the corresponding weight and the weights of its criteria.

Evaluation Criteria		Weight
Task performance		**50%**
TP1	The students' individual and group problem-solving capabilities and learning outcomes (*acquisition* metaphor)	40%
TP2	The students' contributing behaviour during task realisation (production function and use of active learning skills)	40%
TP3	The students' individual and group ongoing (and final) performance in terms of self-evaluation	20%
Group functioning		**20%**
GF1	Active participation behaviour	30%.
GF2	Social grounding (well-balanced contributions and role playing)	20%
GF3	Active interaction or processing skills that monitor and facilitate the group's well-being function	30%
GF4	Group processing (examine whether each member learnt how to interact and collaborate more effectively with his/her teammates)	20%
Social support		**15%**
SS1	Members' commitment toward collaboration, joint learning and accomplishment of the common group goal	30%
SS2	Level of peer involvement and their influential contribution to the involvement of the others	30%
SS3	Members' contribution to the achievement of mutual trust	10%
SS4	Members' motivational and emotional support to their peers	20%
SS5	Participation and contribution to conflict resolution	10%
Help services		**15%**
HS1	Help is timely	25%
HS2	Help is relevant to the student's needs	10%
HS3	Help is qualitative	30%
HS4	Help is understood by the student	25%
HS5	Help can be readily applied by the student	10%

Our analysis was carried out at both the general and the private group spaces, using specific evaluation criteria as parameters to measure the groups' real effectiveness regarding learning and collaborative skills, as explained in the following sections.

2.2 Defining Principled Evaluation Criteria and a Qualitative Evaluation Process

The need for a mixed evaluation scheme comes forth from identifying the most important indicators related to group activity which, due to its variety, can not be covered and satisfied by a unique analysis method. The first step toward establishing a

well suited and effective integrated evaluation approach is identifying principled evaluation criteria or group activity indicators (and their estimated weights) that capture and describe group interaction and performance sufficiently in the context where group work and learning is situated. The second step consists in presenting different analysis techniques to meet the evaluation criteria proposed and testing that neither technique alone nor a combination of any subset of them can account for all the indicators, thus prompting the necessity of an integrated approach. Consequently, the weight (and importance) of each technique within the integrated approach is estimated in terms of the indicators that accomplishes. Next we proceed to define the criteria and the first level of our approach, a basic qualitative evaluation process.

Based on the theoretical principles and indicators of effective collaboration of [10, 11, 14,15,17] we specify four important levels or aspects of collaborative learning analysis: *task performance* (or learning outcome), *group functioning* (or participation/ interaction behaviour), *social support,* and *help supply* (or task/process scaffolding); for more details on these levels see [6].

To measure and evaluate each level, we need to define generic evaluation criteria that describe and capture its important features as fully as possible. Each criterion is also assigned a specific weight. This is an important feature of our approach since it determines not only the importance of each evaluation means but also the way these means can be combined to carry out the analysis and evaluation process. The criteria identification is based on the above theoretical principles and our lengthy experience with online collaborative learning teams; the latter is also an influencing factor for specifying the weights of each criterion. Moreover, particularisation of the evaluation criteria and their associated weights depends on premises, such as the evaluation goals, the context or situation surrounding the collaborative learning experience and its specific tasks, as well as the available evaluation techniques and data sources.

In general, what we describe below shows a way to set weights for the analysis and assessment of our particular case study. We are currently exploring a more principled mechanism (such as a regression statistic model) to derive relative weights for each indicator. Table 1 presents both the evaluation criteria and the assigned weights.

In this experience we consider task performance an important factor of the student evaluation; in addition, we can analyse it through a variety of data sources and methods, thus we assign it a 50% weight. Among the other three factors, we consider that, at the time being, group functioning can be measured more easily and effectively than social support and help supply. Taking this condition into account, we assign it a 20% weight, whereas the other two factors are assigned an equal 15% weight each.

Our case study offers the tutor the context to perform a continuous qualitative evaluation of the students' work and collaborative activity. Thus, a formative qualitative evaluation takes up an important value and constitutes the basis of the evaluation method. In fact, all the four analysis aspects are measured and assessed qualitatively by the tutor at the end of each project phase.

In particular, as concerns task performance analysis, each group delivers the tutor its learning outcomes (the solution of the current sub-problem). The tutor corrects and assesses it thoroughly, assigns it a mark and sends his/her feedback to the group (criterion TP1). In addition, during task realisation the tutor performs a selective qualitative examination of the students' most significant contributions to the task, with the aim to reason out both the specific production function and the active learning skills exhibited by each group member (criterion TP2). Finally, the tutor carries out a quali-

tative analysis and assessment of two self-evaluation report types delivered by the students. The first one is elaborated by each group at the end of each problem-solving phase, whereas the latter is addressed to each student individually at the end of the project. Both reports are guided by specific questions that aim at knowing the students' personal opinion, perception and impression about individual contribution and overall group performance regarding the task (criterion TP3).

Such a qualitative evaluation is not only important to assess task performance issues but also to explore and qualify aspects related to the other three analysis levels (group functioning, social support and help supply). Individual and group self-evaluation reports proved to be a valuable information source for the tutor to delineate and get a basic comprehension of the groups' internal workings such as to explore: the members' participation trends and interaction or processing skills (criteria GF1, GF3), the achievement of a well-balanced group and adequate role playing (criterion GF2), the members' ability to reflect upon the way they learn to collaborate with each other (criterion GF4), the level of group cohesion accomplished (criteria SS1 to SS5), and the degree and suitability of the help provided by each member (criteria HS1 to HS5).

As a result, at this stage of our work, given that the qualitative evaluation is spread along all analysis levels, it is considered as a basic initial layer upon which further evaluation techniques can be incorporated and applied so that to fill the gaps left and evaluate all those issues that could not be covered sufficiently by the qualitative evaluation method alone, thus giving rise to a mixed evaluation approach.

2.3 Further Evaluation Techniques, Data Sources and Supporting Tools

Evidently the above qualitative evaluation method alone does not suffice to provide a complete analysis and assessment of all the indicators we identified at the four analysis levels. Indeed, self-evaluation reports (as well as further possible interviews or questionnaires) can give important information to the tutor, especially as regards issues related to social support and help provided by group members, which otherwise is difficult to extract from other sources or methods. However, there are still some important indicators, which need a further in-depth analysis in order to be evaluated completely. Such indicators are: TP2, GF1, GF2 and GF3.

The nature and objective of these four indicators suggests us to group them into two categories. The first one comprises criteria TP2 and GF3 that intend to measure specific students' active skills employed either to support task realisation and learning or to facilitate better interaction respectively. The second category involves criteria GF1 and GF2 that study participatory issues (like active peer involvement) and social grounding aspects of learning (like comparison of group activity, well-balanced interaction and role playing) respectively.

On the one hand, SNA has proved to be an adequate and sufficient technique to analyse the structure of the social interactions that take place in the virtual workspaces (criteria GF1 and GF2). This structure allows for the study of individual properties (prominence of the actors), small groups and the whole network.

On the other hand, to account for and evaluate the many different types of students' active learning or interaction/processing skills (criteria TP2 and GF3), we proceed to measure (quantify) the types of student contributions that imply a particular

skill; to that end, due to the big amount of interaction data produced, a quantitative (statistical) technique can provide better and more reliable results.

The basic source that provides data for both types of analysis is the BSCW *daily log files.* Every log file records all the interaction data (events) occurred in all active BSCW workspaces. BSCW distinguishes and generates four generic types of events (or actions) related to an object: *Create, Change, Read* and *Move* events.

In this study we take three major objects into account: *folder* (related to task and knowledge management), *document* (related to task realisation and learning), and *note(s)* (related to communication processing). The rest, which are considered minor objects, are grouped into a class called *others* and include, among others, concepts such as URL, Appointment, User, or Group Agenda.

On the one hand, to facilitate the quantitative analysis of the many different action types a student can perform, we initially use a specific software tool that extracts and filters the data contained in the event logs according to desired parameters defined by our analysis needs (for instance, events can be classified by user and action type, or can be distributed in specific periods of time).

On the other hand, to support SNA automatic processing, we use a tool called SAMSA (System for Adjacency Matrix and Sociogram-based Analysis). This tool contains several input modules, one of which takes data from BSCW event logs and transforms them into an XML file representing the interactions. Then, SAMSA allows us to select and configure the network we want to study (selecting dates, actors, and relationship type). The tool builds the matrix that represents the network, known as *sociomatrix,* and computes the indexes chosen for the analysis. It also shows the sociogram representing the network, and allows for visualising the actors' attributes.

Both analysis techniques are discussed in the next section and confirm that if the qualitative method were used alone, the students' evaluation would be insufficient and subjective. Instead, the combination of all three techniques together provides a more effective, complete and correct evaluation. This is particularly shown by the results obtained from the analysis of a specific effective group.

3 Analysing Group Interactions and Social Aspects of Group Work

This section describes the analysis results obtained by the SNA and descriptive statistical quantitative techniques. Each technique aims at achieving specific evaluation criteria set in the previous section and thus is complementary to the other. The obtained results are finally compared to the qualitative evaluation results and definite conclusions are drawn with respect to the real effectiveness of the learning group that was analysed and evaluated as an example. First the SNA technique is described, showing how SNA can be used in a top-down analysis approach so that the evaluator can start from a very general perspective of the classroom interactions and detect which are the groups or actors that need further analysis.

3.1 Analysis of the Participatory Aspects of the Collaborative Learning Processes by Social Network Analysis

In order to perform Social Network Analysis, it is necessary to define the networks and the set of indexes that will be used for the study.

Networks are *relationships* established among a set of *actors.* In this study we considered the relatioships composed by the *indirect links* between an actor that creates an object in the BSCW workspace and those that access this object in order to read it. This is by far the most frequent type of interaction in the context of the use of BSCW as shown by the daily report and the log files maintained by the system. The set of actors included both the students and the teacher or teachers.

We have identified a set of SNA indicators for the study of participatory aspects of learning, which were the ones used in this study, namely: *Network density* (D), *actor's degree centrality* ($C_D(n_i)$), and *network degree centralization* (C_D) [16]. D measures how knitted a network is, with values ranging from 0 (most sparse) to 1 (most dense). *Degree centrality* is an index of the actor's prestige. Given an actor n_i, $C_D(n_i)$ is the proportion of actors that are adjacent to n_i. It reflects the activity of the actors. In the case of directed relationships that consider the direction of the link, two degree indexes are defined: *indegree,* or the number of links terminating at the node; and *outdegree,* or the number of links originating at the node. Finally, *network degree centralization* (C_D), is a group-level measure based on actor's degree centrality. It gives an idea about the dependency of the network on the activity of a small group of actors. Its values range from 0 (even distribution of activity) to 1 (most centralized network). Directed networks define the corresponding indexes of *indegree centralization* (C_{ID}) and *outdegree centralization* (C_{OD}). All of these indexes and ranges apply to dichotomous relationships that can have only one out of two possible values: 0 when there is no link and 1 when there is a link between two actors. It is also possible to consider valued relationships that include a number showing their strength. The indexes computed on these relationships are more difficult to generalize than those computed from the dichotomous relationships, but sometimes are important to provide additional information. All of these indexes provide basic information about the activity of the actors in the network and about the global structure of the network according to different relationships. Moreover, they are simple to understand and to interpret, which are important features for facilitating their use by evaluators, who are not expected to be experts in SNA methods.

These indexes were applied in order to study and compare interactions at the general and at the private workspaces, as well as to identify who were the more and the less active students at both levels. The following two subsections develop these issues.

3.1.1 Student Participation in the General Workspace

The first aspect we wanted to analyse was the general structure of the relationships in the classrooms, which was studied by the indirect relationships network at the general workspace of the virtual classroom.

A first analysis can be performed by studying the network of the aggregated relationships during the last four phases of the course, once the groups had been created

and the students were focused on their project-oriented tasks. The indexes of this network (Δ=14,24%, C_{ID}=42,22%, C_{OD}=63,33%) show that the indirect links considered in this network were quite frequent (if we take into account the size of the network) but too much centralised as regards both reading (C_{ID}) and specially writing (C_{OD}), which means that the activity was concentrated on a very reduced set of actors.

The examination of the sociogram of this network (Fig. 1) allows us to go deeper in this issue in a very intuitive manner that can be easily built by the teacher or evaluator by means of SAMSA. The actors are represented by different shapes according to the group they belong to, and the links between them as directed lines that go from the actor that creates a document to the one(s) that read it. While the high concentration of the lines and arrows makes difficult a detailed analysis of the specific links, it is still possible to draw some initial conclussions that complement the values mentioned above. Firstly, it can be observed that some actors appear as isolated nodes at the left, which means that they did not intervene in the shared workspace at all. It is also possible to see that the teacher (*ifa*) has a central position in the network, shared with an important number of students. It is possible to identify at a glance who were the most active students (at the centre) and the less active ones (at the periphery). For example, actors like *xva, cao, dac,* and *fca* played a prominent role, while others, like *jga, Ton, ngu, etc.* played a peripheral role, as their only connection to it consists of a single link to another actor.

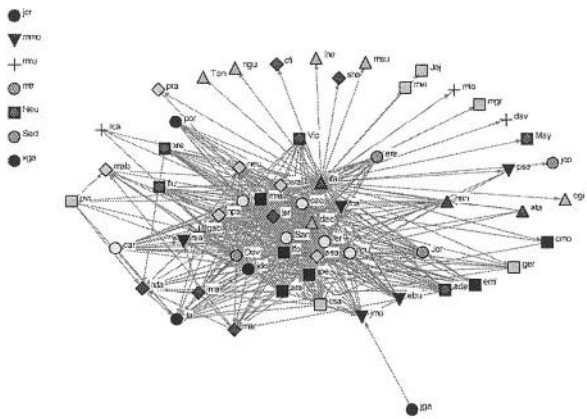

Fig. 1. Indirect relationships network at the general workspace during the course.

The exploration of the actors' centrality values, which is also calculated by SAMSA, complements the analysis of the sociogram, as they allow identifying the most active students, with the added value that these indexes inform also about the reading and writing activities. It is especially interesting to detect who were the students with a higher value of out-degree centralization, which means that they wrote documents that were read by more students. According to [3], this index can be considered as a measure of actors' prestige, and can help a teacher to detect who are the students which act as leaders in the sense measured by the network. In our study the students with a higher out-degree centrality value were *san, fer, jur, fgu,* and *car.*

The analyisis performed so far shows a static view of the aggregated relationships during the course. This view was complemented with an analysis of the evolution of the networks, which is performed by a similar study of each the phases in which the course was divided. It is not possible to show the details of this evolution for lack of space, but we comment here the main results: density remained stable throught the course (with values around 5-6% in each one of the phases), with a slight drop at the last phase; and the prominent actors we have identified above showed a quite regular participation throughout the course. Interestingly the teachers, in spite of their high level of activity in the general workspace, are never on the top position, which is considered a positive indicator in the sense that the students (or at least some of them) indeed got involved in the classroom activities.

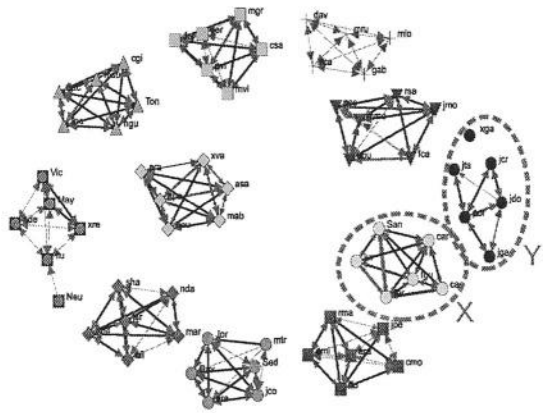

Fig. 2. Sociogram of indirect relationship network at the group workspace.

3.1.2 Study of the Activity at the Private Workspaces

As it has been explained above, the activity at the *private workspaces* has a different nature that the one performed at the general workspace. While the analysis of the general workspace shows the structure that yields from the interactions that take place in the context of discussions that are indirectly related to the fulfillment of the assignments, collaboration at the private group level is mainly focused on the development of a common product (or a set of common products), namely: the set of assignments required by the teacher for the fulfillment of the project tasks.

In order to analyse this workspace, we performed a similar study to the one previously presented. Fig. 2 shows the indirect relationships network at the group private workspaces during the same period of time as the one previously considered. As expected, it displays a rather different structure, due to the restricted access to each group workspace, which results in a set of independent sub-networks that represent the relationships in each group. By simple observation of the sociogram it is easy to detect which were the most and the least active and homogeneous (balanced) groups. These groups are respectively labelled *X* and *Y*. The evolution of these groups is shown by the indexes in Table 2. We can observe that, while group X has a 100% of density and a 0% of centralization during all the phases of the course, group Y never achieves these desired values, with the lowest value at the last period of the course.

This difference is even more outstanding if we compare the valued densities obtained by both groups thoughout the course. According to [18], group X shows an ideal pattern of interaction, with all the members interacting with each other, and none of them taking a too central position. Moreover, if we go back now to the indirect relationships at the general workspace network analysed in the previous subsection, we can observe that most of the members of group X are at the center of the network and belong to the list of prominent actors (actually, only *car* does not occupy a central position at the sociogram of Fig. 1). This can mean that the members of group X obtained a good result regarding criteria GF1 and GF2.

Table 2. Evolution of the values of density and centralisation for groups X and Y.

	Group X				Group Y			
Phase	Δ	Δ_v	C_{iD}	C_{oD}	Δ	Δ_v	C_{iD}	C_{oD}
Ph2	100 %	2833 %	0 %	0 %	63,33 %	186,67 %	25 %	25 %
Ph3	100 %	1493 %	0 %	0 %	46,67 %	76,67 %	50 %	50 %
Ph4	100 %	1696 %	0 %	0 %	6,67 %	13,33 %	50 %	20 %
Ph5	100 %	980 %	0 %	0 %	23,33 %	43,33 %	55 %	25 %

However, the fact that a group showed to be effective at this level does not imply that its members show the same effectiveness regarding the acquisition and use of active learning and interaction or processing skills (criteria TP1 and GF3). A tutor or evaluator certainly needs to make use of this important information in order to evaluate each active group member more objectively. To that end, a quantitative (statistical) evaluation technique offers the means to decode the students' active behaviour both at task performance and group functioning level, exploring whether their active participative behaviour at the general and group workspaces also indicates an active and equiponderant contribution to the group's production and well-being functions. Among the groups we analysed statistically, we next chose to discuss the results obtained by the analysis of the interaction behaviour of the members of group X, which presented a particular interest to our research and shows the necessity of applying different evaluation techniques into an integrated approach.

3.2 A Quantitative Statistical Analysis of Task Performance and Group Functioning of an Effective Group

The quantitative analysis performed here is a simple descriptive statistical analysis that aims to provide a complementary and more refined analysis of the contributing and interaction behaviour of the members of an effective group so that to explore and understand the real performance and achievement of each member. This analysis proves to be a necessary aid to the SNA carried out before, especially for identifying particular attitudes of group members not been able to be tracked through SNA.

To illustrate this, we chose to analyse the real effectiveness of group X as concerns criteria TP2 and GF3 (which were partially measured by the qualitative method). On the one hand, the qualitative method assessed this group with an A mark[1] at all evaluation levels; on the other hand, SNA showed it to be one of the most active and

[1] We use a 5-point scale mark from A (excellent) to D (fail).

well-balanced groups of the experience whereas its members were also considered as prominent actors (to a greater or lesser degree) in the general classroom workspaces.

The quantitative analysis of criteria TP2 and GF3 depends on the information provided by the data sources available. In our case, BSCW data logs account for various action types performed on several object types.

Consequently, a sufficient measurement of criterion TP2 involves the following generic actions: Create and Change (also called *active learning actions*) as well as the Read action. In particular:

- Create: Document, Note (implies knowledge or information generation that contributes to task performance and group product)
- Change: Document (implies active participation to further elaboration, refinement or revision of existing knowledge)
- Read: Document, Note.

To measure criterion GF3 we can use the following actions which when performed on particular objects enable us to draw conclusions about specific active interaction or processing skills exhibited by each member, such as:

- Task processing skills: can be measured by the actions *createfolder,* that indicates task (and indirectly knowledge) management, and *create others* that involves task plannig and preparation of virtual meetings or group agenda).
- Workspace processing skills: can be estimated by the action *move "all" objects* that describes the workspace organisation and maintenance.
- Communication processing skills: can be quantified by the action *change Note/Folder/Other* that adds appropriate meta-information to an object to facilitate and promote better understanding of the objects the members share in the group workspace and thus to achieve more effective interaction.

Table 3 presents a detailed description of the four generic actions and the percentage of objects each member contributed with respect to the other members. For instance, the member *cao* contributed to the creation of the 6.5% of all the documents produced in the group.

The analysis of the data provided in Table 3 allows us to draw a number of conclusions regarding the achievement of criteria TP2 and GF3 by group X.

As for criterion TP2, we observe that there are important differences regarding the contributing behaviour of some members during task realisation; only Read seems to be the most balanced action for all members. In particular, a broad use of active learning skills is shown by member *fer* in generating knowledge (document creation), and by member *fgu* and *fer* as well in generating information (note creation) and document modification (change document). In other words, *fer* and *fgu* show a distinguishing productivity in comparison to their peers as task achievement concerns.

Among the other members, *jur* and *san* have a rather homogeneous, middle-rate contributing activity to the task regarding all actions involved in TP2. In contrast, the use of active learning skills of *cao* and *car* is shown to be the lowest in the group, except *car's* contribution to knowledge generation (had the second highest rate, 15.45%). To interpret *car's* particular attitude to task realisation, we consulted the qualitative evaluation of the individual and group reports, which explained that *car's* focused attention to document creation was primarily due to his particular interest to

specific topics of the case study. In contrast, *cao's* contributing behaviour to knowledge generation deviates significantly from the one of his peers; he also gives the lowest rate at the rest of TP2 actions.

Table 3. Percentage of objects created, read, modified, or moved by each member of group X.

ACTIONS	MEMBERS / OBJECTS	Cao	car	fer	fgu	Jur	san
Create	Document	6.5	15.45	38.21	14.63	13.82	11.38
	Note/s	10.05	10.32	20.63	21.69	18.78	18.51
	Folder	17.65	5.88	52.94	5.88	5.88	11.76
	Others	40	26.67	33.33	0	0	0
Read	Document	12.36	16.30	20.90	18.16	15.54	16.74
	Note/s	14.07	14.07	18.98	17.49	17.76	17.61
Change	Document	6.42	8.02	25.67	28.89	13.90	17.11
	Note/s	0	0	20.83	37.50	33.33	8.33
	Folder	0	0	66.67	0	0	33.33
	Others	0	0	100	0	0	0
Move	'all objects'	10.28	4.25	61.54	5.50	7.90	10.50

To measure the achievement of criterion GF3, looking at the use of task processing skills (create folder/others) we observe the distinguished contribution of member *fer* as well as the supporting attitude of the members *cao* and *car,* especially to task planning (Create Others). As for workspace processing skills (*move* action), we notice that member *fer* is the one who cared more for organising and maintaining the group's workspace; a more refined analysis (not shown for lack of space), showed that other members gave only some (very specific) support, like *san* in document organisation, *jur* in message organisation and *cao* in group meetings and agenda organisation. Studying the use of communication processing skills (Change Note/Folder/Other), we again observe the notable contribution of member *fer,* the specific support of other members, like *fgu* and *jur,* to aspects like "communication improvement" (Change note) and member *san* to group workspaces description (Change Folder), and the null contribution of members *cao* and *car.*

As a result of this analysis, group X does not meet criterion GF3 satisfactorily, since its members did not contribute equally to task, workspace and communication processing; indeed, just one member (*fer*) carried the greatest part of the responsibility burden for this particular collaboration feature; the others showed a rather irregular and uneven contribution, supporting only specific aspects as discussed above.

3.3 Discussion: Reflections on the Analyses Performed

Our study showed that evaluation of a real collaborative learning situation is a very complex task, since one has to consider a variety of aspects and thus to integrate several analysis techniques, data and tools into a mixed evaluation method. To that end, after identifying a set of potential indicators of group performance, we first went through a formative qualitative evaluation of all of them. Four of them clearly showed to need a further in-depth analysis to be evaluated completely. The use of two further

different techniques, which proved to be complementary, was guided by the indicators that each technique best accomplished. Classifying each indicator into a specific category (aspect) of the collaborative learning process and assigning it a relative weight dictate the way each technique is used and influences the evaluation process and how it is positioned and related to the others. Exemplifying the evaluation on a specific group showed that the application of different techniques is essential to unfold the group's internal workings and achieve a more objective interpretation of each member attitude and competence.

While the qualitative evaluation and SNA techniques indicated group X as an effective group (excellent group learning outcome, active, well-balanced, with an ideal pattern of interaction and group processing as well as a sufficient social support and help supply behaviour), a further quantitative analysis showed that two specific criteria (TP2 and GF3) were not completely achieved by all members. There were some members who exercised low active learning or interaction/processing skills, like member *cao* who got a low score in specific competences, like knowledge generation, knowledge modification and communication processing, and member *car* whose performance was insufficient regarding knowledge modification and communication processing. In fact, a further refined qualitative study of the individual and group reports confirmed this particular situation.

Indeed, based on an integrated analysis of the interaction of an effective group, we could see that some of its members, who proved to be prominent and influencing at the general workspaces, continued to act at the same level in the group workspace, while others did not act as well at some aspects. In particular, we could deduce that member *fer* kept on distinguishing as a prominent actor both at the general and the group workspaces, yet at all aspects of collaboration. Instead, the activity of members *cao* and *car* at the group space was eclipsed by some other prominent actor (*e.g., fer*). In other words students who played a prominent role at the general workspaces, when put together to collaborate in the same group, managed to develop an excellent group product, however, they did not achieve an ideal synergy at all aspects.

Consequently, it is important for an evaluation method to enable the evaluator to distinguish such particular cases of insufficient contributing or interaction behaviour and allow him/her to infer correct conclusions for the performance and competence of each group member. Doing so, he/she will be able to intervene adequately to monitor and provide the most proper support and guidance when and to whom is needed.

4 Conclusions and Future Work

In this work we have presented an integrated approach to be used by tutors and evaluators of group interaction in order to monitor and assess the performance of virtual learning groups effectively, especially in the case of real, complex and long-term collaborative learning experiences. We demonstrated that particular evaluation techniques tackle specific aspects or indicators of the evaluation process, so an adequate combination of them is needed to cover, complement and verify the gaps or flaws in the analysis results obtained by one technique or another. The conceptualisation and implementation of an integrated approach aims at providing the evaluator flexible options and well-adapted means for building a principled framework that achieves a more complete and effective evaluation of group interaction. We are con-

scious that in order to accomplish an in-depth and objective analysis and evaluation of the entire collaborative learning practice, we need to provide the evaluators efficient, automatic and effective tools to carry out this task within the proposed framework. Given that the theoretical principles and conceptual bases are set and successfully tested by our integrated approach, our objective is now turned to the development of automated tools that will make the tutor's evaluation task much easier and successful, especially as the qualitative and quantitative analysis concerns. As for the former, a couple of tools are currently designed and implemented for the automated processing and analysis of questionnaires and reports related to self-evaluation, group processing as well as to all criteria associated with social support and help supply. As for the latter, the descriptive statistical analysis performed is currently insufficient, since it does not allow us to establish hypothesis that relate the interactions with the individual or group learning outcome, so our aim is to provide a more comprehensive multivariate statistical analysis and build a tool that automates the whole process as much as possible. These tools are to be integrated with SAMSA and with the CSCL tools that provide the interaction data, which makes the definition of clear interfaces between them an interesting line of future research.

Once these new tools and further data sources are available, it would be the right moment to test the framework in a real situation, as the experience is taking place. This can provide the tutor and the group themselves the means that will help them identify the *weaknesses* and the exhibited in a collaborative learning team. As a consequence, this will assist the tutor to understand, monitor, support and assess group interaction and performance in a more efficient way.

