2009 Fourth IEEE International Conference on Global Software Engineering

CAMEL: A Tool for Collaborative Distributed Software Design

Marcelo Cataldo
Charles Shelton
Reseach and Technology Center
Robet Bosch LLC
Pittsburgh, PA, USA
{marcelo.cataldo,charles.shelton}@us.bosch.com

Yongjoon Choi
Yun-Yin Huang
Vytesh Ramesh
Darpan Saini
Liang-Yun Wang
Carnegie Mellon University
Pittsburgh, PA, USA
{yongjoon,yunyinh,vramesh,dsaini,liangyuw}
@andrew.cmu.edu

***Abstract*—Software design activities require rich communication channels where developers can exchange information in multiple ways. It is well established that geographic distribution impacts negatively on the effectiveness of design meetings. In this paper, we present a tool for supporting virtual software design meetings. The features of the tool address four fundamental challenges identified from the literature: (1) information sharing, conflict resolution and development of consensus among geographically distributed designers, (2) availability of sufficient and organizable drawing surfaces for graphical representations, (3) developing shared understanding and managing focus during the discussion and (4) appropriate capturing and storing of all design-relevant information.**

Keywords – Collaborative Software Engineering, Distributed Software Design Meetings, Collaboration Tools.

I. INTRODUCTION

Over the past couple of decades, geographically distributed software development organizations have become pervasive. Factors such as access to talent, acquisitions and the need to reduce the time-to-market of new products are the driving forces for the increasing number of geographically distributed software development projects [12, 18]. Development work, typically, tends to be divided across locations in terms of development process stages (e.g. design in Location 1, implementation in Location 2 and testing in Location 3) or across sub-systems boundaries, for instance, Location 1 develops Sub-System or Module A and Location 2 develops Sub-System or Module B (see e.g. [10]). Architectural and high-level design activities are crucial to the development effort and even to the success of the project, particularly, in the context of distributed development organizations [2]. Although at the present time most of these design activities tend to be carried out in a co-located and face-to-face environment [8], there is no reason to believe that such a situation will continue in the future, particularly as technical or system competences are developed in different locations of a distributed development organization. However, effectively supporting distributed software design activities is a major endeavor.

Software design activities require rich communication channels where developers can exchange information in various ways [9]. Geographic distribution impacts negatively on the effectiveness of design meetings in two fundamental ways. First, participants in software design meetings typically use graphical representations or software prototypes to convey the technical properties of potential solutions [10, 32]. Access to stored contextual information, such as diagrams on a whiteboard, provide participants of meetings instant access to key pieces of information relevant to the problem and the discussion at hand. Second, individual participants will propose different software design solutions, which necessitates effective conflict resolution techniques to achieve consensus. This leads to a process of negotiation and exchange of information where all the critical aspects of each alternative must be easily shared in order to arrive at an appropriate decision.

In this paper, we present a tool, called CAMEL, for supporting collaborative software design meetings. The tool provides a number of features that address the major challenges that geographically distributed design meetings face. CAMEL provides an environment where multiple diagrams (UML and free-style) can be sketched and discussed. The tool also supports mechanisms to manage the focus of the dialogue during the meetings. Additionally, CAMEL provides mechanisms for capturing all the relevant information arising from the design meetings and permanently storing such information in a repository, allowing designers and engineers to revisit the content of the design meeting at any point in time during the development effort.

The rest of the paper is organized as follows. We first present a discussion of past research in the area of geographically distributed meetings. Secondly, we describe how CAMEL addresses the major challenges in virtual software design meetings. Finally, we present the conclusion and discuss our future research plans.

978-0-7695-3710-8/09 $25.00 © 2009 IEEE
DOI 10.1109/ICGSE.2009.16

II. THE NATURE OF DISTRIBUTED DESIGN MEETINGS

Software design meetings, particularly those focusing on early stages of a development project, represent a classic exampled of complex ill-structured tasks [24]. In order to design tools for supporting such setting in a distributed environment, it is crucial to understand the important characteristics and goals of those meetings. In this section, we summarize the relevant research on design meetings, virtual teams and computer-mediated communication (CMC) and we identify key challenges to be addressed by a tool that supports distributed design meetings.

Studies of co-located software design meetings have shown that a significant portion of the time, almost 60 percent, is spent on coordination of the discussion, clarifications, and summarizing proposed ideas [24]. Such patterns highlight the importance of face-to-face interactions in order to share information effectively and appropriately manage potential conflicts or disagreements arising from misunderstandings or competing solutions. There is a rich body of research in the virtual teams and computer-mediated communication literature that suggests that geographically distributed groups are penalized, as compared to co-located teams, when it comes to information sharing and conflict resolution. For instance, CMC reduces the number of cues that regulate flow of conversation, provide feedback, and convey subtle meanings [19, 24]. The absence of such cues reduces the amount of the information discussed [20] and makes the establishment of common ground more difficult [6]. Furthermore, difficulty in establishing common ground and limited exchange of non-verbal cues can lead to higher levels of conflict [14]. Geographically distributed teams are also less likely to reach consensus [13, 20] and more likely to share less information than co-located teams [15, 21]. All these results suggest that distributed design meetings face the following highly inter-related challenges:

Challenge #1: Information sharing, conflict resolution and development of consensus.

More recent work has focused on how designers utilize artifacts like diagrams and documents for discussing proposed solutions during software design meetings. Wu and Graham's [32] review of the literature reports that designers make extensive use of multiple diagrams for discussing proposed solutions. Dekel [8] reported that designers follow multiple organizing and drawing strategies for presenting diagrams on physical mediums such as whiteboards or canvases during design meetings. For instance, designers organized canvases in a hierarchical fashion making extensive use of portioning and versioning schemes [8]. The author also found that the size of the canvas was a key factor that influenced the level of detail represented by the diagrams. These findings highlight an important challenge that tools for design meetings should address:

Challenge #2: Sufficient and organizable "real estate" for graphical representations.

