15th ICCRTS

"The Evolution of C2"

"Using Transcription and Replay in Analysis of Collaborative Applications"

Topics: C2 Assessment Metrics and Tools, Info Sharing and Collaboration Processes and Behaviors

Seth Landsman, Ph.D., The MITRE Corporation
Richard Alterman, Ph.D., Brandeis University

Point of Contact: Seth Landsman, Ph.D.
The MITRE Corporation
MS 325
202 Burlington Road
Bedford, MA 01730
+1 781 271 7686
landsman@mitre.org

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Using Transcription and Replay in Analysis of Collaborative Applications	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **The MITRE Corporation,MS 325,202 Burlington Rd,Bedford,MA,01730**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Proceedings of the 15th International Command and Control Research and Technology Symposium (ICCRTS '10), Santa Monica, CA, June 22-24, 2010

14. ABSTRACT
As organizations become more distributed enterprises, the use of collaborative applications as part of day-to-day activities becomes more prevalent. How to build collaborative applications so that they work as expected, both as the developer expects and the community of users expect, is a challenge that must be addressed in order for these applications to be adapted as part of organizational activities. In practice, assumptions made by the developer may not match the expectations of the users and may result in a collaborative application being used sub-optimally, at best, or abandoned altogether, at worst. This paper details a model for collecting a transcript of online collaboration. The transcript, as collected, contains sufficient information to recreate the precise state of the collaborative application at any given time, and, therefore, the collaborative activity can be recreated for analysis, using techniques such as ethnography.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**35**	

Abstract

As organizations become more distributed enterprises, the use of collaborative applications as part of day-to-day activities becomes more prevalent. How to build collaborative applications so that they work as expected, both as the developer expects and the community of users expect, is a challenge that must be addressed in order for these applications to be adapted as part of organizational activities. In practice, assumptions made by the developer may not match the expectations of the users and may result in a collaborative application being used sub-optimally, at best, or abandoned altogether, at worst.

This paper details a model for collecting a transcript of online collaboration. The transcript, as collected, contains sufficient information to recreate the precise state of the collaborative application at any given time, and, therefore, the collaborative activity can be recreated for analysis, using techniques such as ethnography.

1 Introduction

As organizations become increasingly distributed, different-place collaboration becomes increasingly critical for day-to-day business. This need is prevalent in the command and control enterprise, which is a prime example of a distributed, collaborative enterprise. However, collaborative processes that govern these enterprises are not fixed entities. There is a constant and considerable need to study and evaluate how people work together and how group work can be improved to better fit the community of users. A number of techniques and approaches have been explored in this area, from the understanding of interaction (Clark 1996), distributed cognition (Hutchins 1996), and ethnographic study of an artifact's use in group activity (Suchman and Trigg 1991).

A collaborative process that is mediated by a software application (i.e., a C2 system) also needs to be studied and improved. External perspectives of activity, such as video taping a user, become less useful as the interaction may be distributed and requires both context of a user's activity and correlation between users' activities. Ethnographic analysis has been shown to be effective in the study of off-line interaction (i.e., not mediated by software), and can be applied to online behavior given the proper software support.

Ethnography for off-line collaboration generally captures the activity of the users (transcription) and provides a mechanism for further review (replay). Our approach is to provide similar capabilities for the analysis of online collaboration by collecting the activity of the users as they collaborate into a continuous transcript. Review of the transcript is accomplished by a software system that replays the transcript in the same context as the transcript was captured.

We have built several applications to study collaboration. In each case, the application produced complete, replayable transcripts. The replay of these transcripts was used for a variety of tasks: as a basis for redesigning an application, as part of a research study on online collaboration, as data that is used for teaching analysis techniques, and as resource data for academic projects. We have developed a pair of toolkits to facilitate the development of collaborative applications: THYME (Landsman 2006) and SAGE. THYME is used to generate collaborative application shells that automatically produce complete, replayable transcripts. SAGE provides the basis for constructing a replay application. Both toolkits have been used to create numerous applications that illustrate different principles of collaborative applications.

This paper extracts a general model for building collaborative applications that automatically produce replayable transcripts. This model supports the transcription of user activity, both “low-level" activity such as mouse clicks and “task-level" activity such as chat events. Replay of the application occurs through a transformation of the basis application used to collect the transcript. The model provides methods to enhance the replay of the transcript given the structured, introspectable nature of the transcript, such as searching for types of events in the transcript and annotation of interesting aspects of the transcript. The THYME and SAGE toolkits are an implementation of this model.

The remainder of this paper starts with a simple example of how transcription and replay can be used to support the redesign of the collaboration. This example shows some of the capabilities of the model of transcription and replay. In the following section, the model of

transcription and replay are detailed, including the parameters on how a transcript is collected, how a collaborative application is transformed to support replay, and how the replay of the transcript is accomplished. This paper then concludes with a discussion of future work and how this model can be brought forward to more complex command and control collaborative enterprises.

2 ANALYSIS OF COLLABORATION

As part of a term project, abl Human-Computer Interaction (HCI) class held at Brandeis University implemented synchronous collaboration applications as part of their term projects. These students were given access to the THYME framework and instruction on using the framework. The class was divided into fourteen teams of 2 - 3 students per team. The majority (twelve out of fourteen) teams successfully implemented an application that could collect replayable transcripts of use.

As an illustration of the types of applications that can be built using this framework, one group in this class produced an application called the **O**nline **R**esearch **A**ssistant (ORA). This application allows a more experienced researcher (such as a librarian or subject matter expert) to help another participant locate information via a web browser.

The ORA application contains several common collaboration components, including a shared whiteboard, a chat room, and a shared web browser; illustrated in Figure 1. The left side of the screen (callout 1) is the web browser with a shared whiteboard in a transparent overlay. The browser is a relaxed What You See Is What I See (WYSIWIS, Stefik, Bobrow et al. 1987) component, in that the web browser context is synchronized between users, but the scrolling of the web page is not. The right side of the screen (callout 2) is the chat room. The bottom of the screen (callout 3) contains the tools used to manipulate the shared whiteboard, including the palette of artifacts and button to toggle the whiteboard's display.

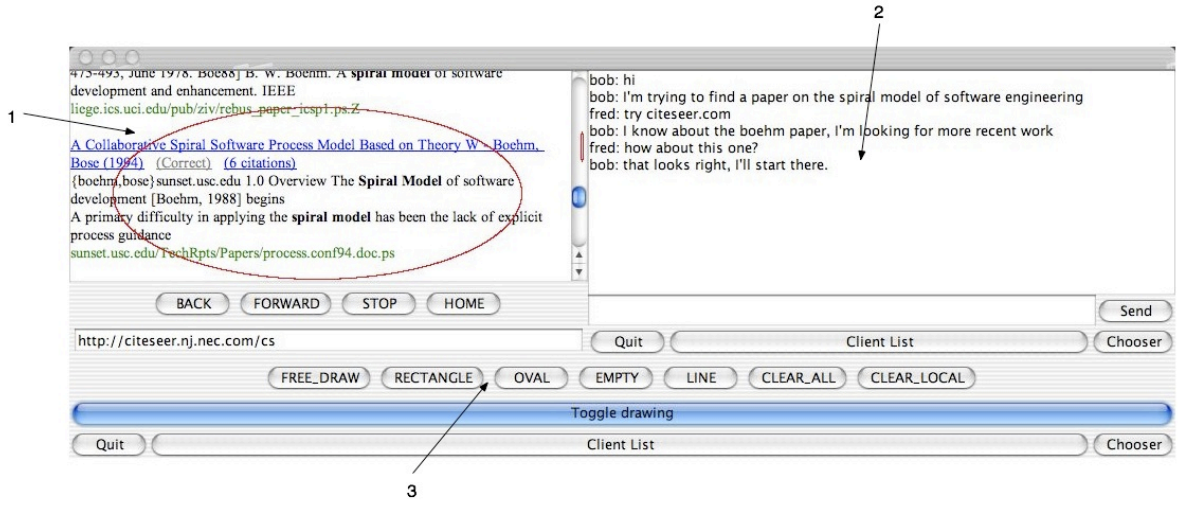


Figure 1, Screenshot of the Online Research Assistant

From the ORA transcript, there are three types of task events: Chat Events, Shared Whiteboard Events (i.e., drawing a new artifact on the glass pane or manipulating an existing

one), and Shared Web Browser Events (i.e., entering a new URL, and going forward and backwards in the history of visited URLs).

From a comparison of the ORA replay application (Figure 2) and the basis ORA application (Figure 1), it is clear that they share a common lineage. As shown in Figure 2, the center replay window is a direct analogue of the ORA client screen, and contains individual components that are leveraged from the basis application.

