In the *Proceedings of the 10th International Conference on Artificial Intelligence in Education*.

A Framework for Asynchronous Collaborative Learning and Problem Solving

Bradley Goodman[1], Marty Geier[2], Lisa Haverty[3], Frank Linton[1], and Robert McCready[4]
[1]*The MITRE Corporation, 202 Burlington Road, Bedford, MA 01730 USA*
[2]*Georgia Institute of Technology, 148B College of Computing, Atlanta, GA 30332 USA*
[3]*Outside the Classroom, 275 Grove Street, Auburndale, MA 02466 USA*
[4]*Predictive Networks, Inc., 689 Massachusetts Avenue, Cambridge, MA 02139 USA*

Abstract. Research has shown that classroom learning improves significantly when a student participates in learning activities with small groups of peers. This educational value of student collaboration has led to the development of computer-supported collaborative learning (CSCL) tools. These tools enrich learning in a setting that encourages students to communicate with their peers. In typical web-based collaborative learning environments, however, it is not always possible for all learners to gather and participate in a learning activity at the same time. Asynchronous and not just synchronous learning must be addressed. In this paper a framework to provide rich asynchronous services to web-based students is described. A key component of this environment is the use of replay to enhance asynchronous learning. Replay provides persistence of objects and actions over time.

1. Introduction

As academic, business, and government communities move from institutional classroom instruction to single-pupil web-based distance learning, students risk losing critical opportunities to collaborate with other students. Yet research in education has shown that classroom learning improves significantly when students participate in learning activities with small groups of peers [2, 8]. Students learning in small groups encourage each other to ask questions, explain and justify their opinions, articulate their reasoning, and elaborate and reflect upon their knowledge, thereby motivating and improving learning. They can bring different strengths and expertise to bear. Recognition of the educational value of student collaboration has led to the introduction of conventional groupware tools - such as chat, threaded discussions, and email - into distance-learning environments. While these tools can facilitate didactic interactions between learners, they cannot ensure productive learning dialogues between participants and they do not address how to provide as rich a learning environment for asynchronous students as synchronous students. We believe that facilitating problem-based learning between peers [5, 10] by having them solve real world problems is a key to providing effective web-based distance learning for both synchronous and asynchronous students. In this research we have focused on instituting an environment to promote effective collaborative distance-learning through the establishment of rich communication and a common workspace for synchronously and asynchronously

connected students. All communication and workspace objects and actions from both synchronous and asynchronous sessions can persist over time for examination by students.

Today's collaborative learning and problem-solving environments afford the opportunity to bring together different learners to jointly tackle a problem. A student in one location can connect over the web and interact with students in other locations. Current collaborative environments concentrate on providing communication between participants and tools to facilitate collaborative activities such as shared whiteboards and shared applications. As the use of collaborative environments becomes more ubiquitous, we can expect many of the same problems facing colleagues physically meeting together to arise in cyberspace. Web-based collaborative learning and problem-solving typically require participants, just like co-located learners and problem solvers meeting to work on a common task, to schedule and gather on-line in a collaborative environment at a preset time and virtual location. Working around these constraints in most collaborative systems typically entails the use of email and threaded discussions. These approaches help address asynchronous collaboration but lack the persistence of objects and actions over time from synchronous and asynchronous sessions that can provide the full learning and problem-solving context most helpful to learners. Records of synchronous interactions other than the current state of a solution of a problem are not stored and available for review and continuation by asynchronous participants.

There have been a number of approaches taken to providing synchronous collaborative services to users. We discuss here how synchronous interaction is commonly provided. Application sharing applications like Netopia's Timbuktu' [20] and Microsoft's NetMeeting' [13] achieve a common perspective by providing a view onto applications running on one of the user's computing environments. Each user can remotely take control of the applications but that user is dependent on the owner of the shared application keeping the environment running. Application sharing provides a general solution across an operating system and associated windowing environment. NetMeeting relays window event actions between computers to provide a consistent view onto a set of running applications. A user interface gesture is transmitted from the originating computer to the running application for interpretation. The results of those actions are mirrored on the other computers.

The Habanero system from the National Center for Supercomputing Applications (NCSA) follows a different approach [3, 7]. It views collaboration in a client-server paradigm. A Habanero collaboration server is launched. Users then run their own Habanero client. Each user can log his client into the Habanero server when he wants to collaborate with other users. When a user logs in, his workspace is configured to that of the other users – ie., all tools running on the client are synchronized to the same state as the tools running on the other clients. Any action taken by a user on one of his tools is sent to the server for broadcast to all clients so that they may execute the same action to keep the whole group's tools in synchronization. The entrance and exit of a user will not affect the state of the tools of the rest of the group.