Acknowledgments

This work has been partially supported by Spanish MCYT project TIC2002-04258-C03-03.

References

1. Barros, M. & Verdejo, M.: Analysing student interaction processes in order to improve collaboration. The DEGREE approach. Int. J. of Art. Int. in Education. 11 (2000) 221-241
2. Bentley, R. Appelt, W. Busbach. U. Hinrichs, E. Kerr, D. Sikkel, S. Trevor, J. & Woetzel, G.: Basic Support for Cooperative Work on the World Wide Web. Int. J. of Human-Computer Studies 46(6) (1997) 827-846
3. Cho, H. Stefanone, M. & Gay, G.: Social information sharing in a CSCL community. In: G. Stahl (ed.) Computer Support for Collaborative Learning: Foundations for a CSCL community. Erlbaum, NJ (2002) 43-50
4. Collazos, C. Guerrero, L. Pino, J. & Ochoa, S.: Evaluating Collaborative Learning Processes. In: J. M. Haake & J.A. Pino (eds.) Proc. of the 8th Int. Workshop on Groupware (CRIWG 2003). Springer, Berlin (2002) 203-221
5. Daradoumis, T. Guitert, M. Giménez, F. Marquès, J.M. & Lloret, T. : Supporting the Composition of Effective Virtual Groups for Collaborative Learning. In Proc. of the Int. Conf. on Computers in Education (ICCE'02). IEEE Comp. Soc. Press (2002) 332-336
6. Daradoumis T. Xhafa, F. & Marquès J.M.: Evaluating Collaborative Learning Practices in a Virtual Groupware Environment. In: Proc. of the Int. Conf. on Computers and Advanced Technology in Education (CATE 2003), ACTA Press (2003) 438-443

7. Daradoumis T. Xhafa, F. & Marquès J.M.: Exploring Interaction Behaviour and Performance of Online Collaborative Learning Teams. In: Proc. of the 9th Int. Workshop on Groupware (CRIWG 2003), Springer, Berlin (2003) 203-221
8. Dillenbourg, P. (ed.): Collaborative Learning. Cognitive and Computational Approaches. Elsevier Science Ltd. (1999) 1-19
9. Frechtling, J. & Sharp, L. (eds.): User-Friendly Handbook for Mixed Evaluations. Directorate for Education and Human Resources Division of Research, Evaluation and Communication, NSF (1997)
10. MacDonald, J.: Assessing online collaborative learning: process and product. In Int. J. of Computers & Education, 40 (2003) 377-391.
11. McGrath, J.E.: Time, Interaction and Performance (TIP). A Theory of Groups. Small Group Research, 22, (1991) 147-174
12. Martínez, A. Dimitriadis, Y. Rubia, B. Gómez, E. & de la Fuente, P.: Combining qualitative and social network analysis for the study of social aspects of collaborative learning, Computers and Education, 41(4) (2003) 353-368
13. Mühlenbrock, M.: Action-based collaboration analysis for group learning. IOS Press, Amsterdam (2001)
14. Sfard, A.: On two metaphors for learning and the dangers of choosing just one. Educational Researcher 27(2) (1998) 4-13
15. Soller, A. Supporting Social Interaction in an Intelligent Collaborative Learning System. Int. J. of Artificial Intelligence in Education, 12, (2001) 40-62
16. Wasserman, S. & Faust, K.: Social Network Analysis: Methods and Applications. Cambridge Univ. Press, Cambridge (1994)
17. Webb, N.: Testing a theoretical model of student interaction and learning in small groups. In: R. Hertz-Lazarowitz and N. Miller (Eds.), Interaction in Cooperative Groups: The Theoretical Anatomy of Group Learning. Cambridge Univ. Press, NY (1992) 102-119
18. Wortham, D.W.: Nodal and Matrix Analyses of Communication Patterns in Small Groups. Proceedings of the Computer Support for Collaborative Learning (CSCL) 1999 Conference. Palo Alto, CA (1999) 681-686

A Tailorable Collaborative Learning System That Combines OGSA Grid Services and IMS-LD Scripting

Miguel L. Bote-Lorenzo, Luis M. Vaquero-González, Guillermo Vega-Gorgojo, Yannis A. Dimitriadis, Juan I. Asensio-Pérez, Eduardo Gómez-Sánchez, and Davinia Hernández-Leo

School of Telecommunications Engineering, University of Valladolid
Camino Viejo del Cementerio s/n, 47011 Valladolid, Spain
{migbot,guiveg,yannis,juaase,edugom}@tel.uva.es
{lvaqgon,dherleo}@ulises.tel.uva.es

Abstract. This paper presents Gridcole, a new collaborative learning system that can be easily tailored by educators in order to support their own CSCL scenarios, using computing services provided by third parties in the form of OGSA grid services. Educators employ scripts in order to describe the sequence of learning activities and required tools, with standard IMS-LD notation. Thus, through the integration of coarse-grained tools, that may even offer supercomputing capabilities or use specific hardware resources, educators do not depend on software developers to easily configure a suitable environment in order to support a broad range of collaborative scenarios. An example of a learning scenario for a Computer Architecture course is described to illustrate the capabilities of Gridcole.

1 Introduction

Computer Supported Collaborative Learning (CSCL) [1], [2] is a mature research field of increasing interest in recent years. As a result, many learning systems have been developed in order to promote and support collaborative methods of learning. These systems typically provide an environment with several tools in order to support a given learning scenario. For instance, C-CHENE [3] is a system aiming to teach modeling and the concept of energy in physics that provides an environment consisting of a structured chat, a free chat, and a collaborative energy chain editor. However, a number of drawbacks may be found in most of these systems.

First, most systems *are not tailorable.* A computer system is said tailorable if it provides users with some means to modify its functionalities in order to better suit their needs [4]. In this sense, tailorable collaborative learning systems typically enable easy integration of suitable tools within a single environment in order to support a given collaborative learning scenario. Examples of tailorable collaborative systems include DARE [5] and Symba [6].

Besides, most collaborative systems *do not allow the interpretation of collaboration scripts.* A collaboration script is a set of instructions prescribing how students

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 305–321, 2004.

should form groups, how they should interact and collaborate and how they should solve a problem [7]. Scripts can be enacted by script interpreters integrated within collaborative systems in order to manage the sequence of learning activities leading students to desired objectives. This way, structured interactions between students can be enabled in a collaborative system, thus enhancing the effectiveness of collaborative learning [7]. COW [8] and CopperCore [9] are two examples of collaboration script interpreters.

Moreover, most systems *do not enable the use of tools requiring supercomputing capabilities.* Therefore, the use of collaborative learning systems is limited in many natural sciences and medical areas in which such tools would be needed. For instance, consider a surgery school where students learn collaboratively to operate. In such a scenario, supercomputing capabilities are required in order to compute high-quality visualizations of a complex human body model, which is collaboratively manipulated by students in real time, and to display the computation result on remote screens. CoVis [10] is another example of a collaborative system requiring supercomputing capabilities.

Finally, most systems *do not allow the use of tools requiring specific hardware resources.* Again, there are many collaborative learning scenarios in which the support provided by tools requiring specific hardware resources is needed. For example, consider a Computer Architecture course in which students collaborate to decide the best computing solution for the requirements posed by a given customer. In order to support this scenario, a learning system should provide not only collaborative tools such as debate and voting tools, but also benchmarking tools that use specific hardware resources such as the different real machines to be tested by students. Ref. [11] also describes a collaborative learning system that integrates micro-robots.

Then, although several CSCL systems have partially tackled these issues, none of them has provided a solution for all of them. In this paper we present Gridcole, a new collaborative learning system that can be easily tailored by educators in order to support their own collaborative scenarios. Tools integrated by our system are not restricted in terms of supercomputing capabilities or specific hardware needs. Furthermore, interpretation of collaboration scripts is enabled. Gridcole is based on two standards that have appeared recently. On the one hand, the IMS Learning Design (IMS-LD) specification [12] provides an Educational Modeling Language (EML) that enables formal description of teaching-learning processes for a wide range of pedagogies in online learning, including collaborative learning [13], [14]. In this sense, IMS-LD can be employed for the description of collaboration scripts. On the other hand, the Open Grid Services Architecture (OGSA) [15] has emerged as the *de facto* standard [16] for the construction of grid infrastructures [17]. OGSA-based grids enable seamless integration of resources to allow the delivery of large amounts of computational power and use of applications requiring specific hardware resources.

The organization of this paper is as follows. Section 2 first shows how IMS-LD specification and OGSA-based grids can be employed in order to tackle the shortcomings of current collaborative systems. Next, Gridcole, a collaborative system building on both specifications is presented. Section 3 introduces and discusses Gridcole architecture. Moreover, a prototype of our system is described. Section 4 further

illustrates the use of Gridcole system by presenting and discussing a real collaborative learning scenario for a Computer Architecture course that can be supported by our system. Finally, conclusions and future work can be found in section 5.

2 Gridcole Technologies

OGSA-based grids and the IMS-LD standard can be combined to address the shortcomings of current collaborative learning systems identified above. This section first shows how IMS-LD and OGSA grids can be employed within the context of a collaborative learning system. Next, Gridcole, a new collaborative learning system that combines both IMS-LD and OGSA concepts, is outlined.

2.1 IMS-LD Documents

The IMS-LD specification defines a structured XML-based language that can be employed to formally express *learning designs.* A learning design is a description of a method enabling learners to attain desired learning objectives by performing predefined learning activities in a certain order within the context of a given learning environment [12].

More specifically, a learning design describes a *learning scenario* in terms of a *learning flow,* and a set of *environments.* The learning flow specifies the sequence of *activities* that learners should perform in order to reach predefined learning objectives according to the different roles that they may play in a learning design. Environments are described in terms of resources, i.e. *tools* and *contents,* that should assist learners during the realization of each activity according to the role played.

Particularly, IMS-LD can be used to describe *collaborative learning scenarios* [13], [14]. It enables the design of learning processes that include several roles, each of which can be played by several people (a group). A collaborative learning experience can be described by associating multiple people and/or multiple roles to the same learning activity. Furthermore, as it has been already mentioned, IMS-LD enables their activities to be specified in coordinated learning flows that are analogous to groupware workflows [12].

An IMS-LD document can thus be employed in a collaborative learning system in two different ways. First, as a collaboration script that could be interpreted by the system in order to structure interactions between students and thus enhancing collaborative learning. Second, as a tailoring script describing the tools that should be integrated by the system within a single environment in order to properly support each activity defined in the learning design document.

2.2 OGSA-Based Tools

The term *grid computing* [17] is commonly used to refer to a large-scale infrastructure that allows the sharing of both software and hardware distributed heterogeneous resources [18]. In analogy with the electric power grid that provides pervasive access

to electric power, the computational grid provides ubiquitous access to software and hardware resources.

Recently, the emergence of OGSA as the *de facto* standard [16] for grid middleware has turned in a noticeable shift towards service-oriented architecture [15]. Following OGSA, all resources in a grid must be offered in the form of a *grid service,* which can be regarded as a web service with some additional features including instance creation, lifetime management, notifications and security [15].

More specifically, a grid service is a software or hardware resource offered by a third-party provider that is exposed through a standard interface adhering to OGSA specifications. The creation of an *instance* of a given grid service can be requested to its corresponding *grid service factory,* which is in turn another grid service. The grid service instance can then be invoked using standard protocols. Providers typically publish their services in well-known directories thus enabling service discovery.

Grid middleware allows seamless integration of grid services. This way, providers are enabled to supply to the grid community any tool requiring supercomputing capabilities or specific hardware resources in the form of grid services. However, it must be noticed that tools without such requirements can also be offered as grid services in an OGSA-based grid.

A grid could thus be employed in order to provide collaborative learning systems with a large pool of tools offered by third-party providers in the form of OGSA-compliant grid services [19]. Significantly, tools would not be limited because of the need of certain computing power or specific hardware resources needs. Following OGSA standards, collaborative learning systems could integrate these tools in order to support different collaborative activities.

2.3 System Outline

Gridcole is a new collaborative learning system in which the use of IMS-LD documents and OGSA tools described in previous subsections is combined. This subsection outlines the way Gridcole operates.

Gridcole enables educators to provide in an IMS-LD document the description of a collaborative learning scenario. Such scenario is expressed in terms of a sequence of activities to be performed by students, as well as a generic description of the tools that make up the collaborative environment that should be available for the realization of each activity. IMS-LD documents are stored in the system for later retrieval and use.

The educator can then choose one of the existing learning designs, so that Gridcole looks for suitable tools supplied by third-party providers in an OGSA-based grid according to the IMS-LD document selected. In case more than one tool service is found matching a tool specification of the learning design, the educator can opt for any of them.

Next, the educator provides the list of participants that will be allowed to join the execution of the learning design (i.e. the realization of a collaborative learning scenario) as well as the roles they will play in such execution. Once this operation is

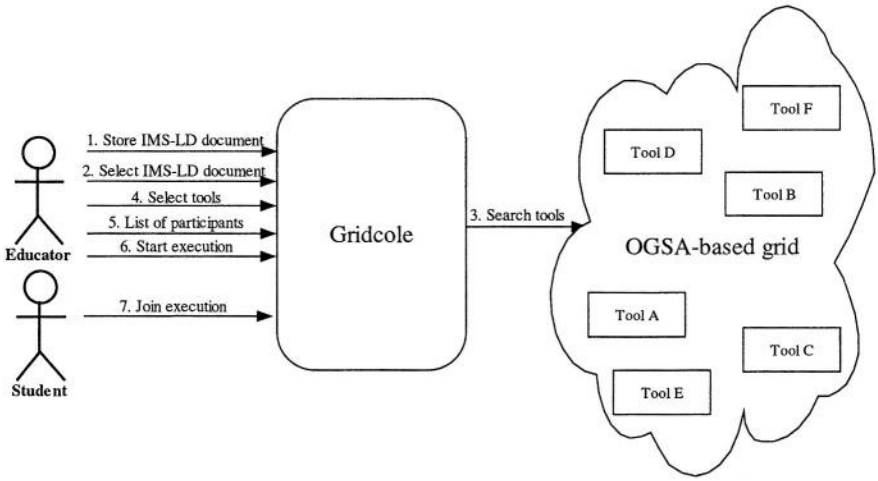

Fig. 1. Typical interactions of educator and student users with Gridcole system before execution of a learning design

completed, the educator can launch the execution of the learning design and students can join this execution and start to collaborate. Fig. 1 shows this sequence of interactions.

During execution, Gridcole determines the sequence of activities to be performed by each participant leading to desired learning objectives. For each participant, the system provides an application desktop in which the collaborative environment defined for the current activity is reified. The desktop integrates the graphical interface panes of grid service tools supporting the realization of current activity. Participants can interact with grid services through these interfaces in order to fruitfully collaborate with other users by means of collaborative tools, or to work with non-collaborative tools. Fig. 2 illustrates these ideas.

3 Gridcole Architecture

This section introduces an architecture for the collaborative learning system outlined before. With this aim, the requirements that have driven the architectural design are stated. Besides, the architecture is presented and is shown how the elements of the architecture interact in order to perform Gridcole main operations. Next, the architecture is discussed and a Gridcole prototype is introduced.

3.1 Requirements

The architectural design of the Gridcole system has been driven by the following requirements, most of which were implicitly introduced in previous subsections:

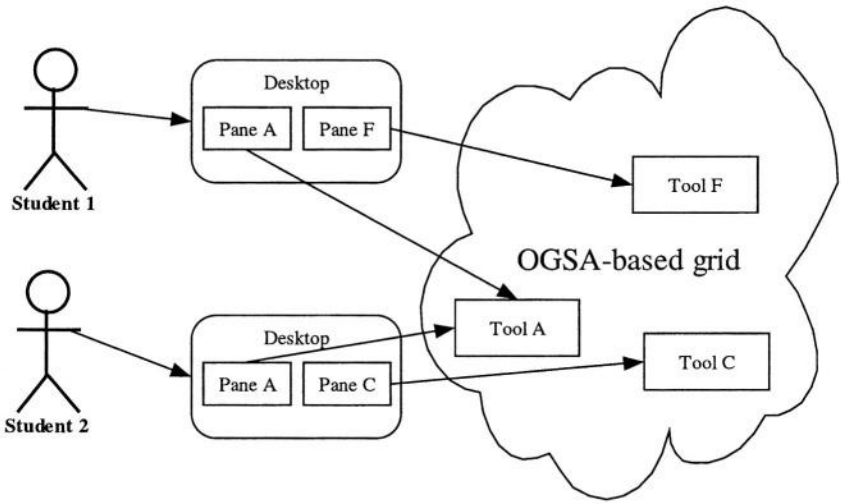

Fig. 2. Two students collaborate using personalized desktops during execution of a learning design. Pane A enables the use of a collaborative tool while Panes C and F allow using different non-collaborative tools

- Support environment tailorability. The system must be able to find suitable tools and to integrate them in collaborative environments as specified by educators in IMS-LD documents.
- Support collaboration script interpretation. The system must be able to interpret the collaboration script described in an IMS-LD document. This way, Gridcole should coordinate the executions of multiple activities defined in such document by instructing *who* (users) does *what* (activities), using *which* (tools and documents) and *when.*
- Provide for new capabilities, such as supercomputing and use of specific hardware units, while neither requiring any specific infrastructure nor restricting the use of tools that do not employ such capabilities. The system must be able to integrate collaborative environments using tools supplied by third-party providers in the form of OGSA-compliant grid services.
- Place low requirements on service providers. Grid services are supplied by third-party providers. This promotes the existence of a wide range of tool services that can be employed by Gridcole for integration purposes. However, service providers are not likely to adhere to non-widely accepted standards. Thus, minimum requirements should be placed on providers in order not to discourage them from offering services that can be integrated in Gridcole.

3.2 System Architecture

In order to meet the requirements mentioned before, Gridcole architecture has been conceived as shown in Fig. 3, consisting of four main elements: a Web Portal, Learning Object Index Services, Applications, and Application Clients.

The *Web Portal* provides a single access point to the system using a simple web browser. Moreover, it is responsible for assisting authenticated users when operating within Gridcole environment. The portal uses a *Database* to keep administrative and system-related information. A *Repository* is also employed to store learning design documents as well as information regarding the location of resources that make up the collaborative environments tailored by these documents. The *Resource Searcher* is used to find suitable tools and content specified in learning design documents by querying learning object index services.

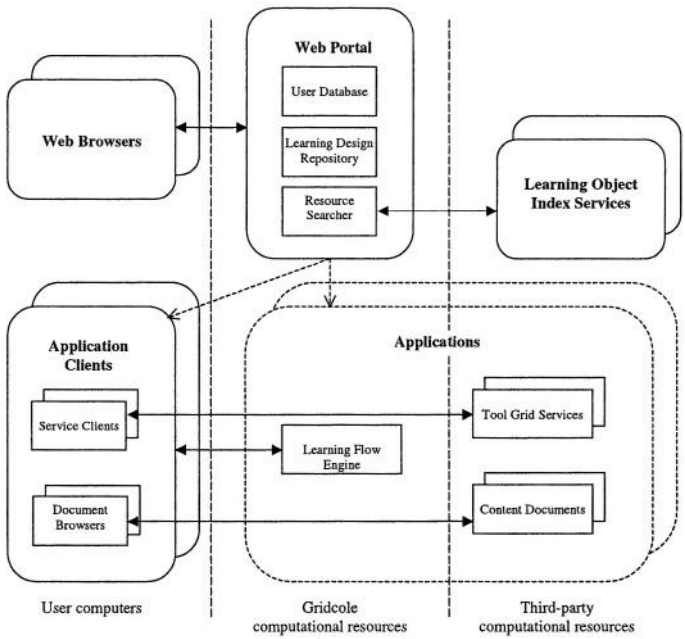

Fig. 3. Architecture of Gridcole system

A *Learning Object Index Service* (LOIS) is a grid service in which third-party providers can register either tools offered in the form of OGSA-compliant grid services or content documents supplied as URL-addressable documents. Each LOIS entry includes a tool or a document description following the IMS Learning Resource Metadata specification [20] and a pointer to a resource location. The location of the LOIS is known *a priori* by the portal. The use of LOIS thus enables the discovery of OGSA-based tools and content documents to be integrated in collaborative environments.

An *Application* is the assembly of a *Learning Flow Engine* (LFE), some *Tool Grid Services,* and *Content Documents* that support the realization of a given collaborative learning scenario following its corresponding IMS-LD learning design document. The LFE is a grid service that interprets the collaboration script described in a learning design. It is responsible for automatically scheduling the activities to be per-

formed by each user as well as providing suitable tools and contents in order to properly assist him during the realization of each activity. Applications are launched by educators using the web portal.

The *Application Client* provides a desktop-alike graphical user interface (GUI) that allows users easy collaboration and interaction with tool services and content documents. Moreover, it is responsible for automatically downloading necessary service clients and document browsers. *Service Clients* are software components providing a graphical interface that allows user easy interaction with specific tool services, while *Document Browsers* enable viewing content documents. Both service clients and document browsers provide GUIs, which are properly integrated within the application client desktop in the form of panes. Application clients are automatically downloaded and installed in each user's computer before starting to use a Gridcole application.

3.3 System Functioning

Gridcole basically performs two main operations: the location of tools and contents specified in a learning design in order to support a collaborative scenario, and the execution of such learning design. Next, it is shown how the elements of the architecture introduced above interact in order to support both operations.

The resource searcher element of the web portal is responsible for locating the tools and contents prescribed by an educator in a given IMS-LD document. In order to perform such a task, the searcher queries all LOISs known by the system. Sometimes slightly different tools may match with IMS-LD document specifications. In this case, the educator can opt for any of them. Once the tools and documents to be used for the support of a given learning design have been determined, a copy of the IMS-LD document and the location of the corresponding grid services are stored in a single file according to IMS Content Packaging [21] specification. Following IMS terminology, this file is called a unit of learning.

The educator may start the execution of a learning design after providing a list of participants that are allowed to join it. As a result of this operation, a new LFE that must interpret an IMS-LD design is created. Subsequently, the LFE is registered in the system and provided with all information necessary to coordinate the run of the learning unit. Then, when users join the run, the location of the LFE is provided to the application clients of participants in the form of a Grid Service Handle (GSH) [15].

After that, both application clients and the LFE perform a sequence of low-level activities in order to allow application execution. These activities are shown in Fig. 4 and described next.

First, application clients register in the LFE. Registration is only possible if the user is in the list of allowed participants. If registration succeeds, the LFE provides the application client with a list of URLs pointing at the service clients and document browsers required for current activity. While the application clients are downloading service clients, the LFE creates instances of the tool services scheduled to support this learning activity, using the GSHs of the corresponding grid service factories. If the

tool is collaborative, one instance of the corresponding service is created and shared by each group of collaborators. Otherwise, different instances are created for each user.

Next, the LFE provides the application clients with a list of GSHs and URLs pointing to the tool service instances and content documents supplied in order to assist the user while carrying out the scheduled learning activity. Then, application clients instantiate the downloaded service clients and document browsers and provides them with the corresponding GSHs and URLs.

Users can then perform the learning activity assisted by scheduled documents and tools. The application client presents all service clients and document browsers as graphical interface panes integrated within a single user desktop. Users can thus employ panes to fruitfully collaborate with other participants by means of collaborative tools, to work with non-collaborative tools, or to browse content documents.

The LFE may assign a new activity to a user when he has completed the current one or when time assigned to it is over. In this case, the LFE notifies the application client that current activity is over. As a result, the application client disables the use of current tools and document browser. Service clients and document browsers which are going to be used in the following activities are kept in the application clients, while the rest are removed. Then, the process of loading new service clients and document browsers and binding them to the corresponding tool services is carried out for the new activity.

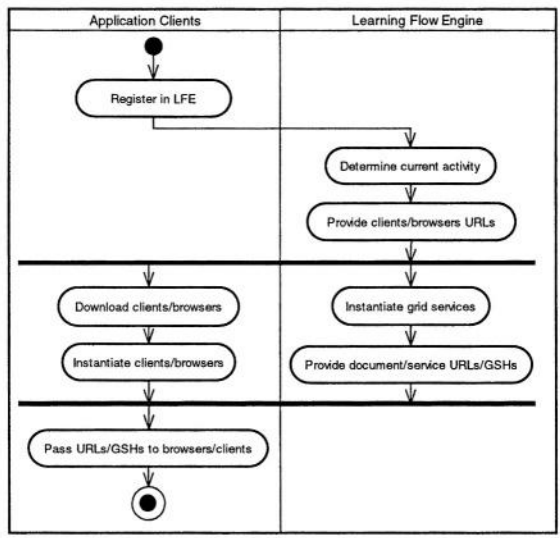

Fig. 4. Activities performed by application clients and LFE

3.4 Discussion

Gridcole can be considered to overcome the drawbacks of current systems identified in the introduction of this paper.

In this sense, the system can be tailored by means of an IMS-LD document in which the educator specifies the tools required to support a given activity. These tools are located by querying a LOIS in which service providers register their OGSA-based tools. Later, the corresponding collaborative environment is provided in the form of an application desktop that integrates the interfaces of all tools defined to support the activity to be realized. This way, our system can be said to enable tailoring by integration [4].

Furthermore, collaboration scripts can also be provided to Gridcole following IMS-LD standard. The interpretation of these scripts is performed by the LFE. The LFE is thus an IMS-LD engine similar to Cow [8] or CopperCore [9], although there is one main difference: the LFE operates in a grid-based environment. Nevertheless, it must be noticed that the use of scripting feature is not mandatory. IMS-LD can also be used to define an collaboration environment, supported by computational tools found through Gridcole, but used to collaborate freely without any prescription on the learning activities to be carried out.

Gridcole also enables the use of tools with supercomputing or specific hardware requirements that can be integrated in collaborative environments if desired. This way, new collaborative learning scenarios can be supported. Again, the use of this type of tools is optional in Gridcole, since tools without such requirements can also be offered as OGSA-based grid services and thus integrated by the system.

Moreover, a number of additional advantages derived from the use of OGSA and IMS-LD standards can also be identified.

First, grid services represent high-level abstractions that can be employed by educators as building blocks to assemble customized collaborative learning environments in order to support a given scenario. Following [22], high-level abstractions are closer to educators' mental model and thus enable and promote educational software reuse in the integration of learning environments. In addition, IMS-LD also promotes the reuse of learning designs and, as a consequence, the reuse of software tools that may be employed to support them.

Furthermore, OGSA service-oriented philosophy pushes deployment and set-up responsibilities to the grid service provider. This contributes to avoid the *technification* problem identified in [23], which refers to the need of technical skills that make it difficult for teachers and students to use learning systems. Finally, it is expected that IMS-LD community will develop author tools soon. These tools will offer educators easy edition of learning designs in order to describe collaborative learning scenarios to be supported by a grid-based collaborative system. Again, this would contribute to avoid the technification problem.

3.5 Gridcole Implementation

An implementation of Gridcole is currently under development using Globus Toolkit 3 (GT3) [24] and Java technologies. However, a prototype has already been built in order to show the feasibility of the proposed system.

The prototype includes a web portal providing basic learning unit creation, instantiation, launching and joining assistance facilities. Hence, some technical skills that

educators and students are not expected to have are still required in order to operate this prototype Gridcole system. Fig. 5 shows a snapshot of the web portal.

The portal has also been partially integrated with a first version of the resource searcher and a learning object index service. Currently, the latter only allows queries on a keyword-basis. An application client that is able to host service clients in the form of Java Beans is available too. This client currently provides a simple application desktop. Furthermore, a limited LFE, which is able to interpret a simplification of IMS-LD language, has been developed.