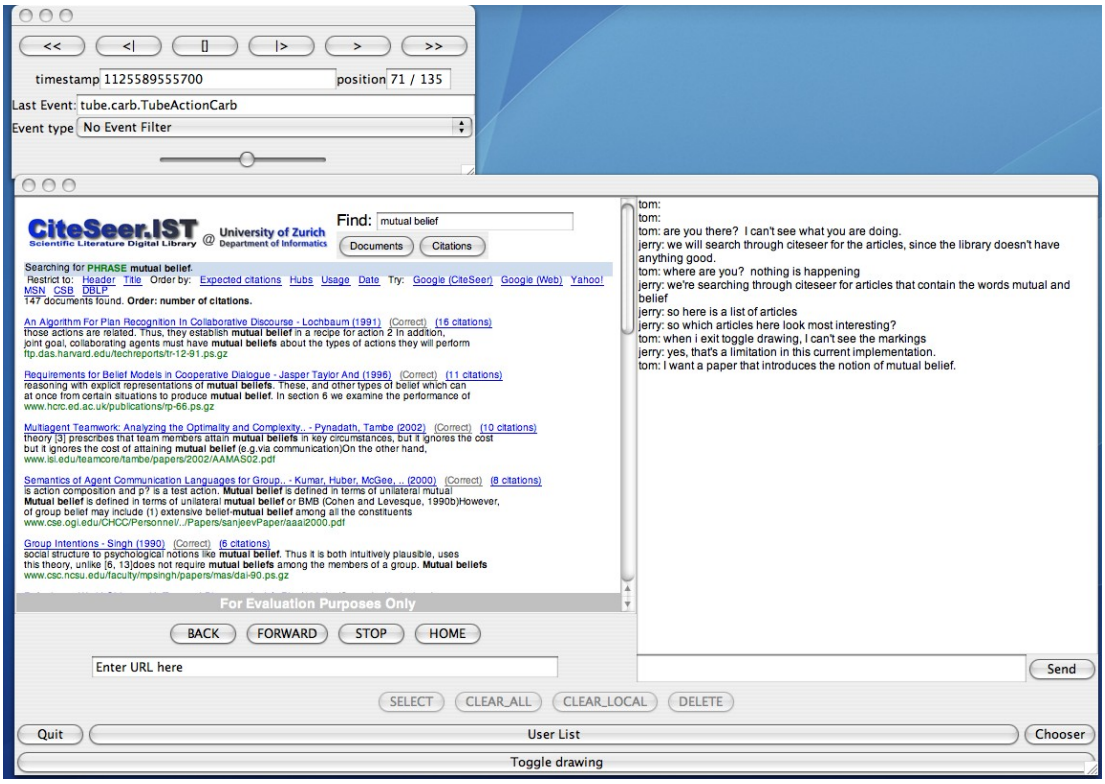


Figure 2, an analysis session for the Online Research Application

The top left object in Figure 2 is the replay controller. It allows basic playback of a transcript through a VCR-like metaphor. The top buttons are recognizable as varying speeds of replay, both forward and backward, as well as a button to stop the replay. Also of note is the *Event Type* field, which allows the analyst to move forward or backward in the transcript to the next instance of a specific type of event. More detail on how this controller operates is discussed later in this paper in section 4.2.2.2.

2.1 Example Analysis

We performed data collection and analysis on usage of the initial design of the ORA application. The analysis is used to direct the further development of the application, allowing conclusions to be drawn about how the application is to be used by a sample set of end-users. Based on these conclusions, the development directions can be determined with a large degree of precision and accuracy, responding to the elicited and observed needs of the community.

In the example analysis shown, two users with the pseudonyms *Tom* and *Bob*, are using the baseline version of ORA. Bob is an ORA developer and researcher who is assisting Tom in finding information relevant to a specific topic.

Throughout this discussion, we will emphasize how the analyst manipulated the replay application by *italicizing* his actions. Figure 2 shows a segment from this analysis session.

This example of use has three major stages:

1. Initial exploration of a university library, which fails because of limitations in the web browser associated with the ORA application.
2. Transition to another citation database (the *Citeseer* database) and resynchronization of the user's common ground
3. Successful competition of the task

The first part of the session starts with the interaction between Tom and Bob (taken verbatim from the transcript):

Tom Hi. Can you help me to find articles or books on mutual belief? I am particularly interested in representation and mutual belief. But first the general concept of mutual belief

Bob sure. let's start in the library.

Tom wait what did you just do??

To obtain this dialogue, the analyst starts *playing* the transcript. By observing the replay, the analyst can see that this exchange corresponds with Bob going to the library website. The page loading gives no feedback to the user who did not initiate going to the new website. This confusion as to what Bob is doing results in a question from Tom, which requires a repair.

The analyst continues to observe by *playing until the next chat event*. It becomes clear that the use of the library does not yield any results. Bob moves on to the CiteSeer website. Again, because of the lack of feedback to Tom as to what the web browser is doing, the following interchange occurs:

Tom are you there? I can't see what you are doing.

Bob we will search through citeseer for the articles, since the library doesn't have anything good.

Tom where are you? nothing is happening

Bob we're searching through citeseer for articles that contain the words mutual and belief

Once Tom's second set of questions has been asked, the analyst performs a *rewind until the previous web browser event*. Now the state of the replay is at the web browser event prior to Tom's question and the analyst can ascertain why Tom would not see any feedback from Bob's actions. He continues to *step forward*, observing that Bob attempted to use the library website to search for books, but the web browser component failed to interact with the custom JavaScript on this website. He then *plays until the next web browser event* after Tom's last utterance, and can see that Bob switched to the CiteSeer website.

Through the session, a number of desired features are specifically identified. For example:

Tom can I look at the pdf?

Tom any version of the paper?

Bob unfortunately, there is no PDF viewer built into this application. We can add it in a few spirals.

In viewing the activity in the replay tool, a workaround is identified by Tom. By continuing to *play until web browser events*, it is observed that Tom discovers that CiteSeer can display images of papers, which worked sufficiently for this task.

Another request was made, to allow searching for specific subject matter of papers, which was clearly not within the scope of this tool, and identified as such during the conversation.

Tom	is there a way we can filter for philosophy and not ai papers?
Bob	I do not think citeseer has that capability.
Bob	I don't think any paper search engine has that capability, currently.
Bob	how can you tell that a paper is philosophy, and not AI?

Based on the observations of use, pointing to the web page and aspects thereof was done frequently, but the drawing tools, which existed to aid in the referencing of objects in the web page, were not used. In fact, only 8 of 135 events were shared whiteboard events in this session, as reported by the replay application. Because of how the drawing tools were implemented, in that they required changing the application's mode of use from browsing to drawing, their use may have been too cumbersome. It may also be that in this collaboration, which had only two users and relatively manageable web pages, the pointing capabilities were not needed. As the application is used further and more data is collected, the drawing mechanism may see more use and will need to be refined. Further, even though much of the analysis of this transcript discusses limitations of the mediating tool, it does bring to the forefront the specific needs of the user, perhaps directing activity to documented user needs, instead of only speculated user needs.

3 MODEL OF TRANSCRIPTION AND REPLAY

Transcription of the use of the collaborative application and the replay of this transcript provide a means of enabling ethnographic analysis. Transcription is the mechanism that records the interaction that the user has vis-à-vis the application. As discussed in this section, there are a number of vectors through which the transcription capabilities of the application can be described. How the transcript is collected influences the available replay options, including the fidelity and precision available to the analyst.

Similarly, there are several different properties associated with the replay application. Some capabilities depend on the corresponding properties of the transcript, while others are inherent in the replay application itself. We will discuss these properties and compare requirements for online ethnographic analysis with the properties that have been achieved by other efforts.

3.1 Transcription

A transcript is the record of the user's activity, as mediated by the collaborative system. The way a transcript is collected can have direct influence on the quality and quantity of replay techniques available to analyze the system. Our set of identified properties include:

- *Collection of Online and Off-line Activity:* Transcripts may encode online behavior, which is the activity that is directly mediated by the software system. They may also encode off-line behavior, which may be part of the collaboration, but not directly mediated by the software. Eye tracking, for example, is usually a collection of off-line behavior, where mouse actions are online behaviors. While off-line behavior can add

value to observation, the quality of the online behavior collection is most important for our analysis techniques.