Habanero was developed in the Java framework and works only with user applets written in Java. NCSA Habanero requires Java applets to be modified to run collaboratively as "Hablets" in the Habanero environment. Any (preplanned) action that changes the state of the Hablet in the environment can be sent to all other clients so that they execute the same action to provide a synchronous environment. Developers must prepare their Java code to utilize Habanero code to maintain state.

The MatchMaker system [21, 22] provides synchronous collaborative services in a manner closest to the methodology described in this paper. MatchMaker provides

synchronous services by coupling user interface objects. All user interface events affecting one user interface object are broadcast to the other coupled objects [12]. Hence, any action by a user on one application's user interface is executed on the other user interfaces to keep the applications synchronized. MatchMaker provides coupled netorked activities as part of a computer-integrated classroom [22].

This paper describes our collaboration environment and our asynchronous learning tool, the results of a pilot study, and contrasts it with other approaches to asynchronous interaction.

2. Tools for asynchronous interaction

Asynchonous contact between users is an important part of collaborative distance-learning. Yet, not all collaboration environments provide asynchronous services. NetMeeting and Timbuktu were not designed for asynchronous interaction. Habanero, however, had asynchronous interactions in mind from the beginning. For example, Habanero is capable of catching up a newly connected user to the same state as all current synchronous users. A simple replay tool was also developed that captured all actions that were sent from the Habanero server to a client so that those actions could be played back at a later point in time. An initial version of the Habanero replay tool came out but further development did not occur until recently – leading us to build our own.[1]

Apple Computer developed an educational research tool, Media Fusion, which was integrated with Apple Quicktime™. Media Fusion took an initial look at rich, asynchronous interaction. The tool integrates video, data analysis, and communication under a paradigm called model-based communication [1]. Model-based communication allows one to construct digital video or text email messages that contain embedded pointers to application software [1]. The sender advocates an opinion in email; the linkage of the email to the data and the data analysis tool supports student reflection through direct manipulation of the data to support the sender's opinion. Media Fusion expands an email message to permit the inclusion of event-based interactions with the data analysis tool. To accomplish this feat, the data analysis application software was modified so that an appropriate Application Programmer Interface (API) existed to permit control of the software externally. The Apple software demonstrates the power of providing a rich environment for replay and presents one potential solution to furnishing a replay feature.

Another approach to the use of replay has been in collaborative educational simulations. Plaisant et al. [14] developed a simulation environment that can record simulation events and actions for later replay. The environment permits recording of actions, annotation of the recorded environment, revision upon replay to the environment, search facilities, and macro capabilities. The environment provides many benefits by permitting students to review and reconsider their work, pass work to peers and mentors for comment, and a rich way for instructors to monitor students to look for problems [14]. A pilot study demonstrated the value of replay to users of the simulation environment. Students felt replay made instruction stronger [14].

Our Synchronous and Asynchronous Interactive Learning Environment (SAILE) and its Asynchronous Replay Tool (ART) can help address similar issues. They provide a general approach to sharing tools and promoting collaborative learning and problem-solving. The

[1] Habanero 3.0 was recently released adding a full complement of asynchronous features. Users can start a session as a synchronous user and switch in the middle to become an asynchronous user. Replay was also greatly enhanced.

next section describes how SAILE and ART can be used to enrich learning and problem-solving for synchronous and asynchronous users.

3. Synchronous and Asynchronous Collaborative Learning and Problem-Solving

Web-based collaborative environments bring students together on-line to discuss topics and solve problems jointly just as they might if they were physically co-located. However, on-line discussion groups can run into the same problems of missing participants and resources as co-located groups. For example, consider the hypothetical situation of a group of students in a web-based course who are being taught the principles of logical reasoning. They are divided into problem-solving teams to work on logic problems. Each team must meet to discuss how to solve a new type of logic problem. Suppose the members of each problem-solving team are scattered around the globe making it impossible to easily meet in person. The students meet to discuss the problem and work out a new approach and solution in a collaborative web session launched to bring them together. Unfortunately, as illustrated in Figure 1, the student most expert in logical reasoning is unable to attend the on-line session. Asynchronous collaborative technology can help address this difficulty.

