D-Meeting: an Object-Oriented Framework for Supporting Distributed Modelling of Software

Naoufel Boulila, Allen H. Dutoit, Bernd Brügge
Technische Universität München, Applied software engineering Chair,
{boulila, dutoit, bruegge}@in.tum.de

Abstract

The distributed development of software is increasingly common, driven by the globalization of companies and business and enabled by the improvements in communication and computing.
The distributed development of software introduce new aspects of cooperative work in which a greater emphasis is placed upon technological support of the software development process. Tools to support distributed collaboration are at present limited to general-purpose groupware involving video, audio, chat, shared whiteboards and shared workspaces. However, we are not aware of any group support framework specialized for distributed software engineering.
In this paper we focus on the development, evaluation, and refinement of the D-Meeting framework for supporting synchronous collaboration among distributed groups of developers.
A prototype called D-UML groupware supports distributed software-modelling meetings by enabling real-time sharing and manipulation of information, the capture and management of rationale knowledge, in which different groups have access to different terminals (e.g., live board, desktop machine, handheld, interconnected over fixed or wireless local area networks). D-UML shows a potential enhancement for supporting distributed meetings.

Keywords: Framework, Distributed Collaboration, Global Software Engineering, CSCW, UML, and Rationale

1 Introduction

Software development performed by traditional co-located teams is an essentially difficult task. The complex nature of the activities being carried out requires strong coordination, collaboration, and communication among developers through numerous meetings. Meetings play typically a critical role in collaboration, as it is much easier to build consensus and reach compromises in face-to-face situations. Moreover, large amount of implicit knowledge is exchanged during negotiation and conflicts resolution.
In a distributed context, where developers are dispersed across different sites and even countries, several problems arise due to the physical, social, and cultural barriers [1]. Coordination of the activities is much more difficult, informal communication among team member cannot happen, and a lot of knowledge is lost due to misunderstandings. Consequently, meetings become difficult, rare, and expensive, as participants have to schedule these meetings and travel. Hence, efforts in supporting distributed development should focus on better supporting distributed meetings.
Previous efforts to develop distributed groupware applications that are interoperable across diverse environments, both research prototypes and products, have encountered difficulties and significant cost [2].
This paper is structured as follows. Section 2 describes a general model of the software engineering meeting activities. Section 3 discusses the issues and challenges in distributed collaborative development. Section 4 introduces the D-Meeting framework. Section 5 describes D-UML, an instance implementation of the framework. Section 6 concludes with future research direction.

2 Meeting activities

Software development activities require strong team collaboration and cooperation. Much of this collaboration occurs during meetings, where designers discuss, argue, negotiate, and reach decisions via compromise and consensus. These meetings represent critical points in the project when knowledge is created, conflicts are identified and resolved and social networks are formed. Figure 1 shows the different software development activities.

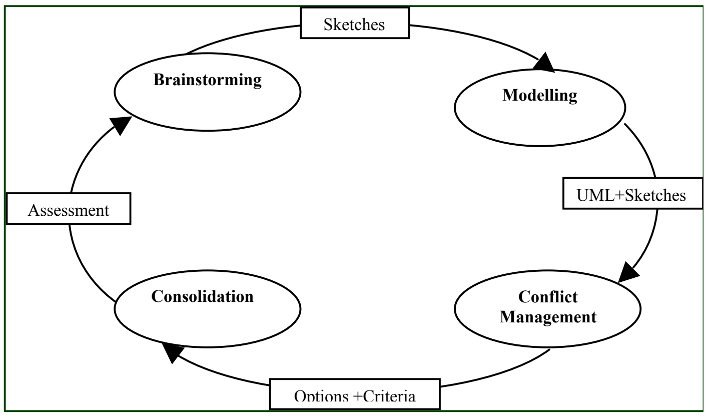


Figure 1: Meeting Activities

2.1 Brainstorming

The early stages of modelling require brainstorming and idea exploration. During this activity, team members explore a wide range of solutions using informal drawings and sketches.