- *Type of Online Information Encoded:* There are two types of online information that a transcript can encode:
    - *User Interface events (UI):* UI events include direct manipulation events in the user interface, such as mouse clicks and key presses.
    - *Task Events:* Task event information refers to interaction that is mediated by the collaborative application. Where user interface events depict the users activities at the level of point-and-click, task events depicts the users activities at the level of plans and communication. The level of event structure is important because it enables the analyst to review and replay the transcript in terms of the semantics of the domain.
- *Online Completeness:* A complete transcript contains information sufficient to re-create the state of the application at any given point in time. A transcript can be complete with respect to user interface events or task events. A transcript that is complete with respect to user interface events is not necessarily complete with regards to task events, and visa versa. For example, a transcript can encode the sequence of keys that were tapped, but that information is not sufficient to reliably reconstruct whether the user's task was planning or chatting without further context.

  Other transcription and replay systems, such as jRapture (Steven, Chandra et al. 2000), Playback (Neal and Simons 1983), and others (Ronsse, Bosschere et al. 2003) provide complete transcripts of user interface events only. jRapture replaces an application's underlying standard Java libraries with ones that transcribe external interactions with the application. The Playback application captures interface events by ``intercepting'' interaction at the device interface level.

  Timewarp (Edwards and Mynatt 1997) and Chimera (Kurlander and Feiner 1992) provide a complete transcript only with respect to an enumerated set of task events. They collect a history of actions within the application by collecting the interaction with the application into a transcript. Chimera uses the transcript as a basis for end-user programming of macros. Timewarp constructs a history of changes to a data object, e.g., a document. The user can then modify a historical data object, and thereby change all of its descendants. Neither Chimera nor Timewarp provide replayable transcripts that could be used for ethnographic analysis.
- *Transitions or States:* As the transcript is generated during a session of use, it can either record *transitions between states* or *individual states*. The advantage of a transcript that encodes data in terms of state is that it allows the replay tool to directly access any state; the disadvantage is that collecting such a transcript is spatially and computationally expensive. Storing transitions result in a smaller transcript and will be computationally cheaper to collect, but at the cost of more expensive post processing and playback of the transcript. Also, rewind may be costly if the transitions are not reversible. Commonly, the application's state would need to be reset and the playback re-run from the start of the transcript until it reaches the state the analyst chose to examine.

In addition to the two extremes of storing transitions or states, a hybrid approach of storing *checkpoints* is also seen in some transcription implementations. Checkpointing is the storing of occasional states of the application's execution as well as transitions between those states. The resulting transcript allows for faster movement between positions in the transcript and definitive points of coordination between different parts of a distributed application. Examples of this technique are found in the BugNet application (Jones, Barkan et al. 1987), as well as Chandy and Lamport's work (1985), and in others (Satyanarayanan, Steere et al. 1992) (Yang and Marsland 1992).

Table 1 summarizes our discussion of prior efforts at transcription in the terms of the criteria we have developed.

Application	Completeness	Info Type	Transitions	Off-line
jRapture	UI	UI	Transitions	None
Playback	UI	UI	Transitions	None
Timewarp	Task	Task	Transitions	None
Videotape	None	N/A	States	Video

Table 1, Transcription Properties

3.2 Replay

Ethnographic analysis of online collaboration requires the ability to replay the transcript of system use. Depending on how the transcript is collected, different capabilities become available to the replay system. Which capabilities the replay system implements affects how the analyst can interact and use the transcript, and include:

- *Search:* What kinds of events the analyst can use the replay tool to search for depends on what kinds of events are in the transcript. For example, a transcript that only encodes mouse clicks and key presses will not provide the basis for the analyst to use the replay tool to search for chat events among users.
- *Annotation:* Annotations give the analyst the ability to annotate, tag, or otherwise mark the transcript as the application session is replayed. In addition to providing information for an analyst to refer to in later analysis sessions, they also provide additional information for the replay tool to search for and notes for other analysts use.

  The video tape solution provided by Suchman and Trigg (1991) provides the means for annotating a video tape transcript during playback, allowing areas of interest to be clearly marked for future reference.
- *Precision:* After the transcript is generated, the analyst can always annotate the transcript noting events of particular interest that can later be returned to for further analysis. Annotation is a time consuming and potentially inaccurate task for the analyst. Ideally, the analyst can replay an unannotated transcript stopping at, for example, each chat event. *Precision* is used to indicate that a transcript is sufficiently encoded with information such that the replay application can accurately differentiate between different features of events.

A replay tool is precise with regards to time if the analyst can replay the transcript (without annotation) to directly display an event that occurred at a specific timestamp. A replay tool is precise with regards to task event if the analyst can replay the transcript to display a task event of a certain type.

The playback provided by jRapture and Playback, for example, allow for precision based on the type of user interface event and the timestamp. However, they do not provide precision based on the task event, since those do not exist within the transcript.

- *Aggregate Information:* Some replay tools allow the analyst to summarize and display quantitative data of system use. For example, this data can include a count of window events or a count of collaboration failures.

  Applications such as CollabLogger (Morse and Steves 2000) are specifically designed to allow for the collection of aggregate information. These applications collect transcripts and analyze them to gather statistics as to how the application was used, how particular participants performed as a measure of their interaction with the application, and other similar measures.

Table 2 summarizes our discussion of prior efforts at playback in the terms of the criteria we have developed. In particular, ethnographic analysis is best served by search, precision, and annotation capabilities.

Application	Search	Precision	Annotation	Aggregate Info
jRapture	No	Time, UI	None	None
Playback	No	Time, UI	None	None
CollabLogger	No	None	None	Yes
Videotape	No	Time	Yes	None

Table 2, Replay Properties

4 ENGINEERING TRANSCRIPTION AND REPLAY

To engineer online ethnographic analysis capabilities into an application and its development lifecycle, the appropriate transcription and replay technologies must be put into practice. Our implementation provides the feature set necessary to do the level of replay necessary for ethnographic analysis. This section describes how we engineered this technology into our application framework.

4.1 Transcription and Replay Requirements

Many of the features of a replay tool depend on the quality of the transcript. A complete transcript best supports analysis, where each state of the collaboration can be reconstituted. It is important that this transcript is complete for events that describe interaction with the user interface, such as mouse clicks, and events that describe interaction with the task environment, such as chat utterances, as both types of information can be critical to understanding the collaboration process.

The technology we have developed produces a transcript that is complete and encodes both task and user interface events. The transcript itself is encoded as additive transitions. Our framework is based on a message-passing architecture, and collects information from both user interface and task environment actions by collecting messages as they are generated within the application during run-time.

The replay technology processes the transcript so that the analyst can view the collaborative activity from a perspective similar to that of the users who generated the transcript. Our replay tool allows the analyst to search an unannotated transcript for the next task event of a certain type. The analyst can also step through the transcript one event at a time. Each of these features depends on the completeness and level of information encoded in the transcript.

Because the transcripts encode task events, the transcripts can be used to do a quantitative analysis of, for example, how much chatting the users did. Similar, they can also be used to perform various kinds of quantitative analyses on user interface interaction with the application.

4.2 Implementation

Our transcription and replay techniques were realized in two complementary software libraries. The first library, THYME, is a framework for building message-oriented collaboration applications. An application constructed using the THYME framework automatically generates transcripts of their use during the application's run-time. These transcripts, as described above, are complete, encode task and user interface events, and are transitional. They contain only online user interaction.

The second framework, SAGE, provides the foundation for assembling replay applications. The THYME application that was used to collect a replayable transcript is the basis for constructing the replay application. The replay application that is assembled provides the necessary “over-the-shoulder" perspective as to what the user was doing when the transcript was collected.

Some implementations of replay, such as jRapture, use, without modification, the original application to do replay, but at a cost. SAGE allows the replay application to better support the analysts’ work. For example, a SAGE-produced replay application can include useful features like rewind, filtering, and alternative views of shared representations. Thus, while the most basic SAGE application will look identical to the basis application on which it is based, the underlying capabilities support the replay of the collected transcript.

4.2.1 Transcription

A THYME application is defined by a set of components and the messaging connections between them. Components are grouped into structures called *nodes*, each of which has a unique namespace. These components communicate via messages, orchestrated by a per-node object called the *message router*. A component sends an addressed message to the message router. The message router will find the component the message is addressed to and deliver it. The overall design, while not implemented using common technology just as a Java Message Store (JMS) Queue, implements the same common pattern. More details of how THYME works and the types of applications built using THYME can be found in another work (Landsman 2006).

THYME's message-oriented architecture has several useful consequences for collecting a transcript of use:

1. All communication between components happens through messages, so a THYME application only needs to collect messages in order to generate a complete transcript.
2. Since all messages go through routing components, only the message routers need to be accessed in order to record all the messages.