The collaborative environment, SAILE, developed under the research described in this paper functions in the Java framework and follows a server-client paradigm like Habanero and MatchMaker. Unlike Habanero, which sends notice of significant changes of state in a client environment to the server due to wrappers around those portions of Hablet code that cause the change in state, SAILE intercepts all user interface actions on tools without requiring modification of the application code, and sends those actions to the server for broadcast to all other client tools for execution. MatchMaker works in a similar fashion. SAILE's ability to capture tool user interface gestures is due to the instrumentation of the Java environment itself. The Java Observation, Scripting, and Inspection Tool (JOSIT) [9] instruments the Java environment to report all user interface actions, through the Java user interface widgets. JOSIT works fully with the Java Swing components' API and partially with the Abstract Window Toolkit (AWT) components' API. All user interface gestures are captured and reported to the Java Remote Method Invocation (RMI) Server where they are sent to all the other clients so that they can update to the same state as the user. This process is illustrated in Figure 2. One user's tool actions are broadcast to all other users' tools to preserve state. A user's interface gesture, such as clicking a button on the interface, is intercepted by JOSIT and passed to SAILE. SAILE, running in the Java virtual machine (VM) of the client operating system (OS), transmits the captured user interface gesture through the Java RMI server to the clients. The interface gesture is then executed on each client by SAILE and JOSIT. JOSIT grew out of earlier research on the instrumentation of the X-Windows environment. The Widget Observation, Scripting, and Inspection Tool (WOSIT) can intercept all user interface events of an application running in X-Windows without modification of the application itself [23]. WOSIT has been used to develop embedded training systems [4].

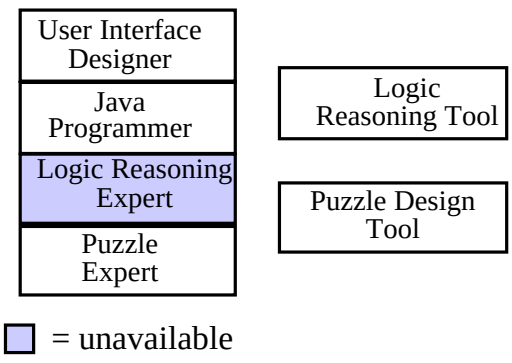


Figure 1: On-line learning situation

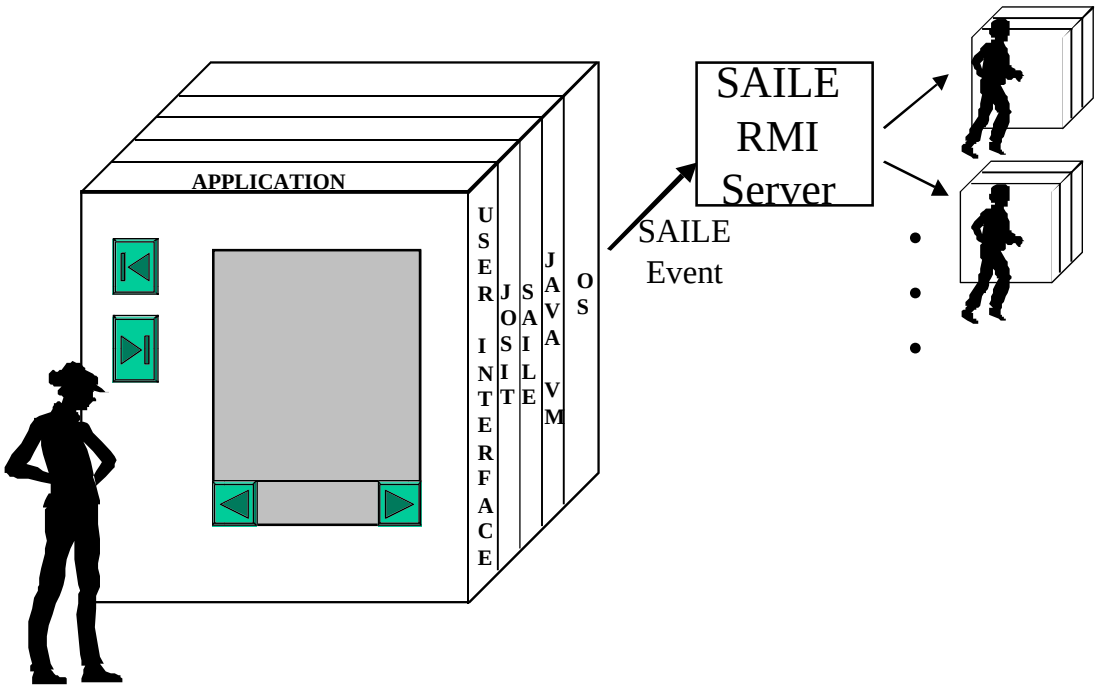


Figure 2: Synchronized actions in SAILE

SAILE differs from Bellamy et al. [1] and Plaisant et. al. [14] by trying to provide a general approach to synchronous and asynchronous interactions. SAILE is composed of a text chat tool that allows users to communicate, a palette to place any Java applications to be shared, a Java RMI server for managing the collaborative connection between users, and the Asynchronous Replay Tool (ART). ART allows an asynchronous learner and problem-solver to become a full participant in a collaborative learning or problem-solving session.