2.2 Modelling

Once team members have explored a sufficient number of ideas, they detail a small number of promising ones. This is usually done with formal artiefacts, e.g. UML diagrams. However, the discussions surrounding the architectural diagrams are often non-linear and sometimes chaotic. Since this is a communication-intensive activity, everyone attempts to talk at the same time. In distributed settings, floor control is the main issue that participants will encounter during these tasks. Moreover, during the modelling activity, team members discuss several options and suggest different design views for each option.

2.3 Conflict identification and resolution

During modelling, participants raise several issues in the form of questions and propose options with different argumentations. Resolution of these issues is done after the evaluation of the pro and cons with respect to criteria. Due to different opinions, conflicts are often raised and need to be addressed. Therefore, structuring issues and the different options and criteria enables team members to quickly identify the source of the conflict and focus on new options to address them. Hence using techniques to capture and maintain rationale can be used to support this negotiation [3].

2.4 Consolidation

At the end of a meeting or after the meeting, team members review their decisions, examine open issues and eventually close them. Finally, this activity ends up with restructuring and documenting the meeting for the following modelling session.

3 Issues in distributed collaborative development

In distributed settings, supporting the collaboration of different individuals and teams over distance and time becomes a increasingly challenging issue.

Many CSCW researchers have investigated the issues associated with collaborative work. Baecker [4] provides several requirements for collaborative writing systems. Nunamaker [5] presents research in developing and using same-time/ same-place and same-time/different-place electronic meeting systems technology; Greenberg [6] describes the issues and experiences in designing and implementing group drawing tools, and Olson [7, 8] presents details of how real groups of expert designers engage in the early stage of software design meetings, Damian [9,10] presents initial studies in investigating groupware support for the interaction in Requirement Engineering processes.

We address the following issues and challenges in building the D-Meeting framework:

Informal/formal communication: knowledge is constructed through group communication and conversation. Communication is a social activity and takes different forms, formal and informal. During brainstorming, a substantial amount of informal knowledge is produced, while during consolidation activity, the generated knowledge is formalized.

Knowledge capture and management: meetings produce a substantial amount of discussion, arguments, and rationale knowledge. The outcome of the meeting, i.e., the knowledge embedded in conversations, should be reusable, organized, and shared within the workgroup to ensure that all participants are working in the same context. In particular, design rationale and its concepts, methods, and techniques will be addressed by the D-Meeting framework.

Awareness: people need to know who else is present at the meeting to guide their work. Peripheral awareness (low-level monitoring of others' activities) is pivotal factor in collaborative work [11]. The tradeoffs inherent in awareness versus privacy and in awareness versus interrupting others will be addressed.

Inflexible floor control policies: in a meeting, there are potential problems if several participants decide to access the same artifact and to manipulate it at the same time, so different systems adopt different floor control policies to determine which participant can take control of an artifact at any time. Most of applied policies are likely to cause the frustration of the users and leads to misuse or to abandon of the system. D-Meeting addresses this issue by providing several control policies.

Heterogeneity: in distributed environments, it is rarely the case that all participants share the same computing environment. A collaborative groupware must be flexible enough to span a variety of devices, from small handhelds to live-board systems.

4 D-Meeting Framework architecture

The key idea behind the D-Meeting framework is that, all the components building a meeting, i.e., Floor Control, Awareness, Design Rationale and the standard meeting activities, are formally modelled as software components that could be used as pluggable strategies (Figure 2).

