Using Groupware Tools to Extend the Organizational Memory with Collaboration Aspects

Marco Aurélio Gerosa, Hugo Fuks, Alberto B. Raposo & Luís Henrique R. G. Mitchell
Software Engineering Laboratory (LES) & Computer Graphics Group (Tecgraf)
Computer Science Department, Catholic University of Rio de Janeiro (PUC-Rio)
R. Marquês de São Vicente, 225, Rio de Janeiro, RJ, 22453-900, Brazil
{gerosa, hugo, raja}@inf.puc-rio.br, abraposo@tecgraf.puc-rio.br

Abstract

In this paper is described how computer supported cooperative work can be used to enhance project development memory, focusing on software development. The collaboration model that is presented is based upon the concepts of communication, coordination and cooperation. Each of these concepts is analyzed regarding software specification. The paper presents some examples of systems that seek to capture and refine group ideas through the process of argumentation. The paper also shows how the building of project memory can facilitate the recovery of the context and the reasoning that led to the taking of project decisions.

1. Introduction

Software specification is a process that involves a number of people from different perspectives, objectives, visions and interests and generates documentation that will comprise the project memory. Analysts, programmers, domain specialists, clients, managers and users, among others, must define the functions, requirements and characteristics of the software in question. The more the software specification is adjusted to reality, the greater will be the chances of its being accepted and put to use by end users, and the development and maintenance costs will be lower.

Crystal Clear is a software development methodology whose main principles and values are its capacity for self-adjustment during a given project, the understanding that the tools, processes and sub-products (for example, documentation) exist to help the members of the group in their work, the providing of effective support to group communication and high tolerance between members of the group. The methodology is described in the following manner: software development is seen in a practical way as a collaborative game of invention and communication whose primary goal is to deliver useful software that works, and whose second goal is to become prepared for the next game [5].

In order to make a good specification, it is not enough to just divide up the tasks among those involved and join them together at the end. The aspects to be dealt with are intimately related and dependent upon each other. Thus, a number of steps of interaction between the persons who are involved are necessary to reach a final product. This interaction must happen in a collaborative fashion; that is, there is a very strong need for communication, coordination and cooperation.

These interactions normally come about through face-to-face contact of those involved in the project. This requires that individuals go to and spend time in meetings, an effort that many clients would rather avoid.

Using the computer as the medium, one can mix together synchronous and asynchronous meetings and live meetings with virtual encounters. The asynchronous virtual meetings are flexible regarding time and location. They allow those involved to work at their own pace and in their own place of work, where they are accustomed to getting on with their tasks and where all of the materials and tools that they normally use are configured and customized for them.

That is why Software Engineering people, which are quite advanced in the development of single-user applications, looked at the field of computer supported cooperative work to find the concepts, techniques and tools that are needed for this reality. Upon using a groupware tool in order to help the cooperative specification of software, the group that is involved in the specification has at its disposal integrated tools for communication, coordination and cooperation among its members. It is through these tools that will take place the interaction between the members of the group, the sharing and management of information, the cooperative construction of documents and digital artifacts and the articulation and the organization of the group. That way the individual efforts will be directed towards the common objective and conflicts will not disturb the collaboration.

2. Collaboration in software specification

At least potentially, group collaboration can produce better results than members working individually. In a group, complementation of skills, knowledge and individual effort can occur, leading to interaction between people with complementary understanding, points of view and abilities [12]. Through collaboration, members of the group get feedback for the early identification of inconsistencies or failures in their thinking and can seek a set of ideas, information and references to help resolve the problems. The group also has more capacity to creatively generate alternatives, assessing the advantages and disadvantages of each member, selecting those that are feasible and then making decisions [27].

Despite the advantages, working in a group necessarily demands extra effort in order to coordinate its members. Without this coordination, a substantial amount of the communication effort will not be taken advantage of for the purpose of cooperation; that is, for the group to be able to operate together in a satisfactory fashion it is necessary that the commitments that are assumed in the conversations between the participants be complied with during the cooperation. Furthermore, the coordination must deal with the conflicts that harm the group.