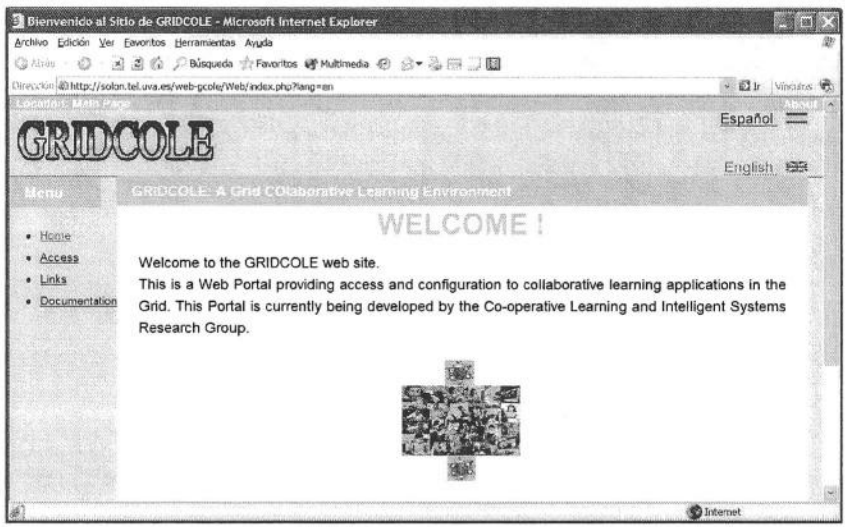

Fig. 5. The prototype provides access to Gridcole functionalities through a simple web portal

4 Sample Collaborative Learning Scenario

In order to further illustrate the use of Gridcole, this section shows how our system could support a real collaborative learning scenario requiring tools with special hardware needs. After describing its educational context, this scenario is introduced. Next, it is shown and discussed the valuable support that Gridcole can provide and our efforts towards the realization of this scenario are described.

4.1 Educational Context

The collaborative learning scenario described in this section is to be applied in a course on *Computer Architecture* for undergraduate students at our University. This course is organized around a computer architecture design and evaluation design project which in turn is divided in three subprojects (see [25] for details). In this project, students organize themselves in groups of four pairs and collaboratively play the

role of *consultants* that have to advise on a computing solution (machine, operating system, software, network, etc.) for a given *customer,* which is played by the teacher.

The scenario considered here concerns the first subproject, in which students get to know the client, model the customer's presumed computational load by mixing standard benchmarks, test *real* machines using the benchmarks, and finally make a recommendation to their client. This subproject pursues clear *learning objectives.* On the content side, it is expected that students learn how to use benchmarks, and get a quantitative impression on a few real machines (with different CPUs, memories, etc.). On the skills side, several abilities, such as interpreting and selecting information, arguing, and taking compromise solutions are promoted. The subproject lasts for six two-hour sessions on approximately four weeks.

4.2 Scenario Description

The collaborative learning scenario described here has been designed by teachers of the course in which it is to be applied. The activities, tools and contents defined for this scenario are briefly outlined next.

For the first activity, students should *study customer needs* while the educator should in turn play the role of the client in order to *clarify customer needs.* This activity is supported by documentation collecting client requirements, a collaborative concept map tool and a debate tool.

In the following activity, students *model the computational load* of the customer with a voting tool that assists the decision-making process and a collaborative questionnaire tool that allows filling in some required forms. In the next activity, students will *distribute* four groups of different real *machines* among them, so that each student benchmarks a group of machines. Here, a collaborative task assignment tool and a chat tool will be employed.

Subsequently, students will *benchmark* those machines that have been assigned to them, collecting the results and studying the documentation of the benchmarks and the machines. Benchmark documentation, and several benchmarking tools that run in computers with different architecture are required to support this activity.

Next, the well-known jigsaw collaboration pattern [7] is applied. Students who have benchmarked the *same machine debate* the suitability *of* such machine for their customer according to benchmark results. This activity will be supported by a debate tool and a chat tool. Finally, students have to *debate* the results with other members of their group that have benchmarked *different machines.* As a result, each group should generate a technical report presenting and arguing the best solution for their customer. This activity will be supported by a debate tool and a collaborative text editor tool.

4.3 Gridcole Support

The support of the scenario presented above requires a number of tools that are not likely to be found in a single non-tailorable collaborative learning system. Hence, most current systems could not be employed to support this scenario, while Gridcole

can be tailored to integrate all tools as long as they are offered in the form of OGSA-grid services by any provider.

The educator can easily tailor Gridcole by providing an IMS-LD document that describes this scenario. Fig. 6 shows an excerpt from such document. The edition of this document may be considered difficult at present for most educators, since it is encoded in XML format. However, authoring tools announced by IMS-LD community will tackle this problem.

Following the specifications provided in this document, Gridcole can locate suitable tools to support the scenario in an OGSA-based grid. Particularly, a tool with special hardware requirements is needed: the benchmarking tool must be executed in machines with different architecture that may be considered of interest by the educators. Again, current collaborative learning systems do not enable the integration of this type of tools, while Gridcole does.

```
<?xml version="1.0" encoding="UTF-8"?>
<learning-design identifier="GRIDCOLE_AO" level="A" uri="">
 <title>Sample Computer Arquitecture Scenario</title>
 <learning-objectives>
    <item identifierref="" identifier="LOB-objectives-list"/>
 </learning-objectives>
...
 <components>
    <roles>
      <learner identifier="R-clientGroup" create-new="allowed">
        <learner identifier="R-machineLearner" create-new="allowed"/>
      </learner>
      <staff identifier="R-teacher"/>
    </roles>
...
  <activities>
    <learning-activity identifier="LA-distribute-machines">
      <environment-ref ref="E-tasks-distribution"/>
        <activity-description>
          <item identifierref=" "/>
      </activity-description>
    </learning-activity>
    <support-activity identifier="SA-supervise-distribution">
      <environment-ref ref="E-tasks-distribution"/>
        <activity-description>
          <item identifierref=" "/>
      </activity-description>
    </support-activity>
...
  </activities>
   <environments>
     <environment identifier="E-tasks-distribution">
       <service identifier="S-tasks-distribution">
         <conference conference-type="synchronous">
           <participant role-ref="R-clientGroup"/>
           <participant role-ref="R-teacher"/>
            <item identifierref="" identifier="I-tasks-distribution-tool"/>
         </conference>
       </service>
       <service identifier="S-chat">
         <conference conference-type="synchronous">
           <participant role-ref="R-clientGroup"/>
           <participant role-ref="R-teacher"/>
           <item identifierref="" identifier="I-chat"/>
         </conference>
       </service>
...
     </environment>
  </components>
```

```
<method>
  <play>
    <act>
      <role-part>
          <role-ref ref="R-clientGroup"/>
          <learning-activity-ref ref="LA-study-customer-needs"/>
      </role-part>
      <role-part>
          <role-ref ref="R-teacher"/>
          <support-activity-ref ref="SA-clarify-customer-needs"/>
      </role-part>
    </act>
    <act>
      <role-part>
          <role-ref ref="R-clientGroup"/>
          <activity-stucture-ref ref="AS-model-load"/>
      </role-part>
    </act>
    <act>
      <role-part>
          <role-ref ref="R-clientGroup"/>
          <learning-activity-ref ref="LA-distribute-machines"/>
      </role-part>
      <role-part>
          <role-ref ref="R-teacher"/>
          <support-activity-ref ref="SA-supervise-distribution"/>
      </role-part>
    </act>
    <act>
      <role-part>
          <role-ref ref="R-machineLearner"/>
          <learning-activity-ref ref="LA-benchmarking"/>
      </role-part>
    </act>
    <act>
      <role-part>
          <role-ref ref="R-machineLearner"/>
          <learning-activity-ref ref="LA-debate-machine"/>
      </role-part>
    </act>
      <role-part>
          <role-ref ref="R-clientGroup"/>
          <learning-activity-ref ref="LA-debate-client"/>
      </role-part>
    </act>
  </play>
 </method>
</learning-design>
```

Fig. 6. Excerpt showing some key elements of the IMS-LD document that describes the learning scenario considered here. Sample definitions of roles, activities, and an environment can be found under <*roles*>, <*activities*> and <*environments*> tags respectively. The description of the sequence of activities to be performed by participants is shown under <*method*> tag

Moreover, in current course configuration, benchmarking is performed in existent machines owned by our University. As a consequence, the range of architectures that

can be chosen for evaluation is not as broad as teachers would like. In this sense, Gridcole increases opportunities for students to benchmark remote machines owned by grid service providers.

Furthermore, the fact that Gridcole interprets the defined collaboration script during the realization of the scenario enables a structured collaboration, which in turn can enhance learning effectiveness.

The support provided by Gridcole in this collaborative learning scenario will be thoroughly evaluated from the educational point of view next fall semester starting in October 2004. Evaluation will be carried out following the method proposed in [26] By this date, it is expected that the development of Gridcole as well as of the tools required to support this scenario will be finished. These tools will be offered in the OGSA-based grid currently shared by the three different Universities that participate in CRAC research project [27].

By the time of writing this paper, a short version of the scenario presented above has been deployed using Gridcole under laboratory conditions. This scenario combines three tools that have already been developed, the collaborative task assignment tool, the benchmark tool, and a chat service. The short scenario includes the following four activities: distribute machines using task assignment and chat tools, benchmark machines supported by benchmarking and chat tool, debate with benchmarkers of the same machine using chat tool, and debate with other members of group again using chat tool. Fig. 7 shows a snapshot of Gridcole application desktop as seen by students during "benchmark machines" activity.

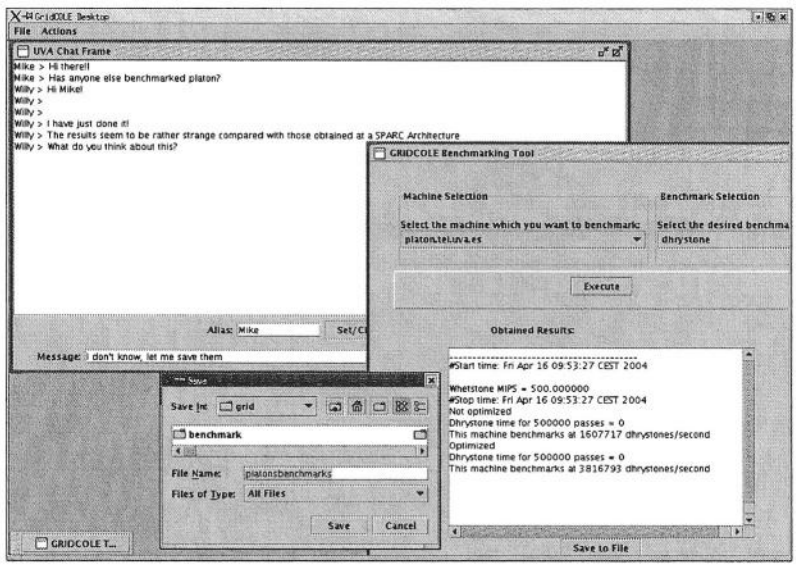

Fig. 7. Gridcole application desktop integrates a benchmarking tool and a chat tool in order to support "benchmark machines" activity

5 Conclusions and Future Work

This paper has presented Gridcole, a collaborative learning system that brings together the benefits of IMS-LD scripting to describe in detail a learning scenario as well as the computational tools needed to support it, along with the OGSA standard to describe educational tools with grid services interfaces, so that the system can automatically integrate third party software without putting severe restrictions on these software providers. This way, Gridcole offers a means for educators to tailor a computational support for their particular learning scenario, benefiting from resources in the grid, thus broadening the range of possible applications to those requiring supercomputing capabilities or special hardware tools.

In addition, since the deployment of the computational support for a learning activity is carried out automatically by Gridcole elements, the technification problem detected for other collaborative learning systems is reduced. Furthermore, tools offered as grid services generally have a coarse granularity, as compared to other software reuse units like components or objects. This granularity is better fit to the way educators describe their computational needs, thus enhancing software reuse, and providing educators with independence from software developers.

The learning scenario described in this paper illustrates some of these properties. For example, two different variations of the learning scenario have been mentioned (though in this case the latter is a simplification of the former to test the prototype) that use the same tools. To do so, educators simply have to write two different IMS-LD scripts describing them, since the service search and the application deployment is performed automatically by Gridcole. Moreover, many other learning scenarios could be devised reusing tools like the chat, the voting tool or the task assignment tool. In addition, educators could choose among several implementations for them, offered by different service providers.

Upcoming work in the near future includes completing Gridcole prototype to have all its elements with full functionality, and evaluate its use in the learning scenario described above, at 2004 fall semester. In addition, other enhancements can be brought into Gridcole, such as a framework of useful collaborative and non-collaborative grid services, that could facilitate the development and communication of third party software. Furthermore, the use of Peer-to-Peer (P2P) tools can offer a complementary channel for collaboration, supporting self-organization and thus overcoming IMS-LD's tightness to organize collaborative activities. Finally, scheduling facilities, which are usually employed in grid computing, will be studied in order to provide collaborative tools according to predefined Quality of Service.

Acknowledgements

This work has been partially funded by European Commission Project EAC/61/03/GR009 and Spanish Ministry of Science and Technology Project TIC2002-04258-C03-02. The authors would also like to thank the rest of "Intelligent & Cooperative Systems Research Group" for their support and ideas.

References

1. Koschmann, T.: CSCL: Theory and Practice of an Emerging Paradigm. Lawrence Erlbaum, Malwah, NJ, USA (1996)
2. Dillenbourg, P.: Collaborative Learning: Cognitive and Computational Approaches. Elsevier Science, Oxford, UK (1999)
3. Baker, M., Lund, K.: Flexibly Structuring the Interaction in a CSCL Environment. Proc. of the European Conference on Artificial Intelligence in Education, Lisbon, Portugal (1996) 401-407
4. Morch, A.: Three Levels of End-User Tailoring: Customization, Integration and Extension. Proc. of the 3rd Decennial Aarhus Conference, Aarhus, Denmark (1995) 41-45
5. Bourguin, G., Derycke, A.: Integrating the CSCL Activities into Virtual Campuses: Foundations of a New Infrastructure for Distributed Collective Activities. Proc. of the European Conference on Computer Supported Collaborative Learning, Euro-CSCL 2001, Maastricht, The Netherlands (2001) 123-130
6. Betbeder, M.-L., Tchounikine, P.: Symba, a Tailorable Framework to Support Collective Activities in a Learning Context. Proc. of the 9th International Workshop on Groupware, CRIWG 2003, Autrans, France. Lecture Notes in Computer Science vol. 2806, Springer-Verlag (2003) 90-98
7. Dillenbourg, P.: Over-Scripting CSCL: the Risks of Blending Collaborative Learning With Instructional Design. In: Kirschner, P. A. (eds.): Three Worlds of CSCL. Can We Support CSCL. Heerlen, Open Universiteit Nederland (2002) 61-91
8. Vantroys, T., Peter, Y.: COW, a Flexible Platform for the Enactment of Learning Scenarios. Proc. of the 9th International Workshop on Groupware, CRIWG 2003, Autrans, France. Lecture Notes in Computer Science vol. 2806, Springer-Verlag (2003) 168-182
9. CopperCore Project Website. http://coppercore.org/
10. Ramamurthy, M. K. , Wilhelmson, R. B., Pea, R. D., Louis M., Edelson, D. C.: CoVis: A National Science Education Collaboratory. Proc. of the American Meteorological Society 4th Conference on Education, Dallas, TX, USA (1995)
11. Despres, C., George, S.: Computer-Supported Distance Learning: An Example in Educational Robotics. Proc. of the 9th International PEG Conference, Exeter, UK (1999) 344-353
12. IMS Global Learning Consortium: IMS Learning Design Information Model V1.0, Final Specification. http://www.imsproject.org/learningdesign/index.cfm (2003)
13. Hernández Leo, Davinia. From IMS-LD to MDA: approaches to component-based CSCL applications modeling (in Spanish). Valladolid, Spain: University of Valladolid; 2003.
14. Caeiro, M., Anido, L., Llamas, M.: A Critical Analysis of IMS Learning Design. Proc. of the Computer Supported Collaborative Learning Conference, CSCL 2003 Kluwer Academic Publishers (2003) 363-367
15. Foster, I., Kesselman, C., Nick, J. M., Tuecke, S.: The Physiology of the Grid. In: Berman, F., Fox, G. , Hey, A. (eds.): Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Chichester, UK (2003) 217-249
16. Foster, I., Kesselman, C., Nick, J. M., Tuecke, S.: Grid Services for Distributed System Integration. Computer. 35 (6) (2002) 37-46
17. Berman, F., Fox, G., Hey, A.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Chichester, UK (2003)
18. Bote-Lorenzo, M. L., Dimitriadis, Y. A., Gómez-Sánchez, E.: Grid Characteristics and Uses: a Grid Definition. PostProc. of the 1st European Across Grids Conference, Santiago de Compostela, Spain. Lecture Notes in Computer Science vol. 2970, Springer-Verlag (2004) 291-298

19. Bote-Lorenzo, M. L., Asensio-Pérez, J. I., Vega-Gorgojo, G., Vaquero González, L. M., Gómez-Sánchez, E., Dimitriadis, Y. A.: Grid Computing and Component-Based Software Engineering in Computer Supported Collaborative Learning. Proc. of the International Conference on Computational Science, ICCS 2004, Kraków, Poland. Lecture Notes in Computer Science vol. 3036, Springer Verlag (2004) 503-506
20. IMS Global Learning Consortium: IMS Learning Resource Metadata Specification V1.2. http://www.imsglobal.org/metadata/index.cfm (2001)
21. IMS Global Learning Consortium: IMS Content Packaging Information Model V1.1.3, Final Specification. http://www.imsglobal.org/content/packaging/index.cfm (2003)
22. Roschelle, J., DiGiano, C., Koutlis, M., Repenning, A., Phillips, J., Jackiw, N., Suthers, D.: Developing Educational Software Components. Computer. 32 (9) (1999) 50-58
23. Spector, M. J.: An Overview of Progress and Problems in Educational Technology. Interactive Educational Multimedia. (3) (2001) 27-37
24. The Globus Project. http://www.globus.org
25. Dimitriadis, Y. A., Martínez, A., Rubia, B.: Cooperative Learning in Computer Architecture: an Educational Project and Its Network Support. Proc. of the Frontiers in Education Conference, FIE 2001, Reno, NV, USA (2001)
26. Martínez-Monés, A., Dimitriadis, Y., Rubia-Avi, B., Gómez-Sánchez, E., Fuente-Redondo, P.: Combining Qualitative Evaluation and Social Network Analysis for the Study of Classroom Social Interactions. Computers and Education. 41 (4) (2003) 353-368
27. CRAC Project Home Page. http://research.ac.upc.es/crac/

A Model for CSCL Allowing Tailorability: Implementation in the "Electronic Schoolbag" Groupware

Christian Martel, Christine Ferraris,
Bernard Caron, Thibault Carron, Ghislaine Chabert, Christophe Courtin,
Laurence Gagnière, Jean-Charles Marty, and Laurence Vignollet

ERTé "cartable électronique", Laboratoire SysCom, Université de Savoie,
Campus Scientifique, F-73376 Le Bourget-du-Lac cedex, France
{christian.martel,Christine.ferraris}@univ-savoie.fr

Abstract. We describe in this paper a model for Computer Supported Collaborative Learning and the corresponding architecture. This model has been designed to take into account the variety of educational activities and cultures. It offers primitives to endusers, mainly the teachers, allowing them to describe a collaborative activity and to regulate it (i.e. modify it dynamycally). It has been implemented within a groupware based on the metaphor of electronic schoolbag, which is used today by more than 40000 users in both University and secondary french schools. Thanks to the model, the developed groupware is flexible and tailorable.

1 Introduction

Today, proposing technical frameworks to support educational activities can no longer be done without considering the relationships that each learner has with the knowledge, the teachers and the other learners in the learning process [1]. In the past, school favoured individual work, competition and personal merit to the detriment of co-operative activity and group work. For many years, the learner was considered as having an exclusive relationship with the teacher. Encouraged to work alone, to listen attentively to his/her teacher, and to compare his/her results with those of his/her peers, the learner can nowadays devote a part of his/her time to carrying out collective and collaborative work, evaluated as such.

This evolution is mainly due to the many works of psychologists, pedagogues, knowledge engineers and didacticians who have highlighted and stressed the importance of social interactions in building knowledge and in the construction by the learner him/herself of his/her knowledge and learning process [2–9].

By affirming that (a) the learning process depends more on the way the knowledge is built than on the nature of the knowledge itself [10], (b) methodological, metacognitive, relational and social capacities are constitutive of knowledge building [5, 6], (c) times and structures for analysis and auto-regulation are essential for the control of knowledge building [11,12], (d) the practices of self-evaluation, formative evaluation with a teacher and co-evaluation with peers

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 322–337, 2004.

are fundamental to the act of learning [13], these researchers have given a new legitimacy to co-operative learning. They have also given teachers the desire to explore new technical frameworks, in particular groupware [14]. These tools, designed to support group work, to facilitate exchanges, communication and information-sharing in organizations, have evolved little since their inception. Their introduction into the world of education was based on simplistic activity models. Educational tasks are more complex than they seem.

In this paper we propose a general framework for the construction of groupware in education. We illustrate this proposition with the description of the electronic schoolbag, which is the name given to the educational groupware used at the University of Savoie and in several French secondary schools.

2 Requirements

2.1 Requirements for CSCW

One of the clearest lessons of groupware research to date is probably the importance of taking into account the social context in which software is to be used. To get away from the traditional approach which consists in providing only tools to produce, communicate and coordinate, groupware developed at present tries to integrate this social dimension. This is done either in an ad hoc way or by basing the development on a theoretical model of group activity. In the latter case, constructing expressive, generic, instanciable and reusable models that can be implemented has become a challenge for groupware researchers.

These models are generally inspired by work in social sciences. They provide concepts which make it possible to describe what a group activity is and consequently to consider its instrumentation. Among the systems featuring such models, we find meta-groupware such as Worlds [15,16] founded on the theory of "locales", Prospero [17] which is based on ethnomethodology [18] or DARE [19] which refers to the Activity Theory.

Our point of view about groupware is identical: we think that these systems must clearly be based on and integrate an explicit model of the activity of the group which allows the activity's co-construction (configuration of the workspace by the group members themselves) and which takes into account the phenomenon of co-evolution. This term was chosen by [20] in order to "express the fact that cooperative systems must be continuously evolutive, but not in an autonomous or auto-adaptive way, since they must consciously give an account of the evolutions of the needs, attitudes and skills of the users, individually or collectively". In a dual way, co-evolution also states that the system will have an influence on the users which will lead them to adapt.

Our approach is founded on the explicitation of such a model and on its being put at the users' disposal. As users have the primitives of the model at their disposal, they can manipulate them. The aim is first to allow them to build a description of what they think their activity is going to be within the framework of a group workspace, which amounts to building an initial configuration of that group workspace. The second objective is to allow the regulation of the activity

which is going to take place in the workspace. What we mean by regulation is the capacity given to a group or to a person managing a group (for example a teacher within the framework of a collaborative learning situation) to observe the ongoing activity and to influence it. Influencing the activity will here consist in changing its description and making it evolve dynamically, and thus the configuration of the workspace too: it amounts to changing the "rules of the game".

2.2 Requirements for CSCL

Taking into Account Various Educational Cultures and Practices. In the field of Education the activities are complex to describe because of the variety which characterizes them. From one discipline to another, from one school to another, teachers vie with each other in imagination to create varied learning scenarios from scratch or to adapt existing ones. The educational practices which result from this are just as varied, being based in addition on teaching cultures which themselves also vary from one country to another. It is of primary importance that CSCL software allow for this variety and offer a framework making it possible to take this variety into account and to express it. That supposes that this software be conceived in a flexible and modular way: the teachers must be able to have functionalities which will enable them to configure the workspaces of the groups of pupils according to the educational methods that they wish to apply. These functionalities will however not be the prerogative of the teachers alone: the pupils will also have to be able to use them, within the framework of increasingly common practices which put them in the concrete situation of team work, and which require them to work in a group and to organize this work (e.g. collaborative knowledge building - see FLE in Itcole project [21] - or project based learning as used in the Netpro[1] project). These functionalities must therefore be simple to understand and use.

Configuring the Workspaces Starting from the Teaching Scenarios. Upstream of the work of the pupils, the teacher carries out preparatory work which consists in building learning situations adapted to the defined educational objective, situations described by the means of scenarios. The teacher imagines the activities that he/she wishes to implement during the situation, their sequencing, the way in which he/she and the pupils can intervene and interact, as well as the timing of these interventions. He/she then identifies the resources (content, material, tools, software) which are necessary to the situation created and searches for these resources. Once they have been found, he/she will install and arrange them in situ (in class). He/she will then be able to implement with the pupils the situation thus built, planned, organized and prepared, by distributing the tasks to realize and the roles that each person will have to play. Finally, he/she evaluates the progression of the activity, to rectify his/her initial vision, to modify it and to improve it.

[1] http://netpro.evtek.fi/team/

This work, whether it takes place in the traditional context of classroom teaching or by means of Communication and Information Technologies, is a true representation of pedagogic situations. It consists in preparing and configuring the workspace according to the teaching scenario. In the same way, CSCL environments must in our opinion provide the means to create virtual group workspaces which, just like the class, must be able to be configured starting from the scenario and then populated with suitable material and tools. Once again, that highlights the need for flexibility and modularity.

Facilitating the Exchanges of Scenarios and Resources. The development of varied teaching scenarios presupposes the availability of a great quantity and variety of material (educational resources and content, tools...) which can be regarded as the building blocks from which the scenarios will be built. Elementary scenarios can also form part of these bricks. The work then consists in assembling these resources, if they are available, to produce an original scenario, in creating the resources necessary if they are not available, or in adapting an existing scenario. Certain teachers engage readily and in a generally voluntary way in the process of constructing scenarios and resources; editors are also producers of content as are research laboratories in the field of Education sciences (see for example the work of the tecfa on "C3MS building blocks" within the framework of the European project SEED[2]).

The current tendency is for the mutualisation, sharing and exchange of the scenarios and resources produced. Testimony to this are the many initiatives, at the European Community level for example but also internationally, to define and build "learning objects" (e.g. CELEBRATE project), and description standards for resources and learning scenarios (IMS-LD), or to set up resource mutualisation systems (e.g. ARIADNE project). CSCL tools must accompany this effort at mutualisation and provide the means to exchange resources easily which will no longer merely be simple files but objects as complex as scenarios or the means to configure group spaces. The scenarios could thus be wholly or partially re-used from one group of pupils to another, from one workspace to another. That implies an approach "by components" of the CSCL environments, which must be designed in an open way and guarantee an interoperability making it possible to integrate existing pedagogic objects (scenario, resources...). That also presupposes the use of standards for interoperability.

Allowing the Regulation of the Activity. Once the activity has been set up in accordance with the chosen teaching scenario, the teacher is placed as an observer of this activity and can intervene if he/she or the pupils consider it necessary. His/her interventions can be anything from the simple contribution of specific help to the total or partial modification of the scenario considered, if he/she detects a problem or a situation which is not in accordance with what he/she expects, i.e. the pedagogic objectives laid down. Observing the activity, to then be able to intervene and possibly modify its parameters dynamically, during the activity itself, constitutes an activity in itself which we call "reg-

[2] http://tecfaseed.unige.ch/door/

ulation". In the context of Education, the teacher regulates naturally: he/she makes sure that the learning situations that he/she sets up succeed (i.e. the pedagogic objectives are reached) and works to this end. That presupposes having these scenarios to hand and being able to modify them dynamically. Within the framework of CSCL environments, that implies being able to modify dynamically the configuration of the workspaces of the groups of pupils and the way in which they are arranged. For that, it is necessary to have a formal description (a model) of the configuration of these spaces. That in addition implies the setting up in these environments of mechanisms which will allow the observation and the intervention of the teacher.

3 A Model for CSCL

We have proposed an original theoretical model to model joint activity: the *participation model* [22, 23]. It had first been designed to take into account the requirements for CSCW presented above. It has then evolved by considering the educational context and its specific requirements to the architecture presented in section 3.3 below.

