AN AGENT-BASED SYSTEM FOR CAPTURING AND INDEXING SOFTWARE DESIGN MEETINGS

TRACY HAMMOND, KRZYSZTOF GAJOS
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139, USA
{hammond, kgajos} @ai.mit.edu

and

RANDALL DAVIS, HOWARD SHROBE
{davis, hes} @ai.mit.edu

Abstract. We present an agent-based system for capturing and indexing software design meetings. During these meetings, designers design object-oriented software tools, including new agent-based technologies for the Intelligent Room, by sketching UML-type designs on a white-board. To capture the design meeting history, the Design Meeting Agent requests available audio, video, and screen capture services from the environment and uses them to capture the entire design meeting. However, finding a particular moment of the design history video and audio records can be cumbersome without a proper indexing scheme. To detect, index, and timestamp significant events in the design process, the Tahuti Agent, also started by the Design Meeting Agent, records, recognizes, and understands the UML-type sketches drawn during the meeting. These timestamps can be mapped to particular moments in the captured video and audio, aiding in the retrieval of the captured information. Metaglue, a multi-agent system, provides the computational glue necessary to bind the distributed components of the system together. It also provides necessary tools for seamless multi-modal interaction between the varied agents and the users.

1. Introduction

Design rationale has been defined in a variety of ways, but all definitions concur that design rationale attempts to determine the *why* behind the design (Louridas and Loucopoulos, 2000; Moran and Carroll, 1996). Design rationale is the externalization and documentation of the reasons behind

Appeared in *Proceedings of International Workshop on Agents In Design, WAID'02,* pp.203-218

design decisions, including the design's artifact features. For the purposes of this paper, we choose the following definition borrowed from Moran and Carroll (1996) for design rationale: Documentation of (a) the reasons for the design of an artifact, (b) the stages or steps of the design process, (c) the history of the design and its context. Louridas and Loucopoulos claim that the design rationale research field includes all research pertaining to the capture, recording, documentation, and effective use of rationale in the development processes. The researchers state that a complete record, by which they define to be a video of the whole development process, combined with any materials used and produced, could, in theory, be used to find the rationale behind the decisions taken. However, they claim that this unformatted data would be unwieldy through which to process and search. Thus design rationale research has generally encouraged the structuring of design to provide a proposed formalism using a small set of concepts appropriate for representing the deliberations taking place.

A considerable body of effort has been devoted to capturing and indexing design rationale. One part of design rationale is documentation of the design history (Louridas and Loucopoulos, 2000; Moran and Carroll, 1996). While videotaping a design session can capture the design history, retrieval may require watching the entire video. Retrieval can be made simpler by structuring the design process, but this can hold back fast-flowing design meetings (Shum et al., 1996). There is an apparent tension between the simplicity of design rationale capture and effectiveness of design rationale retrieval (Shipman and McCall, 1997). We hope to bridge this gap by allowing designers to design as they would naturally, yet also supply the tools that understand those designs and allow the designer to use this understanding to help in retrieving appropriate moments of a design meeting.

This paper addresses the use of advanced multi-modal tools to aid in collaborative design meeting indexing. In particular, we are concerned with the MIT AI Lab's Intelligent Room (Hanssens, et al. 2002), a mature, yet still evolving, system. The software infrastructure behind the Intelligent Room is a multi-agent system called Metaglue (Coen, Phillips, et al. 1999). Metaglue currently supports robust communication among distributed agents, complex resource discovery and management mechanisms, as well as support for multi-modal interactions through speech, gesture, graphical UI's, web interfaces, and other sensory channels.

Traditionally, when new components need to be added to the Intelligent Room's software, a small number of designers gather in the Room and sketch the new design on the whiteboards while discussing their decisions. What gets recorded after those sessions is the final design and the

explanation of the mechanisms employed. What gets omitted, however, are the reasons *why* those particular solutions got employed.

In response, we have created a system that allows software designers to design agents naturally. The designers can draw UML-type free-hand sketches on a whiteboard using an electronic marker whose "strokes" are digital ink projected onto the board rather than drawn on it. These sketches are recorded and interpreted in real-time to aid in the later retrieval of design history. The system allows the users to design as they would naturally, requiring only that they learn the UML syntax. Information extracted from the diagrams can be used by the system to generate stub code, reducing some of the routine part of the programming process. The recognition also allows us to flag, label, and timestamp events as they occur, facilitating retrieval of particular moments of the design history.