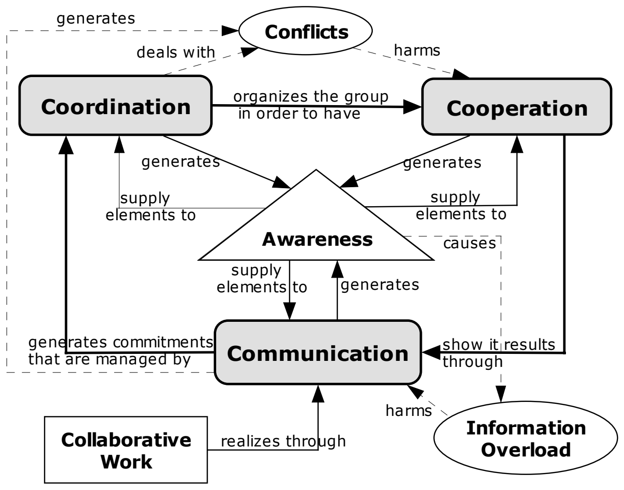

Figure 1. Diagram of collaborative work

The diagram of Figure 1 summarizes the concepts that have been discussed. This diagram is a refinement of the model presented in [11], that is based on [10]. Later we will see the diagram's main elements in greater detail and its relations with software collaborative specification. It should be remembered that despite the fact that we are separating the concepts in order to analyze them, it is not possible to consider them monolithically, since they are intimately dependent and inter-related.

2.1. Communication

To communicate is to share. The participants of a work team must communicate to accomplish tasks that are interdependent, not completely described or that require negotiation [13]. During the collaborative specification of software there is a very strong demand on communication to build common understanding and share ideas, discuss, negotiate and make decisions.

The cultural context, the domain in question and the individual knowledge influence the language to be used in the conversation. Once common language is established, people communicate without noting the expressions or the awareness elements that are used because their attention is concentrated on the purpose and the effects of the messages. However, when there is some type of confusion or problem, the structures of language and the awareness elements that are used are fully brought into focus in an attempt to repair the misunderstanding.

For there to be understanding, and for communication to fulfill its objective, it is necessary that there be understanding on the part of everyone about the use of the means of transmission and reception of data as well as the active participation of the receiver. The receiver must pay attention to the awareness elements and to the information transmitted in order to make communication possible. Furthermore, all of those involved in the communication must be aware of the structures of the language and expressions that are used. The communication is successful if there is understanding of the message being transmitted and if the content received was the equivalent to that which was transmitted in the sense of causing the expected effect.

Since the environment defines the shared information space between the individuals, it can supply additional non-verbal elements to the structure of the language used in the conversation. This simplifies verbal communication, since it is complemented by the elements that are present in the environment [15].

The members of a team that has been formed to specify software normally need to communicate in a number of different ways. The coordinator of the team must choose the proper communications tool for each situation and objective. The groupware that is used must supply a range of tools in order to offer this flexibility. Sometimes, a synchronous communications tool is most appropriate while at other moments an asynchronous tool is better. Normally, asynchronous communications tools are used when one wants to enhance reflection on the part of the participants, since they will have more time before they have to act. In a synchronous communications tool, interaction is more important giving that the response time between a participant's action and that of his/her companion(s) is short.

Also to be considered upon choosing a communications tool is the structuring of the discourse and the interface that will make it possible. Some communications tools are designed for unstructured conversation, while others favor a structured list, tree format or graph [14]. The tool's interface normally is projected for a specific type of use. Nevertheless, one must evaluate if the type of structuring and the tool's interface satisfy the group's discussion requirements at any given moment.

Some examples of communications tools that may be used are e-mail, discussion lists, forum, CSCA(Computer Supported Collaborative Argumentation) tools, voting tools, chat, electronic brainstorming, videoconferences, teleconferences and instant messaging [18].