3.1 The Participation Model: Modelling Collaborative Activities

This model is inspired by locales theory, ethnomethodology, linguistics, pragmatics and differential semantics. It is based on:

- the *arena* concept, which provides the spatial limits of collaboration: a collaborative activity will take place in an arena. This concept is for us a means to give concrete expression to the fact that group activities are situated;
- an analysis of group activity from the viewpoint of participation: The members of a group are thus considered as active participants in a joint activity in which they will define and negotiate the conditions of their commitment, before the activity starts and also and mainly during the activity itself. Negotiating means defining who is going to participate (who is an "actor" in the considered arena), who is going to do what, what the objects and tools needed to carry out the activity are and what the "rules of thumb" of the activity are. This is represented in the model by the concepts of *actor, thematic role, tool, object* and *scenario.*

An Actor has a thematic role in an arena. It corresponds to the main mission with which the group has entrusted him/her. It can change during the activity. A thematic role gives the actor some rights over the objects and tools which have been put in the arena. These rights are expressed by means of permissions regarding objects and tools.

The *scenarios* describe norms, constraints, recommendations, orders, rights, duties... that govern the interventions of the actors in the group. They constitute the "directions for use" of the group, a description of its functioning. They mainly describe the possible interactions between the actors in the group and their intertwining. Scenarios and roles are tightly linked as roles are used in scenarios to express the rules governing the activity.

3.2 Functionnalities Needed to Manipulate the Model

In this context, regulating means instantiating the participation model to create a first configuration of the shared workspace and then eventually acting upon the concepts of the model and modifying them to make the workspace evolve. From a static point of view, this implies providing the following functionalities:

- creation of a group workspace (i.e. creation of an arena),
- insertion of the required tools and objects in the created arena,
- social configuration of the created arena: defining who belongs to the arena (who the actors are, what their features are), attributing roles to the actors (and at least rights through the allocated roles), defining the rules of the game of the group (constructing the scenarios).

From a dynamic point of view, it implies developing mechanisms:

- to put the configuration into practise (the activity will take place in the arena according to the social configuration defined),
- allowing to dynamically modify the configuration and leading, as a consequence, to the modification of the on-going activity,
- allowing to observe the on-going activity.

3.3 An Architecture for CSCL Based on That Model

A Three-Levelled Architecture. To support the functionalities described above and the underlying participation model, we propose to structure a collaborative workspace (an arena in the participation model) according to the three-levelled architecture presented in fig.1.

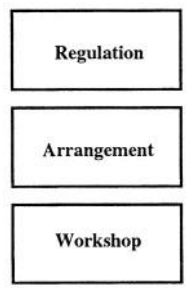

Fig. 1. A three-Levelled Architecture.

The first level, called a "workshop" (see fig.2(a)), is the space where the collaborative activity will actually take place. It is populated with the actors, objects and tools relevant to the activity. The actors will use the tools to produce new objects or even new tools.

The top level is the "regulation" one (see fig.2(c)): a particular actor, whose thematic role is as a *mediator,* will be in charge of regulating the activity, of

"mediating" the interactions between actors. For that purpose, he/she will be aware of what is happening in the workshop by means of observations. Actually, he/she will have at his/her disposal a tool called an "observatory" that will provide a synthetic and pedagogically founded view of the on-going activity in the workshop. A description of the pedagogical objectives will also be available. Observations and pedagogical goals are information which will allow the mediator to decide if an intervention in the on-going activity is required. He/she will then be able to effectively intervene in this activity and to modify its functioning by acting on the configuration parameters and modifying them, or by modifying the configuration of the workshop (adding new tools and/or objects or suppressing some of them). The dynamics of the regulation activity are sketched out in fig.3. It is clear that, in most educational settings, the role of *mediator* will be attributed to the teacher.

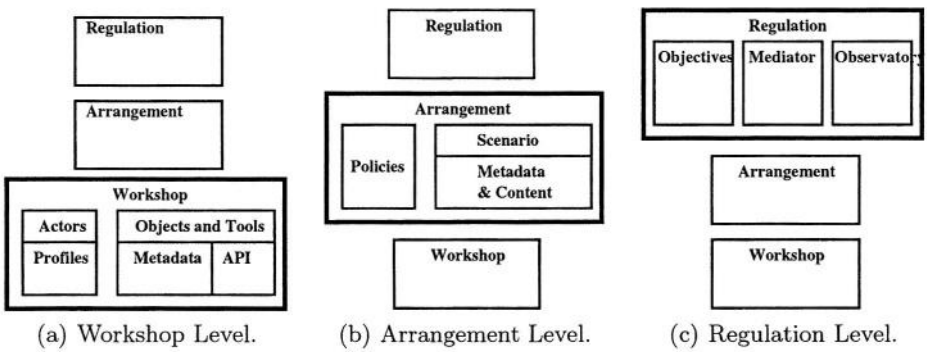

Fig. 2. Description of the three levels.

The configuration parameters are stored in the mediate level: the "arrangement" one (see fig.2(b)). This level has been defined to store what corresponds to the *scenarios* and the *thematic roles* in the participation model. We have identified different kinds of scenarios regarding the various dimensions of collaboration. They may concern the rights of actors within the arena, the way tools and objects will be seen by the actors (the interface dedicated to each category of actors), the way awareness will be managed, the way observation will proceed (what to observe on which component of the workshop) and organisational and social features. We will call a set of scenarios concerning one of those dimensions a *policy*. For each category of actor, i.e. for each thematic role defined, there might be some scenarios concerning that category in each policy.

Initial Configuration by a Pedagogical Scenario. To complete the architecture, we propose to use pedagogical scenarios as inputs to build the first configuration of the arena (see fig.4). In this way, the teachers will be able to adapt the arena to any pedagogical activity. Actually these pedagogical scenarios will describe all the elements necessary for the pedagogical activity to take

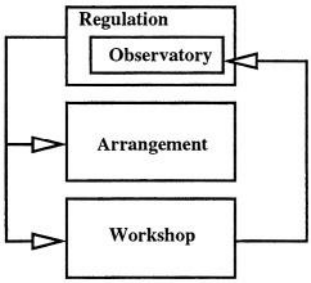

Fig. 3. Regulating the activity.

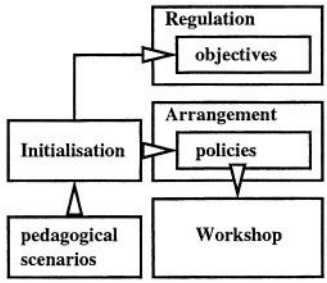

Fig. 4. Initial configuration of the arena by means of pedagogical scenarios.

place. This implies defining a rich and expressive formal language to describe those pedagogical scenarios which will furthermore have the feature of being implementable within a CSCL framework. For us, this means that formal language to be based on the concepts of the participation model.

4 Implementation Within the Electronic Schoolbag

4.1 Description of the Electronic Schoolbag Project

Begun in 1999, the "Cartable Electronique®"[3] project has combined the efforts of the University of Savoie "SysCom" laboratory and "TICE" team, the Savoie General Council "Savoie R&D" team[4] and the National Education Institution's rectoral and departmental services. They have worked jointly on the definition, design and creation of new educational services intended first of all for schoolchildren and university students, but also for all of the actors in the Education field (teachers, administrators, sociocultural partners, etc.) and families.

The ambition of this project is to contribute to the development of new educational practices introducing information and communication technologies in the educational area, from school to university. For the schoolchildren, one of the objectives is also to increase the links between the institution and the children's families, and between the institution and its traditional sociocultural partners. For the students, we are witnessing a mutation of education practices: students have quite different learning curricula, with a mixture of distance and traditional teaching. We have to take into account the different needs of each individual, their various backgrounds and the variety of courses they follow. Moreover, during the course of their scholarship, students belongs to different communities: institutional, cultural, sports, organizational, etc.

[3] "electronic schoolbag" in English.

[4] The "TICE" team and the Savoie General Council "Savoie R&D" one don't exist any more. Computer engineers involved in the project since its beginning, who were members of those structures, and the researcher who originated the project, Christian Martel, have recently set up an enterprise to continue the development of the project, in collaboration with the laboratory.

We have developed within this project a web-based system called "electronic schoolbag" [24] which can support collaborative activities as well as individual ones. It is a framework which gives access to many services and objects concerning basic group work functionalities and specific educational ones. It is intended to offer the learners a permanent access to the contents and the personal tools which are useful for them at home or at school.

This platform has been developed in accordance with the architecture and the underlying model described above, which are used as guides for its development. In what follows, we describe its current state, showing how the concepts presented in the architecture and the model have been implemented and thus putting emphasis on its tailorability and its flexibility.

The Electronic Schoolbag project is supported by the French National Education Ministry and its Technology Executive, the Savoie General Council, the University of Savoie and the Caisse des dépôts et consignations (State's Bank).

4.2 The Model and Architecture Within the Electronic Schoolbag

Two Kinds of Arenas: The Virtual "Schoolbag" and Group Workspaces. A private arena is offered to each learner: the virtual *schoolbag.* It groups personal tools and objects accessible by several protocols (WebDAV, HTTP, HTPPS, ftp) through a desk. By extension, this object is also offered to the parents, to the teachers and to the technical and administrative staff of the schools. It enables them to have a single access point to their private data or to the services useful for the exercise of their profession or their responsibility. Obtaining such a virtual schoolbag is subordinated to the membership of an educational community organized around a school. An LDAP directory is used as a frame of reference for the delivery of the access authorizations.

Figure 5 shows a screenshot of one schoolbag. It is currently viewed as a folder which contains objects and tools. Objects can vary from simple documents of whatever type you want (html, simple text, .pdf, .doc, ..) to structured ones such as photo album, encyclopedia, school report, homework notebook,….Tools are personalised services such as a webmail or an address book manager. One user has functionalities at his/her disposal to manage the contents of its schoolbag as he/she wants: adding objects or tools any time; copying/cutting/pasting those objects or tools; editing some of them; sending them to other users, even the structured objects (by means of a communication tool called a "dropbox" which saves the structure of the structured objects); organizing the contents by adding folders inside the schoolbag space;... These functionalities are for us the first degree of tailoring of the platform, at an individual level, as they allow users to be the masters of their private spaces and data.

As users belong to several institutional or interest groups within a school or a University (for example, one curriculum class, the group of the teachers, the teachers of History, the management staff,…) we have defined "workshops". They are arenas dedicated to group work support. They are described in the next section.

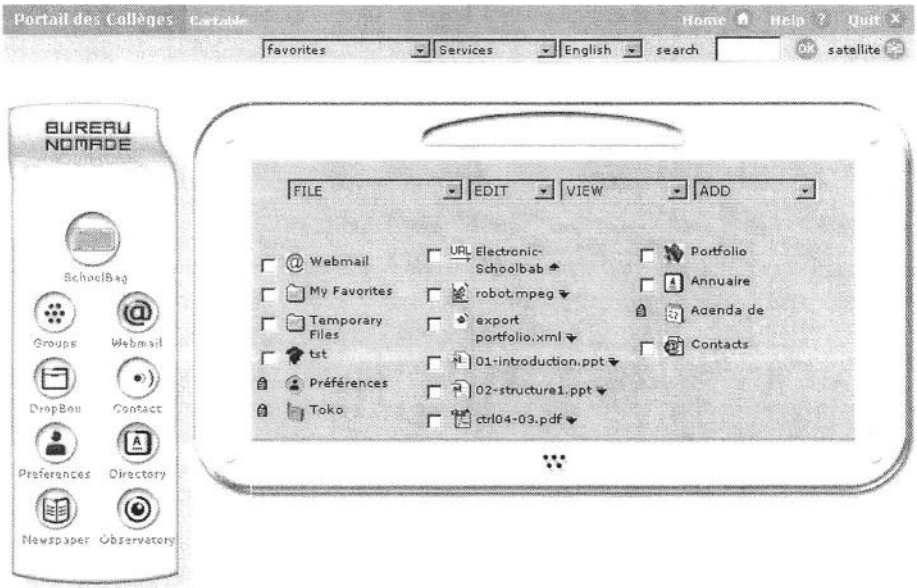

Fig. 5. A view of one virtual schoolbag.

The Workshop Level. The workshop level is reified in the electronic schoolbag platform by means of workshops. They may contain the same objects and tools as a schoolbag arena does plus specific ones concerning collaborative work (for example, wikis, chat, forum, ...). Users may create and manipulate those objects and tools within a workshop as they do in their own schoolbag. However the distribution of functionalities depends on the role attributed to one person (see the next section).

Workshops are created automatically when they correspond to institutional groups (they match the organisational structure of the school described in its information system) or they may be created by users themselves whenever they want, to whatever purpose. This is the case for example when a group of students is involved in project-based pedagogy: they create a workshop to support the activity they will carry on during the project. This can also be the case for people who share a hobby and want to share something (for example, exchange ideas, photos, ...) about this hobby. Users here have great freedom in creating groups and in arranging their contents. So a teacher can create groups and associated workshops to support collaborative learning activities and populate them with the tools and objects available in the platform. This is another way of tailoring the platform, at a collective level this time.

The Arrangement Level. The arrangement level is reified by the policies accessible via a menu in a workshop (see fig.6) describing the scenarios which will govern the entry into one workshop, the registration process, the distribution of roles and permissions and the display of the workshop itself.

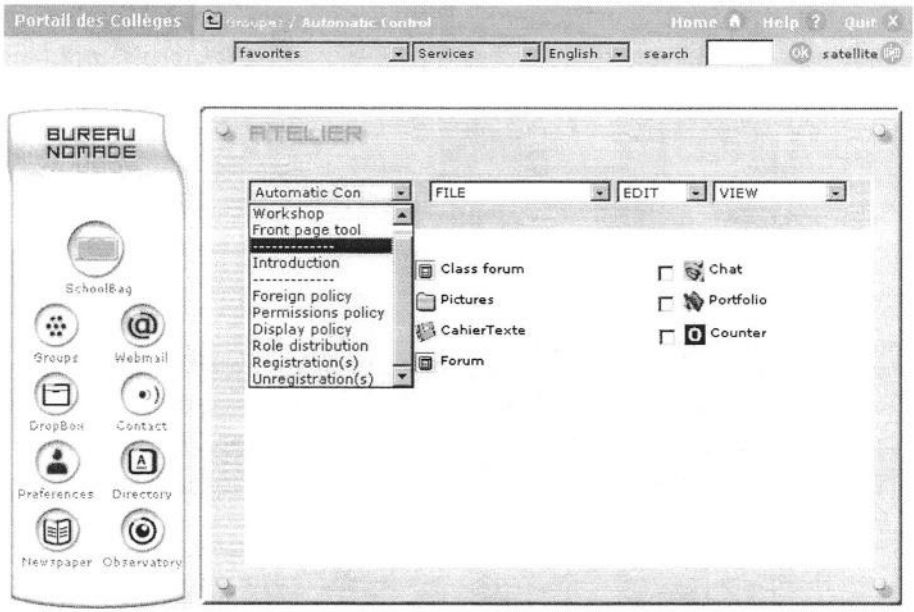

Fig. 6. A view of one workshop.

The registration process is dedicated to one actor having the specific role of "group manager". The group manager will be able to choose, among the people registered in the LDAP, who will be a member of the group. He will also be able to assign a role to each member. There are some predefined roles such as "group manager", "member", "teacher", "learner", ... Some have been defined to control the basic functions of group management; others are dedicated to pedagogical purposes. A "group manager" can create as many roles as he wants and distribute them to group members. For each role, he also defines the *permissions* associated with this role. Permissions describe what a person having a particular role is allowed to do or not in a workshop (see fig.7). They are attached to each tool and object present in the workshop, to the workshop itself and also to the folders within a workshop as a feature of their API. So the matrix you can see in figure 7 is not static but is dynamically built according to the configuration of the workshop. That's why a dynamic definition and distribution of roles is possible.

When a person is a member of one group, he/she can enter the workshop associated with the group whenever he/she wants and manipulate its contents according to his/her role in the group. For a non-member, entry is subject to the *foreign policy.* This policy indicates how the group, its contents and its members will be accessible or available for non-members. If the group is " Public" (as opposed to "Private"), information will be accessible by non-members; if it is "open" (as opposed to "Closed"), everybody will be allowed to subscribe to the group and will be free to leave the group, if it "Can be reached" (as opposed

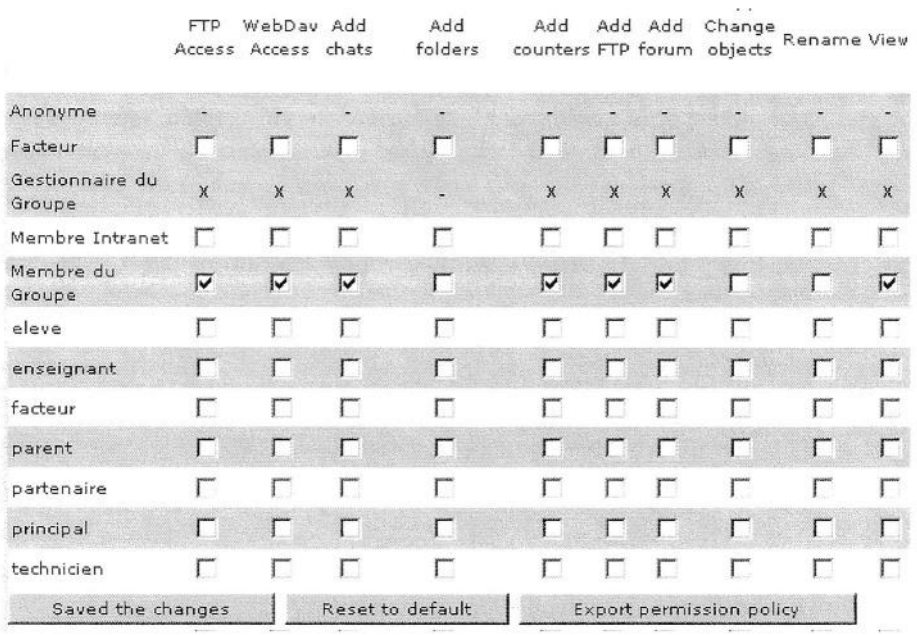

Fig. 7. The permission matrix.

to "cannot be reached"), the non-members will be allowed and able to send documents to the group.

Finally, the last remaining policy is the *display* one, which will allow the "group manager" to describe, for each existing role, a specific presentation of the workshop content. By default, a file system view such as the one presented in figure 6 will be used. An html file present in the workshop could be identified as being the entry page.

The policies are objects that can be manipulated by the members of the group like the others. This allows one to copy one policy from one workshop to another. It then becomes the actual policy in use, in the workshop in which it has been copied. As the policies will be one means of translating some aspects of pedagogical scenarios, this will be a means of reifying those scenarios and sharing them.

The Regulation Level. Regulation is handled within the electronic school-bag platform by modifying the policies, roles and permissions described at the arrangement level as well as by acting upon the tools and objects present in the workshop. Modifying the policies is done by the group manager who has at his/her disposal a "satellite" to observe what has happened in the workshop. This is a basic tool which has been developed to capture the basic events concerning actions which have been done in the workshop. These events are not semantically and pedagogically founded information and work should be done to compute them into such information, which would have sense for a teacher

managing a group activity. At the present time, this has not been done and we consider the "satellite" as a base on which something richer would be built during future work. Starting from the declaration of observables in an observation policy, the future satellite would be able to recover the corresponding events and to associate useful treatments with them for the production of the reports/ratios provided to the observer.

4.3 Advantages and Limits of the Current Implementation

At the University of Savoie, the platform is currently used every day by more than 15000 people[5] organized in 2300 working groups. Because of the technologies used for the development of the platform (Zope and Python) and the component-based approach adopted, new functionalities can be easily added. Because of the architecture and model presented above, we have a guide for the development and we can propose a more and more flexible and tailorable framework. Because of the ongoing experiments and actual use of the platform with a lot of users in a lot of diverse pedagogical situations, we have inputs to modify not only the platform itself but also the underlying model and architecture. This should lead to their constant evolution and improvement.

We here want to make a comparison between the implementation realised within the platform and the underlying model. The objective is to state explicitly what exists and what has not yet been implemented, in order to measure the completion of the implementation, and so the distance between the level of tailoring we want to reach within the model and the currently implemented one. We are thus going to see if the platform offers the functionnalities and mechanisms described in section 3 and if so, to what extent.

Static Configuration of Group Workspaces. He have seen in the previous section that anybody can create a workshop. The creation functionnality is thus provided, with no restriction. So does the insertion of tools and objects one, with no restriction for objects (every kind of files can be added in a workshop), with a restriction for tools: only predefined tools, which have been integrated into the platform as a Zope "product", may currently be added. Three levels of integration for existing web applications or components have been defined and corresponding integration patterns have been designed. This, plus the component-based approach, makes it easy to integrate existing web-software but the intervention of a computer engineer is still required. We are presently exploring the way of providing extensible workshops, with no restriction for tools.

Social Configuration of Group Workspaces. The social configuration of group workspaces mainly resides in the definition of roles, distribution to participants and definition of "scenarios" of whatever kind. In the platform, end-users can easily deal with roles and distribute them to who they want (restriction:

[5] The other 25000 users are people from secondary schools, who use a version of the platform dedicated to those schools: they have specific objects at their disposal, such as a homework book for example, that are not relevant in the University context.

they must have the role of "group manager", which is automatically attributed to the creator of a group).

What has been implemented for "scenarios" is quite far from the theory. We have chosen, for some kinds of scenarios and in a first approach, to consider a few predefined hard-coded policies. On the one side, this has the advantage of being easy to develop, easy to put at one user's disposal and easy to manipulate for end-users (they just have to select the suitable policy). It so offers a certain degree of tailorability. On the other, it is not possible for end-users to modify the existing policies or to create a new one. That's why we have continued the work on formalisation and implementation of scenarios begun in a previous work[23]. We have today at our disposal a formal description for scenarios (a grammar, an XML DTD and a UML formalisation) which has been improved for both social scenarios (concerning awareness or the group functionning, for example) and pedagogical scenarios (see section 4.3 below).

Dynamic Evolution of Group Activity. On the dynamic point of view, we have seen that the users can manipulate and modify dynamically all the elements implemented (objects, tools, adding members to one group or suppression of members, policies, roles, etc.) present in one workshop, according to the workshop policies, of course, with no restriction. The development has here reached the highest level of tailorability and has met the regulation requirement for the part that concerns the intervention in activity.

As regards observation, we have already mentioned above that work remains to be done to improve the satellite and mainly define what can relevant observables be, how they can be computed from basic events and how they can be presented to a group manager or a teacher.

Initial Configuration from Pedagogical Scenarios. The initial configuration from pedagogical scenarios is not achieved yet. Teachers, and other users, currently configure the workshops and policies at hand; there is also no way of enacting the current scenario and once again the teacher has to manage by himself.

From a theoretical point of view, we are currently working with the Arcade team from CLIPS-IMAG laboratory in Grenoble on the definition of a formal language to describe pedagogical scenarios. We are considering as references the IMS/LD metadata recommended to describe pedagogical scenarios, the approach of the TECFA laboratory within the European project SEED based on reusable educational blocks to build original scenarios[6] and the formal description of scenarios in the participation model. We are presently working on the correspondence between the concepts proposed by both projects to structure pedagogical scenarios and the concepts of the participation model, to see if this model would be a good one to describe not only scenarios representing social or organisational features but also pedagogical scenarios formally. This has led us to define a first version of an XML-based language taking into account the common concepts from IMS/LD, TECFASEED and the participation model,

[6] http://tecfaseed.unige.ch/door/

completed by the ones useful for operationalization. We are currently improving this language on several scenarios in order to verify this result. We have also planned to prototype a engine for the enactment of one particular pedagogical scenario described with this language. The implementation should be finished in September and we would then be able to begin experiments and improvements.

5 Conclusion

We have presented in this paper a model of a collaborative activity and the architecture we proposed to structure group spaces in groupware according to that model. This architecture is used as a guide to the development of the electronic schoolbag platform, a tailorable groupware we have developed to provide support to both collaborative and individual pedagogical activities. This groupware is today commonly used at the University of Savoie and several secondary french schools by more than 40000 persons. It progressively evolves by taking into account the reactions and comments of these users (it is adapted to their needs and uses) and the evolution of our researches. It is so a real, great and very interesting experimentation ground for us. We are currently working on the modelling of pedagogical scenarios which should be used to initialize automatically the configuration of the group work spaces. We are also exploring different ways of enacting these scenarios by considering component based and workflow based technologies.

References

1. Lave, J., Wenger, E.: Situated Learning: Legitimate Peripheral Participation. Cambridge: Cambridge University Press. (1991)
2. Baker, M., Lund, K.: Promoting reflective interactions in a learning computer-supported collaborative learning environment. Journal of Computer Assisted Learning **13** (1997) 175–193
3. Brown, A., Campione, J.: Communities of learning and thinking, or a context by any other name. In Kuhn, D., ed.: Contributions to human development. Volume 21. (1990) 108–126
4. Brown, A., Palincsar, A.: Guided, cooperative learning and individual knowledge acquisition. In Resnick, L., ed.: Knowing, Learning and Instruction: Essays in Honor of Robert Glaser. Hillsdale, NJ: Lawrence Erlbaum Associates (1989) 393–451
5. Braten, I.: Vygotsky as precursor to metacognitive theory: I. the concept of metacognition and its roots. Scandinavian Journal of Educational Research **35** (1991) 179–192
6. Braten, I.: Vygotsky as precursor to metacognitive theory: Ii. vygotsky as metacognitivist. Scandinavian Journal of Educational Research **35** (1991) 305–320
7. Braten, I.: Vygotsky as precursor to metacognitive theory: Iii. recent metacognitive research within a vygotskian framework. Scandinavian Journal of Educational Research **36** (1992) 3–19

8. Dillenbourg, P.: What do you mean by collaborative learning? In Dillenbourg, P., ed.: Collaborative learning: Cognitive and computational approaches. Oxford: Elsevier (1999) 1–19
9. Vygotsky, L.: Mind in society: the development of higher psychological processes. Cambridge, MA: Harvard University Press (1978)
10. Salomon, G., Perkins, D.: Rocky roads to transfer: Rethinking mechanisms of a neglected phenomenon. Educational Psychologist **24** (1989) 113–142
11. Zimmerman, B.J.: Self-regulation involves more than metacognition: A social cognitive perspective. Educational Psychologist **30** (1995) 217–229
12. Berardi-Coletta, B., Buyer, L., Dominowski, R., Rellinger, E.: Metacognition and problem-solving: a process oriented approach. Journal of Experimental Psychology **21** (1995) 205–223
13. Nelson, T.O.: Judgements of learning and the allocation of study time. Journal of Experimental Psychology: General **122** (1993) 269–273
14. Cohen, A., Scardamalia, M.: Discourse about ideas: Monitoring and regulation in face-to-face and computer-mediated environments. Interaction Learning Environments **6** (1998) 179–192
15. Fitzpatrick, G., Tolone, W.J., Kaplan, S.M.: Locales and distributed social worlds. In: Proceedings of ECSCW '95, Stockholm (1995) 1–16
16. Tolone, W.J.: Introspect: a Meta-level Specification Framework for Dynamic, Evolvable Collaboration Support. PhD thesis, University of Illinois at Urbana-Champaign (1996)
17. Dourish, P.: Using metalevel techniques in flexible toolkit for cscw applications. ACM Transaction on Computer-Human Interaction **5** (1998) 109–155
18. Dourish, P., Button, G.: On "technomethodology": Foundational relationship between ethnomethodology and system design. Human Computer Interaction **13** (1998)
19. Bourguin, G., Derycke, A.: Integrating the CSCL activities into virtual campuses: Foundations of a new infrastructure for distributed collective activities. In: Proceeding of EuroCSCL'2001, Maastricht, The Netherlands (2001)
20. Bourguin, G., Derycke, A., Tarby, J.: Beyond the interface: Co-evolution inside interactive systems. a proposal founded on activity theory. In Blandford, V., Gray, eds.: Proceedings of Human Computer Interaction 2001 (HCI'2001). Number 15 in People and Computer, Springer Verlag (2001) 297–310
21. Stahl, G.: Groupware goes to school. In: Proceedings of CRIWG'2002, 6th international workshop on groupware. Number 2440 in LNCS, La Serena, Chili, Springer Verlag (2002) 7–24
22. Martel, C.: La modélisation des activités conjointes. Rôles, places et positions des participants. PhD thesis, Université de Savoie, France (1998)
23. Ferraris, C., Martel, C.: Regulation in groupware: the example of a collaborative drawing tool for young children. In: Proceedings of CRIWG'2000, 6th international workshop on groupware, Madeira, Portugal, IEEE (2000) 119–127
24. Martel, C., Vignollet, L.: An educational web environment based on the metaphor of the electronic schoolbag. In: Proceedings of ARIADNE workshop 2002, Lyon, France (2002)

This page intentionally left blank

Representing Context for an Adaptative Awareness Mechanism

Manuele Kirsch-Pinheiro, Jérôme Gensel, and Hervé Martin

Laboratoire LSR – IMAG
BP 72 – 38402 Saint Martin d'Hères Cedex, France
{Manuele.Kirsch-Pinheiro,Jerome.Gensel,Herve.Martin}@imag.fr

Abstract. The application of mobile computing technologies to Groupware Systems has enforced the necessity of adapting the content of information by considering the user's physical and organizational contexts. In general, context-aware computing is based on the handling of features such as location and device characteristics. We propose to describe also the user's organizational context for awareness purposes. Our objective is to permit Groupware Systems to better select the information and then to provide mobile users with some adapted awareness information. This paper presents a representation of this notion of context to be used by awareness mechanisms embedded in Groupware Systems. Then, we show how this representation is exploited for filtering the content of information inside the awareness mechanism.