Figure 1 is a photo of people who are designing agents in the Intelligent Room. Figure 2 illustrates a free-hand sketch drawn by a designer.

The system presented here is itself an Intelligent Room application composed of a number of agents. The Design Meeting Agent extends the Meeting Management System (Oh, Tuchinda, and Wu, 2001) for capture of non-design information such as the structure of the design meeting. It initializes the Tahuti Agent, which controls the sketch recognition and the timestamping of significant events. It also controls the video and screen capturing agents.

In this paper we focus on the understanding, capture and retrieval of design-related information. The paper begins by exploring the previous work done in this area. Section 3 describes Metaglue, a multi-agent system. Section 4 denotes the agent components involved in the system described here. Section 5 provides further detail on the Tahuti Agent. Section 6 explains the algorithm for ranking significant sketch recognition events. Section 7 defines the user interaction with the system thus far. Section 8 presents the current system use, future work, and contributions.

Figure 1. People designing agents in the intelligent room

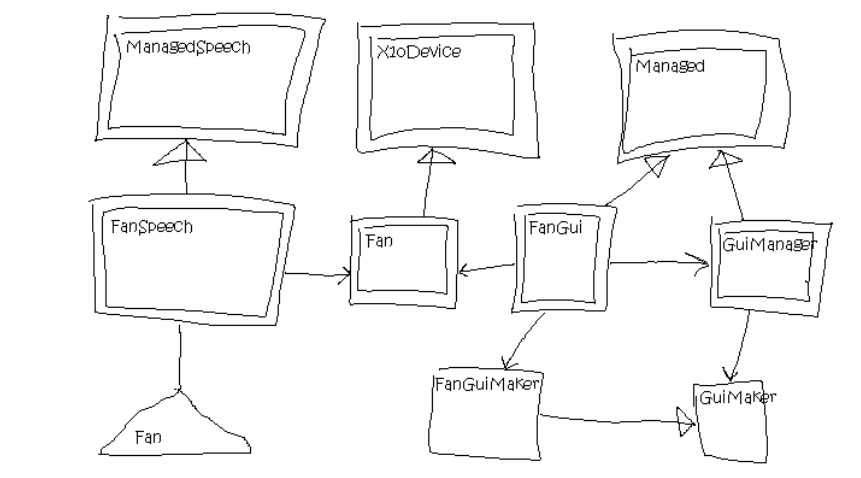

Figure 2. Sketch of Design Diagram

2. Previous Work

Much research has been done on indexing audio-visual material (Brunelli, Mich, and Modena, 1996). Researchers have attempted to label the video with salient features within the video itself, focusing on the recognition and description of color, texture, shape, spatial location, regions of interest, facial characteristics, and specifically for motion materials, video segmentation, extraction of representative key frames, scene change detection, extraction of specific objects and audio keywords.

While not much research has been done using sketch recognition to label and index a particular moment in video, a considerable body of work has been done using sketch recognition to find a particular moment in a pre-indexed video (Kato, Kurita, Otsu, and Hirata, 1992; Cho and Yoo, 1998; Jacobs, FinkelStein, and Salesin, 1995).

UML diagrams have been found lacking simple ways to describe agent-based technologies (Odell, Parunak, and Bauer, 2000). Bergenti and Poggi (2001) have created a CAD system to input UML diagrams for agent-based systems. The system requires designers to enter their diagrams using a rigid CAD interface rather than allowing designers to sketch as they would naturally.

3. Metaglue and the Agent Architecture

This section describes Metaglue, the underlying software infrastructure that the design meeting capture system presented in this paper is built upon. Metaglue, a multi-agent system (MAS), is a foundation of all software developed for the Intelligent Room Project. The rationale for choosing the MAS approach to building software for smart spaces has been explained by Coen (Coen, 1998) but the impact of the approach on our design meeting capture application will be illustrated in this section.

The most important features of Metaglue are:

- **Support for synchronous and asynchronous communication among distributed agents.** The synchronous method calls allow tight coupling among closely collaborating agents that need to exchange large amounts of information quickly. An example would include the central speech recognition engine and the individual speech interface agents controlling spoken interactions with various applications. When the speech recognition engine recognizes a spoken utterance and determines which agent is the intended recipient, it makes a direct method call to that agent passing the information about the recognized phrase. In this case there is only one intended recipient of the communication and timing is critical. In contrast, when a hardware device changes its state, it sends

out a state change notification through the publish-subscribe mechanism. Varieties of meta agents may subscribe to this kind of messages and trigger reactions or simply record the event for future retrieval.