ART provides an opportunity to move beyond time and physical location barriers to address problems such as exemplified in Figure 1. Synchronous learners and problem solvers can work on-line together and still reach out beyond the current session to other learners or experts. The synchronous group can use ART to record their interactions, including text chat, tool usage, and they can add annotations to their recorded interactions pointing out particular events or questions to their missing colleagues. The recorded session built from JOSIT information on each member of the group's user interface actions can be emailed to asynchronous users for later replay.

Asynchronous replay in ART can be accomplished in two distinct modes - as a single user session without other participants or as a synchronous "asynchronous" session in which multiple participants can observe and interact with the replay through the SAILE environment. Playback in ART can take place in several forms. ART permits fast review of all actions, replay of actions in segments delimited by chat events, or step-by-step replay of each action. Problem-solving and learning in SAILE can proceed in a back and forth fashion where one or more participants work on a problem and record their interactions: (1) the recorded session is passed to others for viewing at a later point in time (the asynchronous group could include a subset of the original group) and (2) the asynchronous group works on the problem. The asynchronous group can review the work of the earlier group. They can add their own comments about earlier actions. The group can demonstrate alternative actions by branching off the replay path at some point during the replay (as shown in Figure 3). They can continue the problem-solving activity if it was not completed in an earlier session. Learning and problem-solving could carry on this way until all parties are satisfied with the end result of their efforts. Figure 4 illustrates potential interaction between eight students using SAILE and ART. Each student is indicated by a letter;