To collect the transcript for an application's session of use, the message routers collect all messages at their point of origin, defined as the first message router that handles the message. This approach ensures that every message is logged once and only once. As a router collects messages, it sends the message to the *transcript collector* component. This component will store all messages it receives, thereby building the transcript. This component forms a unified interface to the transcription subsystem, abstracting the means by which the transcript is stored and reconstructed and allowing different transcription formats and strategies to be used without changing the application. Figure 3 illustrates the interaction of a THYME application with the transcription subsystem.

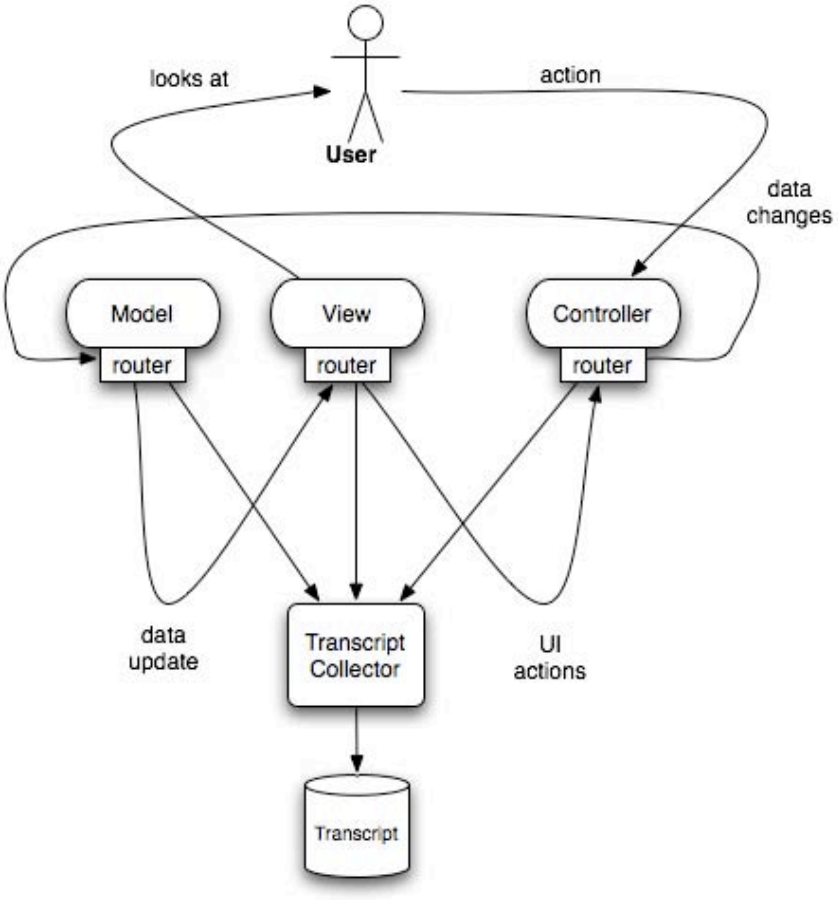


Figure 3, Collecting messages in a THYME application to build the transcript

Each transcribed event in the THYME framework is stored with two timestamps, when it is first handled by a router, and again when it is actually transcribed. The two timestamps gives sufficient data for clock skew correction to be performed, if necessary. The timestamp is the discrete points in the timeline of the session, and is used to explicitly order messages as they get injected into the replay application.

Within the THYME application, transcripts can be accessed during the run-time of the application through a component called the transcript emitter. This component provides access to previous transcribed messages in the current session of use. Transcript collection is ended when the session of use is over, and archived so that it includes session summary information (called *meta* information) in addition to the transcript of events. Archived sessions of use can be loaded into the transcript emitter explicitly. A SAGE application makes use of the transcript emitter to playback the messages contained in the archived transcript, in the order those messages were generated by the participants in the archived session.

In a THYME application, all changes to the application state occur through the reception and processing of messages. A *complete* transcript is, therefore, the collection of all messages that are sent between components between two starting and ending messages. A transcript is represented as $TRANSCRIPT_{T \in [X,Y]}$ and is the transcript for all messages between the ordered messages M_X and M_Y inclusive. *TRANSCRIPT(X)* is used for referring to the message M_X that is stored in the transcript.

The transcript collected contains both interface events and domain actions. Interface events can be encapsulated as messages to an *interface controller* and thereby are collectable in the transcript. Because messages between components are encoded in terms of the representation system of the application, the information contained in the message includes domain action information. For example, a message passed from one client's chat room component to another client's chat room component contains information that the message is a chat message. Thus, during replay, the analyst can run the replay until it comes to a *chat message*. Alternately, if users are collaboratively constructing a document in a shared window, messages between client planning components will contain information that enable the precise replay of document construction actions.

4.2.2 Replay

The SAGE framework enables replay of a transcript. The framework provides access to the collected transcript and enables a replay application to rebuild the state of the basis application at a precision level of individual messages. Additionally, these replay applications can be built relatively cheaply by leveraging the basis application, yet still enjoy many of the expected advantages of a customized replay application, such as reviewing past states and customized views as warranted.

The state of the application is built from the application of messages. During a THYME application's run-time, a message, *M*, is applied to a component *C*, resulting in a component *C'*. Succinctly, this process is represented as $C(M) = C'$. An application consists of a set of related components, $C_0, C_1, \ldots, C_n$. Therefore, applying a message *M* to an application *A* is represented as $A(M) = A'$, which is also written as $A(M) = C_0(M), C_1(M), \ldots, C_n(M)$.

Given a basis application *A* and a collected transcript *T* of size *S* replaying *T* on *A* is done by applying the elements of the transcript on each component in the application. This is done as follows:

$$REPLAY(A,T,0,S) = \forall_i : 0 \leq i \leq S; A_{t=1} = A_{t=i-1} + T(i)$$

Where the statement $A_{t=1} = A_{t=i-1} + T(i)$ is expanded to

$$A_{t=1} = A_{t=i-1} + T(i) = \forall_C : C \subset A_{t=i-1}; C = C(T(i))$$

Where $C_{t=x}$ is the component *C* after *T(x)* has been applied. Replay to a specific message *X* is done by applying all previous events (as defined by the order of execution) from the transcript, up to and including M_X , to the application.

Technically, SAGE only provides event-level *precision*. Time-level precision within a collected transcript is limited to the instants of time that are collected in the transcript. While each message has an associated timestamp that refers to when it was collected, the granularity is limited to the points in time where the messages were actually collected. For example, if the time between a message M_{i-1} and M_i is 10 minutes, there is no way to display any activity between those two events. However, if the transcript is complete, it is possible to go to a specific point in time by progressing to the last message that occurred before the requested timestamp. If, in the example given, the transcript is complete, it can be deduced that no system activity occurred in the intervening 10 minutes, so there is no real precision lost. From these conditions, it can be shown that SAGE emulates time-level precision.

Supporting user interface or task-level precision occurs through being able to search for specific types of events. Given a transcript, the set of possible event types is represented by the set *EVENT-TYPES(T)*, which are mined from the transcript *T*. The available set of *EVENT-TYPES* is related to the level of task information in the transcript. If the transcript only contains UI events, then that level of granularity is available to the analyst through the replay tool. However, if the transcript contains events that illustrate interaction with the task environment, such as chat messages, then the replay tool can use those events to show event boundaries. This information would allow the analyst to say ``skip to the next chat utterance", for example.

4.2.2.1 Milestoning

The approach of encoding transitions between states is done for two major reasons. Encoding the entire state at each change will take up a large amount of disk and processing resources and will require a level of introspection and access to all aspects of the application that may not be easily available. Instead, encoding transitions is accomplished through the use of the existing message passing infrastructure, without needing information about the components, their state, and how their state can be captured.

In encoding transitions, the ideal situation would have each transition reversible. That is, $C' = C + M_i$ and $C = C' - M_i$. However, messages in our framework, as is true in the majority of message-passing frameworks, are not designed to be reversible, as doing so puts a large burden on the developer to track state and limit actions on the application data. Without reversible messages, actions such rewinding an application's state is difficult. A brute force example of rewinding state could consist of selecting the desired point in the transcript,

resetting the application and applying all messages up to the newly desired timestamp. Early versions of SAGE provided such a mechanism, which was quickly deemed unsatisfactory.

Since the data that makes up a THYME application is stored in components, to implement a proper rewinding mechanism for a transcript requires each component to be capable of rewinding its state. To accomplish rewind across all types of components, SAGE provides a set of component wrappers, based on a design pattern called *mementos* (Gamma, Helm et al. 1995). A wrapper's internal state is actually a collection of milestones, which are indexed instances of the component it is wrapping. Each index refers to a specific message number in the transcript. Milestones are laid out so that to retrieve an instance that corresponds to a timestamp previous to the current one, it is only necessary to find the component that is closest to, but previous to, the desired timestamp. That component is then copied and any messages that exist between the desired timestamp and the current component's timestamp are retrieved and applied. This process is shown in Definition 1.