- **Mechanisms for resource discovery and management** (Gajos, 2001). This feature allows agents to refer to one another by their capabilities rather than location or name. For example, an email notification agent may request a text message delivery service, regardless of how it is provided. Depending on context and available resources, this service can be provided by the text-to-speech agent, a scrolling LED sign or an on-wall projected display. In some cases, the pager service might even be used. This level of indirection frees the application creators from having to anticipate or reason about the varying capabilities of different physical environments. It also allows environments and their occupants to exercise their personal preferences on how services are rendered. For example, if the user is on the phone, the resource manager will favor visual over audible renditions of the message delivery service.

  Resource discovery and management services are critical for our project as our software has been deployed in a number of very different spaces such as offices, a conference room, a living room and a bedroom. All of these spaces have very different intended uses and thus the kind, quality and amount of equipment available in them differs dramatically. Metaglue is also capable of arbitrating among conflicting requests from numerous independent applications running in any given environment.
- **Robust recovery mechanisms for failed components** (Warshawsky, 2000). Metaglue adds an extra layer of indirection to all direct method calls. It is used to detect any problems with the target object or the communication channel. In cases where the remote object has failed, Metaglue will attempt to restart it and retry the call before giving up. This feature of Metaglue makes applications relatively immune to many hardware and software failures while keeping the code of the applications simple. Combined with the persistent storage capabilities described below, this makes most of our agents "invincible." Provided they checkpoint their state frequently, in case of failure, the agents will be automatically restarted and given a chance to reload their state before continuing.
- **Built-in persistent storage**. Metaglue provides a convenient mechanism for storing and retrieving arbitrary (serializable) objects. As mentioned above, persistent storage is often used by agents to check point their state. It is also used to store customization information and special purpose application data. Our application is also using this mechanism to store information about the meeting flow and the design process

(Peters 2002). Captured video and audio information are stored directly to a disk location.

- **Support for multimodal interactions through speech, gesture and graphical user interfaces.** Just as popular operating systems provide mechanisms for communicating with users through standard input and output mechanisms available on desktop computers, Metaglue provides means for managing interactions through such channels as speech input and output, distributed graphical interfaces, environmental displays, simple sensors, and complex perception mechanisms based on computer vision. In order to interact with users via speech, Metaglue-based applications need only to provide a grammar describing a set of expected utterances and a handler for speech input events (Coen, Weisman, et al., 1999).

Perhaps the most important feature of Metaglue for the presented system is the run-time composition of elements that comprise the full application through the resource discovery and management system. That implies that the core of the application comprises of just a few lightweight elements. All of the remaining capabilities, such as capture, presentation and storage resources, are obtained at run time from the environment. This allows our system to be run in a variety of environments ranging from relatively impoverished offices where only a single large display is available with no cameras, to the original Intelligent Room lab equipped with 5 projectors, multiple cameras, microphones, etc.

4. Components of the System

In this section we describe the major components of the system, including the mandatory core components as well as the optional but desirable services obtained from the surrounding environment. In the later parts of the paper, where we describe interactions with the system, we will assume that a full suite of desired resources is available. In other environments, the interactions may be scaled down.

The core elements of our system are the Tahuti Agent and Design Meeting Manager, which need to be always present as they manage the entire application. The other components, such as communication, capture and playback services, are dynamically discovered and incorporated into the application based on their availability.

Design Meeting Manager

The Design Meeting Manager extends our earlier Meeting Manager (Oh, Tuchinda, and Wu, 2001). At startup, it is responsible for obtaining resources necessary for running a basic meeting (a display for keeping track of the agenda, issues, commitments, etc) and for starting Tahuti, the sketch recognition part of the system. It is also responsible for negotiating with the environment the use of available audio, video, and screen-capture devices. During the meeting, the Design Meeting Manager, will keep track of the organizational aspects of the meeting such as moving through and augmenting the meeting agenda. It also provides means for querying previous meetings.

Tahuti Agent

The Tahuti Agent is a white-board sketching application for UML based design sketches. The application's primary use is to aid in software design meetings in the Intelligent Room. Since many of the applications designed in the Intelligent Room are perceptually enabled agent based systems, we have included symbols for specifying Agents and Speech Grammars. The Tahuti Agent watches as people in the room write on the white board in the room using, for example, a Mimeo mouse, which sends stroke data to the Tahuti Agent. The Tahuti Agent recognizes UML diagrams as they are sketched, and identifies and time-stamps events as they occur (see Section 6). These timestamps are used to index the video of the design meeting.