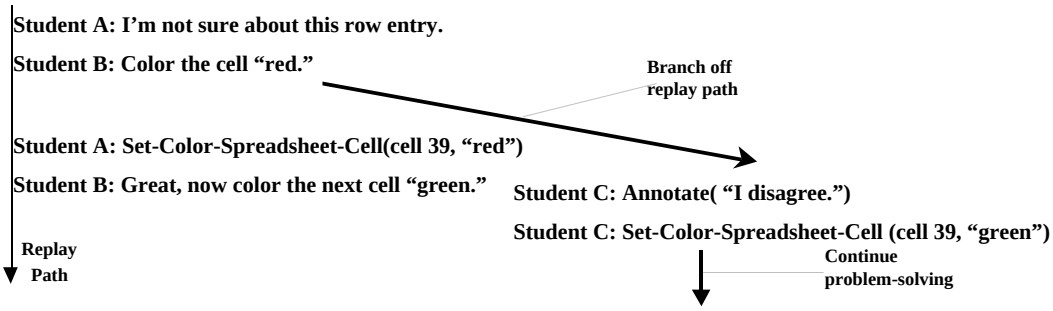


Figure 3: A replay path

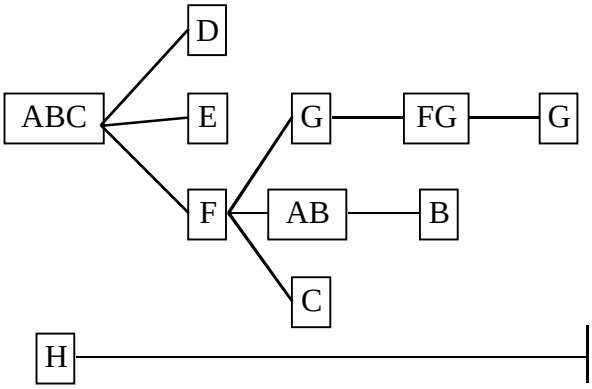


Figure 4: Students collaborating to solve a problem

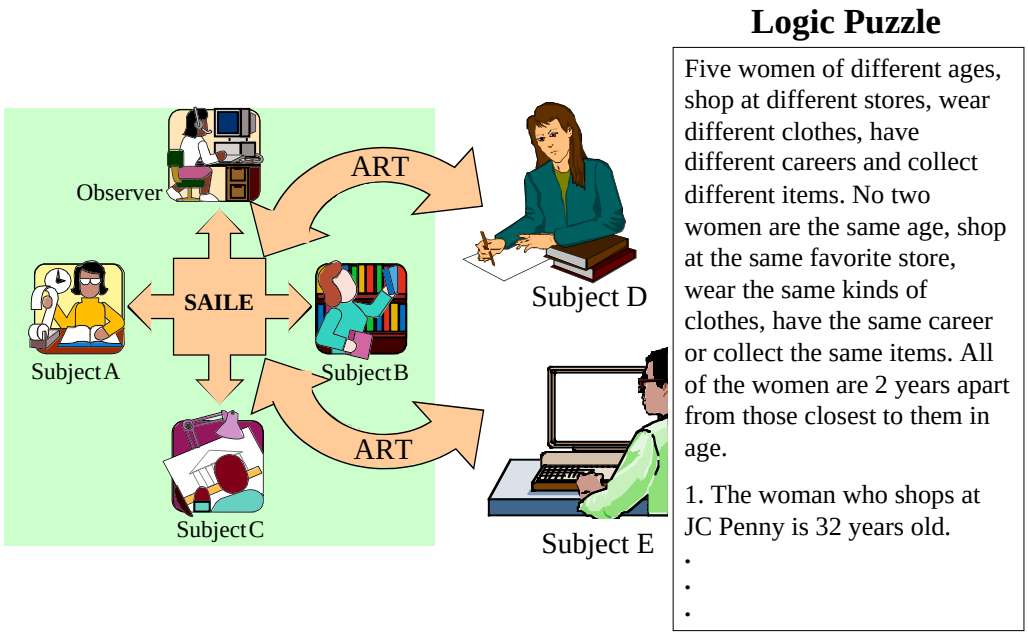


Figure 5: Using SAILE and ART for collaborative problem-solving

students working together synchronously are shown enclosed by a rectangle; asynchronous sessions are connected to their originating session by a line. Time starts on the left side and progresses to the right. Note that student H works on the problem by himself. Students D, E, and F separately tackle the problem by examining the replay of the synchronous problem-solving session of students A, B, and C. Table 1 lists typical group actions and how they might be accomplished in either a synchronous or asynchronous fashion.

SAILE and ART can potentially promote learning and problem-solving along several dimensions. Stahl [19] defines a conceptual framework for collaborative knowledge

Table 1: Potential group actions

Collaborative Learning & Problem-Solving Activity	**Collaborative Interaction** **Synchronous**	**Asynchronous**
Discuss problem and approach	**Text chat** Lisa: I figure we could have a header for age/jewlery/store/occupation Brad: I agree. Mind if I start up the table? Lisa: Okay, go ahead. **Add comments** Brad: Bob, we felt a table was the best representation to use for the data.	**Replay text chat and read comments** **Add comments** Bob: I also think building a diagram can be very helpful in spotting connections.
Attempt solution step	**Perform tool action** Brad: Enter-spreadsheet-cell (“house”, spreadsheet-tool, cell5)	**Replay tool action** **Branch off replay path and perform tool action** Bob: Add-object(square, diagram-tool)
Remediate group interaction	**Text chat** Frank: Lisa and Brad, you’re wrong here. Shouldn’t “house” go in cell 6? **Perform corrective tool actions** Frank: Enter-spreadsheet-cell (null, spreadsheet-tool, cell5) Frank: Enter-spreadsheet-cell (“house”, spreadsheet-tool, cell6)	**Add comment** Bob: I think that cell 6 is where “house” belongs. **Branch off replay path and perform corrective tool actions** Bob: Enter-spreadsheet-cell (null, spreadsheet-tool, cell5) Bob: Enter-spreadsheet-cell (“house”, spreadsheet-tool, cell6)
Describe status for group	**Text chat** Lisa: I think we’re now on course.	**Add comments** Bob: I think you’re on track following this course of action.

building environments that can be used in their assessment. Table 2 illustrates where SAILE and ART support learning along some of Stahl’s phases of knowledge building [19]. SAILE and ART encourage discussion and promote the consideration of alternative courses of action.

In contrast to asynchronous interaction in SAILE and ART, Apple’s Media Fusion accomplishes asynchronous interaction by interweaving tool action with email. For example, a Quicktime email message could be sent to an asynchronous user. The author of the email could be orally describing in the email an event that occurred while running data analysis software. As the event is being discussed in the email, the actual data analysis application software is launched on the email recipient's computer and put into the same state described in the email. The receiving student can modify and manipulate the data analysis environment and respond in the same fashion to the email sender.

4. The Pilot Experiment

Table 2: Computer support for learning and problem-solving

Phase of knowledge building	Form of support
Articulate ideas	SAILE Chat tool ART Annotation tool
Compare perspectives	ART Branching
Consider alternatives	SAILE Chat tool ART Annotation tool
Argue, clarify and negotiate	SAILE Chat tool ART Annotation tool ART Branching
Promote shared understanding	SAILE Chat tool ART Annotation tool ART Branching

Many questions faced us when we built SAILE and ART. Would users recognize the potential for working with others across distance and time? Would users bother annotating their actions to facilitate participation by asynchronous teammates? Would the time spent replaying an earlier session take too long negating the value of observing how a (partial) solution was derived as opposed to just working with the solution itself? Would the folding in of multiple synchronous and asynchronous sessions become too confusing for learners? We ran a pilot experiment to provide some initial answers to the questions and to help us design a more controlled experiment. The primary objective of the experiment was to compare performance and participation levels of synchronous and asynchronous students working on a common problem. A determination whether synchronous collaboration yielded better performance than asynchronous or solitary students was sought. Our hypothesis was that asynchronous collaborative sessions are richer than solitary ones and could approach the level of synchronous collaboration.