4.1 D-Meeting framework components

Meeting: is the core engine of the framework, built as a mediator class that defines an interface for communicating with the *MeetingComponent*. Built as black box component that could be subclassed. The reuse of D-Meeting however, is done through composition and delegation more than inheritance. This class implements a cooperative behavior by coordinating the elements of a meeting.
MeetingComponent: represents a generic component that makes up a meeting. It communicates with *the Meeting* object whenever an event of interest occurs. By subclassing and overriding its behavior, new components could be added to the system without changing the behavior of the *Meeting* component.
Awareness: is an essential component in the D-Meeting framework as it is always required to coordinate group activities. This component cooperates closely with the *Floor Control* component. A default implementation is provided.
Modeling: supports the creation and manipulation of informal structures such as free-hand diagramming and textual annotations to communicate, as well as formal artefacts such as UML models.
Floor control: Default behaviour is provided which consist in allowing users to manipulate objects without any kind of locking relying only on socially accepted practices to ensure consistency.
Design Rationale: this component addresses an important issue such as linking the design rationale to the concrete and visible artifacts through embedding communication and history in the design process. This enables developers to collaboratively build a UML diagram and attach additional knowledge to the diagram. Techniques like QOC [12] are intensively used to support knowledge management activity.

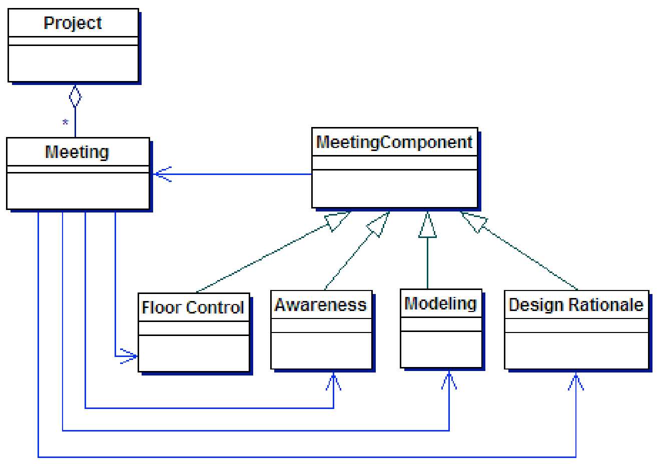


Figure 2: D-Meeting Architecture

In designing D-Meeting, we considered the flexibility of extending it with new components or behaviour without breaking its structure. The Meeting class localizes behaviour that otherwise would be distributed among the concrete *MeetingComponent* subclasses. This enforces decoupling of the different components, in that the *Meeting* class as well as *MeetingComponent* class can vary independently.

4.2 D-Meeting Components Collaboration

The D-Meeting framework provides default behaviour that can be used for experimentation with. Every component subclassing the *MeetingComponent*, implements a *RemoteObservable* interface and the *Meeting* class implements a *RemoteObserver* interface to support distributed real-time meetings. Whenever a component changes state (after a remote user initiates an action, such as adding an object, marking it, scribbling etc), it sends a notification to the *Meeting* object, which responds by propagating the effect of change to the concerned *MeetingComponents*(Figure 2,3).

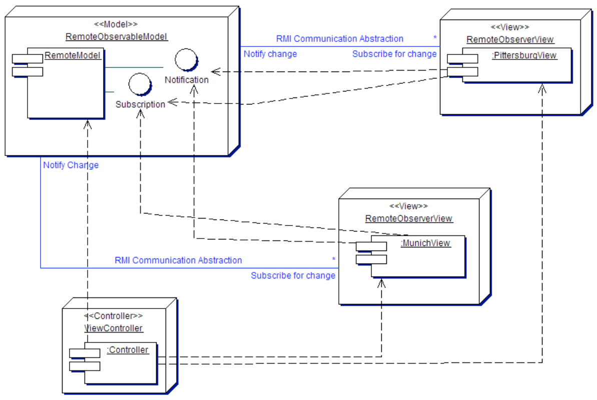
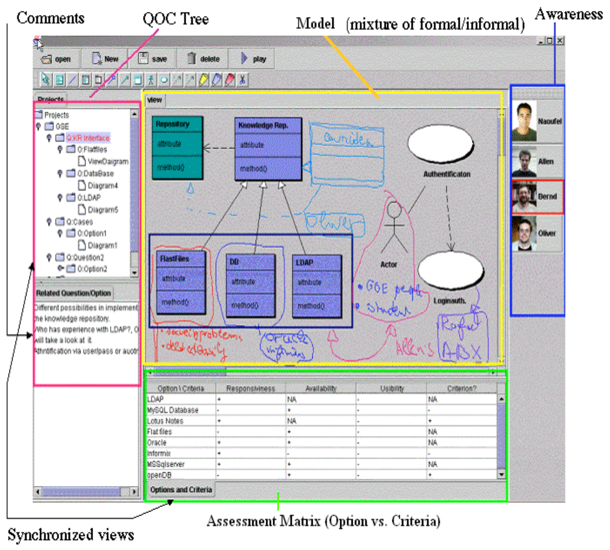