An important aspect of design meetings is achieving a shared understanding of a proposed solution. Olson and colleagues [24] found that participants in co-located software design meetings spent a third of the time in those meetings on clarifications. Designers continuously use physical means such as fingers or markers to guide or focus the discussion around the relevant graphical artifact when presenting a solution or discussing or clarifying attributes of that solution [8]. These patterns represent major challenges for virtual meetings. As described in the previous paragraphs, the constraints in communication bandwidth significantly increase the need for clarifications, and mechanisms for guiding or focusing the discussion are more limited. The problem is augmented by the fact that software designers tend to make extensive use of multiple diagrams when discussing potential solutions [8, 32]. Then, a third challenge is:

Challenge #3: Shared understanding and focus in the discussion

Dekel and Herbsleb [9] found that participants in co-located software design meetings improvise graphical representations and organize information based on the ad-hoc needs that arise as the design discussion evolves. Moreover, the authors found that the evolving patterns exhibited by the designers create significant grounding difficulties for which the participants attempt to compensate by relying on memory and contextual cues. In the context of virtual meetings, such behaviors impose important challenges in traditional collaborative or CASE tools that typically provide standardized ways of creating diagrams. Furthermore, as suggested by Dekel and Herbsleb [9], collaborative design tools need mechanisms for capturing contextual information as well as accommodating for flexible graphical representations in order to capture the evolution of the design discussion. Those findings suggest the following challenge:

Challenge #4: Capturing and storing all design-relevant information

Often developers or implementers have questions about design decisions and their rationale later in the project. It would be beneficial to maintain a repository of design artifacts that captures all of the changes in the design as it evolves during these software design collaborative meetings Such a repository would record not only the final decisions derived from these meetings, but also the entire discussion of design alternatives and explorations along with the reasons for choosing one alternative over another

III. SUPPORTING DISTRIBUTED SOFTWARE DESIGN MEETINGS

In this section, we describe a tool, called CAMEL, we designed and implemented to support distributed software design meetings. CAMEL is a combination of a client interface and a server backend that provides a number of features that address the 4 challenges outlined in Section II.

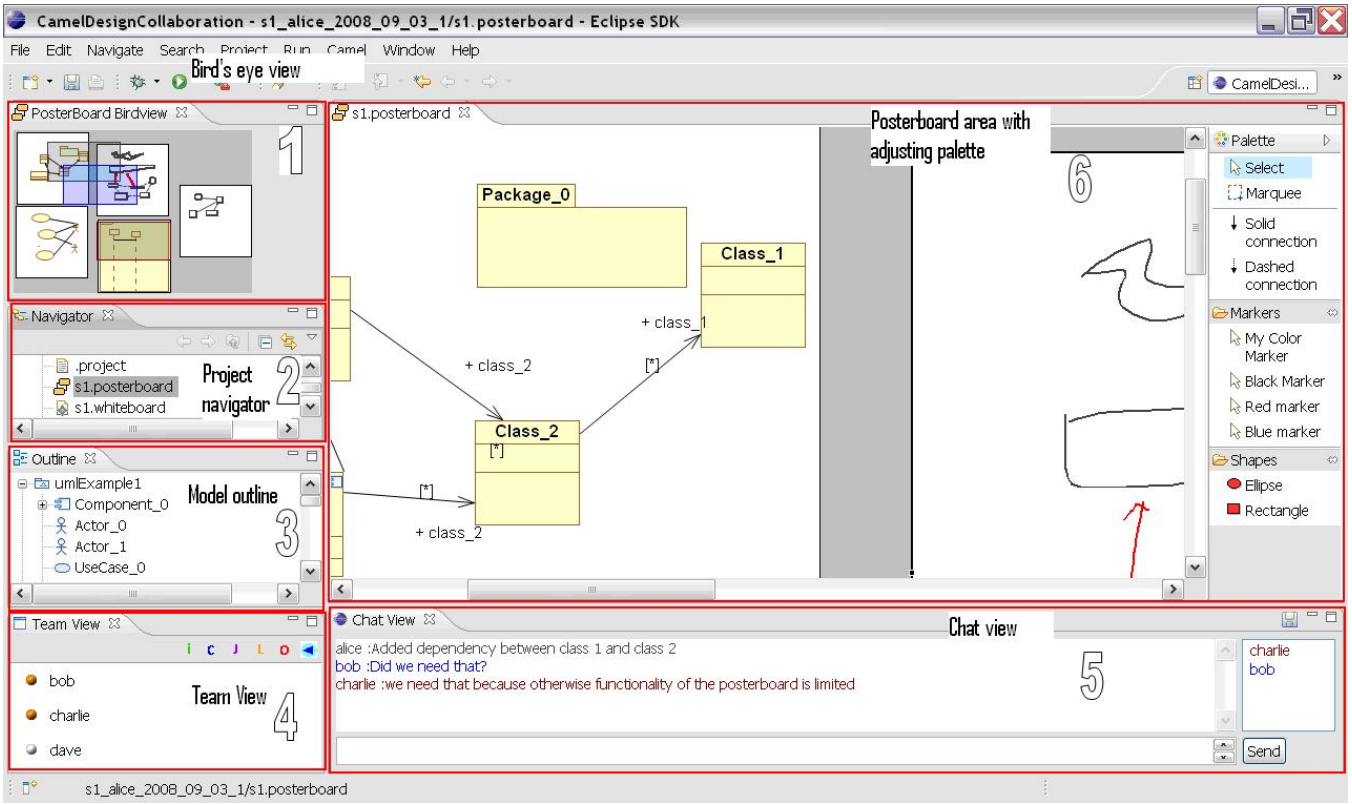

Figure 1. The CAMEL Design Tool

CAMEL is designed not to replace communication mediums such as traditional or VoIP phones or video-conferencing but to significantly improve the richness of the communication among geographically distributed software designers by recreating in a virtual environment, key characteristics of co-located design meetings. Figure 1 shows the overall appearance of the tool where the key features are marked with red boxes each one identified with a number from 1 to 6. A Team View displays the group of individuals that were invited to the meeting and their presence information (box number 4 – lower left corner). CAMEL also provides a group editor, the Chat View (box 5), where participants can take notes during the meeting.