2.2. Coordination

Conversation for action generates commitments [28]. In order to ensure compliance with these commitments and the carrying out of collaborative work, through the adding together of individual work, it is necessary to coordinate activities. This coordination organizes the group and avoids that efforts of communication and cooperation are lost. It also ensures that the tasks are carried out in the correct order, in the correct timeframe and in compliance with the restrictions and objectives [21].

In the collaborative specification of software, decisions and success depend upon the integration of the different members of a group, and thus it is important that each one understands the progress of the work being conducted by his or her companions: what was done, what remains to be done, what are the preliminary results, etc. Furthermore, it is necessary to provide information about what to do and what the others are doing. Without this context, the individuals are not able to measure the quality of their own work compared to the objectives and progress of the group, which might lead to unnecessary duplication of effort [9].

The group coordinator normally is in more need of this type of information. He or she needs to know, for example, who is or is not working, where there are conflicts of interest and what are the skills and experiences of each one of the members. A groupware environment normally offers awareness elements that help make this information available in the way that helps in coordinating the group.

However, the flow of information that goes to the coordinator must be planned very carefully. In principle, most of the information about what is happening, has happened or will happen in the group has some type of importance. Furthermore, an excess of information will make decision-making more difficult.

A failure in coordination can take place because of a problem in communication or awareness or because of differences in the interpretation of the situation or of interest. The coordinator must promote the conflict resolution among the participants in the attempt to re-establish the communication channel and the collaboration [20]. Furthermore, the coordination must deal with the conflicts that harm the group performance, such as competition, disorientation, hierarchical problems, the spreading out of responsibility, etc. [22].

2.3. Cooperation

Cooperation is the joint operation in the shared information space between the members of a group. In a virtual information space individuals cooperate by producing, handling and organizing information that is normally obtained through conversation and argumentation among the members of the group. By recording the information exchanged during communication, the group is able to count upon collective memory, which can be consulted whenever necessary to recover the history of a discussion or the context in which a decision was made.

The recording of the information is aimed to increase the understanding between people, reducing uncertainties (related to the lack of information) and the level of errors (related to the ambiguity and the existence of conflicting information) [8]. The individuals handle information and communicate in an attempt to solve misunderstandings.

The individuals also may cooperate to build and refine digital artifacts, such as documents, spreadsheets and charts. The environment can offer a number of tools for managing these artifacts, such as the recording and the recovery of previous versions, access control and permission, etc.

In the specification of software, the normal way to ensure the memory of the project is to preserve the documentation produced by the participants. This type of captured knowledge can be seen as formal knowledge. However, so-called informal knowledge — that is, the ideas, the facts, the questions, the points of view, the conversations, discussions and decisions, etc. that occur during the process and end up defining it — is difficult to capture.

This knowledge can be recorded, manipulated and linked to digital artifacts. Thus, one may investigate the thinking that led the project to a given artifact (design rationale) and subsequently check in a new context if the motives for the project decisions that were taken continue to be valid. If the new context invalidates the arguments the decision was based upon, one can redesign the artifact. When the thinking behind the decisions is not available, the identification of the motives for the design of the artifact as it is becomes more difficult. The importance of the artifact prototype in the development process should be remembered since this enables the developers to reflect upon the consequences of the specification [24][25].

There are a number of tools in the literature that use hypertext to organize group memory [26]. Some of these make it possible to use digital artifacts within a shared information space, making it possible to explain their links with the interaction that they set off and those that originated them. As a result, the context of the artifacts and the interaction are preserved, facilitating understanding and subsequent recovery. And the group memory is formed by the artifacts and by the information networks composed of the facts, hypotheses, restrictions, decisions, arguments, concept meaning, etc.

Some of these tools are designed to support software development. Beyond-Sniff [2] lets users of a software program record notes associated with its functions. Evolving Artifact [19] merges hypertext-based documentation with prototypes and, based upon the documentation, the prototype is inserted within the context of the documentation and the users can record their comments and criticisms while they interact with the software.

2.3. Awareness

Awareness, in this context, is to acquire information through the senses about what is happening and what other people are doing, even without communicating directly with them [4]. Awareness, which is inherent in humans, thus becomes central to a workgroup's communication, coordination and cooperation. Through their senses, individuals become aware of the changes caused in the environment by the action of the participants, allowing them to adjust their attitudes and acquire new information.