Given a component wrapper *CW* that wraps a component type *C*, upon receiving message *M*, which is position *I* in transcript *T*, the following takes place:

1. if there is a milestone *MI* in *CW* that corresponds to $C_{T=I}$, activate that milestone and exit
2. if there is no such milestone, find the milestone MI_J that has the closest index less than *I*
3. copy MI_J to a new component $C_{T=I}$
4. apply every message M_X where $X > J \geq I$ to $C_{T=I}$
5. activate $C_{T=I}$
6. if a new milestone is desired at *I*, store $C_{T=I}$ into a milestone MI_I

Definition 1, Milestoning process

The decision to store milestones depends on the application and storage needs. Generally, milestones exist at increasing intervals. For example, the first milestones would be at *t + 1* and *t - 1*, the second at *t + 5* and *t - 5*, the third at *t + 20* and *t - 20*.

The milestone process is related to *checkpointing* (Lorie 1977) a database to ensure consistency of the database state. Milestones may, unlike checkpoints, change in number and temporal location during execution. Global system snapshots (Chandy and Lamport 1985 and Yang and Marsland 1992), a technique used in distributed debugging, is also similar, in that it looks to collect a consistent state across multiple systems. The milestones described here are not used to create a unified system state, although they do that as a consequence because of the simple nature of the SAGE application. Instead, milestones are designed to provide a means to access a set of temporal positions within a single model.

A component can implement its own state mechanisms so that it can provide its own reverse functionality. The wrapper approach is a general solution if the component does not have this capability already implemented.

4.2.2.2 Analysis Interface

Using the replay application, an analyst can perform precise analysis of the usage of the basis application. The SAGE Playback Controller (shown in Figure 4) gives the analyst control over

the flow of the playback of the transcript. The playback controller has standard VCR-like controls: play, rewind, fast-forward, and stop (see 1 in the figure). The controller adds two other standard movement controls: step forward and step back, which move one event forwards and backwards, respectively. The controller allows movement through the transcript by searching for types of messages that are in the transcript's set of *EVENT-TYPES* (see 3). The list of types is populated from the transcript at the run-time of the replay application. (Note that the message displayed in this example is the Shared Browser message, more meaningful event names are provided at the discretion of the component being transcribed.)

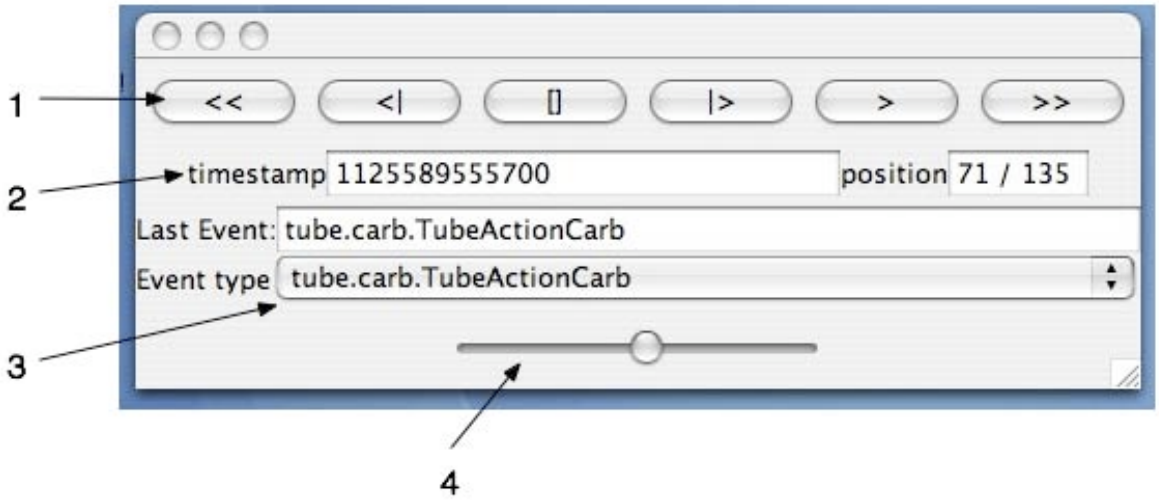


Figure 4, SAGE Playback Controller

The controller also provides the analyst feedback as to where he is in the session. The information underneath the VCR controls (see 2) shows the current timestamp of the session, based on the timestamp of the last event replayed, currently shown as the unix standard milliseconds since epoch. Next to the time display is the current message and the total number of events in the session. Movement within the session can also be controlled via a slider (see 4), at the bottom of the window. The slider provides feedback as to where in the session the current timestamp is, with the far left of the slider being the beginning of the session and the far right being the end. The analyst can manipulate the slider, causing the replay tool to go to the event closest to the timestamp selected.

5 VesselWorld Example

Another example of using transcript and replay in the design and redesign of a collaborative system was implemented in the VesselWorld group problem solving system (Alterman, Landsman et al. 1998 and Landsman, Alterman et al. 2001). The problem domain encoded in the VesselWorld application has three participants engage in a computer-mediated problem solving session. To complete a set of tasks in this simulated environment, the participants must communicate and jointly problem-solve. The only avenue of communication is via the application client. Access to the environment, and objects in the environment, is also mediated through representations provided by the software application. The problem solving sessions require cooperation, coordination and collaboration.

There have been multiple experiments run with the VesselWorld application (Alterman, Feinman et al. 2001) (Feinman and Alterman 2003) (Introne and Alterman 2003), all of which have leveraged the analysis capabilities of the application. The replay application associated with VesselWorld can be seen in Figure 5. The replay of transcripts was used for both redesign tasks and analysis of experimental data. Over the course of these experiments VesselWorld has been modified several times to support different types of collaborative interactions and different types of analysis. Over two hundred hours of VesselWorld data has been collected across these versions of the VesselWorld application.