The main feature of CAMEL is the PosterBoard (box 6 in Figure 1) which provides a logical framework for creating and editing any number of UML and free-form virtual whiteboard diagrams. On the upper left corner, Figure 1 shows the Bird's Eye view (box 1) of the posterboard. We organized the rest of this section around detailed discussions of the features of CAMEL and how they support distributed software design meetings.

A. Design Meetings as Sessions

A collaborative meeting starts with the creation of a session, which logically represents the meeting, and invitations are automatically sent to the other relevant users. A session is identified by a unique identifier, a name, and a description. Every session contains a posterboard (described later in Section 3.2), which represents the primary drawing surface. Each invited user has to join the session (Figure 2) and after successfully becoming part of the distributed meeting, the user is presented with a user workspace like the one depicted in Figure 1. In addition, each user is assigned a unique color that will identify the individual's contributions to the different elements of the user workspace such as objects he or she draws in the free-form whiteboard drawings, notes he or she adds to the chat view, and his or her viewport location in the posterboard.

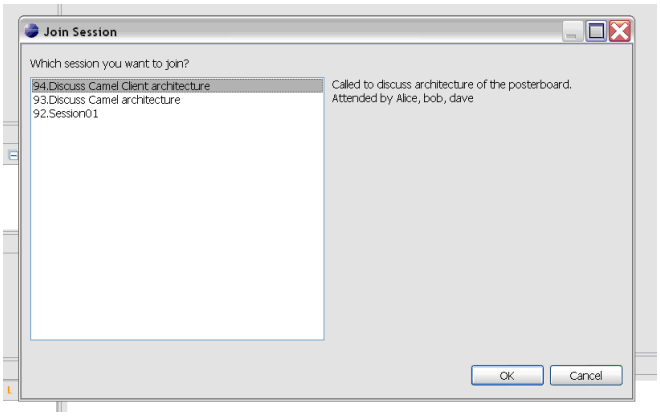

Figure 2. Join Session Dialog Box

The discussion in a session is centered on a collection of artifacts, such as UML models, hence the session creation process asks the user to provide a UML model. As a good practice in distributed development [27], we would expect such a UML model be part of a version-controlled repository. At all times during a session, each user has a local copy of the UML model, the posterboard and the whiteboard (described in sections 3.2 and 3.3, respectively) in their workspace. This allows each user to view any part of the posterboard recreating the conditions of a co-located meeting where participants rely on a collection of diagrams during the design discussions [5, 8]. This feature in particular represents an important improvement over traditional screen-sharing tools where users are constrained to watching only the portion of the screen that is being shared at any particular point in time.

Every session has the role of the facilitator who represents the user that has write privileges on the UML models. The facilitator role is transferable among users during the session. All users have write privileges on all other artifacts such as the free-form whiteboard diagrams and the chat tool.

The state of all the elements depicted in the client workspace (e.g. UML and free-form diagrams, chat tool, etc) is maintained and synchronized across all users at all times during the session. The synchronization of the users' workspaces is achieved through the collaboration server which saves the entire proceedings of a session and distributes the changes to any of the artifacts made during a session among the users' workspaces. The saved session information is also critical for the playback feature that allows a user to revisit any session after the meeting is concluded.

B. The Posterboard

The posterboard is the main drawing component of the CAMEL design tool. In any session, a posterboard consists of a collection of UML diagrams and free-form whiteboard diagrams. The UML diagrams are part of the UML model associated with the session. CAMEL supports class, sequence, state machine, activity and use case UML diagrams. The left-hand side of Figure 3 shows an example. The free-form diagrams are represented as whiteboards (see right hand-side of Figure 3) and allow the meeting participants to sketch draft diagrams of any kind. Each user's contribution to the whiteboard drawings is represented with the user's color assigned at the time the user joined the session. The posterboard also provides a palette with the necessary drawing elements for each type of diagram. For instance, Figure 3 shows the palette for the whiteboard. If the editing focuses on, for instance, a UML class diagram, the palette would change to show the appropriate set of elements for that UML diagram.

Dekel [8] found that people in co-located software design meetings often use a hierarchical canvas. The posterboard simulates that concept by creating a container canvas that allows different kinds of diagrams to be placed on it for collaborative editing. However, the posterboard itself cannot be drawn upon. When a session is a created, the user is presented with an empty posterboard. Then the users are free to drag from the Project Navigator (see Figure 1) different kinds of diagrams such as UML class, sequence, use case, activity, and state machine diagrams as well as whiteboards and drop them onto the posterboard. Once a diagram is added to the posterboard, it becomes visible to all other participants in the session. The facilitator is the only user allowed to edit the UML diagrams but no such restriction is placed on the whiteboards.

One challenge in distributed design meetings is providing the appropriate "real-estate" for representing all the necessary diagrams that designers typically use in their co-located discussions (Challenge #2 in Section 2). CAMEL's posterboard allows meeting participants to operate on multiple kinds of diagrams and juxtapose them on a single large canvas. Users can also set the appropriate zoom level depending on the size and number of diagrams.

Another challenge in distributed meetings is maintaining focus in the discussion (Challenge #3 in Section 2). The posterboard addresses this problem by providing the Bird's Eye View of the posterboard which is a thumbnail of the entire posterboard area showing the locations of the viewports of all the users in a session (Figure 4). The viewport of a user is drawn with the same color that was originally assigned to that user during session creation. This feature allows the participants to be aware of what part of the posterboard is currently visible to each user. In addition, the facilitator has the possibility to call for the attention of specific users by double-clicking on a user's viewport which will result in a signal (viewport blinks for 10 seconds) to that particular user's workspace to focus the viewport to the appropriate portion of the posterboard. Then, the facilitator can further focus the attention to a more specific part of the diagram with the mouse pointer (in the UML diagrams) or the drawing pointer (in the whiteboard).