Awareness elements are elements of a shared space where information is transmitted that is designed to provide awareness. This information helps individuals plan their action, interpret events, forecast possible necessities and communicate in an orderly fashion. To be aware of the activities of the other individuals also is essential for the flow and for the naturalness of the work and to diminish the feelings of impersonality and distance, which are common in virtual environments.

That is why environments for the collaborative specification of software must provide awareness elements in order to provide the necessary information for the collaboration between the members of the group and individual work. Guided by their awareness, the participants may build a shared understanding and set up their work context. Furthermore, they can coordinate themselves so that the individual communication and cooperation efforts add value to the collaborative work. Some examples of awareness information that can be provided by virtual environments are: a common objective, the role of each member within a context, what to do, how to proceed, what is the impact of the actions, where to act, who is nearby, what can a companion do, what the other people are doing, the location, the origin, the importance, the relationships and the author of the objects.

3. Use of argumentative processes to capture and refine group ideas

Besides the navigation and information resources normally offered by hypertext technology, group memory must provide mechanisms to capture information. A number of different software programs carry out capturing through the recording of the argumentative processes [6] [3][1]. For this, they make use of categorization of the participant message exchange. Through categorization, upon preparing their messages the authors must select a pre-defined set of appropriate categories.

With the use of categorization and structuring of the messages, the participants provide explicit information about message content. As a result, the environment can take advantage of the previously known semantics of the categories and the explicit relations between the messages in order to organize and deduce information that will help the participants [14]. The environment may also provide reports and statistics about how the participants use the categories, which can help to understand behavior and check about compliance with tasks.

One of the first software programs to work with hypertext and message categorization was gIBIS [6], which was designed for the collaborative specification of software. It is based on IBIS (Issue Based Information Systems) [16], which preaches message categorization through *Question*, *Position* and *Argumentation*. *Question* is used to propose questions and discussion topics; *Position* is used to express an opinion and to respond to a question; and *Argumentation* is used to supply the rationale upon which the positions are based. In IBIS, besides the messages being categorized, they are linked to each other in hypertext and the links themselves are also categorized.

According to Conklin and Begeman [6], the use of gIBIS favored the resolution and understanding of problems through the use of argumentative discussion. They also reported that the discussion was more explicit and the participants were able to more clearly structure their ideas, better understanding the points of view of their colleagues and arriving at conclusions faster.

QuestMap (http://www.gdss.com), which can be seen as an evolution of gIBIS, uses some categories other than those proposed by IBIS. It also is based on the use of an argumentative process for decisions in a project.

SISCO [3] also uses hypertext and a set of IBIS-based categories. Its purpose is that the members of a group can conduct a distributed and asynchronous pre-meeting in order to understand the questions that are involved, the points of view of colleagues and their own points of view. Sibyl [17] seeks to facilitate and document the thinking that went into the making of decisions through a structured discussion based upon a set of categories and relationships denominated DRL (Decision Representation Language). Compendium [23] reconciles the structured and de-structured representation of the information.

The majority of these systems use asynchronous communication as the base for the argumentative process. It is through asynchronous communication that participants have more time to reflect before exposing their points of view or arguments. This is desirable in an argumentative process since what is wanted is clarification, refinement and debating of the ideas of the group. In order to generate ideas, synchronous communication tools can be used, such as electronic brainstorming.

In asynchronous communication one can require the authors to provide extra information about message content, since the response time is not the most important factor in this form of communication. The author can decide about the title, the priority, the category, the relationships with the other messages, etc., as well as the content in and of itself. This also contributes so that the participants may reflect before acting, since the authors of the messages must understand and think more about what they are writing in order to supply meta-information, categorizing the messages and inserting them into the structure of the pre-existing discourse.