We chose a complex logic problem as a test of solving a problem over multiple sessions and with multiple users. The logic problem (shown in Figure 5) presented partial information that had to be carefully integrated into a common framework to reveal missing information. Inferences could then be made to fill in the missing information. There were eight subjects (labeled A through H in Figure 4) that participated in our pilot experiment – two synchronous users, 2 asynchronous users, 3 synchronous and asynchronous users, and one solo user. Subjects were all high performers – engineers and scientists. The experimental infrastructure included SAILE, to provide a multi-user synchronous environment, and ART, to provide “record” and “replay” for asynchronous sessions. To facilitate in the solution of the problem, we provided two collaborative tools that could be employed: a diagram drawing tool and a spreadsheet tool. As one user entered information into a tool, the same information was automatically entered into the corresponding tool of the other participants. The integrated chat tool in SAILE afforded textual communication between participants. Figure 6 shows the SAILE environment configured for the pilot experiment. We wanted to observe numerous potential completions to the problem-solving task so we timed the end of the first session so that the logic problem would only have been partially completed. We encouraged participants to annotate their work for asynchronous partner(s).

Experimental method: Figure 5 illustrates the setup we followed in the experiment. Synchronous subjects were connected to each other using SAILE. An observer was present in the environment to monitor the experiment and to note any problems with the software.

Figure 6: Synchronous and asynchronous interactive learning environment (SAILE)

Students could use the chat tool to communicate with each other. The diagram drawing and spreadsheet tools could be used to help solve the problem. Students saved their session to a file for distribution to asynchronous students (D and E in Figure 5). ART was employed by them to replay the session. The asynchronous students had full access to chat and tool events between synchronous students. They could add comments or take control of the tools to solve the problem their way at any point in the replay.

The experiment was designed so that we could examine a number of potential solution paths. Figure 4 shows the paths collected during the pilot experiment. Each student is represented by a letter; students working synchronously together are grouped in a box; asynchronous sessions are connected to their originating sessions by a line; time proceeds to the right. A solo student solved the problem to provide a baseline for time to completion of the problem. A synchronous group of students partially completed the problem and passed the results to a single asynchronous student. Lastly, a synchronous group of students worked on the problem and sent a partial solution to multiple asynchronous and synchronous sessions to develop a full solution. Some members of the original synchronous session participated in the follow-up sessions. Data was collected on whether or not a successful solution was derived, time to complete a full solution, time on task, replay time, and interaction form (i.e., communication between participants and the thrust of that communication). A parser was written in Perl to extract data for analysis from the SAILE logs of each protocol.

Experimental Results: While the primary goal of this pilot study was to help shape future experiments that provide a more controlled study, some interesting observations are worth pointing out.

1. People tended to wait until a whole session was replayed before jumping in and taking control of the tools. One possible explanation is that no glaring mistakes occurred in the protocols that might warrant someone immediately stopping a replay and taking control of the tools.

2. Some asynchronous students would switch representation formalism upon the completion of replay (e.g., switching from spreadsheet form to diagram form). Their individual preferences seem to be the driver for the change.

3. As expected multi-session asynchronous/synchronous groups took longer in most cases to solve the problem in total, even after deducting replay time, than someone who solved the problem from beginning to end by himself. Times ranged from 31 to 93 minutes, averaging 54 minutes (Figure 7). This increase of 23 minutes is not necessarily unreasonable given the value received by asynchronous users from reviewing earlier sessions. Replay time itself was not a significant factor (ranging from 11 to 19 minutes per session as seen in Figure 8).

4. Participants varied in their use of chat and annotation and the employment of the spreadsheet tool and diagram drawing tool during problem-solving. Figure 9 follows one solution path in the pilot study and indicates the number of times each technique was utilized.

5. Users reported at the conclusion of the experiment that they liked the replay tool and felt it did not detract from success of problem-solving. They preferred replay to receiving only end-state information from an earlier session.

6. Users in all conditions (synchronous, asynchronous, and synchronous asynchronous) worked out a solution to the problem successfully. All participants expressed a positive attitude towards the collaborative tools provided in SAILE. Replay was effective for bringing users up-to-speed.