C. The Whiteboard

Software designers usually deviate from strict UML semantics when creating early designs [9, 32], and, typically, later convert those designs to formal notations. Over the past decade, a host of tools that support quick and fluid diagramming have been proposed or for sketching UML diagrams (see [31] and [32] for a review). CAMEL provides a feature, the whiteboard, for sketching free hand drawing using a pencil element along with simple geometric shapes such as rectangles, circles and diamonds. Each participant can draw using their assigned color or choose a marker with a specific color from the tool palette (see right side of Figure 3).

The purpose of adding a whiteboard diagram to the posterboard is two-fold. First and consistent with past research, a distributed meeting support tool should facilitate early design by providing a semantic-less drawing surface. Secondly, such a tool should provide for a less constrained discussion environment where all participants are free to contribute at any point in time during a meeting.

The posterboard allows the creation of multiple whiteboards that can be used in different ways. Users can

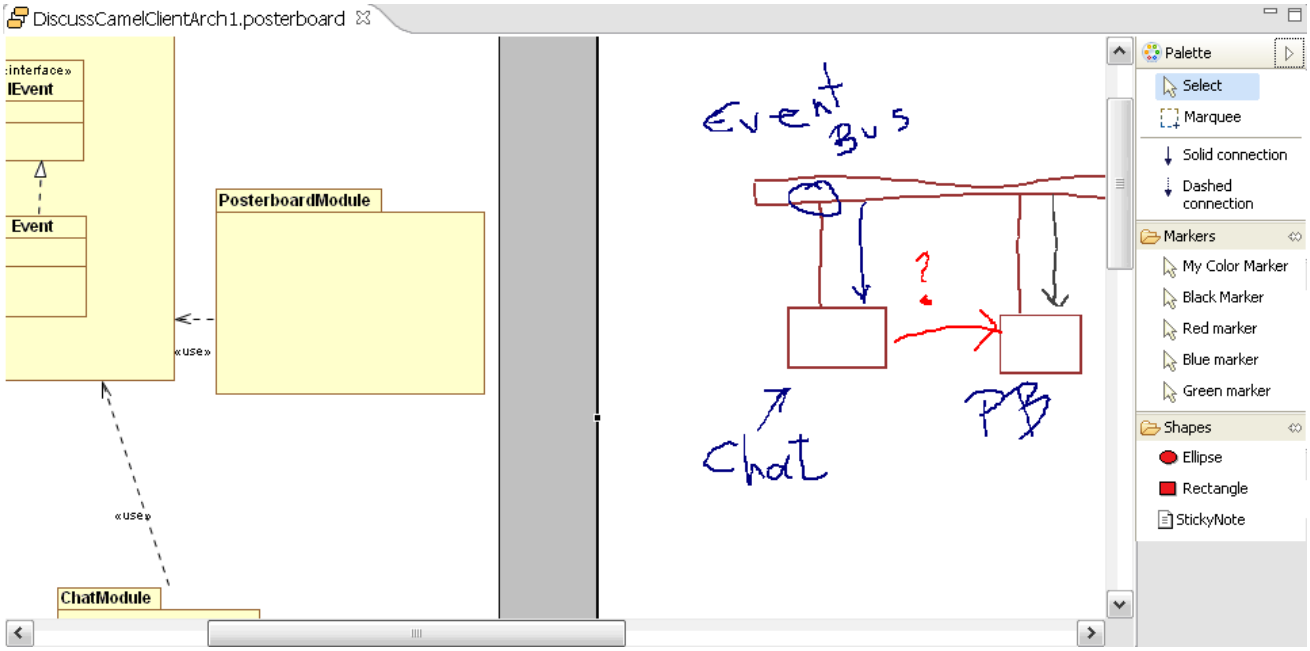

Figure 3. The PosterBoard.

discuss multiple designs by creating multiple whiteboards, one for each competing design proposal. On the other hand, participants could create one or more whiteboard diagrams for each UML diagram to save associated contextual information related to those diagrams. Dekel [8] found that software designers in co-located meetings did extensive usage of artifacts like post-it notes to capture notes about particular UML diagrams. CAMEL supports a post-it type of object, called a StickyNote, in the whiteboards.

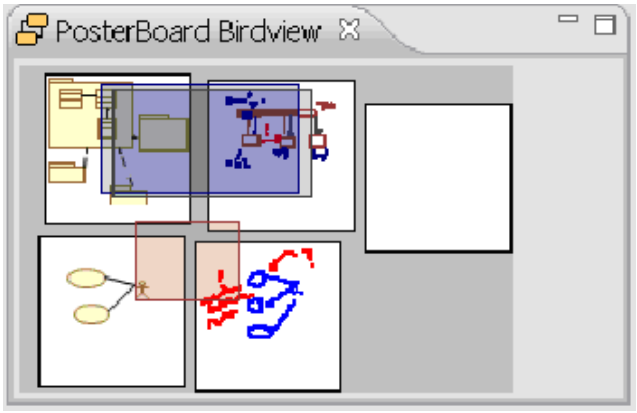

Figure 4. Bird's Eye View of PosterBoard.

Another set of patterns exhibited by co-located designers is the use of contextual information to capture an issue that is revisited later during the meeting [8, 9]. In that context, people can record the issues that arise during a discussion on the whiteboard and later revisit them. In order to better support such behaviors, the whiteboard is not implemented as a bitmap but as a model much like a UML model. This approach allows easy scaling of shapes, deletion of objects instead of having to erase them, renaming of objects and the capability to add an existing whiteboard to a new session. If required, the whiteboard can also be exported as an image file in JPEG format.

A feature that many UML sketching tools provide is automatic conversion of sketches into formal UML diagrams [7, 31, 32]. We decided initially not to include such functionality in CAMEL's whiteboard because, in the context of our research questions, converting of sketches into UML diagrams is not critical to supporting distributed design meetings; rather it is a productivity enhancement feature. However, we plan to add such functionality as we continue developing our research program as we intend to empirically evaluate CAMEL in real world software development projects.