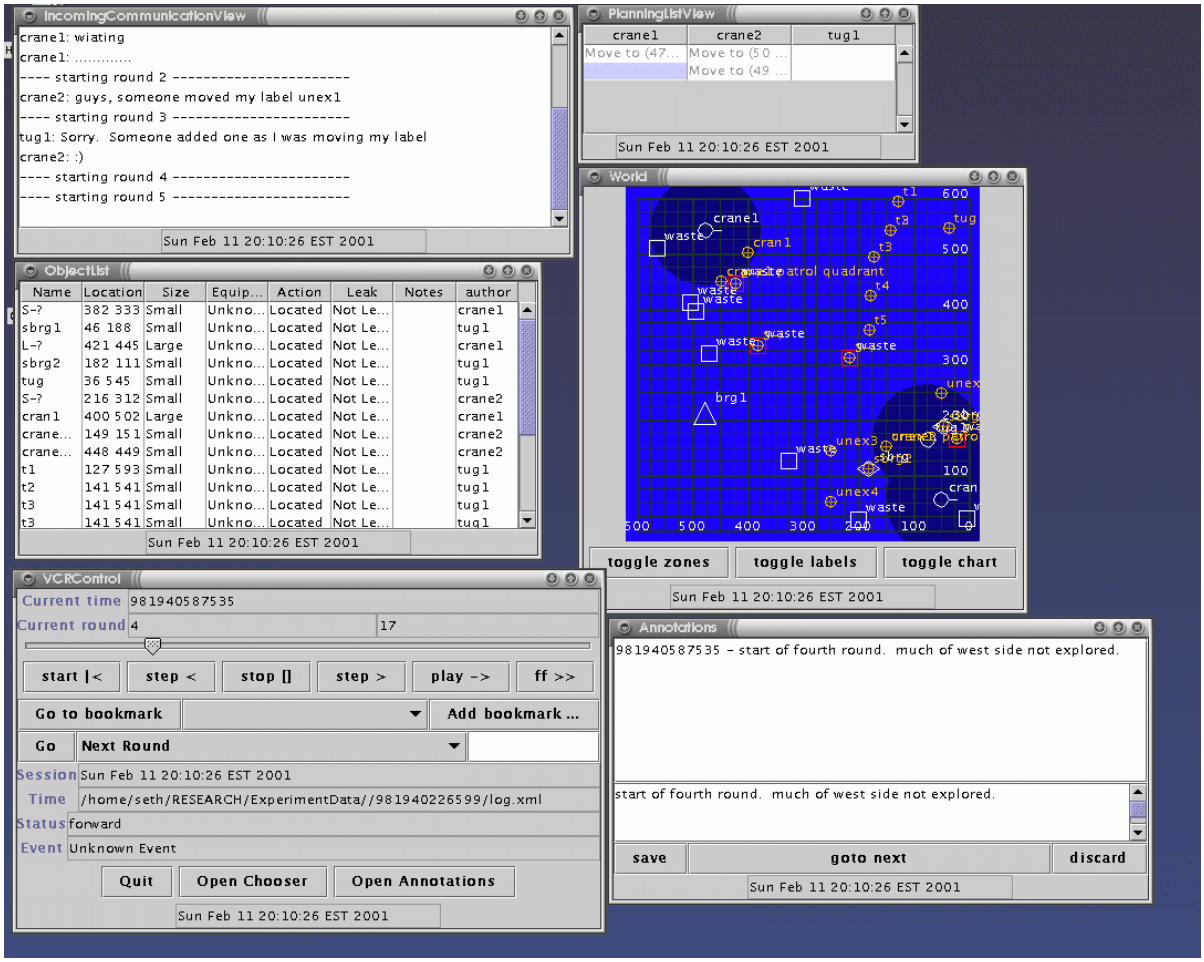


Figure 5, The VW-SAGE system

6 CONCLUSIONS

This paper discussed our approach to online ethnographic analysis of collaborative applications. In the analysis of applications, an ``over-the-shoulder'' perspective of the user's activity is given to an analyst, allowing him to draw conclusions by observing how the application was used. Contrasted to quantitative methods, this perspective allows insights to be drawn that may not be otherwise available, especially in how collaborative breakdowns occur. When paired with quantitative analysis of the transcript the after-execution analysis process can be greatly enhanced (Feinman and Alterman 2003).

As computer mediated collaboration becomes a larger part of the day-to-day activity of C2 enterprise, being able to validate the utility and ability to execute efficiently of the collaborative system becomes increasingly important. If the development of the replay applications can be reduced to be a small part of the total development costs of the application itself then analysis can be realized as part of the software lifecycle. In another work (Landsman 2006) we discuss some techniques in place to allow SAGE to generate the majority of the replay application, based on a well-instrumented THYME application.

The concepts presented in this paper, and especially the example implementation, represent an initial research foundation. Future work needs to focus on how to bring these techniques to systems that do not require a common foundational backend (i.e., THYME), but instead bring the ability to collect a transcript of a slightly modified existing collaborative application. Ongoing work has focused on instrumentation of message-oriented applications (e.g., enterprise service bus oriented applications) and applications that depend on common platform for all communication (e.g., container-oriented applications).

7 ACKNOWLEDGEMENTS

This work was supported under ONR grants N00014-00-1-8965 and N00014-96-1-0440, and NSF grant EIA-0082393.

REFERENCES

Alterman, R., A. Feinman, et al. (2001). Coordination of Talk: Coordination of Action, Brandeis University.

Alterman, R., S. Landsman, et al. (1998). Groupware for Planning, Computer Science Department, Brandeis University.

Boehm, B. (1988). "A Spiral Model of Software Development and Enhancement." IEEE Computer **21**(5): 61-72.

Chandy, K. M. and L. Lamport (1985). "Distributed snapshots: Determing global states of distributed systems." ACM Transactions on Computer Systems **3**(1): 63 - 67.

Clark, H. H. (1996). Using Language, Cambridge University Press.

Cunningham, W. WikiWiki.

Edwards, W. K. and E. D. Mynatt (1997). Timewarp: techniques for autonomous collaboration. Proceedings of the SIGCHI conference on Human factors in computing systems, Atlanta, Georgia, United States, ACM Press.

Feinman, A. and R. Alterman (2003). Discourse Analysis Techniques for Modeling Group Interaction. Ninth Internation Conference on User Modeling.

Gamma, E., R. Helm, et al. (1995). Design Patterns, Addison-Wesley.

Hutchins, E. (1996). Cognition in the Wild, MIT Press.

Introne, J. and R. Alterman (2003). "Leveraging Collaborative Effort to Infer Intent." Ninth Internation Conference on User Modeling.

Johansen, R. (1984). Teleconferencing and beyond: communications in the office of the future. New York, NY, USA, McGraw-Hill, Inc.

Jones, S. H., R. H. Barkan, et al. (1987). Bugnet: A Real Time Distributed Programmng Environments. SRDS.

Kurlander, D. and S. Feiner (1992). A history-based macro by example system. Proceedings of the 5th annual ACM symposium on User interface software and technology, Monteray, California, United States, ACM Press.

Landsman, S. (2006). A Software Lifecycle for Building Groupware Applications: Building Groupware On THYME, Brandeis University.

Landsman, S., R. Alterman, et al. (2001). VesselWorld and ADAPTIVE, Dept of Computer Science, Brandeis University.

Larusson, J. A. and R. Alterman Integrating collaborative technology into the interdisciplinary classroom.

Lorie, R. A. (1977). "Physical integrity in a large segmented database." ACM Trans. Database Syst. **2**(1): 91-104.

Morse, E. and M. P. Steves (2000). CollabLogger: A Tool for Visualizing Groups at Work. Proceedings of the IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Neal, A. S. and R. M. Simons (1983). Playback: A method for evaluating the usability of software and its documentation. Proceedings of the SIGCHI conference on Human Factors in Computing Systems, Boston, Massachusetts, United States.

Ronsse, M., K. D. Bosschere, et al. (2003). "Record/replay for nondeterministic program executions." Communications of the ACM **46**(9): 62-67.

Satyanarayanan, M., D. Steere, et al. (1992). Transparent logging as a technique for debugging complex distributed systems. Proceedings of the 5th workshop on ACM SIGOPS European workshop.

Stefik, M., D. G. Bobrow, et al. (1987). "WYSIWIS Revisited: Early Experiences with Multiuser Interfaces." ACM Transactions on Office Information Systems **5**(2): 147 - 167.

Steven, J., P. Chandra, et al. (2000). jRapture: A Capture/Replay tool for observation-based testing. Proceedings of the International Symposium on Software Testing and Analysis, Portland, Oregon, United States, ACM Press.

Suchman, L. and R. Trigg (1991). Understanding Practice: Video as a Medium for Reflection and Design. Design at Work: Cooperative Design of Computer Systems. M. Kyng. Hillsdale, New Jersey, Lawrence Erlbaum Associates**:** 65-89.

Yang, Z. and T. A. Marsland (1992). Global snapshots for distributed debugging. Fourth International Conference on Computing and Information.

Using Transcription and Replay in Analysis of Collaborative Applications

Seth Landsman, Ph.D. – The MITRE Corporation

Richard Alterman, Ph.D. – Brandeis University

MITRE

Overview

- Motivation and Problem Statement
- Analysis of Collaboration
- Software Model for Transcription and Replay
  - Instrumentation
  - Generation
- Conclusions and Ongoing Work