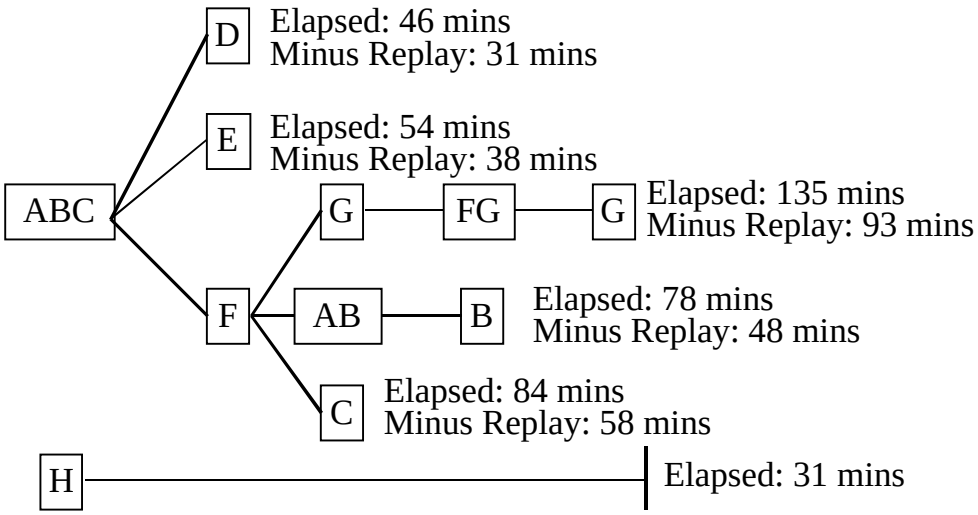
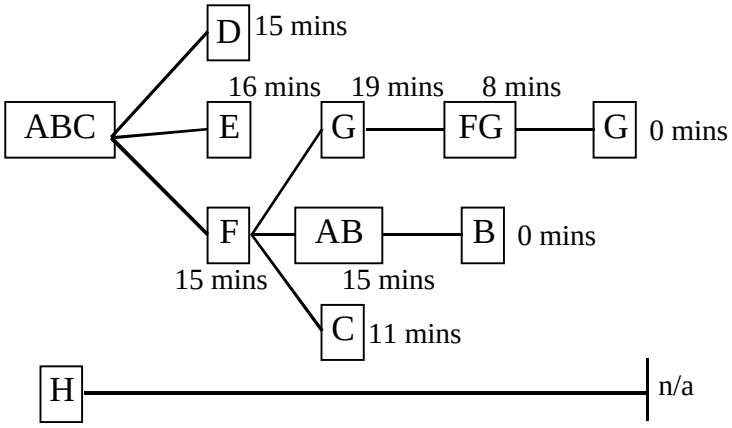
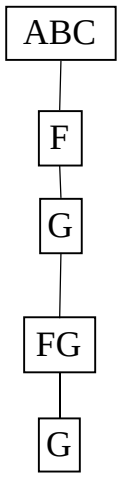


Figure 7: Time to solution



Range: 11-19 minutes

Figure 8: Time to replay prior sessions



	Chat	Spreadsheet entries	Diagram
A	17	7	0
B	25	17	0
C	18	16	0
F	5	4	68
G	12	17	23
F	37	0	16
G	21	0	17
G	25	23	60

Figure 9: Tool usage on one solution path

5. Conclusions and future directions

Our pilot study provided initial evidence that SAILE and ART can add benefit to collaborative learning sessions – both synchronous and asynchronous. A more controlled study is now required to determine the effectiveness of the Asynchronous Replay Tool. In the full study, we will use only asynchronous students. Subjects will be provided with a synchronous session to replay. The replay session will include errors to see if they prompt corrective action by the subject in the form of annotations and branching off the replay path to fix the mistakes.

SAILE and ART are one piece of a multi-prong approach to providing collaborative learning services that we have undertaken. We have studied the dynamics of collaborative learning groups [16, 17] by tracking the communicative and tool actions of participants. Our studies have shown the value of employing the underlying communicative acts of collaborative learning dialogues as a predictor of the effectiveness of a collaborative learning session [18]. In earlier research we developed a simulated learning companion capable of acting as a peer in an intelligent tutoring system [6]. The presence of the simulated peer can help ensure the availability of a capable collaborator to enrich learning through the promotion of reflection and articulation in the human student. SAILE was developed as a platform to integrate our research in collaborative learning. The instrumentation of student interaction in SAILE using JOSIT and a communicative dialogue act tagger (cf. [15]) will feed an algorithm to determine the effectiveness of a collaborative learning session. Our overall goal is to develop an intelligent agent that can co-exist in cyberspace with human collaborators and interact as a full partner to promote effective collaborative learning and problem-solving When learning is not proceeding on course for the participants, the simulated peer can interact with the students to help steer the group back on track.

6. Acknowledgements

The research described in this paper was performed while the authors were at the MITRE Corporation. Funding was provided by the MITRE Technology Program, grants 51MSR80C and 51MSR104.

References

[1] Bellamy, Rachel, Grant, Wayne, Cooper, Eric, Borovoy, Rick, and Adams, Steve (1994). Media Fusion: A Tool that Supports Experience, Reflection, and Collaboration, Apple Computer ACOT Report, Cupertino.