D. The Group Editor

Shared workspaces are very important support tools for virtual meetings and mechanisms, such as a group editor, that allows all participants to contribute content or take notes collectively, are very effective tools for facilitating the flow of the virtual meetings [25]. CAMEL provides such functionality through the *Chat Tool*. The implementation of the feature resembles a chatting tool that allows all participants to take notes and contribute comments during the meeting. Although actual interaction among the meeting participants could take place through the chat tool, we envision its use primarily as a group editor.

As part of a related research project, we interviewed 42 software architects, software designers and developers, and they all indicated that they prefer voice interaction (e.g. by phone or VOIP-phones) over chatting or IM tools when discussing designs. In fact, the interviewees indicated that when they participate in a distributed meeting, they typically

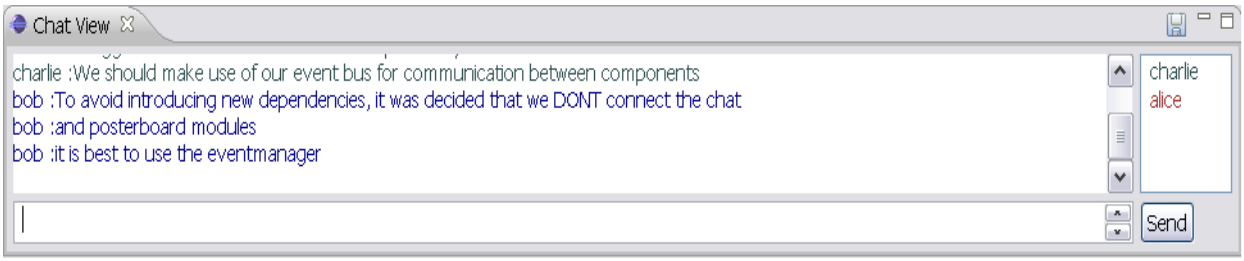

Figure 5. The Chat View

use the phone in combination with a tool such as Netmeeting [22] or Interwise [16]. Thus, our intention for the chatting tool is for the users to take notes during the session, particularly, focusing on capturing the rationale behind specific design decisions. Using the chat view in the envisioned way represents a mechanism for capturing relevant design information and, since sessions are recorded, that information could be later retrieved for a multitude of purposes (Challenge #4 in Section 2).

Figure 5 depicts how users can leverage the group editor for taking notes. The color used for the group editor is consistent with the color assigned to the user when he or she joined the CAMEL session, which helps to identify messages from each user and correlate them to contributions in the whiteboard. The group editor also provides participants with quick feedback regarding who has joined the session, and the color assigned to those users (see right-hand side of Figure 5). Moreover, the group editor helps those users who join a session later to catch up with the current state of the discussion. The following section discusses the details of how the CAMEL design tool uses the group editor to facilitate the playback function.

E. Recording and Playback of Design Meetings

CAMEL records sessions in order to facilitate capturing, persistently storing, and retrieving all relevant design information (Challenge #4). The recording is centered on capturing and storing the changes to the artifacts (e.g. UML diagrams, whiteboards, chat view, etc). This approach is significantly more efficient than capturing entire snapshots of the artifact on a periodic basis and it has the additional benefit of providing the basis for the synchronization of the information on the users' workspaces. In addition, CAMEL allows a user to playback what transpired during a session. Using the playback feature (see Figure 6), users can view the proceedings of an older session timeline and navigate either forwards or backwards through the sequence of changes to the artifacts (see buttons in the top-right corner of Figure 6). In order to facilitate the association of changes to the models with contextual information captured in the chat view, CAMEL interleaves chat messages with model changes. Thus, the navigation is enhanced by displaying the changes and pausing on a chat comment, allowing the users to read the contents of the chat view and connect them to the changes depicted in the posterboard.

IV. Implementation Challenges

A critical requirement for CAMEL is to synchronize the information displayed to the distributed user work spaces in a sufficiently short amount of time such that it does not disrupt the flow of the meeting. Several design decisions were made in order to satisfy this requirement and the rest of this section describes them in detail.

A. Distributed System Architecture

The camel design tool uses a client-server approach where each user represents a client and a central collaboration server coordinates communication between those clients. We chose this architecture for two reasons. First, in order to facilitate playback, we need to store complete session information in a repository that is accessible after the session ends. Second, in the case that a user faces a network outage, we need a place to cache all the session information for this particular user.

A key quality attribute was the time it took to capture changes and distribute them to the user workspaces. The approach selected to make the protocol as efficient as possible was to use rich clients capable of drawing UML diagrams. Then, all the editing and rendering of the diagrams on the shared posterboard would be done by the clients. The editing party only sends out the result of editing after it has been completed locally. The receiving parties decide how to render the result onto their local screen resolution, window size, zoom in factor, and time zone settings, all without intervention from the server.

The rich client is based on an open source UML editing tool named Papyrus [27]. The Papyrus tool builds on the Eclipse Modeling Framework (EMF) and Eclipse Graphical Editing Framework (GEF) technologies. EMF is used to store all UML related components, relationships, and their graphical representations, and it provides object persistence and change notification mechanisms. GEF groups editing operations into commands, and serializes them through a command stack.

The role of the collaboration server is to provide participant authentication, message relaying, information storage and token locking services during the session. In addition, the server stores all changes, including chat messages and model differences, into the server database as session commands, and the client polls from the server multiple times a second.