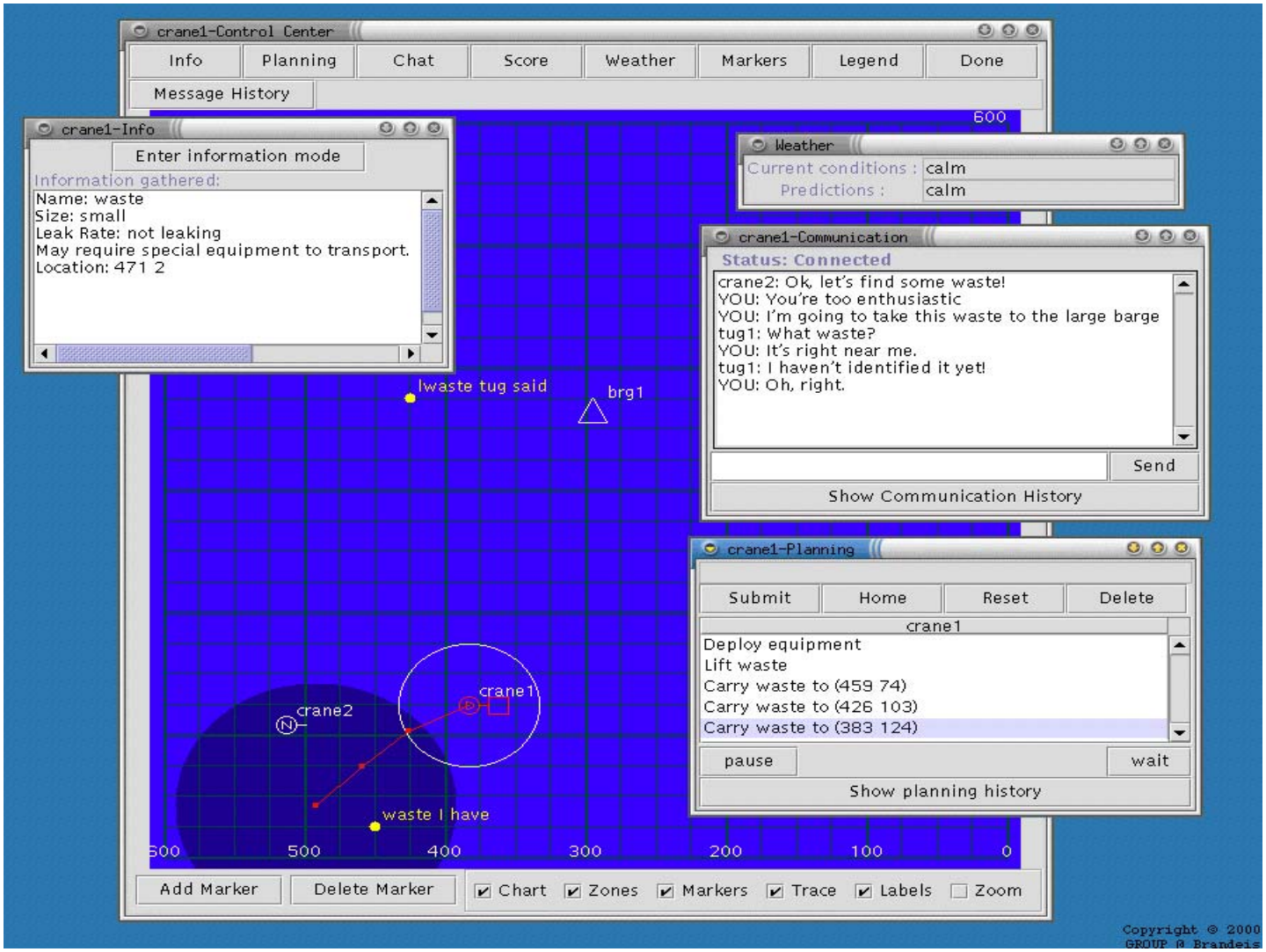
Motivation

- Successful execution of a C2 operation is increasingly a distributed enterprise mediated by a software system
- Success of an operation, therefore, depends on
  - Success of the underlying software and task environment
  - Success of the collaboration mediated by the software system
- How can we increase the likelihood of success in a collaborative environment?
  - How can we ensure that collaboration is tailored to the task environment and user community?
- Being able to understand how the user community uses the C2 system may be able to aid in the engineering of more successful systems
- Our approach is the in system replay of collaborative activity to support ethnographic analysis of the users' actions

VesselWorld

- The VesselWorld system is a simple, synchronous collaborative application used to study challenges in software mediated collaboration
  - Three users work to solve a cooperative problem
    - Each user has a different role in the problem solving
      - Each role has different capabilities
      - Explicit coordination of activity is required to complete the task
- To understand the collaboration as mediated via the application, ethnographic analysis was performed
  - Analysis indicated that users structured their communication over domain objects and planning
  - Enhancements were added that provided tracking of domain objects, and short term and long term planning
- Long development cycle followed by imprecise analysis
  - Experimenter notes, observations of collected data

VesselWorld



Lessons Learned from VesselWorld

- Adding new capabilities to VesselWorld was expensive:
  - Time consuming to build
  - Hard to enhance once built
- How can we be more precise in the enhancements or changes made to a collaborative application?
  - Shorten the feedback loop between implementation and analysis
- How can we be more precise in how collaborative improvement and issues are observed
  - Improve the tools used to study collaborative activities
  - Improve the analysis methods we have available, as supported by the tools
- As the complexity of collaborative applications increases, the need for techniques to construct applications that are appropriate for the task and user community become more critical

Analysis of Collaboration

- Existing techniques help in the construction of collaborative application
  - Rapid development techniques (Roseman and Greenberg, 1992, Pedersen, et al, 1993, Li, et al, 1999)
  - Ethnography (Suchman and Trigg, 1991, Neal and Simons, 1983)
  - Analysis Techniques (Feinman and Alterman, 2003, Feinman, 2006, Lárusson, 2010)
- Each of these techniques provide a piece of the puzzle
  - How can we engineer applications quickly, figure out what information to collect, and do something with the information once it is distilled
- However, the fundamental question of how to collect and work with the user activity is unanswered

Software Model of Transcription and Replay

- Our approach is what we call a “within system perspective” of user activity
  - Compared to having a video camera focused on a user’s screen
  - Over the shoulder view of the user’s activities
- The user activity from the perspective of system events, not UI events, is captured and transcribed
  - Capture chat utterances or planning activity, not key presses and mouse clicks
- The result is that the user activity can be replayed from an individual user perspective or an omniscient perspective
- Our model is implemented into two frameworks
  - THYME is the collaboration construct toolkit that generates the transcripts
  - SAGE is the set of replay components that are applied to a THYME application

Transcription

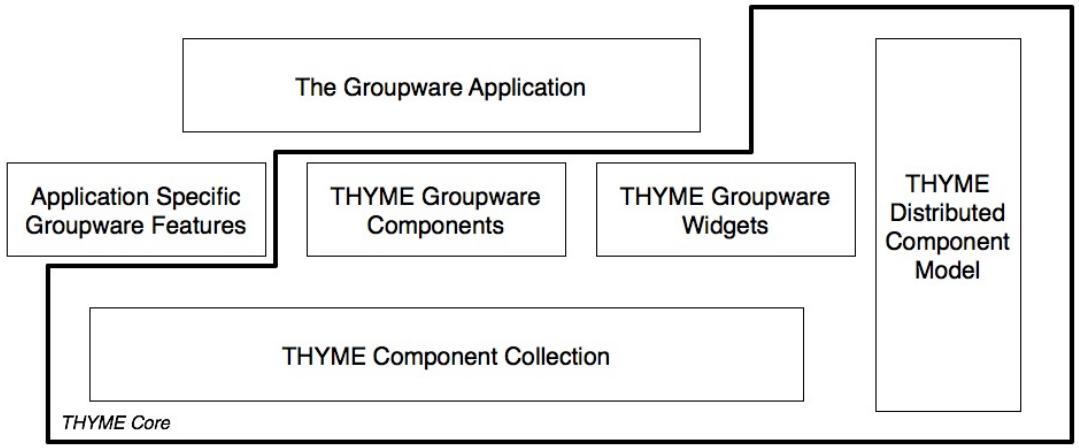
- Collection of interaction between the application and the users
- Features of the transcription capability influence the replay capabilities
  - Completeness
    - Both the amount of information and details
  - Types of information collected
    - Mouse events, chat events, etc
  - Transitions versus States
    - Each atomic unit in the transcript is the system state or an event
- Customized transcription gives most fidelity of information, but is expensive to implement on a per-application basis
  - Internal transcription is next best (e.g., Morse and Steves, 2000)
  - External transcription lacks information context (e.g., Suchman and Trigg, 1991)

Replay

- Allows ethnographic analysis of groupware application use
  - Online behavior can be captured and recreated exactly through a transcript
- Basis replay capabilities are similar to playing a video tape
  - Features enhance the analysis
    - Precision
    - Search
    - Annotation
- How can transcription and replay be accomplished without significant impact to deployment schedule?
  - Leverage system infrastructure
  - Make replay cheap

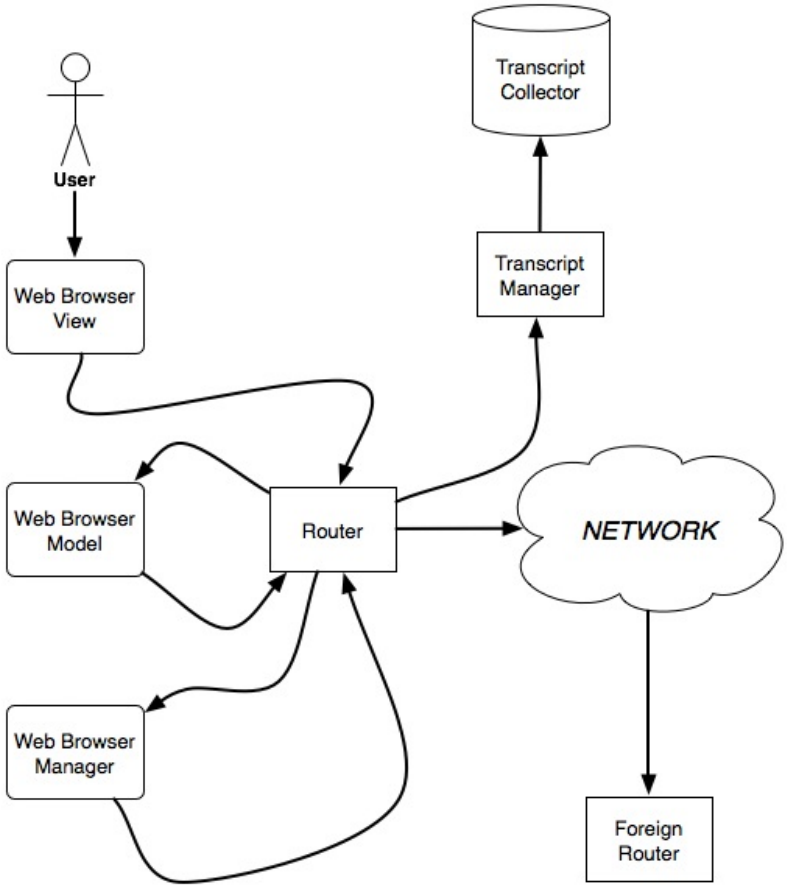
Frameworks

- THYME
  - Framework for building component-oriented groupware applications
    - Includes transcription capabilities
    - Model of development encourages localized changes
  - Rich library of groupware widgets and components
- SAGE
  - Class library for replaying THYME applications
    - Includes capability to generate replay applications from a THYME application