Speech Interfaces

Both the meeting manager and the Tahuti Agent can interact with the users through speech. The grammar of the Design Meeting Manager contains vocabulary for controlling the flow of the meeting and querying previous meetings. Tahuti's speech interface allows users to interact with the sketch (e.g. provide feedback in case of misrecognition of drawn shapes) and to query earlier designs (e.g. "What where we talking about when we added this class?").

Meeting Capture Services

There are a number of agents deployed in Metaglue-enabled spaces that can, depending on the availability of hardware and software resources, provide capture services to the Design Meeting Manager. In spaces equipped with cameras, the video capture agent can capture video of the design meeting. In most rooms, audio may also be captured by the audio capture agent. Finally, if the sketching is done on a machine with appropriate software (such as Camtasia), the sketching process can be captured directly from the computer's display. Ideally, all of those capabilities would be present. In

fact, when we run the system in our Intelligent Conference Room, we have two cameras recording the progress of the meeting in addition to audio and screen capture. As explained before, however, in some of the spaces, not all of these services will be available. For example, in some spaces where screen capture is not available, a similar service may be provided by a laser pointer tracking camera that watches the projected display. The quality of the recorded picture is not as good, but the content is still readable. Conversely, when the system is ran in a standard Metaglue-equipped office, no cameras are available and only audio and the screen get captured but not the video of the interaction among the participants.

5. The Tahuti Agent: Sketching As The Main Design Medium

When designing new components for the Room, the designer can draw a variety of symbols from UML notation, including class (rectangle), interface (circle), interface association (line), dependency association (arrow), inheritance association (arrow with triangle head), and aggregation association (arrow with diamond head). She can also use special additional symbols we have introduced: a double-edged rectangle to denote agents, and a triangle (shown in the interpreted view as a triangle with an extended bottom, or a pentagon, to fit more text) to denote grammars for speech-enabled agents. These symbols simplify our diagrams by providing certain syntactic shortcuts. In reality, an agent's implementation is always accompanied by an interface and the inheritance structure of interfaces usually parallels that of agents (see Figure 3). In our sketches we omit the interfaces. The interactions among agents involve a complex pattern of proxy objects and helper classes, which we also omit in our sketches (Figure 4). Finally, reliance on a grammar always implies use of a special proxy agent and interaction with Metaglue's speech facilities (Figure 5).

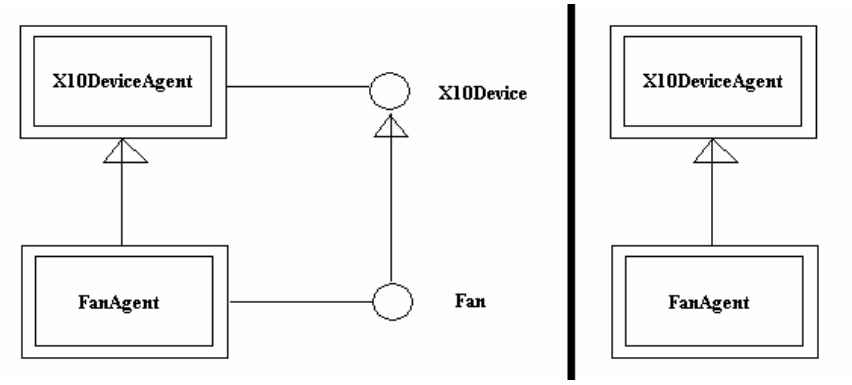

Figure 3. Each agent implements an interface with a corresponding name. If an agent inherits from another agent, so do the interfaces of the agents (left figure). In our sketches, the interfaces are assumed and not drawn (right figure).