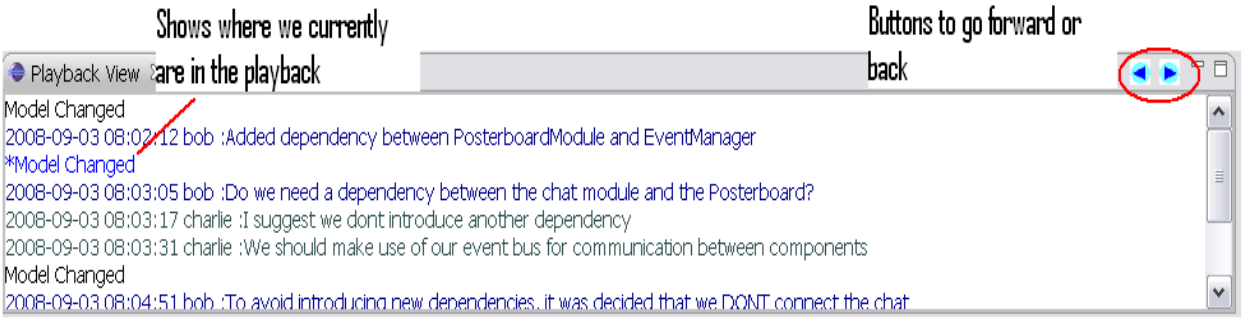

Figure 6. The Playback View.

B. *Distributing Change Information*

In order to satisfy the requirements in terms of response time to updates in the models, we opted for distributing the lower level model differences between the "writer's" model and the previously shared one. Since EMF constrains the type of operations you can do to an EMF object to simple primitives, handling this small number of operations correctly allowed the server to efficiently exchange changes to any artifact based on EMF. The posterboard and the whiteboard are both models constructed with EMF, and can be synchronized across the users' workspaces using the same core functionality in CAMEL.

We conducted several scalability and performance tests and all results showed update times that were barely noticeable by the users. In particular, we conducted a test involving four users: two users were located in Pittsburgh, Pennsylvania, one user was in Anchorage, Alaska, and the forth user was in Stuttgart, Germany. All participants reported no significant delay (approximately 1 second) in seeing new messages in the chat view or seeing changes in the posterboard diagrams.

V. DISCUSSION

In this paper, we have presented a tool, CAMEL, for supporting distributed software design meetings. The collection of features incorporated in CAMEL addresses a set of four critical challenges identified from the literature that distributed design meetings face. In particular, CAMEL complements traditional communication means, such as phone or video conferencing, by providing a rich communication environment through features like the posterboard, the whiteboard, the chat view and the Bird's Eye view. Table 1 summarizes the mapping of CAMEL features to the challenges described in section II. For instance, a collection of features facilitates information sharing during the meetings and contributes to reducing potential conflicts that might emerge when designers evaluate competing solutions to a particular problem (Challenge #1). In addition, the posterboard provides an almost unlimited surface for UML diagrams and free-form graphical representations in an organizable way (Challenge #2), facilitates capturing contextual information through the use of whiteboard diagrams combined with the chat view and allows managing the flow and focus of the discussions through the posterboard's Bird's Eye view (Challenge #3). Finally, the recording and playback functionality provides the basic infrastructure for storing and retrieving any type of design information (Challenge #4).

TABLE I. MAPPING OF CHALLENGES TO CAMEL FEATURES

Feature	Chal.#1	Chal.#2	Chal.#3	Chal.#4
Posterboard		✓		
Whiteboard	✓	✓	✓	
Team View	✓			
Bird View	✓		✓	
Group Editor	✓		✓	✓
Session Recorder				✓
Session Playback				✓

A. *Limitations*

The goal of the CAMEL tool was to address specific and critical challenges in conducting software design meetings in a distributed environment. Certainly there are other challenges, more technical in nature, that can have an important impact on the effectiveness of the meetings. Those particular issues were beyond the scope of the work reported here. For instance, the ability to draw diagrams very quickly is hindered when using input devices such as mouse or keyboards. In this regard, several of the authors utilized tablet PCs with pen-based input devices during testing and demonstration of the tool with great success. We found that pointing to objects in the diagrams and drawing free-form diagrams was almost as natural as utilizing traditional markers and whiteboards. However, we encountered some difficulties performing drag and drops with the pen-based devices.

Another limitation of this study is the lack of rigorous empirical evaluation and validation of the tool. Later in the discussion section, we present a detailed description of the plans we devised to evaluate the contributions the tool would have in a distributed meeting setting as we argued in this paper.

B. Related Work

Over the years, many tools that provide support for collaborative software design have been proposed. Wu and Graham [32] reviewed the literature and identified 14 different research efforts proposing tools that they organized in three groups: Informal Case Tools, Electronic Whiteboards and Shared Drawing Tools. CAMEL incorporates elements from all those three types of tools (e.g. the whiteboard which is shared among the meeting participants and a UML drawing environment). Whitehead [31] also reviewed the landscape of collaborative tools for design and he identified 13 commercial and research tools. More recently, closely related tools to CAMEL have been proposed in the literature. Boulila and colleagues [3, 4] developed a framework for supporting distributed software modeling and utilized it in the realization of two tools, GroupUML [4] and D-Meeting [3]. Those tools incorporate a shared drawing space for UML and free-form diagrams, a chatting tool and an awareness mechanism that indicates which members are participating.

CAMEL provides a number of novel features that differentiates it from the vast landscape of tools proposed in the past. First, CAMEL focus primarily on facilitating and enhancing the dynamic of distributed design meetings by proposing features that resemble how engineers conduct co-located design meetings. For example, engineers utilize multiple drawing surfaces for semantic-enforced diagrams (e.g. UML) and semantic-less diagrams (e.g. free-form) [9]. CAMEL's posterboard and whiteboard recreate that environment. Moreover, CAMEL's Bird's Eye View allows a meeting participant to explore the landscape of diagrams simultaneously making the other meeting participants aware that such actions are taking place, hence, facilitating the identification of loss of focus in key points during meeting discussions. Another novel feature of CAMEL is the Recording and Playback of sessions which provide the mechanisms for storing and revisiting the content discussed and generated during the meetings.

