Distributed Synchronous Collaborative Modeling Supporting System for UML Diagrams

Dongmei XU† Jun KUROGI‡ Yoshihide OHGAME‡ Atsuo HAZEYAMA‡

‡Graduate School of Education, Tokyo Gakugei University
†Department of Information Education, Tokyo Gakugei University

Abstract

With the advancement of network technology and globalization of business, distributed software development has become in common. Along with this trend, distributed synchronous collaborative modeling supporting systems were proposed. This study points out two issues traditional systems have; they don’t relate conversations with model diagrams, and they can’t replay histories of modeling by associating contents of conversations with elements of model diagrams. The authors developed a distributed synchronous collaborative modeling supporting system for UML diagrams, called Libra-on-Chat, which solved the issues. As the result of experiment, the system allows participants to focus on elements in a diagram under discussion. It also enables developers to grasp design rationale in review.

1. Introduction

With the advancement of network technology and globalization of business, distributed software development has become in common. In case of collaborative software development under geographically dispersed environments, lack of informal communication among team members causes difficulty of collaborative development [1]. In particular, work of upstream phases such as requirement analysis and/or design is communication intensive. The outcomes of those phases are usually represented using diagrams. In recent object-oriented software development, UML (Unified Modeling Language)[7] are used, which have several types of diagrams. On the other hand, face-to-face meetings took expensive costs.

Along with this trend, distributed synchronous collaborative modeling supporting systems were proposed. However traditional systems have the following issues for artifact creation support:

(1) Modeling tools are not associated with communication tools [2]

D-UML is a synchronous modeling support system for UML diagrams. Although D-UML provides a function of drawing free-hand annotation in “shared model viewer”, it does not provide a function of reviewing of the annotations. Although a comment tool is provided, comments are not associated with model elements. Some commercial UML tools are provided (for example, Elapiz [3]). Elapiz provides two types of modes for model diagram creation. One is off-line mode in which each developer creates model diagrams by individual. Another is on-line mode in which plural developers create model diagrams in a synchronous and collaborative manner. Although Elapiz provides Bulletin Board System (BBS) as a communication tool, it does not associate model diagrams with conversation contents as D-UML does not.

In a CHI (Computer Human Interaction) area, some systems are proposed which can associate a communication tool with a specific location in an artifact in real-time [2, 4]. However, they can’t co-edit shared artifacts in a synchronous manner, therefore it is not suitable to create UML diagrams in a synchronous manner under a distributed environment.

(2) It is not possible to catch history of collaborative modeling by associating model elements with conversation contents [8]

In design work, people may require design rationale when they review afterward. The persons who were absent from a design meeting and/or reviewers may require design rationale. However as traditional systems don’t manage conversations in a meeting by associating with model elements, they can’t meet such a requirement.

This study proposes a distributed synchronous collaborative modeling supporting system (we call Libra-on-Chat), which enables developers to share and co-edit UML diagrams in a synchronous manner, and manages conversation contents by associating it with model elements of a UML diagram. By managing conversation contents associated with model elements and time information, developers can conduct focused discussions on specific model elements and retrieve conversation contents in an efficient way. By time information, developers can refer to conversation contents in a chronological way. As this system holds conversation contents chronologically, users may be able to grasp design rationale.

This paper is organized as follows: Section 2 describes a distributed synchronous collaborative modeling system. We evaluated the system. Section 3 describes the experiment
method and results. Finally we conclude this paper.

2. System

In collaborative modeling in practice, developers often prepare their diagrams and create a unified model diagram by referring to them and discussing in a meeting. Libra-on-Chat supports synchronous collaborative modeling by sharing UML diagrams prepared by individuals respectively and associating conversations and their contents with model elements. Figure 1 shows a system architecture of Libra-on-Chat.

![System Architecture](image)

**Figure 1: System architecture of Libra-on-Chat**

The architecture of the system is client-server. The client side is a GUI application constructed with the Swing components. We reused a UML editor that was constructed by another project [5]. We used JSDT (Java Shared Data Toolkit) provided by Sun Micro Systems for synchronous communication between clients [6]. Figure 2 shows a screen shot of Libra-on-Chat.

A UML diagram created by a developer in an asynchronous manner can be shared with others by opening it in the “shared viewer” (Figure 2a). “Shared viewers” of more than other two developers are shown by switching tabs. “Shared modeling editor (Figure 2b)” enables developers to add, modify, and delete model elements in a synchronous and collaborative manner. Model elements of a diagram shown in the “shared viewer” can be integrated into this editor by clicking the right button. This function contributes to improve efficiency of development by avoiding redundant data entry. (Both D-UML and Elapiz don’t provide this function.)

In addition, Libra-on-Chat has the following features:

1. **Association of conversations with UML model elements (Figure 2c)** (we call this tool “UML chat”)

   This tool supports conversations for specific model elements. Icons that correspond to conversation contents are attached to the model element. Users can browse the corresponding conversation contents and time by double-clicking an icon. Plural “UML chats” can be attached to a model element chronologically. They can be used as design rationale.