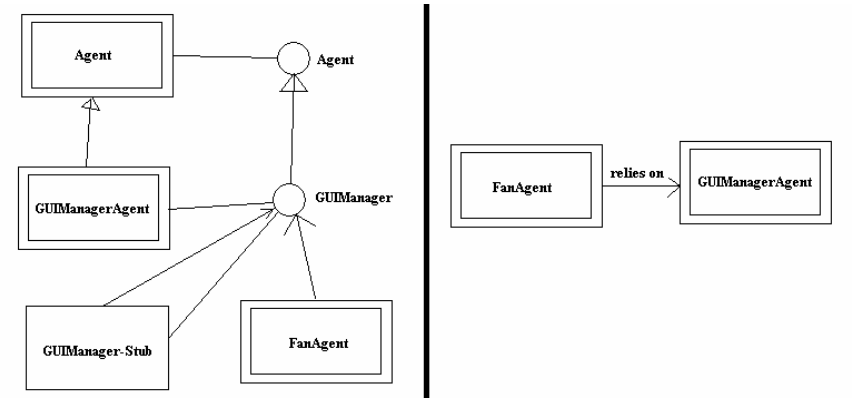

Figure 4: The figure on the left displays the actual interaction between the two classes. The figure on the right displays the abstraction for "relies on".

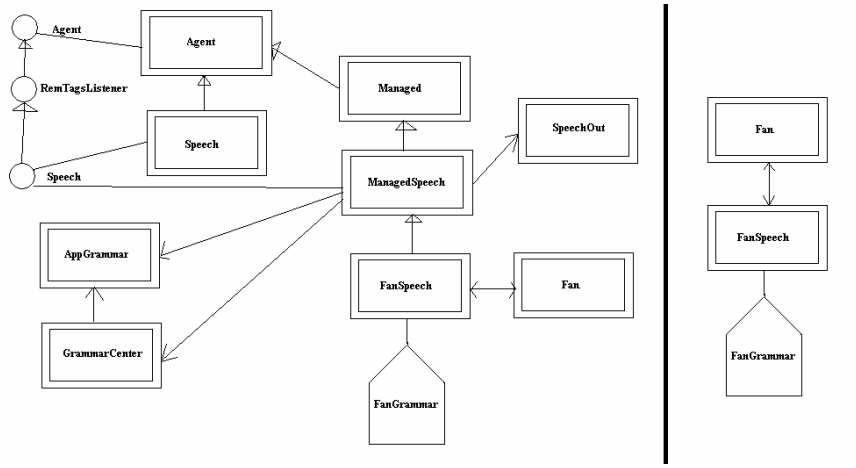

Figure 5. The left shows the speech grammar along with connected classes. The right shows the simper version with the grammar symbol implying the relationships and agents on the left.

5.1. UNDERSTANDING SKETCHES

The sketches drawn during the design process are interpreted in real time, e.g., rectangles are understood to indicate classes, etc. While drawing, the designers can alternate between viewing their free-hand sketches, or their interpreted drawings. Figure 6 shows the interpreted drawing of the design in Figure 2. The interpreted drawing neatens the sketch if desired and provides recognition feedback to the designer.

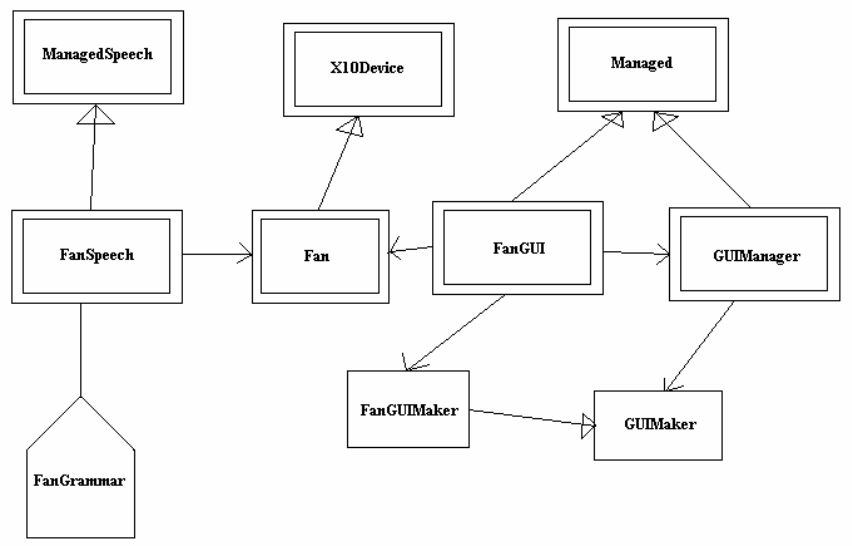

Figure 6. Interpretation of Sketch of Design in Figure 2 after Recognition

The free-hand sketches can be edited because the diagrams are interpreted. Classes, agents, speech grammars, and associations can be moved or deleted while viewing either the free-hand sketch or the interpreted structures. Drawn objects are deleted by scribbling them out. When a class, agent, or grammar is moved, the original, as well as the interpreted, strokes of the associations are stretched and skewed to remain attached to the appropriate object. This is described further in Hammond and Davis (2002).