C. Empirical Evaluation of the Tool

An important next step in our research project is the empirical evaluation of the CAMEL tool. We have designed two complementary ways of empirically evaluating and validating the tool. First and as part of a larger project studying the challenges of geographically distributed software development, we are working closely with a business unit to carry out a field experiment where architects and designers will use the tool for their design meetings related to the development of the next generation of an embedded system for hydraulic applications. There will be 9 architects and designers involved in the experiment who are distributed in 5 development locations (1 in India and 4 in Western Europe). We have interviewed 6 of those 9 individuals using a semi-structure interview approach to understand their work environment and the way those individuals approach software design. We have also developed a questionnaire that assesses a wide range of topics such as architecture, processes, collaboration and coordination across sites, tool environment, and work environment as well as demographic data. The questionnaire will be administered to the entire project which includes the 9 architects and designers. We have also done a seminar where we presented the tool and we are planning two training seminars before the project starts.

The second approach to evaluate and validate the tool involves controlled laboratory experiments. In such a setting, we are planning to carry out experiments were we compare three conditions: face-to-face meetings, conference calls with a shared presentation (e.g. NetMeeting) and conference calls with CAMEL. The first two conditions resemble traditional settings in corporate development organizations. We have developed pre- and post-experiment questionnaires. The pre-experiment questionnaire collects demographic information, in particular, experience related to software design, use of collaborative tools, and involvement in design meetings. The post-experiment questionnaire represents a collection of self-assessment questions related to the 4 challenges the CAMEL tool addresses. The questionnaires have been tested in mockup experiment runs using 5 graduate students not aware of the research questions. The experiments will consist of 3-person groups randomly selected from the pool of participants. We are planning to use graduate-level students attending professional programs (e.g. Master of Software Engineering) at Carnegie Mellon University. Those students are required to have at least 2 years of professional experience which increases the likelihood of the study population to be similar to those found in corporate development organizations. The experiment will involve a design task that the groups will tackle in two meetings of approximately 2 hours each. Each group will also have an initial meeting where they familiarize themselves with the features of the tool.

We consider that these two validation paths will provide us the opportunity to thoroughly examine the features of the CAMEL tool and evaluate the benefits it can bring to distributed software development projects.

D. Future Research Directions

There are a number of areas in which the tool could be extended. The rest of this section discusses two areas for enhancement that we consider to have the potential to be quite beneficial in the context of distributed design meetings.

1) Architectural Conformance Checks

A critical issue in distributed software design is the problem of "architectural drift" [28, 30]. When the design of individual software components and subcomponents is assigned to remote teams, they should adhere to the constraints and interfaces specified in the software architecture. The lack of conformance to the architectural specifications can potentially cause severe integration problems when the finished components are assembled into the software system.

Thus, one enhancement we envision for the CAMEL tool is to enable an architect to specify architectural constraints in the form of UML model constraints and have the tool automatically check whether a UML diagram violates those constraints. Such a conformance check feature could be

provided both in an online mode that continuously checks the changes in the model and warns the designer when a violation occurs, and in an offline mode during session playback or viewing the UML model outside of a collaboration session to highlight all the conformance violations of a UML model.

Such a capability for architectural conformance checks in UML diagrams would be a crucial aid to software design, particularly in the context of large-scale systems where the designers and architects might face challenges keeping up with all the constraints imposed by the functional and non-functional attributes of a system's architecture.

Furthermore, the conformance checks could be used during an online collaboration session to drive negotiations between the software architect and component designers for relaxing or changing architecture constraints. For example, if software Component A needs access to a certain data element that is not exposed by the interface of Component B due to an architectural constraint, Component A's designer could have a collaboration session with the architect and Component B's designer to negotiate changing the architecture. In the CAMEL session the architect and component designers could view the UML class diagram of the architecture, and Component A's designer could argue why the change is necessary in the diagram by making temporary changes to the model and showing their results. The tool would report an architectural constraint violation when the model is changed, and the architect could decide whether to change the constraint in the UML model. The architect and the component designers could see the effects on the system and their components, and negotiate whether or not to change the architecture constraints and/or the component designs.

2) Supporting Decision Making

An important portion of design meetings is dedicated to discussing and clarifying the merits of different solutions [24]. The discussion and evaluation of alternative design proposals typically ends with the selection of a particular solution. Research in decision making has shown that pressure to conform to the views of particular individuals participating in the meeting might negatively impact the quality of the decision [1]. In the context of distributed meetings, anonymity has been found to reduce the need for conformance [23] and individuals are more likely to evaluate alternatives on their merits than on the status of the person that presented the information or supported the alternative [17, 33]. Hence, extending CAMEL with voting mechanisms that emphasize anonymity and assist participants in the selection of competing solutions represents an opportunity to empirically explore the contribution of GDSS-type features in product design meetings.

3) Enhancing Meetings with Auxiliary Information

Informal or impromptu conversations are also an avenue to discuss design decisions in co-located projects [11]. Such interactions are quite limited in a distributed environment, typically circumscribed only to some instant message exchange or similar type of communication medium if such technology is available in the organization. Future research could explore the integration of lightweight communication mediums with the recorded sessions that CAMEL creates. In such a way, a collaborative tool environment could leverage the benefits of those lightweight information exchange mechanisms with the design-specific context that a recorded meeting in CAMEL provides. These integrations could be articulated in many different ways. For instance, StickyNotes could be added to a recorded session as part of a playback or post-meeting editing session. Alternatively, those StickyNotes could represent linkages to other repositories of data (e.g. chat archives or discussion forums). Then, such integrations could represent an opportunity to enhance the content created during the design meetings with additional information collected in informal discussions.

ACKNOWLEDGMENTS

The authors would like to thank the valuable feedback to earlier versions of this work from James D. Herbsleb, Felix Bachman and Mel Rosso-Llopart.

REFERENCES

[1] Baltes, B.B., Dickson, M.W., Sherman, M.P., Bauer, C.C and LaGanke, J.S. Computer-Mediated Communication and Group Decision Making: A Meta-Analysis. *Org. Behavior and Human Decision Processes*, 87, 1 (2002), pp. 156–179.

[2] Bass, M., Bass, L., Herbsleb, J.D. and Cataldo, M. Architectural Misalignment: an Experience Report. *In Proceedings of the 6th Working International Conference on Software Arch. (WICSA'07)*, 2007.