Figure 3: Distributed Architecture

A typical scenario deployment is that small group-work members are physically dispersed in different places and working around a common task to build an architectural design.

5 D-UML: an instance of the D-Meeting framework

D-UML (Distributed UML) is a Java implementation of the D-Meeting framework, based on a Distributed MVC pattern (Figure 3). The Model resides in the *Meeting* object, and can be shared across a number of views composing the D-UML groupware (Figure 4,6):


Comments
QOC Tree
Model (mixture of formal/informal)
Awareness
Synchronized views
Assessment Matrix (Option vs. Criteria)
Options and Criteria

Option \ Criteria	Responsiveness	Availability	Usibility	Criterion?
LDAP	+	NA	-	NA
MySQL Database	-	+	-	-
Lotus Notes	+	NA	-	+
Flat files	-	+	-	NA
Oracle	+	+	-	NA
Informix	+	-	-	-
MSSqlserver	+	+	-	NA
openDB	-	+	-	+

Figure 4: D-UML Groupware

- *UML view*: represents a high-level architecture of a system being modelled which is composed of use cases, class diagrams, objects etc…. The users manipulate the UML view in a similar way as a CASE tool. However the interaction is simpler, enabling the use with a touch screen or a smart board.
- *Scribbling view*: composed of free-hand drawing, handwritten annotations, scribbling and any mean that enables a communication in a team. The scribbling view and the UML view are superimposed, so that developers can use scribbling to draw attention to specific parts of the model.
- *D-UML* is collaboration-aware; several users can simultaneously interact with the application and with each other. Each user that joins a modelling session, his icons is added to the list of the participants and the moment he starts an action his icon is selected with a red square to let the all participants know who is doing what. It was a default implementation of the awareness component.
- *Rationale view*: where we describe in a structured way the Questions that are raised while a brainstorming session, the different Options that could be envisaged and the Criteria that influence a decision. To each artefact being modelled, a related rationale view is created to track its history and the knowledge behind its existence and its relation to other artefacts. D-UML supports conflict resolution activity through an assessments matrix that regroups the different options versus criteria. Criteria are used to selectively identify the acceptance or differentiation of an option. Positive assessment indicates an option satisfies a criterion. A negative assessment indicates an option hurts a criterion (Figure 5).

O/C(examples)	**Criterion1**(platform independent)	**Criterion2**(fast execution)
Option1(C++)	-	+
Option2(Java)	+	-

Figure 5: Matrix option vs. criteria

- *D-UML* provides some history features: replay allows a user that didn't attend the meeting or simply joined it late can replay the session to see what have been done so far, who did what and who were present. The replay functionality is simple and interesting; it shows all the steps the meeting went through till the current status of the brainstorming session. Undo/Redo feature which can be initiated in single as well as in distributed settings.