Keywords: adaptability, awareness, mobile computing, cooperative work.

1 Introduction

In recent years, Groupware Systems have been using the Web to propose a world wide access to their users. With the massive introduction of web-enable mobile devices, such as laptops, PDAs and cellular phones, these users can access the system virtually everywhere. In fact, nowadays, workers may interact with their colleagues and accomplish some tasks (such as to compose messages or to exchange meeting notes), even when they are not at their office, and through a large variety of devices.

However, the use of this kind of mobile device introduces several technical challenges on system design and development, mainly due to the heterogeneity and the physical constraints (limited display size, power and memory capacity...) of these devices. These challenges make adaptation a necessary technique when building mobile systems [9]. In fact, a mobile user, whilst moving or alternating between different devices, often changes the context in which she/he accesses the system. Therefore, the ability of detecting the context of use seems to be particularly relevant for mobile computing systems, which may be used in different locations, by different users (who may access them from different devices), and/or for different purposes [3]. This detection of the context characterizes what is called context-aware systems. In fact, one of the core premises of context-aware computing is that the computing device should be aware of the user's circumstances and should be able to interpret any interaction in a manner appropriate to these circumstances [16].

Considering the notion of context, it is worth noting that there is no unique definition for this concept (see [2] as an illustration). For instance, according to Kirsh [11], "context is a highly structured amalgam of informational, physical, and conceptual

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 339–348, 2004.

resources that go beyond the simple facts of who or what is where and when to include the state of digital resources, people's concepts and mental state, task state, social relations, and the local work culture, to name a few ingredients". This notion of context can be related to the notion of awareness on Groupware Systems, which refers to the knowledge a user has and her/his understanding about the group itself and her/his colleagues' activities, providing a shared context for individual activities in the group (see, for example, [6] and [14]).

Nevertheless, when considering context-aware systems, we notice that the notion of context usually adopted by those systems is limited to some physical aspects, such as the user's location or device (see [3] as an illustration). There are only a few systems that associate the notion of awareness on Groupware Systems to this idea of context-aware computing (for instance [8]). However, this association appears evident to us, once a mobile user is also involved in some cooperative process. And, as any other user in a cooperative environment, a mobile user needs to be aware of what is going on inside the group in order to build a sense of community [18]. This means that Groupware Systems, in order to cater for these mobile users, should provide them with an awareness support adapted to their situation. Consequently, we should consider the activities and the status of the group as parts of the notion of context handled by the system.

In this paper, we explore the hypothesis assuming that the awareness mechanism should exploit the notion of context in order to adapt the information delivered to the user. We propose a context based awareness mechanism which filters the information delivered to the user according a context description. This context description takes into account the concepts related to the notion of awareness (group and role definition, activities and work process, etc.). We use an object-oriented knowledge representation, where these concepts are represented as classes and associations. This paper represents the first part of a work in progress. Such work considers, when defining this context representation, an awareness mechanism embedded on Web-based Groupware Systems, which is accessed through mobile devices. We focus on Groupware Systems that support asynchronous work, such as systems managing group calendar, messages and shared repository (a shared workspace). We assume those systems as composed by many components (such as components for access control policy, for communication facilities, etc.) which are connected and communicate with each other (and eventually with other instances on remote sites). Hence, we assume awareness mechanism as one of such components, and we propose a filtering process that uses the context representation mentioned above to better select the awareness information delivered to mobile users.

This paper is organized as following: first, we present some works related to adaptation for mobile devices (Section 2). Second, we discuss the notion of awareness, our main focus area (Section 3). Then, we present the context description used to represent the extended notion of context (Section 4), and the filtering mechanism based on this description (Section 5), before we conclude (Section 6).

2 Adaptation and Context

The development of software applications for mobile environments involves several technical challenges, which makes adaptation a necessity for the usability of such

systems. Many researches consider this need of adaptation. In their majority, these works deal with the adaptation of multimedia and web-based information content. They usually take into account the technical capabilities of the client device, and try to adapt the content by transforming the original content in such a way that it can be handled by the device (see, for example, [20] and [13]).

Additionally, some works adapt the content by selecting or filtering the content delivered to the user, according to the physical context of the client device. In the latter, the adopted notion of context includes some aspects such as location, time and, of course, the device itself. Some examples of this approach include [3], [16], or [15].

We adopt this approach of selecting the content delivered to the user considering the context where this user finds herself/himself. Moreover, we consider that the organizational context, as well as the physical context, should be taken into account to evaluate what is relevant to a user, and thus, to select the available information for her/him. In fact, according Dourish [5], "context – the organizational and the cultural context, as much as the physical context, plays a critical role in shaping action, and also in providing people with the means to interpret and understand action".

Indeed, this organizational context is important to determine what information is relevant to users. These users are engaged in a cooperative process. Because of this process, such users are particularly interested in information related to their belonging group. More precisely, users, in most circumstances, are interested only in events related to their work context, i.e. events that can lead them to better decisions and/or increase their capacity to decide [4]. In our approach, we represent and explore this context in order to better cater for the user the awareness information to be delivered.

3 Awareness and Adaptation

It is worth noting that the term *awareness* represents a large concept, which can be used in very different situations, as shown by Liechti [14] and Schmidt [21]. The term awareness refers to actors' taking heed of the context of their joint effort, to a person being or becoming aware of something [21]. However, this definition is too vast and we adopt a more concise defined by Dourish [6]: "an understanding of the activities of others, which provides a context for your own activity. This context is used to ensure that individual contributions are relevant to the group's activity as a whole and to evaluate individual actions with respect to the group goals and progress".

There is, in the CSCW community, a consensus about the importance of the awareness support for cooperative work [21][7]. Awareness represents the knowledge about the group, its activities, status and evolution. This knowledge refers to the organizational context where the cooperative work takes place. Thanks to the awareness support, users may coordinate and evaluate their own contributions considering the group evolution. Indeed, awareness support can be used as an implicit coordination mechanism, since whether the members of a team are kept aware of their project status and activities, then they will be able to communicate with each other and coordinate themselves [19].

However, providing an awareness support presents some risks. Espinosa et al. [7], for instance, have obtained encouraging evidences about the benefits of awareness tool use, but they stress the fact that the availability of such tools can turn out to a distraction when not properly used. Awareness tools should fit with the tasks performed by the users.

For users who access Groupware Systems through mobile devices, such as PDAs or cellular phones, the delivered information should also match the constraints of the device as well as the mobile situation of such users. Typically, people using such mobile devices are interested only in information that can help them in their current context. In order to adapt the awareness information to such users, the notion of context should be represented to be exploited for adaptation purposes. In the next section, we present a representation of this.

4 Context Representation

In order to adapt the delivered awareness content to the user's context at a given moment, we have to represent this context in such way that it could be exploited by the awareness mechanism. In order to be useful, we believe that this representation should be limited to relevant aspects of the notion of context. As stated before, we address an awareness mechanism embedded on Groupware System supporting asynchronous work. Thus, the representation of the context described below focus only on concepts relevant to users who access the system through mobile devices.

There are, in the CSCW literature, several propositions of user's context representation. For instance, Leiva-Lobos and Covarrubias [12] propose that the context where cooperating users are situated is tripartite: spatial, temporal and cultural. The spatial context addresses artifacts populating physical or electronic space, while the temporal context refers to the history of performed cooperative processes and to the expected future one. The cultural context gathers users' shared view and practices (i.e., the community practices). Similarly, Allarcón and Fuller [1], define, as principal entities for the work context, the content (tools, shared objects, etc.), the process (activities and their calendar) and the users themselves. In addition, these authors emphasize the importance of the user's electronic location and integrate this concept into the user's context.

Considering these works, we identify five viewpoints containing main entities for a context representation: space, tool, time, community and process. The *space* viewpoint refers to the concept of physical *location,* while the *tool* viewpoint refers to the concepts of physical *device* and *application.* The *time* viewpoint refers to the group *calendar* idea. The *community* viewpoint refers to the composition of the community, including the concepts of *group, roles* and *user.* Finally, the *process* viewpoint refers to the *process* (workflow) performed by the group, including the concepts of *activities* (tasks) and *shared objects* (objects handled by the group).

We consider these concepts as the most relevant ones when defining a user's context for a cooperative mobile environment. Using these concepts as a starting point, we define our representation of the notion of context. We represent this notion through an object-oriented representation, using the UML notation. In this representation, the concepts above (user, group, role, location, etc.) become classes (member, group, role...) and the relationships among them, associations.

This representation of context relies, then, on the concepts identified by the five viewpoints above. We consider these concepts as the basic entities of the context description, which we see as a composition of such entities. Thus, our representation starts by the definition of a *context description* class, which is composed of a set of basic elements and is defined for a user that is currently accessing the Groupware System (see the UML class diagram in Fig. 1).

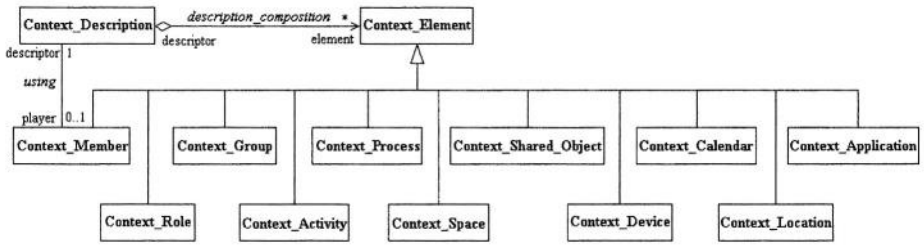

Fig. 1. The context description and the basic elements of the context representation. The prefix "Context" is used in this paper to distinguish the elements of the context representation from those of the awareness mechanism.

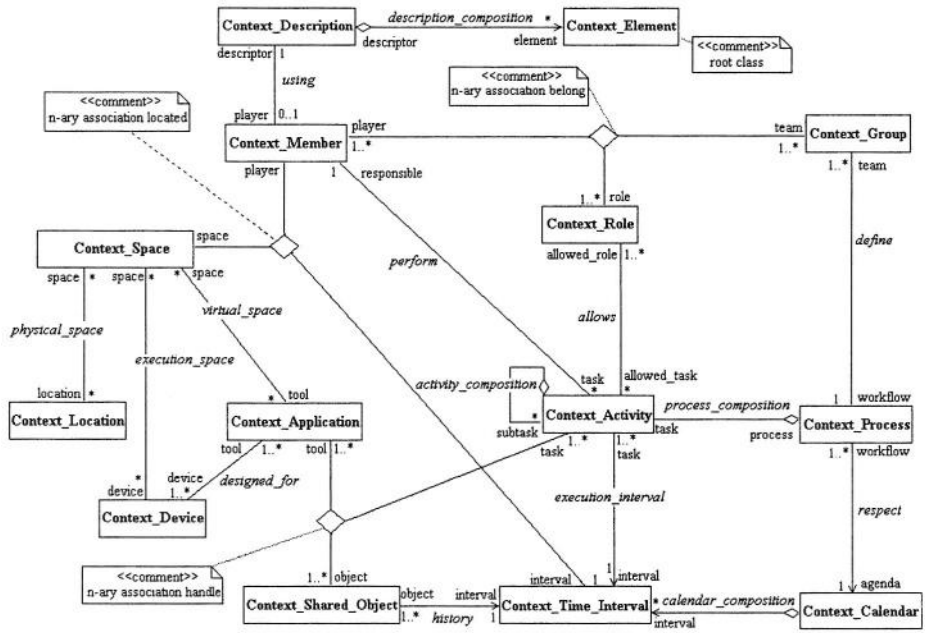

Fig. 2. The context representation with the relationships among all represented concepts.

These basic elements are related to each other, defining relationships between the corresponding concepts, as represented in the Fig. 2. Thus, we define relationships such as the user (called here 'member') that belongs to the group through the roles she/he plays in this group, represented in the model by the ternary association *'belong'* among the classes *'member', 'role', 'group'* (see Fig. 2). We also consider that each group *defines* a *process,* which should *respect* a given *calendar* and is *composed* by a set of *activities* (or tasks, also composed by subtasks). The *roles allow* the execution of an activity, which is effectively *performed* by a member. Each activity *handles* a set of *shared objects* through a set of applications, which are *designed* for specific devices. This *member* is located, during a certain *time interval,* into a certain *space.* This space is composed by a physical space (the member's physical *location*), by a virtual space, comporting the *application* that the member is accessing, and by an

execution space, including the *device* used by the member. The Fig. 2 presents the complete context representation proposed.

These elements, combined through the context description, are able to describe the context of a user that is accessing the Groupware System through a mobile device. These elements, together with the context description, are instances of a knowledge base, which can be exploited by the awareness mechanism. It exploits this context representation through the context description.

As an illustration, let's us introduce a simple scenario of a group member (the team coordinator) who is participating to a business meeting in the company central office. During a break, she/he may access the Groupware System, using her/his PDA, in order to consult the last information about a report that her/his group is writing. For this scenario, the context description will contain the following basic objects: a "member" object describing the mobile user (her/his name, email...); a "role" object, describing the coordination role (rights, etc.); an "activity" object describing the report writing (status, components, deadline...); a "location" object pointing out to the central office; a "device" object defining her/his PDA; an "application" object indicating what application she/he is using; and a "process" object describing the group's main process. Finally, all these objects will compose the context description object associated with this mobile user. The Fig. 3 shows the relationships among these basic objects.

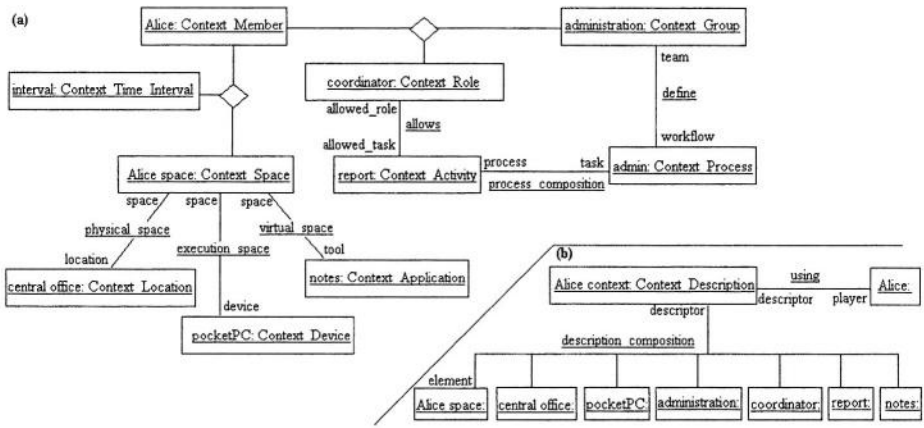

Fig. 3. A scenario of context description. In (a), the relationships among the context elements. In (b), the context description object composed by these context elements.

In addition, since this description is defined as a composition of basic elements, the mechanism can handle partial representations of the context. For instance, a *context description* object may refer, through this composition relationship, only to objects describing the member, her/his roles and the activities she/he is performing, and omit some information about the application or location, or it may include only objects referring to the notion of space (location, device, and application), and ignore all information about the activities and the group process. This omission means that the system does not have enough knowledge to represent these elements, and then it can assume nothing about them. This feature (the omission of some context elements in the context description) is very interesting for awareness mechanism, since often the

system is not able to determine all elements of this context representation. Back to our scenario, if the system cannot determine the user's location and device, context description will be defined without such objects ("central office" and "pocketPC").

We implemented this representation using the AROM system [17] that is an object-based knowledge representation system, which adopts classes/objects and associations/tuples as main entities. This representation allows queries such as if a user is currently using a given device or stands in a specific location. This corresponds to query whether the objects representing such device or location belong to the current user's context description (for instance, if the object "central office" belongs to our mobile user's context description). We now describe how an awareness mechanism can exploit this context representation.

5 Filtering Awareness Information According Context

As we stated before, we consider the awareness mechanism as a component of a web-based Groupware System. This awareness mechanism should be able to analyze the users' activities, as well as those performed by others components, in order to collect information that could be relevant to the performance of the group members. However, the amount of information collected by this mechanism can be very important. Hence, we consider that the awareness information delivered to a group member should be subject of a carefully selection, in order to avoid problems such as an information overload and to better cater for the user's needs.

In this work, we intent to exploit the context representation presented above to perform this selection. We consider an event-based awareness mechanism and we assume that all information that can be delivered to a group member is carried by events. Events are defined by the Groupware developer and each event selects useful information about a specific subject. In order to select the events that should be delivered, the awareness mechanism defines the concept of *"general profile"*. This concept represents the preferences and the constraints that the system should apply for a given element (group member, role, device...). For group members, this concept is specialized on *"preferences",* describing the preferences of the user concerning the awareness information delivery, and for devices, it is specialized on *"characteristics",* describing the capabilities of the referred device. These profiles may define what types of events and information should be delivered, as well as its quantity (maximum number of events or Kbytes supported). For devices, the "characteristics" profiles can take the form of a CC/PP description, whereas the "preferences" profiles may indicate a priority order for the events, the time interval that is suitable for the user, and other conditions related to the context description (for instance, if the current device accepts a given media type, or if a given activity has been concluded).

In order to perform the context based filtering process, we associate those events and general profiles with context description objects (see Fig. 4). In fact, these events should be produced in a certain context, which can be represented by a *context description* object. Further, once a user accesses the system, she/he is doing so through a specific context, which is identified by the Groupware System and represented by a context description. We also associate with the profiles, at least, one context description object describing the circumstances where it can be applied. The Fig. 4 presents the associations among events, profiles and the classes defined by the context representation (Section 4).

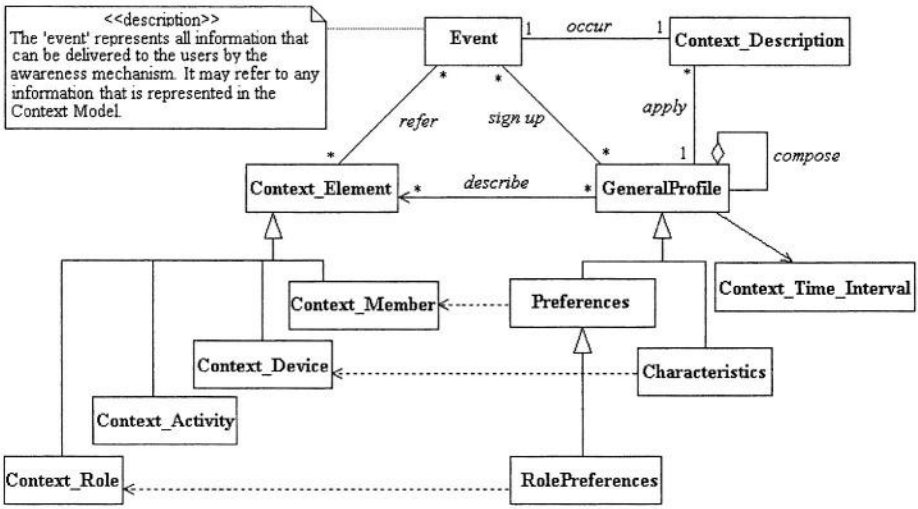

Fig. 4. The application of the context representation on the awareness mechanism.

Then, the proposed filtering process uses these context description objects associated with the group member and with the general profiles to perform the selection of the suitable events for the current context. This filtering process is performed in two steps. First, the awareness mechanism selects the profiles (preferences or characteristics) that are applicable to current user's context description. In fact, we consider that each group member can define for herself/himself a set of profiles and the circumstances, through a context description object, in which each profile can be applied. The selection is performed by comparing the content of the context description objects of both, user and profile: if the context description object of the profile has the same content or is a subset of the current user's context description object, then this profile can be applied. For example; a preference profile for which the context description object is composed by the objects "coordinator" (for the *role*) and "report" (for *activity*) will be selected in the previous scenario (cf. Section 4). In addition, individual elements of the user's context may have its own profiles (e.g. a device which has its own *"characteristics"* profile).

Once all applicable profiles are selected, awareness mechanism can apply them over the available set of events, performing the filtering process. We suggest a gradual application, first applying the selected preferences of the user and then applying the other selected profiles. This can be made respecting a selection order (one profile at a time) or performing first a merge of the content of each selected profiles. This latter is especially interesting for "preferences" objects, which union can form a complete set of the group member's preferences. It is worth noting that, in this case, the awareness mechanism should be able to handle eventual incompatibilities among the selected profiles

The result of such filtering process should be a limited set of events that will be delivered by the Groupware System. However, our approach presents some limitations. The first one relies on the profiles definition that may be a hard and boring task. In addition, the definition of profiles with strict sets of application circumstances (profiles with context description composed by many context objects) may lead to the

selection of no profile at all. If the context description object associated with the profile is larger than the one related to the user, it will never be a subset of the latter and such profile will never be applied. This will block the filtering process, leading to the presentation of all the available events.

Other limitation of our approach concerns, similarly to other context-aware systems, the context detection as all filtering process depends on the user's context description object. Hence, the definition of this object by the system (the context detection process) is very critical.

6 Conclusion

This paper presents a proposition of context base filtering process for awareness mechanisms, using a context representation that uses an object-based knowledge model. This context representation represents the main contribution of this paper. It was successfully implemented on a knowledge base, using the AROM system, and we are now implementing the filtering process, using a framework for awareness support called BW [10]. We expect to perform practical tests using a test application, a cooperative game, specially designed for this purpose. In this game, small teams will use PDAs and laptops to look for (and describe) targets (sights and objects) geographically distributed. Users will then use the system mainly to communicate and build the descriptions of the targets, and the awareness mechanism to coordinate their actions. Through these tests, we expect to evaluate the effective impact of our proposition in a context based awareness mechanism, and the user's acceptation, specially concerning the limitations related to the profiles (cf. Section 5).

Acknowledgements

This work received grants from CAPES-Brazil (BEX 2296/02-0). Authors would like to thanks the referees by their valorous comments.

References

1. Alarcón, R., Fuller, D.: Intelligent awareness in support of collaborative virtual work groups. In Haake, J.M., Pino, J.A. (eds.), CRIWG 2002, LNCS 2440. Springer-Verlag (2002) 168-188
2. Brézillon, P.: Modeling and using context: past, present and future. Rapport de Recherche LIP6 2002/010 (2002) http://www.lip6.fr/reports/lip6.2002.010.html (Jan., 2004, in French)
3. Burrell, J., Gray, G.K., Kubo, K., Farina, N.: Context-aware computing: a text case. In: Borriello, G., Holmquist, L.E. (eds.), Ubicomp 2002, LNCS 2498. Springer-Verlag (2002) 1-15
4. David, J.M.N., Borges, M.R.S.: Improving the selectivity of awareness information in groupware applications. In: International Conference on Computer Supported Cooperative Work in Design (CSCWD'2001). IEEE Computer Society (2001) 41-46
5. Dourish, P.: Seeking a foundation for context-aware computing. Human Computer Interaction, vol. 13, n° 2-4. Hillsdale (2001) 229-241

6. Dourish, P., Bellotti, V.: Awareness and Coordination in Shared Workspaces. Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW'92). ACM Press (1992) 107-114
7. Espinosa, A., Cadiz, J., Rico-Gutierrez, L., Kraut, R., Scherlis, W., Lautenbacher, G.: Coming to the wrong decision quickly: why awareness tools must be matched with appropriate tasks. CHI Letters, CHI 2000, vol. 2, n° 1. ACM Press (2000) 392-399
8. Hibino, S., Mockus, A.: handiMessenger: awareness-enhanced universal communication for mobile users. In: Paternò, F. (ed.), Mobile HCI 2002, LNCS 2411. Springer-Verlag (2002) 170-183
9. Jing, J., Helal, A., Elmagarmid, A.: Client-server computing in mobile environment. ACM Computing Surveys, vol. 31, n° 2. ACM Press (1999) 117-157
10. Kirsch-Pinheiro, M., de Lima, J.V., Borges, M.R.S.: A Framework for Awareness support in Groupware Systems. Computers in Industry, vol. 52, n° 3, Elsevier (2003) 47-57
11. Kirsh, D.: The context of work. Human Computer Interaction, vol. 13, n° 2-4. Hillsdale (2001) 305-322
12. Leiva-Lobos, E.P., Covarrubias, E.: The 3-ontology: a framework to place cooperative awareness. In Haake, J.M., Pino, J.A. (eds.), CRIWG 2002, LNCS 2440. Springer-Verlag, (2002) 189-199
13. Lemlouma, T.; Layaïda, N.: Context-aware adaptation for mobile devices. In: IEEE International Conference on Mobile Data Management (2004)
14. Liechti, O.: Awareness and the WWW: an overview. In: ACM Conference on Computer-Supported Cooperative Work (CSCW'00) - Workshop on Awareness and the WWW. ACM Press (2000) http://www2.mic.atr.co.jp/dept2/awareness/fProgram.html (Dec., 2002)
15. Muñoz, M.A., Gonzalez, V.M., Rodriguez, M., and Favela, J.: Supporting context-aware collaboration in a hospital: an ethnographic informed design. In: Favela, J., Decouchant, D. (eds.), CRIWG 2003, LNCS 2806. Springer-Verlag (2003) 330-344
16. O'Hare, G., O'Grady, M.: Addressing mobile HCI needs through agents. In: Paternò, F. (ed.), Mobile HCI 2002, LNCS 2411. Springer-Verlag. (2002) 311-314
17. Page, M., Gensel, J., Capponi, C., Bruley, C., Genoud, P., Ziébelin, D., Bardou, D., Dupierris, V.: A New Approach in Object-Based Knowledge Representation: the AROM System. In: 14th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, (IEA/AIE2001), LNAI 2070, Springer-Verlag (2001) 113-118
18. Perry, M., O'Hara, K., Sellen, A., Brown, B., Harper, R.: Dealing with mobility: understanding access anytime, anywhere. ACM Transactions on Computer-Human Interaction, vol. 8, n.4. ACM Press
19. Projet ECOO: Projet ECOO: environnements et coopération. Rapport d'activité, INRIA (2001) http://www.inria.fr/rapportsactivite/RA2001/ecoo/ecoo.pdf (Nov. 2002, in French)
20. Schilit, B.N., Trevor, J., Hilbert, D.M. and Koh, T.K.: Web interaction using very small Internet devices. Computer, vol. 35, n° 10. IEEE Computer Society (2002) 37-45
21. Schmidt, K.: The problem with 'awareness': introductory remarks on 'Awareness in CSCW'. Computer Supported Cooperative Work, vol. 11, n° 3-4. Kluwer Academic Publishers (2002)

Opportunistic Interaction in P2P Ubiquitous Environments

Rolando Menchaca-Mendez[1], E. Gutierrez-Arias[1], and Jesus Favela[2]

[1] Center for Computing Research, National Polytechnic Institute, Mexico City, Mexico
{rmen,egutierrez}@cic.ipn.mx
[2] Computer Department, CICESE, Ensenada, B.C., Mexico
favela@cicese.mx

Abstract. In this paper we present the design and implementation of an ubiquitous system that supports the opportunistic collaborative edition of shared documents. Our system is based on the instant messaging metaphor in the sense that it uses concepts and interfaces similar to those used in instant messaging systems. We employ the concept of group to define a cluster of users that work on a particular shared document and use awareness to convey the state of users with respect to the shared documents (editing, reading, not using it). The system is implemented using a peer-to-peer (P2P) architecture and can be accessed by means of mobile devices such as handheld computers or cellular phones as well as from desktop computers. The P2P architecture provides the system with useful properties such as fault tolerance, the possibility of using public key infrastructures to implement secure transactions, scalability, a P2P repository to store shared documents and a distributed awareness service.