6. Documenting Important Stages in Design

6.1. TIMESTAMPING OF SIGNIFICANT EVENTS

In our system, all events in the design process are recorded, labeled, and time stamped. A significant event is defined as the addition or deletion of a general class, interface class, agent, grammar, or relationship. Less significant events include the movement of a class, agent, grammar, or association, or the addition, deletion, or editing of text, such as class, method, or property names. During the development process, the designer may also mark a particular event as particularly significant. The designer

can then later ask questions such as “What was the discussion when this class was created?” and the system can show the appropriate section of video and screen shots.

6.2. RANKING OF SIGNIFICANT EVENTS

Designers may also want to ask the more general question “How did we design this system?” We would like to present to the designer a visual description of how the scene evolved. We don’t want to show the designer all of the significant events. Rather, we want to select a small number of snapshots that when combined together can best display the evolution of the design. We want to select the most significant events to show to the user, and the most revealing snapshot related to those significant events. Significant events are all given a rank, represented as a floating-point number. The number before the decimal place is set according to the type of event. For instance, creation of an Agent is given the highest rank of all sketched objects, with a rank of 10. The table below lists the initial rank of each of the possible events. While the numbers themselves are slightly arbitrary, what is important is the relative ordering of the events.

- Final Design: 12
- User Marked Significant Event: 11
- Creation of an Existing Agent: 10
- Creation of an Existing General or Interface Class: 9
- Creation of an Existing Speech Grammar: 8
- Creation of an Existing Association: 7
- Creation of an Existing Unrecognized Stroke: 6
- Text Update: 5
- Movement: 4
- Creation of Deleted Object: 3
- Deletion of an Object: 2
- Undo/Redo: 1

The logic behind the initial ranking is as follows. The final event is always ranked the highest. The designer selected significant events outrank computer selected significant events. Creation of viewable objects is considered a more significant event than the updating or movement of that object. Creations of objects that no longer exist in the final version are considered to be much less significant than those that remained throughout the entire process.

Within a particular category (e.g., looking only at the Creation of Agent Events), events are again ranked as more or less significant. Events that

affect more objects have a higher ranking. The fraction part of the floating-point number is used to do further ranking. Events specifying the creation of agents, classes, and grammars are further differentiated by the number of associations attached to them. For instance, an agent connected by and association by 4 classes would have a rank of 10.04 (since the number of associations is divided by 100).

A designer may want to see screenshots of the 5 most significant events to get a brief history of the design process. When the most significant events are chosen, the screenshot associated with the event is not the snapshot of the time of the occurrence of that significant event, but rather the snapshot of the moment before the next significant event. The next significant event is defined to be the next event greater than or equal to the lowest ranking in the listing of the most significant event. This allows any smaller additions such as text or movement to be included in the snapshot. Figure 7 shows ranking of each of the significant events of the diagram. Figure 8a-c shows the three most significant events of a diagram.

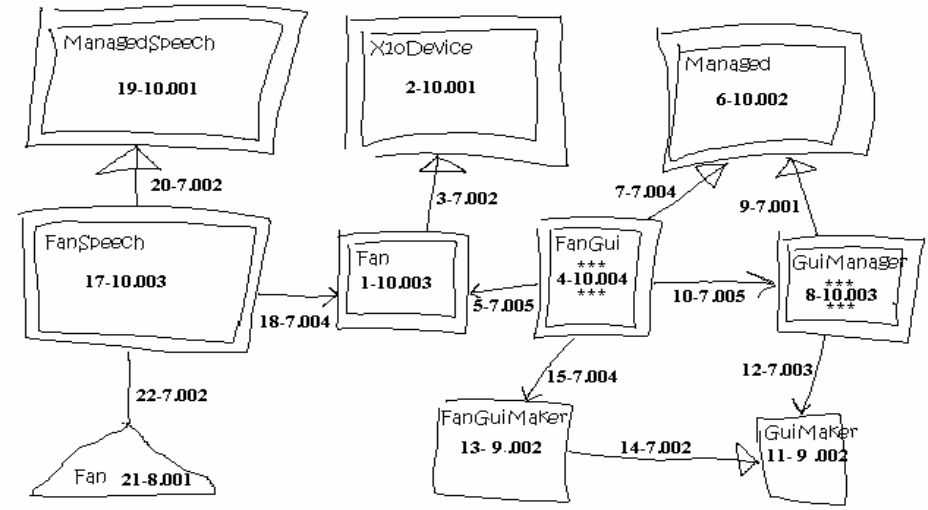

Figure 7: Each class, agent, association, and grammar is marked with a number specifying its order drawn followed by its ranking. Note that the two with the highest ranking are marked with stars.