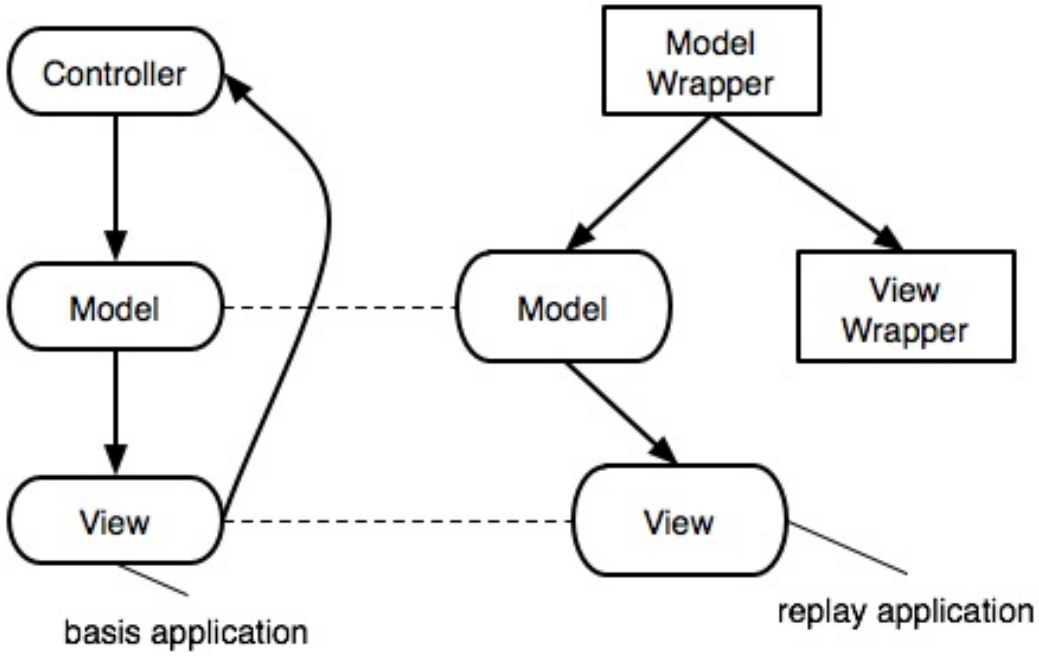
Instrumentation

- Interaction is collected into an ordered transcript of messages
  - Interaction between components
  - Interaction between the user and the system

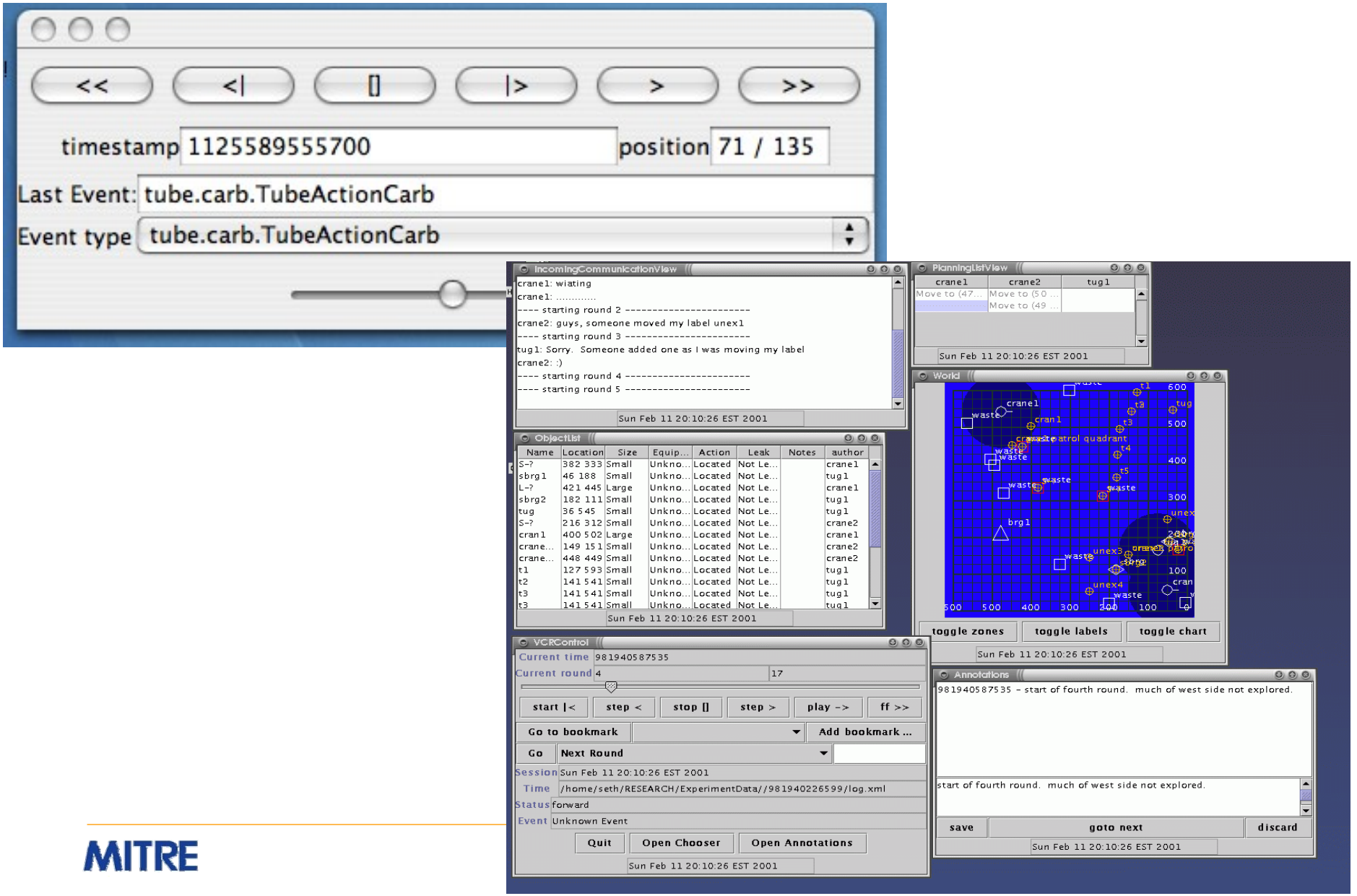


Generation

- Individual components from the basis THYME application are used in the SAGE application
  - Cheaper development
  - Ensures accuracy of the representation



SAGE for VesselWorld



MITRE

Ongoing Work

- There is demonstrated benefit to replay of collected usage data for improving collaborative activity
  - More examples in the paper
- However, doing so requires an investment
  - THYME and SAGE reduce that benefit, but it was still an upfront investment to build the frameworks
- Infrastructure has come a long way since we wrote THYME, specifically
  - More introspectable component architectures in J2EE, Microsoft Web Services, etc
  - More distributed architectures in ESBs and general messaging architectures
- How can we leverage these architectures to enable transcription and replay on more general systems?

Conclusions

- Analysis of collaborative applications is key need for building maintainable, adaptable, and usable applications
  - The application changes during its lifetime
    - Building the application is insufficient, it must be analyzed, modified, and redeployed
    - These activities must be factored into the engineering process
  - The proposed system shows how to accomplish the analysis task
- THYME and SAGE are example implementations of the software support necessary for this analysis
  - Automatic transcription of use
  - Generation of replay application
- This work is a first step on being able analyze and learn from a user community's behavior in situ