1 Introduction

Most documents are created and developed in collaboration [1]. The persons involved in the collaborative authoring of those documents have, at the same time, many other activities. So, it is difficult to arrange real collaborative-editing sessions where authors interchange their ideas about key parts of the document, instead of just reading the changes made or proposed by the co-authors. This situation could lead to a problem that we have called "slow-converging iterations" where the critical parts of the document (those which need the agreement and contribution of many of the authors) are changed many times until the document satisfies all authors involved in that section. This situation could significantly augment the time needed to complete the document.

As a step towards addressing this problem, we propose using the instant messaging metaphor to keep other authors informed if any of them is currently working with a shared document. In this way, authors are able to opportunistically arrange sessions of collaborative edition, particularly useful when a critical part of the document is being edited. This schema has the advantage that authors have the chance to discuss critical changes to the documents while they are in the process of doing them.

This approach is similar to the one proposed in [2] where they argue that it is important to provide support not only for collaborative activities inside collaboration spaces, but also for those that might occur outside of it and that might provide important support for collaboration. They introduce two concepts: *Actual and Potential*

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 349–362, 2004.

Collaboration Spaces. An Actual Collaboration Space is where collaboration activities take place. A Potential Collaboration Space represents an additional space where the possibility of collaboration is identified and from which an initial interaction is established. We implement the ECO Collaborative Editor that can be classified as an actual collaboration space and the ECO Instant Messenger that can be classified as a potential collaboration space because it fulfills the set of requirements identified in [2]:

- In ECO, a user is aware of potential collaboration, without having to leave the current mode of work. In this case we use the instant messenger approach that in spite of not being very intrusive lets users to be aware of the state of other users regarding relevant shared documents.
- Indicate to others, directly or indirectly, the interest or availability (or the lack of it) to get into collaboration. In particular, our system indicates if any other user is currently working on a shared document.
- Be capable of effortlessly launching an initial interaction to negotiate the possibility of collaboration, or to get into collaboration, when the potential for this is discovered. In our case, when a user identifies that another user is editing a relevant document (for example a document in which she has been working), she just needs to double click into the document icon to ask for a collaborative edition session. In this case, as we will see in later sections, the persons involved in this interaction are able to negotiate if they engage or not into the collaborative session.
- Be capable of ignoring any sign of potential collaboration, including direct requests made by other users, effortlessly and without being considered rude. As is in the case of instant messengers, a user can ignore in any time a request for collaboration. This is important for us because, one of our main goals was to design a tool as less intrusive as possible.

Among the advantages of potential collaboration spaces we can mention: They provide support to identify the possibility of collaboration and to establish an initial interaction to collaborate, while working individually. They provide mechanisms for seamless transitions between individual and collaborative modes of work.

On the other hand, in order to maximize the possibility that an author interested in a particular document become aware of changes being made to it, we implement access to our system from a variety of devices, from cellular phones to desktop computers. In the same way, we developed a set of distributed services to support scenarios than can be totally implemented by mobile devices. Examples of these services are: The ECO Distributed Awareness (ECO-DA) Service and the ECO File Repository (ECO-FR).

Shared files in ECO can be accessed in a ubiquitous manner by means of different kinds of devices such as smart cellular phones or PDAs. However, cellular phones and PDAs don't have enough memory to store large files and their network access is usually intermittent. Another key characteristic of this kind of devices is their mobility. A PDA could change its network identifier as it goes into different NAT networks or as it engages itself in different ad hoc networks. All these characteristics impose restrictions to the underlying distributed infrastructure as well as to the behavior of the system. To cope with these restrictions we use Jxta pipes [3] as the main method to send and receive all kind of information in Eco. We chose Jxta Pipes because they provide the illusion of virtual in and out mailboxes that are not physically bound to a specific transport location. Pipe ends are dynamically bound to a peer endpoint at

runtime. Using the pipe abstraction, our applications and services can transparently failover from one physical peer endpoint to another in order to mask the mobility of a device or a service failure. Moreover, the peer-to-peer infrastructure provides the system with useful properties such as fault tolerance, the possibility of using public key infrastructures to implement secure transactions, and scalability.

ECO shared files are managed by a peer-to-peer file repository called ECO-FR. In ECO-FR, shared files are described by a XML document that is also used to advertise the presence of the document in the system. In these advertisements, ECO-FR stores the name of the document as well as some other metadata such as the owner, a list of prioritized authors, the date of creation, the last date of modification, key words, if it belongs to a project, and so on. All these data are intended to be used in more complex searches than just looking for a file with a particular name. To cope with mobility and disconnections, all ECO-FR operations are idempotent and the communication is based on Jxta pipes.

The rest of the document is organized as follows. In the following section we present an analysis of the most important implementation aspects that a Collaborative Ubiquitous Computer System must fulfill. In section 3, we present a detailed explanation of the implementation aspects of ECO. In this section we emphasize on the peer-to-peer nature of the underlying infrastructure and how we solve the issues introduced on section 2. We also present the design of the Distributed Awareness Service and the ECO File Repository. In section 4 we detail the behavior of the potential and actual collaboration spaces of ECO as well as the way in which users move between different working modes (working alone, collaborating). In section 2 we analyze and compare similar projects. Finally, in section 6 we present our conclusions as well as work in which we are currently engaged.

2 Requirements for Collaborative Ubiquitous Computing Systems

A ubiquitous collaborative tool must allow its users to interact with the environment (artifacts and humans) in order to settle down or to carry out collaborative work. In order to do so, we identify seven important characteristics that the implementation of an ubiquitous collaborative tool must meet: scalability, device-independent access, support for disconnections and mobility, context awareness, enhanced security mechanisms, open protocols, and fault tolerance.

Scalability is important because an excessive growth in the response times could prevent the utilization of a collaborative system. This increase in the response time could be caused by a considerable growth in the number of users and shared resources managed by a collaborative system; situation that is not very improbable in nowadays systems. On the other hand, ubiquitous scenarios described by Marc Weiser commonly include hundreds of computers [4] interacting by means of wireless and fixed networks. Thus, architectures must be designed to allow for the dynamic inclusion of software and hardware components in response to an increase in the number of users and shared resources.

In order to be ubiquitous, a collaborative tool must not attach its users to a fixed computer nor have Internet access as a prerequisite to proper functioning. The system must allow to its users to physically move while using it and must have support for a variety of devices such as PDAs, smart phones or laptop computers. On the other hand, collaborative tools must be capable of dynamically find and interact with other

ubiquitous components [5] and collaborative artifacts such as public displays or projectors. So, the underlying distributed infrastructure as well as the key services that implement the application must support the utilization of different kinds of networks (fixed, wireless, ad hoc), manage possible disconnections, and implement standard protocols that allow dynamic discovery of ubiquitous components.

Context-awareness is a key characteristic of a ubiquitous collaborative tool. A collaborative tool must sense the current state of the user's environment and present a summarized view of the most relevant aspects. More specifically, a collaborative tool must make its users aware of interesting situations such as potential for collaboration and the state (physical localization, who is using it, etc.) of relevant shared artifacts. For example, to know that a shared document is being edited using a near public display could be a good evidence of potential collaboration.

Secure communications are necessary when shared documents are confidential and when we need to implement mechanism to assure that an author could not deny a modification made by her to a particular document. Privacy violations could also prevent the utilization of a collaborative tool.

Finally, as in any distributed system, bottlenecks and single-points of failure must to be reduced. So it is important to design architectures that support failures in some of its components without affecting the system overall functioning.

3 Implementation Aspects

To achieve the seven characteristics presented in the previous section, we purpose the implementation of peer-to-peer services using Jxta and a modified version of Jxme [6] that allows dynamic finding of relay peers.

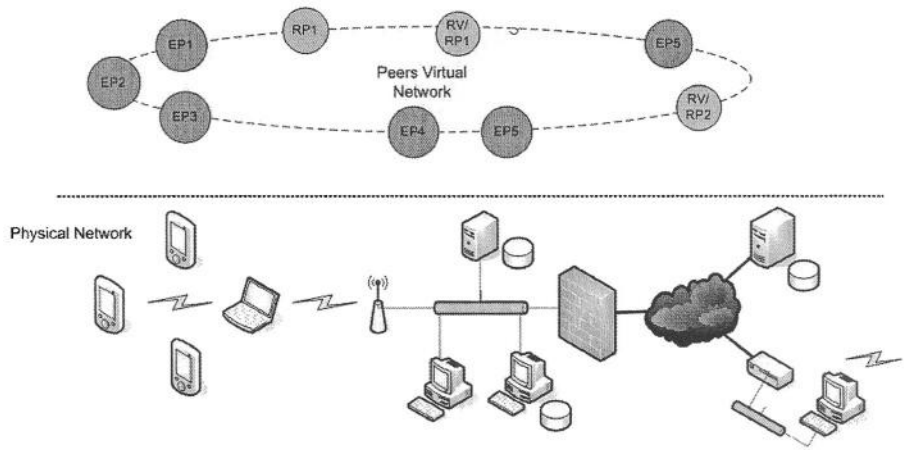

Fig. 1. The project Jxta virtual network

As it is shown on Figure 1, the main advantage of using the Jxta platform is that their protocols establish a virtual network overlay on top of the Internet, allowing peers to directly interact and self-organize independently of their network connectivity and domain topology (firewalls or NATs) [3]. In ECO, services (ECO-FR and ECO-DA) as well as ECO-IM and ECO-CE are implemented as Jxta peers.

3.1 ECO File Repository (ECO-FR)

ECO's file repository service is composed of a variable set of peers which are commonly located on fixed computers. These peers publish advertisements of their service into a distributed hash table implemented by a special kind of peer known as rendezvous peers. Among other information, advertisements contain a description of the resource or service they advertise (for example a file repository service) and a Jxta identificator needed to locate that resource or service. When a new ECO-FR peer is created, and as it is shown on Figure 2, it first has to publish its advertisement (1). To interact with a File Repository peer, ECO-IM and ECO-CE peers query rendezvous peers to find out ECO-FR's advertisements. To do so, (2) ECO-IM (Client1) sends a query to a rendezvous peer (RV1) looking for an ECO-FR. RV1 looks if it has an index of that advertisement. If it does not find an index, it propagates the query (3) to the next rendezvous (RV2). When the query reaches RV2, RV2 finds the index for the advertisement and forwards the query to peer ECO-FR1. When ECO-FR1 receives the query, it sends the advertisement to peer Client1 (4). When Client1 receives the advertisement, it is able to invoke the services implemented by ECO-FR1.

As we mentioned earlier in section 3.1, to create a new shared document, Client1 sends a XML document to the ECO-FR1 peer. ECO-FR1 will use this XML document as the advertisement of the new shared document. So, ECO-FR1 will push the file advertisement into the distributed hash table. To read or write on the newly created document, a Client must find the document's advertisement on the distributed hash table using a process similar to the one just described.

ECO-FR defines four simple operations over shared documents: Idempotent read, Idempotent write, Metadata actualization, and Delete. Finally, at any time, more ECO-FR can be included into the system (17).

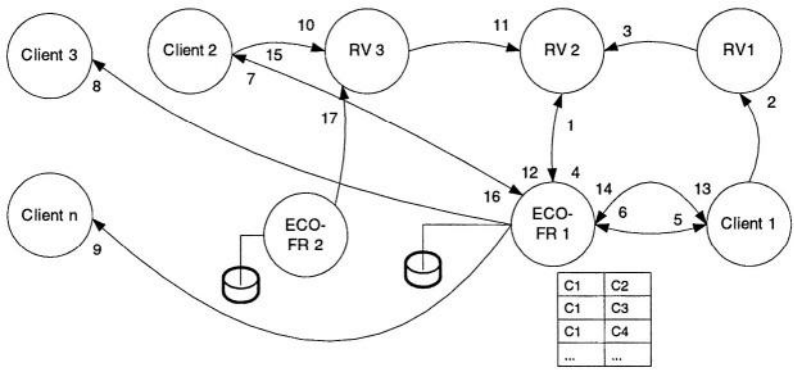

Fig. 2. ECO-FR peers implement the file repository and the distributed awareness services

3.2 ECO Distributed Awareness Service (ECO-DA)

It is important to remember that in ECO, we are interested in the activities performed over shared documents. We leave presence awareness to the traditional services such as Jabber or MS Messenger. The distributed awareness service is implemented by the ECO-FR. When a peer (Client1) accesses a shared document (6), the ECO-FR1 looks

for the advertisements of the co-authors registered in the XML document used to create the shared document. Authors' advertisements contain the information necessary to contact the ECO-IM that is actually using a particular user. If a user is not currently connected, the ECO-FR won't find her advertisement because all advertisements are published with a lifetime that specifies the duration of that advertisement in the network. An advertisement must be republished to extend its lifetime before it expires. So, ECO-FR collects the advertisements of all currently connected co-authors and using a propagating pipe keeps them informed (steps 7, 8, 9 of Figure 6) of the activities being made over the shared file. In this situation, Client2 could get interested in participating in a collaborative edition session and send a request to ECO-FR1 (10, 11, 12). When ECO-FR1 receives the request, ECO-FR1 propagates it to Client1 (13) who decides whether to accept the request or not (14). If the request is granted, ECO-FR1 notifies it to Client2 and the collaborative session starts. Following this schema, an ECO-IM could be receiving awareness information from *n* different ECO-FR if its user is co-author of *n* shared documents located on *n* different ECO-FRs.

ECO-FR and ECO-DA services are scalable because at any time, more ECO-FR can be added to the system without interfering with its current operation. ECO-FR peers use pipes as its main method to interchange information with other peers. As we mention earlier, pipes provide the illusion of virtual in and out mailboxes which are transport independent. So, a ECO-FR or a ECO-DA peer is able to communicate with any other peer not regarding if both are running on a handheld computer or on a desktop computer.

3.3 Secure Interactions

If a secure interaction is needed, the ECO system uses secure pipes to interchange information. When a JXTA secure pipe is created, and the associated peer endpoints are resolved, a virtual TLS transport is instantiated. The transport is bi-directional secured end-to-end with TLS, independently of Jxta relays and the underlying physical transports. In order to participate in a secure communication, peers must possess the X509.V3 root certificate of any peer with whom they wish to communicate securely. The root certificate contains the RSA 1024-bit public keys used to verify the RSA private key signatures of peer X509.V3 service certificates. The service certificates are used to authenticate the communicating peer endpoints by TLS after signatures of the root certificates of their issuers are locally verified.

When a secure shared file is created, a secure peergroup is also created. When a peer accepts to be co-author of a secure shared file, he or she will receive the shared document's creator root certificate under the protection of a TLS connection, and will use a Certificate Service Request (CSR) to acquire a group membership X509.V3 certificate signed with the private key of the group creator's root certificate. When a peer contacts another peergroup member, the former peer can be authenticated by the latter using TLS handshake's certificate request/response and certificate verification. Secure communication is currently implemented only for ECO-IM and ECO-CE running on desktop computers.

4 ECO Collaboration Tools

The ECO system is composed of two collaborating tools: The ECO Instant Messenger (ECO-IM) and the ECO Collaborative Editor (ECO-CE). The main goal of both, ECO-IM and ECO-CE is to penetrate naturally into the lifestyle of users and help them carry out their activities more efficiently. To do so, both tools were designed to model as close as possible the way in which a group of authors interact in a traditional collaborative authoring session, where two or more persons seated by a computer work on a common document, but presenting enhanced facilities in order to make the collaborative sessions more productive.

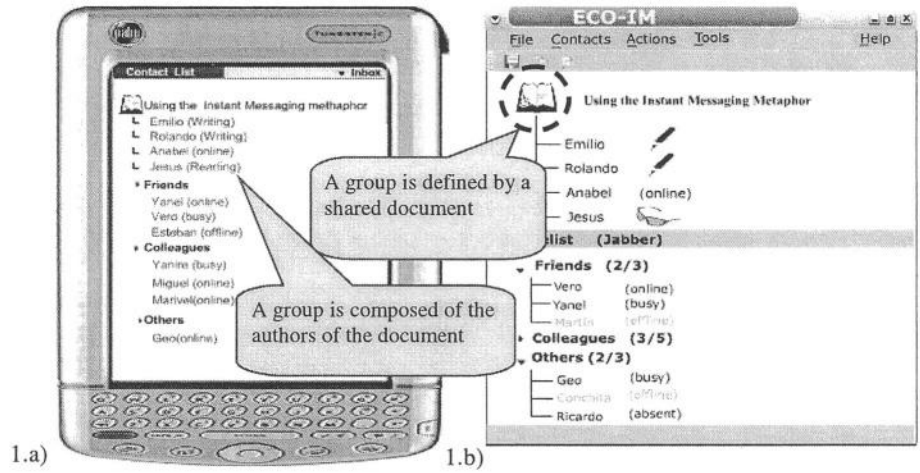

Fig. 3. Main interface of ECO (ECO-IM). a) ECO-IM running on a PDA. b) ECO-IM running on a desktop computer. In ECO the relevant state of a user is the one related to the shared document, i.e. if a particular user is editing, reading or not accessing it

The first of these facilities is the implementation of an ubiquitous working environment that allows users to be aware of the state of current editing or collaborative editing sessions regardless of whether the users are accessing the system by means of a fixed or a mobile computer. As we mention earlier, we choose to make users aware of "edition activities on relevant documents" rather than just "user or document presence or availability" because knowing that a user or a set of users are editing a relevant document is a better evidence of potential collaboration. We show this information in an IM-like tool because instant messengers have demonstrated their capability to penetrate into the lifestyle of their users and they already are a commonly used collaborative tool. As it can be seen in figure 3, the main interface of ECO (ECO-IM) is similar to those used in commercial instant messengers where users arrange its contacts in different groups and the system shows them the current state (on line, without connection, etc.) of their contacts. In ECO each group corresponds to a particular shared document and the members of that group are the authors or reviewers of that particular document.

Figure 4, shows a summarized activity diagram of the ECO system. In it, we can see how the transitions between individual and collaborative work is carried out:

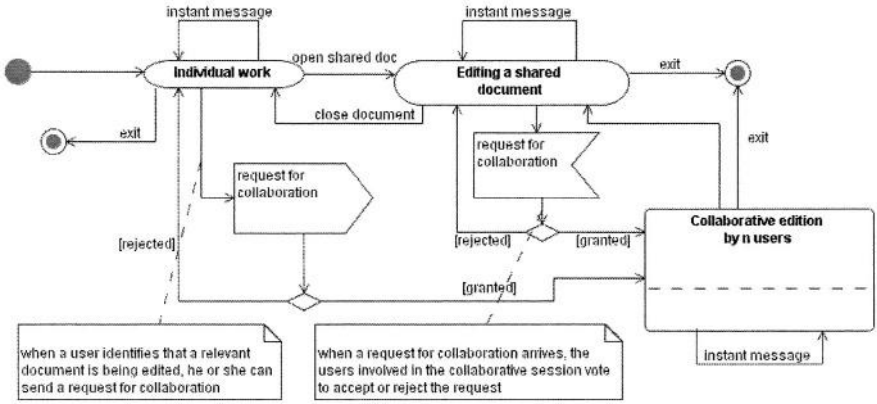

Fig. 4. Activity Diagram of the transitions between individual and collaborative work in ECO

- When a user is doing individual work (any activity that is not related with a shared document loaded into ECO) she can be aware of the status of other users in an unobtrusive manner. So, she can identify a potential collaboration and decide to request her joining on a collaborative authoring session.
- If the request is granted by the current participants of the collaborative session, the ECO-CE of the requesting user will be open and the requesting author will be able to collaborate.
- To go back to individual mode of work, an author just needs to close the ECO-CE.

In the following sections we will explain with more detail the behavior of ECO's collaborative tools.

4.1 The Potential Collaboration Space (ECO-IM)

ECO-IM is our potential collaboration space and it is also in charge of a set activities:

Creation and Elimination of Shared Documents. When a shared document is created, ECO-IM helps the document's owner to produce an XML document with the following information: Name of the shared document, owner, date of creation, a prioritized list of authors, a set of keywords, name of the project to which the document belongs, and if enhanced security is needed an X509.V3 certificate. This information can be modified at any time by the document's owner.

Once the new document is created, ECO-IM asks proposed users if they agree to participate as co-authors of the newly created document. These users can accept or reject the invitation. The creation of the new shared document is reflected by the creation of a new group into ECO-IM. The group will be integrated by those users who have accepted the invitation. By default, the name of the group is the same as the name of the shared document. When a document is deleted from the ECO-FR, the corresponding group is also deleted from the ECO-IMs of the authors related to that document. Only the owner of a shared document has permission to delete it from the ECO-FR.

Manage Transitions from Individual to Collaborative Work. When a user identifies a potential for collaboration, she just has to double-click on the icon that represents the document in order to send a request to participate in the collaborative authoring session. As it can be seen in Figure 5, when a request for collaboration arrives, ECO-IM displays a window asking to the current participants of the collaborative session if they agree with the inclusion of the new participant. The ECO-IM collects the votes to make the final decision. In case of a draw, ECO-IM utilizes the priorities given to the users (when the document was created) in order to give more weight to the vote of the current user with the highest priority. As an alternative, the current user with the highest priority may decide alone if a new co-author is accepted in a collaborative edition session.

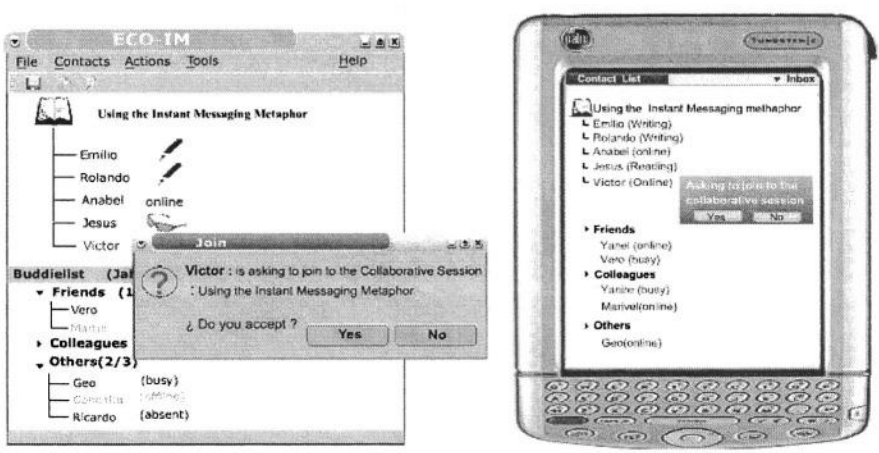

Fig. 5. ECO-IM implementing a voting session to decide if a new member is admitted in a current collaborative authoring session

4.2 The Actual Collaboration Space (ECO-CE)

The actual collaboration space is implemented by the ECO Collaborative Editor (ECO-EC). As we mention above, we define a collaboration schema similar to the one used in traditional not distributed collaborative authoring sessions. In ECO-CE, collaborative edition sessions are directed by the current participating author with the highest priority. At any time, the author with the highest priority decides who and when may modify the document. As it can be seen in Figure 6b, the interface of this author is enhanced with radio buttons that can be used to administer the access capabilities of other users.

If the author with the highest priory leaves the current session, the next author with the highest priority will take control of the collaborative session. In the same way, if a collaborator with a higher priority arrives, she will take control of the collaborative session. When an author no longer has the highest priority, the panel shown in Figure 6b is hidden and a panel for a regular author (Figure 6d) is displayed.

A regular participant may ask at any time for a particular right over the shared document. To do so, the collaborator just needs to activate a radio button labeled with

the permission being requested. This is shown in Figure 6d. Requests for permissions are displayed as text messages in the chat section (Figure 6c) of the current coordinator. This message will be displayed periodically until the request has been granted. Finally, at any moment, the participants of a collaborative session can interchange ideas or comments by means of a chat-like tool that is displayed at the bottom of the window. This can be seen in Figure 6c.

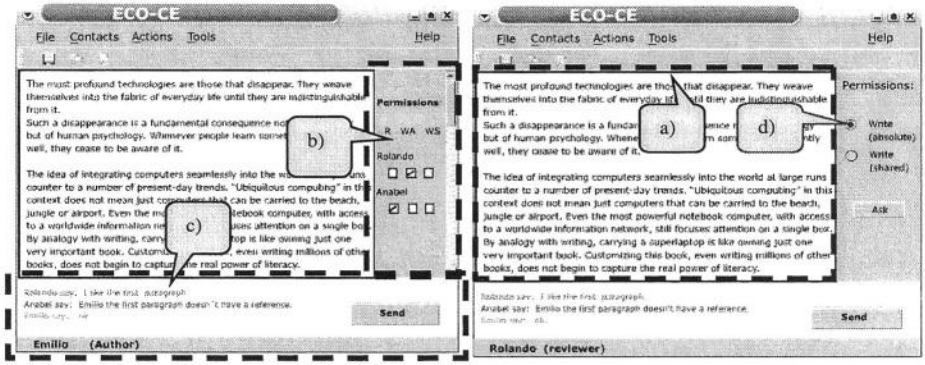

Fig. 6. ECO-CE that implements the actual collaboration space. a) The edition section. b) The session administration panel displayed in the interface of the user with the highest priority. c) The chat section. d) The interface used by a regular participant to ask for a particular right over the shared document

5 Related Work

Many authors [7-14], among others, have identified the need of collaborating authoring tools and many projects have been developed to help carry out collaborative editing sessions. Collaborative authoring requires support for coordination of the authoring activity as well as flexibility to support the dynamically changing needs of the collaborative writing groups over time. The coordination of the authoring activities include the need to support several writing strategies, to communicate changes among co-authors, as well as to provide collaborators with information related to current state of the authoring process. The flexibility requirements include the need for simple methods to support a seamless transitions among writing strategies (sequential, reciprocal and parallel), among writing activities (brainstorm, research, initial plan, writing), between synchronous and asynchronous ways of communication, and from a private to a shared workplace [11]. Another key aspect of CSCW is that collaborating in the authoring of a document does not necessarily mean that users want to be notified of all updates or provided with all the shared data. So, reducing the cognitive load on users by better organizing the coordination activity, by filtering unnecessary information, and by delivering key information can greatly increase the effectiveness of CSCW systems [15].