[3] Boulila, N. Dutoit, A.H. and Bruegge, B. D-Meeting: an Object-Oriented Framework for Supporting Distributed Modelling of Software. In *Proceedings of the International Workshop on Global Software Developmentence,* 2003.

[4] Boulila, N. Dutoit, A.H. and Bruegge, B. Group Support for Distributed Collaborative Concurrent Software Modeling. In *Proceedings of the Automated Software Engineering Conference (ASE'04),* 2004.

[5] Cherubini, M., Venolia, G. Deline, R. and Ko, Andrew. Let's Go to the Whiteboard: How and Why Software Developers Use Drawings. In *Proceedings of the International Conference on Human Factors in Computer Systems (CHI'07)*, San Jose, CA, USA, 2007.

[6] Clark, H. and Brennen, S. Grounding in Communication. In *Perspectives on Socially Shared Cognition*, Resnick, L., Levine, J., and Teasley, S., eds. American Psychological Association, Washington, D.C, 1991.

[7] Chen, Q., Grundy, J. and Hosking, J. An E-whiteboard Application to Support Early Design-stage Sketching of UML Diagrams. In *Proceedings of the IEEE Symposium in Human Centric Computer Languages and Environments*, 2003.

[8] Dekel, U. Towards distributed software design meetings: what can we learn from co-located meetings? *ACM SIGSOFT Software Engineering Notes*, 30, 4 (July 2005), pp. 1-7.

[9] Dekel, U. and Herbsleb, J.D. Notation and Representation in Collaborative Object-Oriented Design: an Observational Study. In *Proceedings of the 22nd Annual Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'07)*, Montreal, Quebec, Canada, 2007, pp. 261-280.

[10] Grinter, R.E., Herbsleb, J.D. and Perry, D.E. The Geography of Coordination Dealing with Distance in R&D Work. In

Proceedings of the Conference on Supporting Group Work (GROUP'99), Phoenix, Arizona, 1999.

[11] Herbsleb, J.D. and Mockus, A. An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 29, 6 (2003).

[12] Herbsleb, J.D.and Moitra, D. Global Software Development. *IEEE Software*, March/April (2001), pp. 16-20.

[13] Hiltz, S.R., Johnson, K., and Turoff, M. Experiments in Group Decision-Making: Communication Process and Outcome in Face-to-Face Versus Computerized Conferences. *Human Communication Research*, 13, 2 (1987).

[14] Hinds, P.J. and Bailey, D.E. Out of sight, Out of Sync: Understanding Conflict in Distributed Teams. *Organization Science*, 14, 6 (2003).

[15] Hollingshead, A.B. The Rank-Order Effect in Group Decision Making. *Org. Behavior and Human Decision Processes*, 68 (1996), pp. 181-193.

[16] Interwise. URL access on September 2nd, 2008. http://www.interwise.com/Interwise_connect.html

[17] Jessup, L. M., Connolly, T., & Galegher, J. The effects of Anonymity on Group Process in an Idea-generation Task. *MIS Quarterly*, 1990, pp. 313–321.

[18] Karolak, D.W. *Global Software Development: Managing Virtual Teams and Environments*, IEEE Computer Society, 1998.

[19] Kiesler, S., Siegel, J., and McGuire, T.W. Social Psychological Aspects of Computer-Mediated Communication. *American Psychologist*, 39, 10 (1984).

[20] McGrath, J.E. and Hollingshead, A.B. *Groups Interacting with Technology*. SAGE Publications, 1994.

[21] McLeod, P.L., Baron, R.S., Weighner Marti, M., and Toon, Y. The Eyes Have It: Minority Influence in Face-To-Face and Computer-Mediated Group Discussion. *Journal of Applied Psychology*, 82 (1997), pp. 706-718.

[22] Netmeeting. URL access on September 2nd, 2008. http://www.microsoft.com/downloads/details.aspx?familyid=26c9da7c-f778-4422-a6f4-efb8abba021e&displaylang=en

[23] Nunamaker, J. F., Jr., Dennis, A. R., Valacich, J. S., Vogel, D. R., & George, J. F. Electronic Meeting Systems to Support Group Work. *Communication of the ACM*, 34 (1991).

[24] Olson, G.M., Olson, J.S., Carter, M.R. and Storrsten, M. Small Group Design Meetings: An Analysis of Collaboration. *Human-Computer Interaction*, 7 (1992), pp. 347-374.

[25] Olson, G.M. and Olson, J.S. Distance Matters. *Human-Computer Interaction*, 15, 2 & 3 (2000), pp. 139-178.

[26] Olson, G.M. and Olson, J.S. Human-Computer Interaction: Psychological Aspects of the Human Use of Computing. *Annual Review of Psychology*, 43 (2003), pp. 491-516.

[27] Papyrus. http://www.papyrusuml.org. URL access on September 2nd, 2008.

[28] Perry, D. E. and Wolf, A.L. Foundations for the Study of Software Architecture", *ACM SIGSOFT Software Engineering Notes,* 17, 4 (1992), pp. 40-52.

[29] Sangwan, R., Bass, M., Mullick, N., Paulish, D.J. and Kazmeier, J. *Global Software Development Handbook*, Auerbach Publishers, 2006.

[30] van Gurp, J. and Bosch, J. Design Erosion: Problems and Causes. *Journal of Systems and Software*, 61, 2 (2002), pp. 105-119.

[31] Whitehead, J. Collaboration in Software Engineering: A Roadmap. *Future of Software Engineering (FOSE '07)*, 2007.

[32] Wu, J. and Graham, T.C.N. The Software Design Board: A Tool Supporting Workstyle Transitions in Collaborative Software Design. In *Lect. Notes in Computer Science 3425*, edited by R. Bastide, P. Palanque and J. Roth, 2005, pp. 363-382.

[33] Zigurs, I., Poole, M. S., & DeSanctis, G. L. A Study of Influence in Computer-Mediated Group Decision-Making. *MIS Quarterly*, 1988, pp. 625–644.