4. Conclusion

Most of the time the design and specification of software is oriented towards artifacts [7]; that is, the emphasis is the generation, manipulation and storing of documents, requirements, spreadsheets, prototypes, user documentation, etc. However, the process through which these artifacts take on their final form is implicit (hidden in the memories of the participants, the exchange of e-mail messages, the minutes of meetings, etc.) and is lost over time, making it difficult to recover and use again [26].

With the use of computers to mediate the collaboration between those involved in a project, the information exchanged in the discussion and the decision-making process that leads to the end product can be recorded, manipulated and recovered. With hypertext or with a tree structure the relationships between the questions, the ideas, the point of view and the arguments can be explicated and separated. Thus, the authors do not need to argument in a linear fashion and the readers can clearly identify the ideas, points of view, arguments, etc.

This separation and categorizing of the different types of contributions lead the participants to organize themselves and to reflect upon how to be clear about the content they are developing, since it will be necessary to categorize it and link it to the pre-existing discourse. This improves quality of the contributions [14] and makes it difficult to use unproductive techniques to improve the acceptance of an argument: such as repetition, the use of rhetoric to distort content, the sophistication of the discourse, etc.

The use of computers also facilitates the implementation of the techniques of anonymous discussion. This can be interesting when one desires to reduce the fear of exposure, either out of inhibition or hierarchical position, and to depersonalize the contributions, de-associating them from their authors in way that they can be analyzed in an independent fashion [3]. Furthermore, the use of computers brings more flexibility to the meetings between the software development team and the clients and/or domain specialists. These individuals often are very busy and try to avoid live meetings, since they believe they are a waste of time. With the use of computers, one can mix localized meetings with distributed ones, synchronous with asynchronous, diverse tools with different functions and objectives, etc.

It should be remembered that in the initial stages of the project, when the concepts are still fuzzy, common language has not been established yet, so one must adopt a tool that offers flexibility and little structuring of information. This tool provides the liberty so that ideas will flow freely and concepts will appear along with their definitions and relations. As the process continues, the degree of communication structuring can be increased so that the concepts are refined in order for inconsistencies to be identified and so that there is an alignment of ideas within the group.

The use of CSCW techniques for the specification of software can facilitate and improve the collaboration between those who are involved in the process. However, it is necessary that the group coordinator knows the environment to be used, selecting tools to be available at the right time and in the right way so that they are appropriate to the objectives and characteristics of the group, of the work and of the accumulated information.

5. Acknowledgements

The AulaNet project is partially financed by the Fundação Padre Leonel Franca, by the Ministry of Science and Technology through its Program of Excellence Nuclei (PRONEX) grant nº 76.97.1029.00 (3366) and also through individual grants awarded by the National

Research Council to: Hugo Fuks nº524557/96-9, Marco A. Gerosa nº140103/02-3 and Luís H. Mitchell nº130225/02-9.

6. References

[1] Araujo, R. & Fuks, H. (1995), "Quorum-W: A group decision support tool for the Internet environment", Proceedings of CRIWG 95, Lisboa, Portugal, September 95, pp. 39-52

[2] Bischofberger, W.R., Kofler, T., Mätzel, K., Schäffer, B. (1994), "Computer-supported cooperative software engineering with Beyond-Sniff", UNILAB Techical Report 94.9.1, Union Bank of Switzerland, Zurich.

[3] Borges, M.R.S., Fuller, D.A., Pino, J.A. & Salgado, A.C. (1999), "Key issues in the design of an asynchronous system to support meeting preparation", Decision Support Systems 27, 1999, pp. 269-287

[4] Brinck, T. & McDaniel, S. E. (1997), "Awareness in Colaborative Systems", Workshop Report, SIGCHI Bulletin.

[5] Cockburn, A. (2001), Agile Software Development, USA, Addison-Wesley Pub Co, 2001. ISBN 0-201699-69-9

[6] Conklin, J. & Begeman, M. (1988), "gIBIS: A hypertext tool for exploratory policy discussion", ACM Transactions on Office Information Systems, Vol. 3, No. 3

[7] Conklin, J. (1989), "Design rationale and maintainability", 22nd Hawaii International Conference on System Science, 555-561

[8] Daft, R.L. & Lengel, R.H. (1986), "Organizational information requirements, media richness and structural design", Organizational Science, 32/5: 554-571

[9] Dourish, P, & Belloti, V. (1992), "Awareness and coordination in shared workspaces", Proceedings of CSCW'92, Chapel Hill NC, 1992.