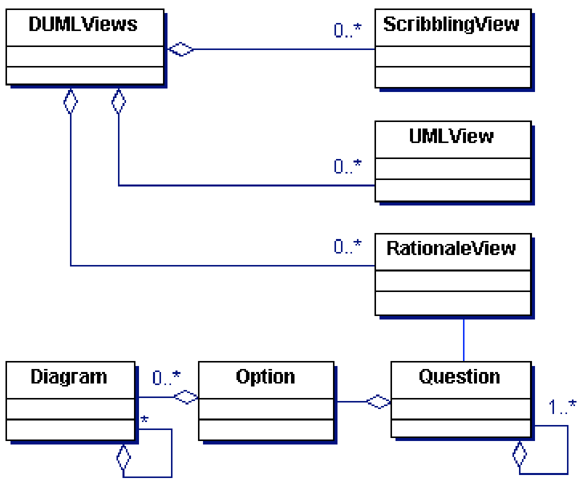


Figure 6: D-UML Views

6 Conclusion and future work

In this paper, we described a generic framework architecture to support synchronous distributed software development meetings. We described D-UML as an instance of this framework. D-UML allows us to investigate the effectiveness and usability of distributed software modelling activity. D-UML addresses the following issues:

- D-UML based on a What You See Is What We Share (WYSIWWS) paradigm in that users don't share windows or applications but rather a model. The advantages are that it provides potentially a lot of flexibility in view sharing and respecting privacy, although we visualize the same view in a given time, scrolling, zooming and moving the view don't involve the rest of the users but rather only the initiator.
- Support for integrated software modeling (UML), rationale capture and management through communication and history embedding.
- Support of replay sessions, where a user, who was not present, can replay the meeting history, as well as Undo/Redo functionalities.
- Support for distributed collaboration over formal and informal artifacts

D-UML has been refined in the context of a small group of researchers. For further usability evaluation, D-UML will be used in an experiment with two sets of groups, one group who uses D-UML and the other who does not, both trying to accomplish the same task to collaborate over a system model from different locations.
In the near future we plan to support the following features:

- Adding an interaction component that enables the use of Augmented Reality technology by augmenting desks with virtual UML objects and video streams to experience the collaboration in another way and to evaluate its usability.
- Automate event trace for automated post-mortem structuring of design rationale capture.

7 Referencess

[1] R.E. Grinter, J.D. Herbsleb, & D.E. Perry. "The Geography of Coordination: Dealing with Distance in R&D Work". ACM. 1999.

[2] I.Marsic. Data-centric collaboration in heterogeneous environments. Submitted for publication.

[3] A.H. Dutoit, B. Paech, "Rationale management in Software Engineering", In S.K. Chang (ed.) Handbook of Software and Knowledge Engineering, World Scientific Publishing, 2002

[4 R. M. Baecker, Readings in Groupware and Computer Supported Co-operative Work, Assisting Human-Human Collaboration, Morgan Kaufmann Publishers, 1993

[5] J. F. Nunmaker et al., "Electronics Meetings Systems to Support Group Work", Communication of the ACM, 34, 7, 1991.

[6] S. Greenberg et al., "Issues and Experiences Designing and Implementing Two Group Drawing Tools", in Proc. Of the 25th Annual Hawaii International Conference on The System Science, pp. 139-250, Jan 1992

[7] G. M. Olson et al, "Small Group Design Meeting: An Analysis of Collaboration", Human-Computer Interaction, 7, 347-374, 1992

[8] G. M. Olson et al, "Designing Software For A Group's Needs: A Functional Analysis of Synchronous Groupware", in User Interface Software, Edited by Bass and Dewan, John Wiley & Sons Ltd, 1993

[9] Herlea, D. (1997). A groupware system for negotiating software requirements. M.Sc. thesis, Dept of Computer Science, University of Calgary, Alberta, Canada.

[10] D. Damian. An empirical study of requirements engineering in distributed software projects: Is distance negotiation more effective? In Asian Pacific Software Engineering Conference, Dec. 2001.

[11] Dourish, Paul, and Bellotti, Victoria. "Awareness and Coordination in Shared Work Spaces." Proceedings of ACM Conference on Computer-Supported Cooperative Work, Toronto, Canada, November 1992

[12] A. MacLean, R.M. Young, V.M.E. Bellotti, and T.P. Moran. Questions, options, and criteria: elements of design space analysis. CHI'91.