[2] Brown, A. & Palincsar, A. (1989). Guided, cooperative learning and individual knowledge acquisition. In L. Resnick (Ed.), Knowledge, learning and instruction, Lawrence Erlbaum Associates, pp. 307-336.

[3] Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., and Seguin, C. (1998). Java Object-Sharing in Habanero, *Communications of the ACM*, 41, pp 69-76.

[4] Cheikes, B. A., Geier, M., Hyland, R., Linton, F., Riffe, A. S., Rodi, L. L., Schaefer, H. P. (1999). Embedded Training for Complex Information Systems. *International Journal of Artificial Intelligence in Education*, 10, pp. 314-334.

[5] Collins, A., Brown, J.S., and Newman, S. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In L.B. Resnick (Ed.) Knowing, learning and instruction: Essays in honor of Robert Glaser, Hillsdale, NJ: Lawrence Erlbaum.

[6] Goodman, B., Soller, A., Linton, F., and Gaimari, R. (1998). Encouraging Student Reflection and Articulation using a Learning Companion. *International Journal of Artificial Intelligence in Education*, 9, pp. 237-255.

[7] Jackson, L., and Grossman, E. (1999). "Integration of Synchronous and Asynchronous Collaboration Activities", *ACM Computing Surveys*, 31.

[8] Johnson, D., Johnson, R., & Holubec, E. (1990). Circles of learning: Cooperation in the classroom (3rd ed.). Edina, MN: Interaction Book Company.

[9] JOSIT (2000). http://www.mitre.org/tech_transfer/josit/.

[10] Koschmann, T. (Ed.) (1996). CSCL: Theory and practice of an emerging paradigm. Mahwah, NJ: Lawrence Erlbaum.

[11] Matsuura, Kenji, Ogata, Hiroaki, Yano, Yoneo (2000). Agent's Contribution for an Asynchronous Virtual Classroom. In G. Gauthier, C. Frasson, and K. VanLehn (Eds.), Proceedings of the 5th International Conference, Intelligent Tutoring Systems (ITS2000), Springer-Verlag, Berlin, pp. 344-353.

[12] Mühlenbrock, M., Tewissen, F., Hoppe, H. U. (1998). A Framework System for Intelligent Support in Open Distributed Learning Environments. In *International Journal of Artificial Intelligence in Education*, 9, pp. 256-274.

[13] NetMeeting (1999). http://www.microsoft.com/window/NetMeeting/Features/default.ASP.

[14] Plaisant, C., Rose, A., Rubloff, G., Salter, R., Shneiderman, B. (1999). The design of history mechanisms and their use in collaborative educational simulations. Proc. of the Computer Support for Collaborative Learning, CSCL' 99, Palo Alto, CA, pp. 348-359.

[15] Samuel, Ken, Carberry, Sandra, and Vijay-Shanker, K. (1998). Dialogue Act Tagging with Transformation-Based Learning. Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1150-1156.

[16] Soller, A., Goodman, B., Linton, F., and Gaimari, R. (1998). Promoting Effective Peer Interaction in an Intelligent Collaborative Learning Environment. Proceedings of the Fourth International Conference on Intelligent Tutoring Systems (ITS 98), San Antonio, TX, pp. 186-195.

[17] Soller, A., Linton, F., Goodman, B., Lesgold, A. (1999). Toward Intelligent Analysis and Support of Collaborative Learning Interaction. Proceedings of the Ninth International Conference on Artificial Intelligence in Education, LeMans, France, pp. 75-82.

[18] Soller, A., and Lesgold, A. (2000). Modeling the Process of Collaborative Learning. International Workshop on New Technologies in Collaborative Learning, Tokushima, Japan.

[19] Stahl, G. (2000). A Model of Collaborative Knowledge-Building. In B. Fishman and S. O'Connor-Divelbiss (Eds.), Fourth International Conference of the Learning Sciences, Erlbaum, Mahwah, NJ, pp. 70-77.

[20] Timbuktu (2000). http://www.netopia.com/software/.

[21] Tewissen, Frank, Baloian, Nelson, Hoppe, Ulrich, Reimberg, Erich (2000). "MatchMaker" Synchronising Objects in Replicated Software-Architectures. Proceedings of the 6th International Workshop on Groupware CRIWG 2000, Madeira, Portugal, IEEE CS Press.

[22] Tewissen, Frank, Lingnau, Andreas, Hoppe, H. Ulrich (2000). "Today's Talking Typewriter" supporting Early Literacy in a Classroom Environment. In G. Gauthier, C. Frasson, and K. VanLehn (Eds.), Proceedings of the 5th International Conference, Intelligent Tutoring Systems (ITS2000), Springer-Verlag, Berlin, pp. 252-261.

[23] WOSIT (1997). http://www.mitre.org/technology/wosit/.