[10] Ellis, C. A., Gibbs, S. J. & Rein, G. L. (1991), "Groupware - Some Issues and Experiences", Communications of the ACM, Jan 1991, vol. 34, nº. 1

[11] Fuks, H. & Assis, R.L. (2001), "Facilitating perception on virtual learningware-based environments", The Journal of Systems and Information Technology, Vol 5, No. 1, ISSN 1328-7265, Edith Cowan University, pp. 93-113

[12] Fuks, H., Gerosa, M.A. & Lucena, C.J.P. (2002), "The Development and Application of Distance Learning on the Internet", The Journal of Open and Distance Learning, Vol. 17, N. 1, ISSN 0268-0513, February 2002, pp. 23-38

[13] Fussell, S. R. et all. (1998). "Coordination, overload and team performance: effects of team communication strategies", Proceedings of CSCW '98, Chapel Hill NC, pp 275-284, 1998.

[14] Gerosa, M.A., Fuks, H. & Lucena, C.J.P. (2001), "Use of Categorization and Structuring of Messages in order to Organize the Discussion and Reduce Information Overload in Asynchronous Textual Communication Tools", CRIWG 2001, IEEE, 6-8 September Darmstadt, Germany, pp 136-141.

[15] Gutwin, C. & Greenberg, S. (1999), "A framework of awareness for small groups in shared-workspace groupware", Technical Report 99-1, Saskatchewan University, 1999

[16] Kunz, W. & Rittel, H. (1970), "Issues as elements of information systems", Working Paper no. 131, Institute of Urban and regional Development, Univ. of California, Berkeley, 1970

[17] Lee, J. (1990), "SIBYL: a tool for managing group decision rationale", Proceedings of CSCW 90, New York, pp. 79-92

[18] Long, B. & Baecker, R. (1997), "A taxonomy of Internet communication tools", Proceedings of WebNet 97, Canada.

[19] Ostwald, J. (1995), "Supporting collaborative design with representations for mutual understanding", CHI'95 Proceedings, Doctoral Consortium

[20] Putnan, L. L., & Poole, M. S. (1987), "Conflict and Negotiation", Handbook of organizational Communication: An Interdisciplinary Perspective, Newbury Park, pp. 549-599.

[21] Raposo, A.B., Magalhães, L.P., Ricarte, I.L.M. & Fuks, H. (2001), "Coordination of collaborative activities: A framework for the definition of tasks interdependencies", CRIWG 2001, September 2001, Germany

[22] Salomon, G. & Globerson, T. (1989), "When Teams do not Function the Way They Ought to", Journal of Educational Research, 13, (1), pp. 89-100, 1989.

[23] Selvin, A. et alli (2001), "Compendium: making meeting into knowledge events", Knowledge Technologies 2001, March 4-7, Austin, TX

[24] Schön, D. and Bennet, J. (1996) "Reflective Conversation with Materials", In: Bringing Design to Software, Edited by Terry Winograd, ACM Press, USA. ISBN 0-201-85491-0

[25] Schrage, M. (1996) "Cultures of Prototyping", In: Bringing Design to Software, Edited by Terry Winograd, ACM Press, USA. ISBN 0-201-85491-0

[26] Shum, S.B & Hammond, N. (1994), "Argumentation-based design rationale: what use at what cost?", Human-Computer Studies 40, pp. 603-652

[27] Turoff, M. & Hiltz, S. R. (1982), "Computer support for group versus individual decisions", IEEE Transactions on Communications, 30, (1), pp 82-91

[28] Winograd, T. (1988), "A Language/Action Perspective on the Design of Cooperative Work", Computer-Supported Cooperative Work: A Book of Readings; ed: I. Greif, Morgan Kaufmann Publishers, 1988