The approach to keep track of shared resources and presenting a simplified view of the collaborative activities presented here, is inspired in instant messaging (IM) or user presence awareness applications such as ICQ, AOL Instant Messenger, Jabber, MS Messenger, Mercury Prime, Quicksilver, etc. This type of application has become

increasingly popular in the last few years and client applications have been developed for handheld computers and smart cellular phones. According to IDC [16], currently, around 18.3 million workers use instant messaging for job-related purposes. That number may seem paltry compared to the 132 million consumers who use IM for fun, but it represents nearly a 300 percent increase from the number of corporate users last year. By 2005, IDC estimates that more than 229 million workers will use instant messaging to get their job done. From the implementation point of view, Instant Messengers are mainly based on synchronous communication over the Internet [17, 18]. Instant messengers have been traditionally developed using client-server architectures but recently, new proposals have appeared that use P2P architectures [19].

Instant messengers however, only provide awareness on the state and availability of other people. For this reason, even when a traditional IM application can be used as a "companion tool" to provide or complement collaborative features of other applications, it will still lack the ability to provide document presence awareness information.

Kirby and Rodden [20] identify the need for lightweight monitoring of shared resources during collaborative authoring. The contact toolkit they developed uses Watch Objects to detect alterations to files or register commands invoked upon files. However, rather than using a non-intrusive interface to monitor activities on documents, users need to explicitly consult the state of a particular document using a Web browser. In contrast, users of systems such as ECO or Doc2U [11] can track the state of several objects and collaborators with a simple glance. The ability to discover opportunities for collaboration and to establish initial contact to start or launch actual collaborative work has also been identified by several authors. Examples of these are the work on casual, informal and lightweight interaction studies and systems [21-24].

The most closely related project to ECO and the one we will analyze in more detail is Doc2U [11]. Doc2U is an Instant Messaging and Presence Awareness tool, which implements a Potential Collaboration Space for the PIÑAS cooperative authoring environment [12]. AIDA is a Doc2U client for mobile devices [10].

Doc2U has three different kinds of paired components and servers: user management, session management and document management. Doc2U and AIDA are implemented using the JabberCom library [25] and their document storage server is implemented as an extended Apache HTTP Web Server. The main difference between Doc2U and AIDA with respect to ECO is related with their implementations. But, these implementation differences also produce different behavior as well as different functional capabilities. Whereas Doc2U and AIDA are implemented using a client-server architecture, ECO is based on a peer-to-peer (P2P) architecture. The P2P architecture allows us to implement ad hoc scenarios because a set of handheld devices using ECO don't strictly need access to the Internet (to contact a web server or an awareness server) in order to participate in a full-featured collaboration session. This is due ECO services can be implemented by any peer running on mobile computers (with reasonable computer power).

Finally, ECO could be complemented with systems that manage early stages of initiating collaboration. For example, I2I [26] provide opportunities for users who may not know each other to collaborate by let them be aware of the activities of other that share common interests.

6 Conclusions and Future Work

We propose an extension of the instant messaging and presence awareness paradigm to support the implementation of a potential collaboration space that lets users become aware of collaborative authoring sessions being carried out, and join them opportunistically. In order to be easy to use and adopt, ECO implements a simple but useful collaborative schema that resembles the way in which a group of authors collaborate when they are physically seated in front of the same computer. We developed ECO clients for a variety of devices such as handheld computers and desktop computers using fixed or wireless networks. On the other hand, by implementing specialized ECO Collaborative Editors, our approach can be easily extended to support collaboration over a variety of shared documents such as UML diagrams, electronic diagrams, computer programs, images, presentations, and so on. All these resources can be dynamically founded using the distributed hash table implemented by Jxta. We are currently working on the dynamic interoperation with ubiquitous components. We want to dynamically integrate artifacts that could be useful in a collaboration session. For example, support collaboration by means of geographically near artifacts such as public displays or projectors.

From the implementation point of view, we identified seven important characteristics that a ubiquitous collaborative tool must meet: scalability, device-independent access, support for disconnections and mobility, context awareness, enhanced security mechanisms, open protocols, and fault tolerance. To cope with these requirements we develop a file repository service and a distributed awareness service based on a peer-to-peer architecture. The P2P architecture allows us to implement ad hoc scenarios because ECO main services (distributed awareness and file repository) can be deployed on a set of mobile devices interconnected by means of an ad hoc network. This way, users using ECO don't strictly need access to the Internet in order to participate in a full-featured collaboration session. Moreover, transport independent identifiers allow mobile devices to access system functionality not regarding the kind of network they are using or if they change of network adapter. On the other hand, modern P2P infrastructures such as Jxta, have implemented virtual transport based on TLS to provide secure communication between peers. For example, the default cipher suite for Jxta is RSA 1024 with 3DES and SHA-1.

We also developed the instant messenger client and the collaborative editor as peers. So, they can be seamless integrated with the P2P distributed services and obtain full advantages of their design. Finally, we have to mention that ECO is still under development, and although it is a suitable demonstration system, it is not ready for deployment.

Acknowledgments

The authors of this paper would like to thank the Centre for Computing Research (CIC-IPN), General Coordination of Postgraduate Study and Research (CGEPI), National Polytechnic Institute (IPN) and the Mexican National Council for Science and Technology (CONACYT) for their support to this project.

References

1. Baecker, R.M., D. Nastos, R. Posner, and Mawby, K. L.: The User Centered Interactive Design of Collaborative Writing Software. Proceedings of the ACM Conference on Human Factors in Computing Systems (INTERCHI'93), Amsterdam, The Netherlands, 24-29 April, (1993) 399-405
2. Morán, A., et-al: On the design of potential collaboration spaces. International Journal of Computer Applications in Technology, Vol. 19, Nos. ¾, (2004) 184-194
3. Traversat, B., et-al: Project JXTA 2.0 Super-Peer Virtual Network. http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf
4. Weiser, M: The Computer for the 21st Century. Scientific American, Vol. 265, No. 3, September (1991)
5. Kindberg, T. and Fox, A.: System Software for Ubiquitous Computing. IEEE Pervasive Computing, Vol. 1, No. 1, January-March (2002)
6. JXTA for J2ME: http://jxme.jxta.org
7. Favela, J., & Ruiz, D.: Collaborative Authoring and Reviewing over the Internet. WebNet Journal: Internet Technologies, Applications & Issues 3(3), (2001) 26-34
8. Fish, R. S., Kraut, E. Mary, Leland D.P. and Cohen, M.: Quilt – A Collaborative Tool for Coperative Writting. Proc. COIS'88 Office Information System. Palo Alto. Ca, March (1988)
9. Neuwirth, C., Kaufer, D., Chandhok, R. And Morris, J.: Issues in the design of computer support for co-authoring and commenting. Proceedings of the third conference on CSCW'90. Baltimore, MD : ACM Press, (1990) 183-195
10. J. Favela, C. Navarro, Rodriguez, M.: Extending Instant Messaging to Support Spontaneous Interactions in Ad-hoc Networks. In Proceedings of ACM 2002 Conference on Computer Supported Cooperative Work (CSCW '02), New Orleans, Louisiana, (2002)
11. Moran, A.L., Favela, J., Martinez, A.M., and Decouchant, D.: Document presence notification services for collaborative writing. Proceedings CRIWG-2001, 7th International Workshop on Groupware, IEEE Computer Society, Darmstadt, Germany, (2001) 12-133
12. Decouchant, D., Favela, J., Martínez, A. M.: PIÑAS: A Middleware for Web Distributed Cooperative Authoring. SAINT 2001, (2001) 187-194
13. Malone, T.W., Grant, K.R., Lai, K.Y., Rao, Rosenblitt, R., D.: Semi Structured Messages are Surprisingly Useful for Computer-Supported Coordination. ACM Transactions on Office Information Systems, Vol. 5, No. 2. Apr. (1987) 115-131
14. Ellis, C. A., Gibbs, S. J., and Rein, G. L.: Design and use of a group editor. Proceedings of IFIP WG2.7 Working Conference on Engineering for Human Computer Communication, August 1989, (1990) 13-28
15. Prakash, A., Shim, H.S., and Lee, J.H.: Issues and Trade-offs in CSCW Systems: IEEE Transactions on Data and Knowledge Engineering, Jan.-Feb. Vol. 11, Issue 1, (1999) 213-227
16. International Data Corp: http://www.idc.com/
17. Grinter, R. E., L. Palen: Instant messaging in teen life. ACM 2002 Conference on Computer Supported Cooperative Work (CSCW 2002); November 16-20; New Orleans; LA; USA. NY: ACM Press; (2002) 21-30
18. Nardi, B., Whittaker, S. and Bradner, E.: Interaction and Outeraction: Instant Messaging in Action. Proceedings of CSCW 2000. ACM Press, (2000) 79-88
19. JXTA Instant Messenger: http://jxtaim.sourceforge.net/
20. Kirby, A., and Rodden, T.: Contact: Support for Distributed Cooperative Writing. In proceedings of ECSCW'95. Stockholm, Sweden, Sep 10-14 (1995) 101-116
21. Whittaker, S., Swanson, J., Kucan, J. and Sidner, C.: TeleNotes: Managing lightweight interactions in the desktop, ACM Transactions on Computer Human Interaction, Vol. 4, No. 2, June, (1997) 137-168

22. Churchill, E.F. and Bly, S.: It's all in the words: supporting work activities with lightweight tools. Proceedings of GROUP'99, Phoenix, AZ: ACM Press, Nov 14-17, (1999) 40-49
23. Tang, J.C., Yankelovich, N., Begole, J., Van Keek, M., Li, F. and Bhalodia, J.: ConNexus to Awarenex: extending awareness to mobile users. Proceedings of CHI 2001, Seattle, WA: ACM Press, March 31-April 4, (2002) 121-128
24. Isaacs, E.A., Walendowski, A. and Ranganthan, D.: Hubbub: a sound-enhanced mobile instant messenger that supports awareness and opportunistic interactions. Procedings of CHI 2002, Minneapolis, MN: ACM Press, April 20-25, (2002) 179-186
25. Jabber SoftwareFoundation: http://www.jabber.org/
26. Buzdik, J., et-al: Clustering for Opportunistic Communication, Procedings of ACM WWW2002, May 7-11, Honolulu, Hawaii, USA, (2002)

Mobile Support for Collaborative Work

Luis A. Guerrero[1], José A. Pino[1], César A. Collazos[2],
Andres Inostroza[1], and Sergio F. Ochoa[1]

[1] Department of Computer Science
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
{luguerre,jpino,neinostr,sochoa}@dcc.uchile.cl
[2] Department of Systems
Universidad del Cauca
FIET-Sector Tulcan, Popayán, Colombia
{ccollazo}@unicauca.edu.co

Abstract. An attempt is made to characterize situations in which the use of mobile devices can be useful for the development of collaborative systems. Mobile devices have advantages, such as small size, low cost, portability. They also have disadvantages, such as small viewing screen, little storage capacity, slow processor, unreliable communication facilities. The idea is to use them when advantages are most relevant and disadvantages do not affect the system under development. A collaborative system for text co-authoring is presented as an example of design for the best conditions of mobile devices inclusion. This system uses the mobile devices for individual tasks performed while away from normal work site in uncomfortable or congested places.

1 Introduction

Many people need to move themselves to accomplish their jobs. For them, hand-held computer devices have convenient small size and there are simple but useful applications already running on these gadgets, such as telephone directories, to-do lists, and calendars.

Hand-helds have recently improved the portability introduced by notebooks some years earlier. The newest models include relatively larger memories than the first ones, better graphic resolution and wireless communication. Therefore, many system developers increasingly look at hand-helds as tentative devices to build with them new applications which might take advantage of their features. In particular, some developers may be interested on designing systems to support people doing cooperative tasks. However, a few relevant questions should be asked: are hand-helds appropriate components of collaborative applications? If they are, which are the tasks they support in the best way?

Of course, it is not obvious hand-helds – also called Personal Data Assistants, or PDAs – may be adequate for ambitious systems involving several people with many complex interactions among them. One of the main advantages of handheld computers is that they are portable. Also, initial vendor-supplied software encouraged individual rather than collaborative use. Even now, most handheld applications often reinforce the idea of a handheld computer as a *personal* digital assistant. Neverthe-

G.-J. de Vreede et al. (Eds.): CRIWG 2004, LNCS 3198, pp. 363–375, 2004.

less, recent articles describe group or collaborative applications based on this kind of devices [20, 26].

Developing collaborative applications including traditional computers is already a hard task [13]. Trying to design useful collaborative applications using PDAs present further challenges, since these devices have several difficulties for group work when compared to normal computers. These restrictions include small screens for visualization and interaction, limited input facilities, short-life batteries and slow processors.

On the other hand, when we observe collaborative work we may notice there are some tasks or activities which are performed individually in many cases. We may hypothesize PDAs as potentially useful devices for these individual tasks within the group context. This hypothesis may be reinforced if these tasks are to be done by people with high mobility.

The purpose of this paper is to attempt to characterize collaborative situations in which PDAs could be used with advantage. It will also present a case in which such conditions are held for one of the most traditional collaborative applications: text co-authoring.

2 Cases Reported in the Literature

The mobile computing concept is quite new and has no clear definition, although some studies have already tried to survey this fast-growing area of information technology. Mobile computing does not only involve mobile computing devices (notebooks, cellular phones, PDAs and wearable computers), which are designed to be carried around, but also the mobile networks to which these computers are connected. Mobile services are the third component, rounding out this definition of mobile computing.

In this sense, mobile computing has been discussed in just a few recent papers from certain typical points of view. Wireless network service problems in so-called wireless personal area networks (PAN) are a fundamental issue [38]. In his paper, Zimmerman examines wireless technologies appropriate for PANs, and reviews promising research in resource discovery and service utilization, including data formats. Zimmerman already emphasizes the role of Extensible Markup Language (XML) as a standard for structured document interchange. Some interesting further problems of mobile network services are discussed by Chalmers [4].

We agree with Zimmerman in emphasizing the evolution of PDAs as the engine driving mobile computing. Pocket-size, low-price units with long-lasting power autonomy and broad functionality guarantee the critical mass of buyers necessary to motivate the industry to produce and further develop this type of devices. (The growth of mobile phone use is a good example of such a stimulant loop). Zimmerman's list begins with the legendary but rather clumsy Newton, introduced by Apple Computer in 1993. The Nokia 9001 communicator, introduced in 1998, was another milestone, representing the first successful fusion of a handheld PC with a mobile phone. Unfortunately, it ran on a non-standard operating system, and it was therefore short of software. The Sagem Pocket PC, introduced by Sagem and Microsoft in 2000, was a hybrid of the same kind, but with a standard operating system: Windows CE.

The opposite of integrating functions in a single device is device modularization, which has some additional positive effects. A small mobile phone with data transmission capability can be connected to a PDA or another more specialized device, thus

offering them networking capabilities. Bluetooth wireless connectivity [2] simplifies and automates the linking of devices in a very convenient way, thereby supporting the concept of modularity. This technology enables users to connect to a wide range of computing and telecommunications devices without the need to carry, buy or connect cables. The new generation of mobile phones, notebooks and PDAs already comes with Bluetooth. However, integration versus modularization is an old dilemma, one which is not very significant in the present case.

A paper by Held and Ziegert describes some of the characteristics of mobile computing, presents a system model, and shows in more detail how one of the features of mobile computing – service mobility – can be realized [15]. Some earlier papers addressed the specific problems of mobile computing, like designing mobile computing systems using distributed objects [6], but the most relevant research reports can be found only in recent publications. Interesting topics include: mobile client-server computing, including mobile-aware adaptation, an extended client-server model, and mobile data access [18], interaction with the World Wide Web via wireless-connected PDAs, problems with bandwidth limitations, screen real-estate shortage, battery capacity, and the time costs of a pen-based search keyword input [3], and client/agent/server adaptation of a framework to enable performance improvement, which is very important on slow or expensive networks [29]. Other subjects include context-aware collaboration [24], and data management, including data dissemination over limited bandwidth channels, location-dependent data querying, and advanced interfaces for mobile computers [1].

A field which could much benefit from mobile devices is electronic commerce (M-commerce). Rao and Minakis examine location-based services and their importance in M-commerce, taking advantage of mobility rather than repackaging old applications in a new format [32]. While old methods can be adapted and retooled to create applications and explain M-commerce successes and failures, new methods, tools, and ways of thinking must be developed and refined to take advantage of mobility and its potential. The same could be said about other PDA application fields.

Although the number of research papers addressing mobile computing is modest, there is no doubt that much research is going on, perhaps even too fast for papers to be published. As one technology overtakes another [17] it is probably wiser to concentrate on more general concepts and problems. One such problem is the adaptation of existing information systems suitable for efficient integration with mobile computing. In this regard, we agree with Vizard that "until then, mobile computing will just remain a troublesome niche application for those who can afford to pay for it" [37].

We believe it is important to gain an understanding of how collaborative processes can be structured using handheld computers. Handheld computers have the potential to impact both the individual and social processes and we need to understand how to best design the inclusion of these devices to support these activities.

Sarker and Wells describe a framework for mobile device usage and acceptance identifying the relationships between the variables motivating people not only to purchase devices but actually to use them for M-commerce [33]. Their model presents an integrated framework for use and adoption of PDAs according to an IPO model. IPO (Input-Process-Output) consists of Inputs (User characteristics, technology characteristics, modality of mobility, and the surrounding context); Process (exploration and experimentation), and Output (outcome of the process). Our strategies have similar goals, but they rather try to characterize favorable contexts for the use of PDAs.

3 A Favorable Case

We consider next a case in which a group of scientific researchers is trying to write a joint technical paper. Scientific papers [35] are an important --although poorly understood--method of publication. A scientific paper is written for the scientific community at large. The contents may be, e.g., a survey, a tutorial, present a theoretical model or discuss new experimental results. The typical paper needs to introduce the subject, place any results in context with other scientific works, and suggest future possibilities for research.

Some roles need to be specified, e.g., scribe, reviewer, coordinator. Also, some goal achievement strategies and social protocols are required. Let us assume not all co-authors are co-located at all times. This may occur because some of the co-authors work in various places. For instance, one author may wish to do part of his work while being at an airport lounge waiting for a plane, or in the plane itself. Another co-author may need to work in a rather crowded underground train while returning home at evening.

Let us also assume the group has decided to divide the initial writing task among some of the co-authors. Each of these writers then has to produce a draft of a part of the article. The writers will need meetings for information sharing and synchronization after they finish their divergent tasks. During or after these meetings, it is expected co-authors will do material reviewing and re-writing and also some planning.

This set-up seems appropriate to incorporate handheld computers into the solution: perhaps the co-author working in an airport or plane may use a laptop computer, but clearly the one working in a crowded underground train needs something smaller and more portable than a laptop. However, the restrictions stated in the previous sections make us aware of the limitations of PDAs. Their small screen and limited viewing angle makes it difficult for a group of persons to collaborate around a shared display when co-located. Also, the handheld input clumsiness is relevant in this case. Current handheld technology introduces other problems, such as the involvement of users in the transfer of information, and the fact communication is primarily peer-to-peer. Thus, users must switch their focus from the productive task to the act of transferring the information [36]. In conclusion, even when we guess PDAs may be useful, we must carefully design their introduction into the solution. Next section describes the strategies we have designed and implemented to take advantage of this technology.

4 A Design Strategy to Support Collaboration with PDAs

Designing work with PDAs means to take into account both opportunities and restrictions. We have already mentioned restrictions due to the reduced screen size, available memory, input devices and processor speed. We should now add small communications bandwidth and eventual interruptions in information interchange as a consequence of wireless communication intermittence [19]. Therefore, it seems reasonable to consider the following design aspects as influenced by these restrictions:

- Design simple user interfaces, including few elements.
- Consider little memory and storage. Most data stored in the PDA is volatile.
- Consider alternative data input devices. PDAs are slow and somehow unfit for large amounts of data to be input.

On the other hand, opportunities provided by handhelds include: great portability, short start-up and response times and ability for gathering and presenting small pieces of information. Thus, the following design characteristics are suggested by these opportunities:

- PDAs may be useful when user mobility is a main consideration during the operation of the system.
- Use PDAs when individual work is required in unusual, congested or uncomfortable places.
- Consider the use of PDAs when users can make timely short annotations which can be expanded into larger contributions afterwards.
- Consider the use of PDAs during periods of individual divergent work. Each of these periods will probably be followed by a period of group convergent work, in which the previous achievements are appropriately merged and improved.

The role of the individual in collaborative work should not be overlooked, as many activities within a complex undertaking may be better done by individuals than by groups. An interesting study exploring the opposing individual vs. collective views in the foundations of group work in the context of an organization has been done by DeSanctis [9].

5 A Solution to the Case

Our solution to the case presented in Sect. 3 considers to support co-authors in order to work in the most convenient place and time. It also distinguishes work which may be done individually from the work which needs synchronous joint participation. Furthermore, we make another design decision by specifying that the synchronous joint participation be a face-to-face meeting. Individual work output does not have to wait for a meeting to be shared: it may be communicated as individual work proceeds, as explained below.

The technology needed to support this solution includes workstations or notebooks to do normal text and multimedia authoring and editing, and PDAs for some of the individual text creation and edition. It is assumed a wireless local network and Internet connection are available. All of this hardware is used by MoSCoW (Mobile Support for Collaborative Writing), our software composed of four modules: a user management component, a Web editing module, a PDA text editing module, and a communication and synchronization component.

Work supported by PDAs may be done in two ways: network-connected and off-line. When working network-connected, the user works in a way resembling workstation use, i.e., document synchronization is automatic. Off-line PDA work occurs when the co-author steps outside the range of the wireless network. In this latter case, the PDA stores a "copy" of the original document; the co-author then does all text editing as desired. Of course, after off-line work, the performed changes must be synchronized with the master document. When this is done, the master document is stored as a new version. In turn, this means all stored versions must be merged (synchronized) at some time. A coordinator must do this merging, and usually this involves discussion with the other co-authors in order to keep document coherence.

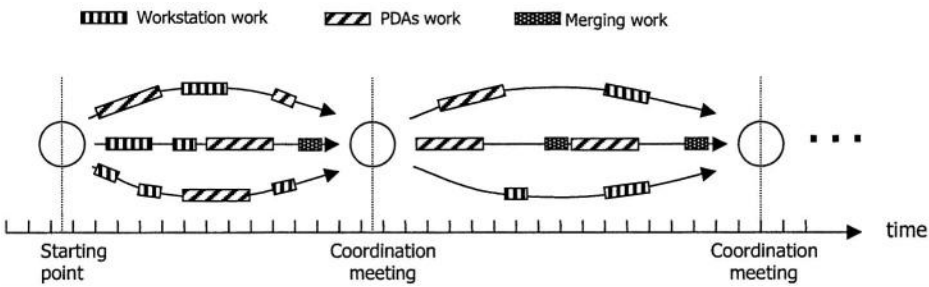

Fig. 1. A sample activities sequence

Figure 1 shows a diagram of a collaborative text-editing job supported by PDAs. Represented activities are editing from a workstation or notebook, editing from PDAs, and merging processes. Large nodes represent coordination meetings. The three depicted co-authors may do divergent work in the most convenient way according to their needs. We may guess that perhaps many of the contributions generated from PDAs are annotations or brief statements which are developed in full when working from workstations afterwards. In Figure 1, a single co-author (coordinator role) is responsible for merging the various existing versions. This latter type of task must be done with the other co-authors being aware and agreeing.

It should be noted the merging process is only needed to incorporate changes made from off-line PDAs because the other ways of work (from workstations or network-connected PDAs) consider an instantly updated shared document. The shared document includes a locking mechanism: only one user can be updating it at the same time. When one user is updating the shared document, the others have reading access. Although the locking mechanism may not be desirable when several users want to update at the same time, the system is actually intended for mostly asynchronous work, and thus, update rights will seldom be denied. Annotations, however, may be done concurrently with one user doing updates on the master document (annotations are actually made on a copy of the master document). Annotations are visible by all group members. Figure 2 shows a diagram of concurrent use of the system.

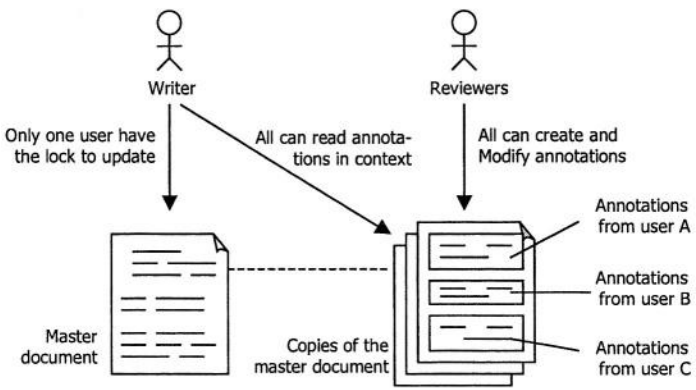

Fig. 2. Concurrent work on a document

The software system consists of four modules. They allow static or mobile operation, as described in the next subsections:

- Web editing module: it is intended as the main program to create, edit and share documents. Roles within the group are also assigned with this module.
- PDA editing module: it also allows creating, editing and sharing documents. It can also manage local documents to the PDA itself.
- Communication module: it manages communication between the PDA and the shared documents database.
- User management module: it handles user creation and privileges within the system. Only an administrator can enter this module.

The system only allows users who have been created with the User management module. The rest of the modules are discussed below.

5.1 Web Editing Module

This module lets users to create, edit and share documents through the Web. When a group member creates a new document, she must provide the list of co-authors and the roles assigned to each of them (the current implementation just considers *reader* and *reader/writer*). A co-author can modify a document by first blocking it; after making changes, she must unblock it. This module also lets co-authors to generate a new document version. Furthermore, the same module allows co-authors to add own annotations and see annotations provided by other users. Figure 3 shows a document being edited via Web.

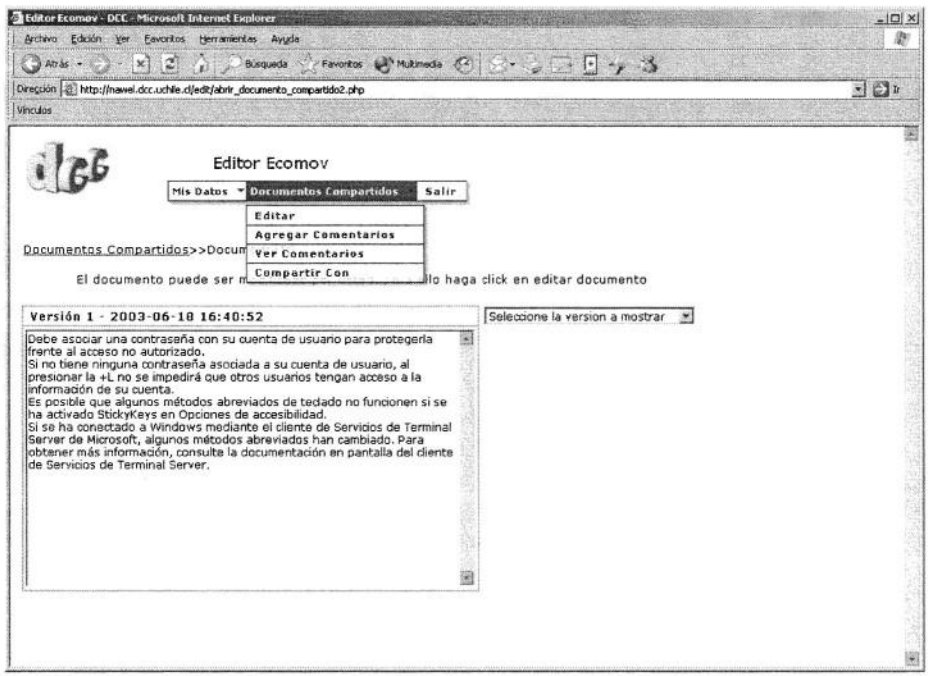

Fig. 3. Editing a document from a Web browser

5.2 PDA Editing Module

The design of our solution tried to make the Web and PDA editing modules as similar as possible. However, taking into consideration the strategy outlined in Sect. 4, the PDA module should be very compact, with a simple user interface, and using little storage. Despite this austere design, the PDA module should still provide the main functionality of the Web module.