2. **A text chat that is not associated with specific model elements (Figure 2d)** (we call this tool “plain chat”)

   This tool is used for conversations, which are not associated with specific model elements. However if a user wants some conversations in this chat to associate with a specific model element after a modeling session, this tool enables such an operation.

3. **History browsing of conversation contents (Figure 2e)** (we call this tool “message area”)

   All chats are listed up chronologically in this area. By clicking a title, the corresponding model element is highlighted and the conversation contents with the date and name of the person who uttered can be browsed. A modeling meeting can be replayed with this tool.

3. Experiment

We conducted a set of experiments to evaluate the system. The following sub-sections describe the method and result of the experiment.

3.1 Method of the Experiment

We conducted two types of experiments. The goal of the first experiment is to validate effectiveness of association of conversations with model elements. The goal of the second experiment is to validate how well the system supports to utilize the stored conversation contents. We describe the detail of each experiment.

1. Experiment of the first phase

   We organized two groups which were consisted of three subjects (five graduate students, and one undergraduate student). Each group conducted two modeling sessions. In one session, the subjects performed their task with the system. In another session, they performed another task with the quasi D-UML (as D-UML is not available, we implemented a quasi system that had the corresponding functions to D-UML). Table 1 shows the task and systems each group
used in each session.

Table 1: Task and assigned tools

<table>
<thead>
<tr>
<th>Task</th>
<th>First session</th>
<th>Second session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A</td>
<td>Quasi D-UML</td>
<td>Libra-on-Chat</td>
</tr>
<tr>
<td>Group B</td>
<td>Libra-on-Chat</td>
<td>Quasi D-UML</td>
</tr>
</tbody>
</table>

The procedure of each session is as follows:
(1) Face-to-face kick-off meeting: In this meeting, explanation of the experiment procedure and tasks, and exercises of tool usage were given.
(2) After the meeting, the subjects started a modeling session by individual respectively without any communication.
(3) Then synchronous collaborative modeling session by group (60 minutes) was conducted.
(4) After the modeling session, we asked the subjects for answering a questionnaire.

(II) Experiment of the second phase
Experiment of the second phase was conducted in one month after the experiment of the first phase. In the second experiment, the following materials were used: requirement specification that a few enhancement requests are appended for the specification used in the first experiment, the model diagrams and conversation logs that were created in the first experiment by the group with Libra-on-Chat. The number of subjects in the experiment of the second phase was three. One subject did not participate in the experiment of the first phase.

3.2 Result
We show the experiment results of two phases.

(I) Result of the experiment of the first phase
Table 2 shows a fragment of typical conversation contents with both tools in synchronous collaborative modeling.

When subjects discussed cardinality, attributes, and so on, about a model element, they asked without describing the target like “what is [1]?”,”Is the type of this ID String?” in case of using Libra-on-Chat. On the other hand, when the subjects used the quasi D-UML system, they discussed by describing the target explicitly like “If the “tennis court” class holds the “time slot” attribute,” or “Is the type of the sex attribute in the membership class integer?”. These facts show that Libra-on-Chat saved the burden of data input.
We show some comments gained from the result of questionnaire with respect to conversations that were associated with model elements: a subject gave the following comment: “by using Libra-on-Chat we could discuss each model element, therefore concentrate on discussions.” In another subject gave the following comment: “I could conduct collaborative modeling close to communication”. We found this system is useful for focused discussions.

From the result of questionnaire with respect to history browsing, a subject gave the following comment: “as the number of classes grew, when other classes had to be modified by revision of a class, this system was effective because it associated conversation contents with model elements, therefore it enabled developers to browse design rationale. On the other hand, in case of using the quasi D-UML, we had to read the discussions again to find the concerned discussions”. This means our hypothesis “users can discover contents of discussion in a prompt manner” was supported.

(II) Result of the experiment of the second phase
In this phase, subjects started their work from ascertaining the conversation histories, which were associated with classes and/or relationships which may have to be revised by their assigned task.

In addition, from the result of the questionnaire, all subjects gave comments that support the history browsing for each model element as follows: “By ascertaining the conversation history that led to the design decision, we could determine the design was incorrect.”, “It was helpful that I could view history of discussions for the corresponding model elements. In particular, it was very helpful for the diagram that I viewed for the first time.”, “I could catch through what communications the model was created.”

4. Conclusion
This paper has described a distributed synchronous collaborative modeling supporting system for UML diagrams.

From the experiment, by associating model elements with communications, we found the following benefits of the system: enabling focused discussions, and it is easy to find rationales of model elements because they are written in the chat associated with the model elements. On the other hand, from the result of experiment, some subjects pointed out some problems for the system: “I wanted awareness information on whether other members are in modifying diagrams, they are considering something, or they are typing comments”, “I had some trouble to grasp where discussions are in progress, when a lot of chat windows were opened”. As future work, we will solve these problems.

References