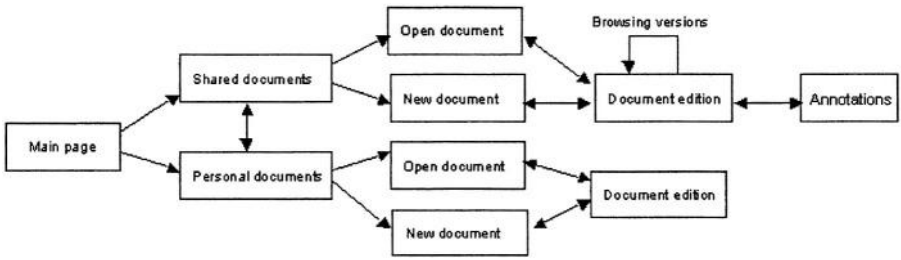

Fig. 4. Navigation model for the PDA editing module

Figure 4 depicts the navigation model of the PDA editing component. Entrance to the system is done through the main page. Here, basic data for connection to the server is initialized. Then, a choice must be made by the user: work on shared documents or personal ones. Shared documents will be the normal choice; personal documents will not be shared when the PDA will get synchronized with the server.

The co-author may create new documents or open previous ones. These may be personal or shared. A typical use may be to create a personal document with an outline of ideas; these are expanded later in a shared document. Shared documents may have several versions, which can be navigated by the co-author. The user can also place annotations on any document from this software module.

Figures 5a and 5b show the PDA editing module user interface. The upper part of the screen has information on the current document. The middle part of the screen presents the document, and the lower part contains the application menu. Fig. 5a shows the "File" menu options. The editing menu has an option to work on the various versions: buttons allow to move forward or backwards on the local document versions.

Annotations can be added to personal or shared documents. In the case of shared documents, a co-author is permitted to include annotations only if he has the corresponding privileges. Annotations creation privileges also include permits to delete them. Annotations are entered as text comments enclosed within braces ({}); see Figure 5b.

5.3 Communication and Synchronization Module

This module allows communication and synchronization between PDAs and the server database. The database is also accessed by the Web editing modules. The main difficulty solved by the communication module concerns concurrency, since several co-authors could be editing the same document at the same time. The module also

solves the document versions management problem. Keys for a simple solution to these problems are the locking mechanism already mentioned and a time stamp associated by the system to each document version. Time stamps are then used by the system itself to guide co-authors on which versions are appropriate for merging.

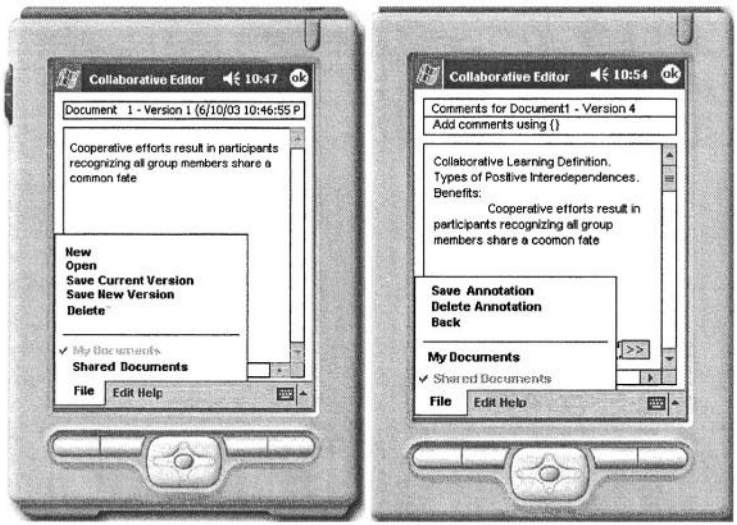

Fig. 5. (a) PDA editing module user interface and (b) adding annotations

The locking mechanism is paired with a unique version of the document, called the master document, as introduced above. When a co-author has blocked the master document, the other co-authors can make annotations over copies of the master document. At a later time, a co-author can modify the master document based on her colleagues' annotations. For such task, the system lets visualize all document copies associated to a master document as separate windows (Fig. 6). Annotations are shown in color to make them easily distinguishable.

6 Related Work

Collaborative text editing has been widely studied. Several tools have also been developed to support it, both for synchronous and asynchronous work, and with various approaches to role assignment, versioning, awareness and editing mechanisms. Some of these tools support our strategy of separating individual tasks from group activities [14,5].

Decouchant et al. have developed a system called Alliance, which allows group participants to ease asynchronous distributed documents edition in a structured way. Stored documents may be remotely accessed through the definition of parts or segments within a document. Such fragments are the basic sharable units [8]. In relation to PREP [28], the approach is to provide information usability and visual representation, emphasized with the use of annotations. Roles are not explicitly supported by this system. GROVE is a distributed synchronic text editor with blocking at a granularity of one character [10].

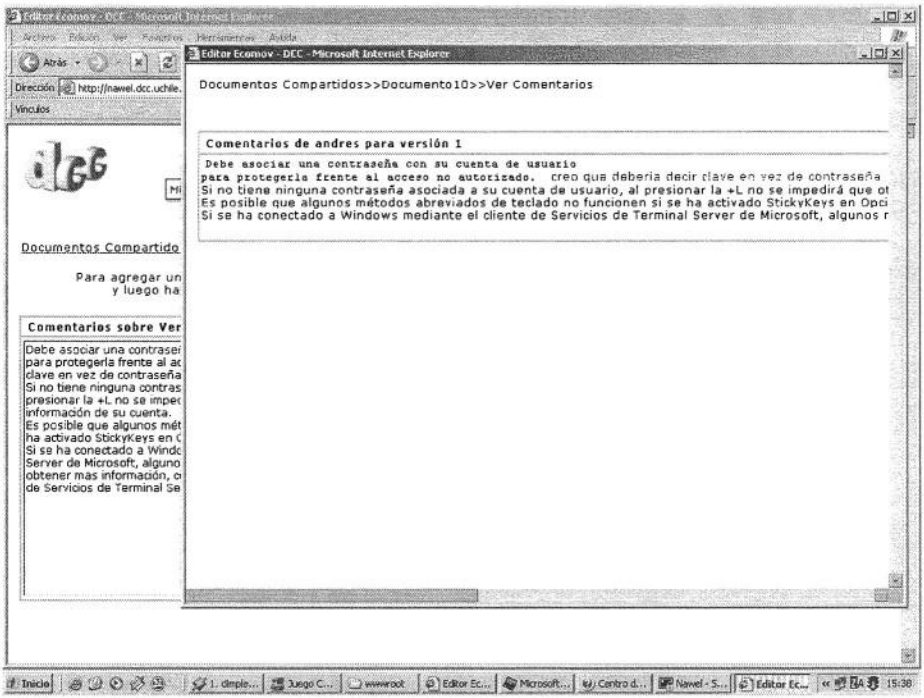

Fig. 6. Improving the master document with a copy containing an annotation

Asynchronous collaborative text co-authoring has been supported by many tools, such as QUILT. This editor bases its support on providing services to very clear roles [22]. Duplex is another such tool, providing support to users connected by Internet [30]. Finally, IRIS allows synchronous or asynchronous collaborative work [21]. Document versioning has long been known. Also, several proposals have been made to present *local versions:* suggested changes to parts of the text. Some of them are change bars [16], flexible diffs [27], active diffs [23] and Stick-Ons [31].

Supporting collaborative work using PDAs is achieved with Shared Notes [12], which allows information interchange among group members. NotePals [7] is a lightweight meeting support system that automatically combines individuals' meeting notes into a shared record. Shared records are essential to maintaining organizational knowledge. Pebbles [26] supports collaborative work both with PDAs and personal computers. Finally, Fieldwise [11] focuses on solving mobility problems; it allows group members to work from their PDAs, coordinating group members for decision making and task performing. None of the previous experiences refers to collaborative text editing.

7 Discussion

We are no longer tied to our desktop computer, since we now have wireless, mobile networking capabilities, and a plethora of new computer technologies. New advances will allow us to access and utilize technology in new ways. Given we often want or need to collaborate with others in many scenarios, it is important we consider the

potential of each technology. In the case of handheld computers and other mobile support devices, each of them has strengths and weaknesses, which should be taken into account to support the collaborative activity. This not only requires communication among devices, but more importantly, it requires an understanding of "how" users can best take advantage of multiple technologies. How should information be distributed across devices? What information is best displayed where? How do users interact with the devices as well as with each other? Which task types should be performed with each device? How should tasks performed in various devices be synchronized?

Of course, some the previous questions may be ignored. It is possible to develop systems which do not take advantage of the devices strengths or without much consideration to the weaknesses. However, the result may be sub-optimal. For instance, it is possible to build a system based on PDAs to be used within a classroom to support typical teaching/learning tasks. In such a case, why not to use a wireless network of notebooks instead, with less problems of batteries running out, transmission interruptions and without so many visualization/input deficiencies? Perhaps the shortcomings outweigh the low cost and small device size of the PDA network in this case.

Joint scientific papers authoring imposes restrictions on the co-authors. They have to be available for discussions, be willing to change some (or all) of the text they have written, etc. One way to facilitate this may be allowing them to work in any place they wish and at any time. It is not an exaggeration to let them do some work while travelling in a crowded underground train: people are increasingly conscious of their time both for work and leisure. PDAs then have something to contribute in this field, since they have such a small size and light weight. Other advantages and disadvantages must be considered, as we have argued, to get a successful system.

Will PDAs evolve to eventually overcome the difficulties we have identified? That cannot be assured, although it is probable some of these current shortcomings will be easied or forgotten. Candidate features to be improved in the short term are batteries, storage capacity and communications reliability.

8 Conclusions

PDAs are appealing and offer many advantages, not the least of which is sustainability. It is their cost effectiveness which makes PDAs an exciting option. The capital investment is low since there is no need for specialized labs, additional wiring, or intensive training. PDA technology is portable, requires little infrastructure and can therefore be readily transferred from one site to the next. It is not platform dependent and readily interacts with any operating system. All these advantages make handheld devices attractive for developers to integrate them in their computer-based systems. However, our argument is PDAs advantages must be considered as well as their disadvantages when designing computer-based systems, in particular, collaborative systems. PDAs must not be used where they are not effective, at least with current technology. We showed an example: the design of a collaborative system for joint document authoring. Although many studies have been done on this subject, none has considered the inclusion of PDAs as part of the authoring process. Our design does not pretend to use these devices in most of the process, but just in those in which they are well suited. Of course, the developed tools are just that: they may be used in the creative process with the frequency or intensity the co-authors themselves consider

most appropriate. As Sharples has pointed out, writing is an open-ended, under-constrained, recursive design task, without formal transitions between states [34].

The example system we developed can certainly be improved: additional awareness, a chat and messaging tool, and blocking granularity at the paragraph level are all new features we plan to incorporate in a second version. Furthermore, experimentation is needed with actual users. Experiments may allow us to answer the question stated in the Introduction concerning PDAs relevance for building collaborative applications. In particular, we wish to have feedback from users concerning various document sizes in order to know whether or not that variable is significant for the application usefulness.

Acknowledgments

This work was partially supported by Fondecyt (Chile) grants No. 1040952 and 1030959 and by MECESUP (Chile) project No. UCH0109.

References

1. Barbara, D., Mobile computing and databases – a survey. IEEE Transactions on Knowledge and Data Engineering 11, (1), (1999), 108-117
2. Bluetooth Website, http://www.bluetooth.com
3. Buyukkokten, O., Garcia-Molina, H., Paepcke. A., Focused Web searching with PDAs. Computer Networks - the International Journal of Computer and Telecommunications Networking 33, (1-6), (2000),. 213-230
4. Chalmers, D., Sloman, M., A survey of quality of service in mobile computing environments. IEEE Communications Survey, (1999)
5. Chambers, R., Crockett, D., Griffing, G., Paris, J., A Java tool for collaborative editing over the Internet. In Proceedings of the ETCE '98, Houston, TX, (1998)
6. Chen L, Suda T. Designing mobile computing systems using distributed objects. IEEE Communications Magazine 35(2), (1997), 62-70
7. Davis, R., Landay, J., Chen, V., NotePals: Lightweight note sharing by the group, for the group. In Proceedings of the CHI'99. ACM Press, (1999), 3388-345
8. Decouchant, D., Quint, V., Romero-Salcedo, M., Structured and distributed cooperative editing in a large scale work. In Groupware and Authoring; R. Rada (Ed.), Academic Press, UK, (1996), pp.265-295
9. DeSanctis, G., Shifting Foundations in Group Support System Research. In Group Support Systems – New Perspectives, L. Jessup, J. Valacich (eds.), MacMillan Pub. Co., New York, (1993), 97-111
10. Ellis, C., Gibbs, S., Rein, G., Design and Use of a Group Editor. In: Engineering for Human-Computer Interaction. G. Cockton (Ed.), Elsevier Science Publisher, North Holland, (1990)
11. Fagrell, H., Forsberg, K., Sanneblad, J., FieldWise: A mobile Knowledge Management Architecture. Proceedings of the ACM Conference of CSCW2000, , (2000), 211-220
12. Greenberg, S., Boyle, M., Laberge, J., PDAs and Shared Public Displays: Making Personal Information Public, and Public Information Personal. Personal Technologies, USA, (1999)
13. Guerrero, L.A., Fuller D. A, Pattern System for the Development of Collaborative Applications. Information and Software Technology Vol.43, No.7, (2001), 457-467
14. Hodel, T., Dubacher, M., Dittrich, K., using database management systems for collaborative text editing European Conference of Computer-supported Cooperative Work, ECSCW CEW (2003), Helsinki, Finland
15. Held, A., Ziegert, T., Service mobility – a new feature of mobile computing. Integrated Computer-aided Engineering 6, (2), (1999), 131-142

16. Irish, P., Trigg, R., Supporting collaboration in hypermedia: Issues and experiences. In Barrett, E. (ed.) The Society of Text: Hypertext, Hypermedia and the Social Construction of Information, MIT Press, (1989), 90-106
17. Jefferson, S., Orubeondo, A., Mobile computing advances on reality. InfoWorld 22, (39), (2000), 57-59
18. Jing, J., Helal, A.S., Elmagarmid, A. Client-server computing in mobile environments. ACM Computing Surveys 31, (2), (1999), 117-157
19. Joseph A. D., Tauber J. A., Kaashoek M. F., Mobile Computing with the Rover Toolkit, IEEE Transactions on Computers Vol. 46, No. 3, (1997), 337-352
20. Kirda, E., Fenkam, P., Reif, G., Gall, H., A service architecture for mobile teamwork. In Proceedings of the 14th international conference on Software engineering and knowledge engineering, Ischia, Italy, (2002)
21. Koch.,M., Design issues for a distributed multi-user editor. Computer Supported Cooperative Work - An International Journal 3(3-4), (1995), 359-378
22. Leland, M., Fish, R., Kraut, R., Collaborative Document Production Using Quilt. Proceedings of the CSCW'88, (1988), 206-215
23. Minor, S., Magnusson, B., A model for semi (a) synchronous collaborative editing. In Proc. ECSCW'93: 3rd. European Conf. Computer Supported Cooperative Work, Milano, Italy, (1993), 219-231
24. Muñoz, M., Gonzalez, V., Rodriguez, M., Favela, J., Supporting Context-Aware Collaboration in a Hospital: An Ethnographic Informed Design, CRIWG 2003. Lecture Notes in Computer Science 2806, (2003), 330-344
25. Myers, B.A., Stiel, H., Gargiulo, R., Collaboration using multiple PDAs connected to a PC, Proc. ACM Conf. on Computer-Supported Cooperative, CSCW'98), (1998), 285-294
26. Myers, B., Using Handhelds and PCs Together. Communication of the ACM 44(11), (2001), pp. 34-41
27. Neuwirth, C., Chandhok, R., Kaufer, D., Erion, P., Morris, J., Miller, D., Flexible diff-ing in a collaborative writing system. Proc. of CSCW'92, the Conf. ofn Computer-Supported Cooperative Work, Toronto, Canada, (1992), 147-154
28. Neuwirth, C., Chandhok, R., Charney, D., Wojahn, P., Kim, L., Distributed Collaborative: A Comparation of Spoken and Written Modalities for Reviewing and Revising Documents. Proceedings of the Conference on Computer Human Interactions, CHI'94, (1994), 51-57
29. Papastavrou, S., Samaras, G., Pitoura, E., Mobile agents for World Wide Web distributed database access. IEEE Transactions on Knowledge and Data Engineering 12, (5), (2000), 802-820
30. Pacull, F., Sandoz, A., Schiper, A., Duplex: A Distributed Collaborative Editing Environment in Large Scale. Departement d'Informatique. Ecole Polytechnique Federale de Lausanne, CH-1015, (1994)
31. Pino, J.A., A visual approach to versioning for text co-authoring. Interacting with Computers 8 (4), (1996), 299-310
32. Rao, B., Minakakis, L., Evolution of Mobile Location-based Services. Communications of the ACM 46 (12), (2003), 61-65
33. Sarker, S., Wells, J.,Understanding Mobile Handheld Device Use and Adoption. Communications of the ACM 46(12), (2003), 35-40
34. Sharpies, M., Goodlet, J. S., Beck, E. E., Wood, C. C., Easterbrook, S. M., Plowman, L., Research issues in the study of computer supported collaborative writing, Sharples, M. (ed) - Computer Supported Collaborative Writing, Springer-Verlag, London, (1993)
35. Schulman, E., How to Write a Scientific Paper Journal of the American Association of Variable Star Observers 17, 130, (1988)
36. Stewart, J., Bederson, B.B., Druin, A., Single display groupware: A model for co-present collaboration. In Proceedings of CHI 99 , (1999), 286-293
37. Vizard, M., The way things work is the fundamental problem with mobile computing. InfoWorld 22, (40), (2000), 83-87
38. Zimmerman, T. G., Wireless networked digital devices: a new paradigm for computing and communication. IBM Systems Journal 38, (4), (1999), 566-574

This page intentionally left blank

Author Index

Lecture Notes in Computer Science

For information about Vols. 1–3073

please contact your bookseller or Springer

Vol. 3207: L.T. Jang, M. Guo, G.R. Gao, N.K. Jha, Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation, and Use. XI, 378 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Vol. 3186: Z. Bellahsène, T. Milo, M. Rys, D. Suciu, R. Unland (Eds.), Database and XML Technologies. X, 235 pages. 2004.

Vol. 3184: S. Katsikas, J. Lopez, G. Pernul (Eds.), Trust and Privacy in Digital Business. XI, 299 pages. 2004.

Vol. 3183: R. Traunmüller (Ed.), Electronic Government. XIX, 583 pages. 2004.

Vol. 3182: K. Bauknecht, M. Bichler, B. Pröll (Eds.), E-Commerce and Web Technologies. XI, 370 pages. 2004.

Vol. 3178: W. Jonker, M. Petkovic (Eds.), Secure Data Management. VIII, 219 pages. 2004.

Vol. 3177: Z.R. Yang, H. Yin, R. Everson (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2004. VXIII, 852 pages. 2004.

Vol. 3174: F. Yin, J. Wang, C. Guo (Eds.), Advances in Neural Networks - ISNN 2004. XXXV, 1021 pages. 2004.

Vol. 3172: M. Dorigo, M. Birattari, C. Blum, L. M.Gambardella, F. Mondada, T. Stützle (Eds.), Ant Colony, Optimization and Swarm Intelligence. XII, 434 pages. 2004.

Vol. 3166: M. Rauterberg (Ed.), Entertainment Computing – ICEC 2004. XXIII, 617 pages. 2004.

Vol. 3158:I. Nikolaidis, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. IX, 344 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004. (Subseries LNAI).

Vol. 3156: M. Joye, J.-J. Quisquater(Eds.), Cryptographic Hardware and Embedded Systems - CHES 2004. XIII, 455 pages. 2004.

Vol. 3155: P. Funk, P.A. González Calero (Eds.), Advances in Case-Based Reasoning. XIII, 822 pages. 2004. (Subseries LNAI).

Vol. 3154: R.L. Nord (Ed.), Software Product Lines. XIV, 334 pages. 2004.

Vol. 3153: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), Mathematical Foundations of Computer Science 2004. XIV, 902 pages. 2004.

Vol. 3152: M. Franklin (Ed.), Advances in Cryptology – CRYPTO 2004. XI, 579 pages. 2004.

Vol. 3150: G.-Z. Yang, T. Jiang (Eds.), Medical Imaging and Augmented Reality. XII, 378 pages. 2004.

Vol. 3149: M. Danelutto, M. Vanneschi, D. Laforenza (Eds.), Euro-Par 2004 Parallel Processing. XXXIV, 1081 pages. 2004.

Vol. 3148: R. Giacobazzi (Ed.), Static Analysis. XI, 393 pages. 2004.

Vol. 3146: P. Érdi, A. Esposito, M. Marinaro, S. Scarpetta (Eds.), Computational Neuroscience: Cortical Dynamics. XI, 161 pages. 2004.

Vol. 3144: M. Papatriantafilou, P. Hunel (Eds.), Principles of Distributed Systems. XI, 246 pages. 2004.

Vol. 3143: W. Liu, Y. Shi, Q. Li (Eds.), Advances in Web-Based Learning – ICWL 2004. XIV, 459 pages. 2004.

Vol. 3142: J. Diaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), Automata, Languages and Programming. XIX, 1253 pages. 2004.

Vol. 3140: N. Koch, P. Fraternali, M. Wirsing (Eds.), Web Engineering. XXI, 623 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004. (Subseries LNAI).

Vol. 3138: A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D.d. Ridder (Eds.), Structural, Syntactic, and Statistical Pattern Recognition. XXII, 1168 pages. 2004.

Vol. 3137: P. De Bra, W. Nejdl (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems. XIV, 442 pages. 2004.

Vol. 3136: F. Meziane, E. Métais (Eds.), Natural Language Processing and Information Systems. XII, 436 pages. 2004.

Vol. 3134: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - XP/Agile Universe 2004. XIV, 233 pages. 2004.

Vol. 3133: A.D. Pimentel, S. Vassiliadis (Eds.), Computer Systems: Architectures, Modeling, and Simulation. XIII, 562 pages. 2004.

Vol. 3132: B. Demoen, V. Lifschitz (Eds.), Logic Programming. XII, 480 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004. (Subseries LNAI).

Vol. 3130: A. Syropoulos, K. Berry, Y. Haralambous, B. Hughes, S. Peter, J. Plaice (Eds.), TeX, XML, and Digital Typography. VIII, 265 pages. 2004.

Vol. 3129: Q. Li, G. Wang, L. Feng (Eds.), Advances in Web-Age Information Management. XVII, 753 pages. 2004.

Vol. 3128: D.Asonov (Ed.), Querying Databases Privately. IX, 115 pages. 2004.

Vol.3127: K.E.Wolff,H.D.Pfeiffer.H.S.Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004. (Subseries LNAI).

Vol. 3126: P. Dini, P. Lorenz, J.N.d. Souza (Eds.), Service Assurance with Partial and Intermittent Resources. XI, 312 pages. 2004.

Vol. 3125: D. Kozen (Ed.), Mathematics of Program Construction. X, 401 pages. 2004.

Vol. 3124: J.N. de Souza, P. Dini, P. Lorenz (Eds.), Telecommunications and Networking - ICT 2004. XXVI, 1390 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004. (Subseries LNAI).

Vol. 3122: K. Jansen, S. Khanna, J.D.P. Rolim, D. Ron (Eds.), Approximation, Randomization, and Combinatorial Optimization. IX, 428 pages. 2004.

Vol. 3121: S. Nikoletseas, J.D.P. Rolim (Eds.), Algorithmic Aspects of Wireless Sensor Networks. X, 201 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004. (Subseries LNAI).

Vol. 3118: K. Miesenberger, J. Klaus, W. Zagler, D. Burger (Eds.), Computer Helping People with Special Needs. XXIII, 1191 pages. 2004.

Vol. 3116: C. Rattray, S. Maharaj, C. Shankland (Eds.), Algebraic Methodology and Software Technology. XI, 569 pages. 2004.

Vol. 3114: R. Alur, D.A. Peled (Eds.), Computer Aided Verification. XII, 536 pages. 2004.

Vol. 3113: J. Karhumäki, H. Maurer, G. Paun, G. Rozenberg (Eds.), Theory Is Forever. X, 283 pages. 2004.

Vol. 3112: H. Williams, L. MacKinnon (Eds.), Key Technologies for Data Management. XII, 265 pages. 2004.

Vol. 3111: T. Hagerup, J. Katajainen (Eds.), Algorithm Theory - SWAT 2004. XI, 506 pages. 2004.

Vol. 3110: A. Juels (Ed.), Financial Cryptography. XI, 281 pages. 2004.

Vol. 3109: S.C. Sahinalp, S. Muthukrishnan, U. Dogrusoz (Eds.), Combinatorial Pattern Matching. XII, 486 pages. 2004.

Vol. 3108: H. Wang, J. Pieprzyk, V. Varadharajan (Eds.), Information Security and Privacy. XII, 494 pages. 2004.

Vol. 3107: J. Bosch, C. Krueger (Eds.), Software Reuse: Methods, Techniques and Tools. XI, 339 pages. 2004.

Vol. 3106: K.-Y. Chwa, J.I. Munro (Eds.), Computing and Combinatorics. XIII, 474 pages. 2004.

Vol. 3105: S. Göbel, U. Spierling, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, A. Feix (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. XVI, 304 pages. 2004.

Vol. 3104: R. Kralovic, O. Sykora (Eds.), Structural Information and Communication Complexity. X, 303 pages. 2004.

Vol. 3103: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. XLIX, 1439 pages. 2004.

Vol. 3102: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. L, 1445 pages. 2004.

Vol. 3101: M. Masoodian, S. Jones, B. Rogers (Eds.), Computer Human Interaction. XIV, 694 pages. 2004.

Vol. 3100: J.F. Peters, A. Skowron, J.W. Grzymała-Busse, B. Kostek, R.W. Świniarski, M.S. Szczuka (Eds.), Transactions on Rough Sets I. X, 405 pages. 2004.

Vol. 3099: J. Cortadella, W. Reisig (Eds.), Applications and Theory of Petri Nets 2004. XI, 505 pages. 2004.

Vol. 3098: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets. VIII, 849 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004. (Subseries LNAI).

Vol. 3096: G. Melnik, H. Holz (Eds.), Advances in Learning Software Organizations. X, 173 pages. 2004.

Vol. 3095: C. Bussler, D. Fensel, M.E. Orlowska, J. Yang (Eds.), Web Services, E-Business, and the Semantic Web. X, 147 pages. 2004.

Vol. 3094: A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval. VIII, 229 pages. 2004.

Vol. 3093: S. Katsikas, S. Gritzalis, J. Lopez (Eds.), Public Key Infrastructure. XIII, 380 pages. 2004.

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3089: M. Jakobsson, M. Yung, J. Zhou (Eds.), Applied Cryptography and Network Security. XIV, 510 pages. 2004.

Vol. 3087: D. Maltoni, A.K. Jain (Eds.), Biometric Authentication. XIII, 343 pages. 2004.

Vol. 3086: M. Odersky (Ed.), ECOOP 2004 – Object-Oriented Programming. XIII, 611 pages. 2004.

Vol. 3085: S. Berardi, M. Coppo, F. Damiani (Eds.), Types for Proofs and Programs. X, 409 pages. 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3080: J. Desel, B. Pernici, M. Weske (Eds.), Business Process Management. X, 307 pages. 2004.

Vol. 3079: Z. Mammeri, P. Lorenz (Eds.), High Speed Networks and Multimedia Communications. XVIII, 1103 pages. 2004.

Vol. 3078: S. Cotin, D.N. Metaxas (Eds.), Medical Simulation. XVI, 296 pages. 2004.

Vol. 3077: F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems. XII, 386 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3075: W. Lenski (Ed.), Logic versus Approximation. IX, 205 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.