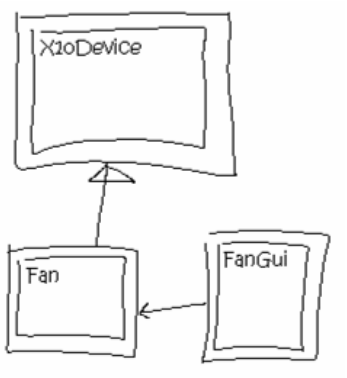

Figure 8a: Significant Design Event 1, the screen shot significant event 4 (which include significant event 5)

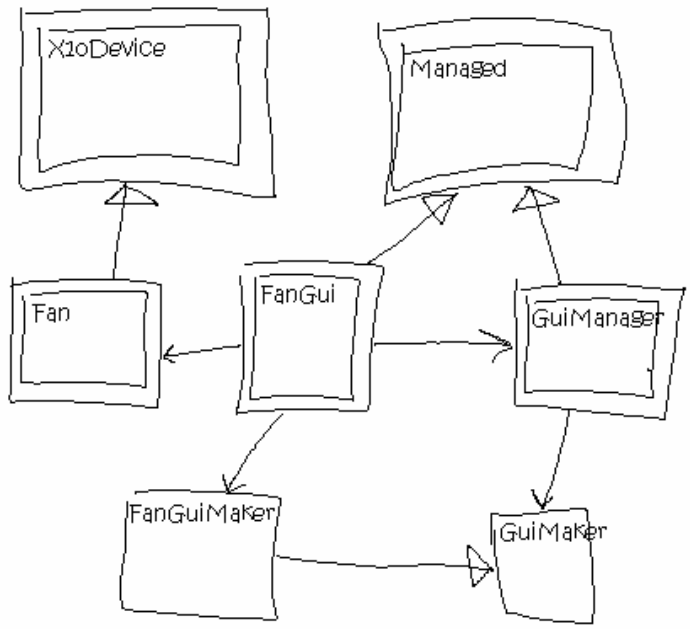

Figure 8b. Significant Design Event 2, after significant event 8 (which includes significant event 15)

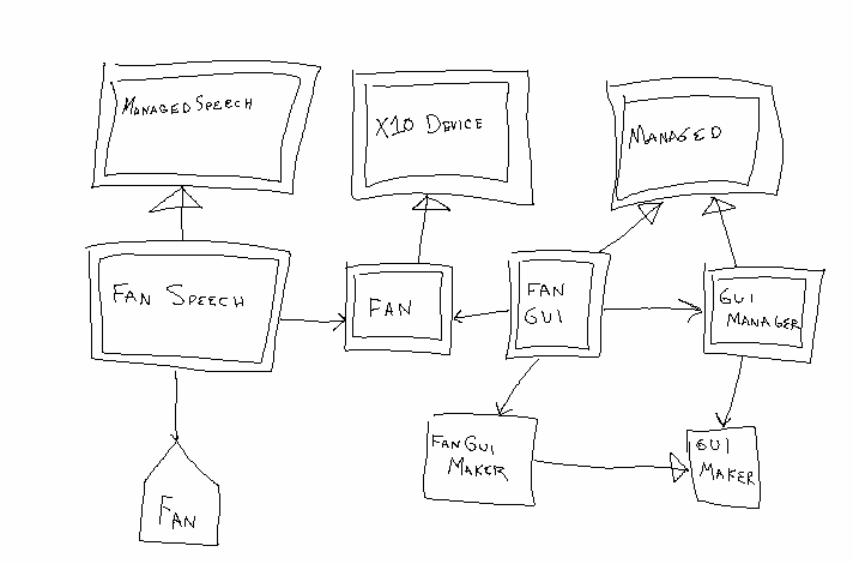

Figure 8c. Significant Design Event 3, the final diagram

7. Conclusions

7.1 CURRENT SYSTEM USE

The Tahuti stand-alone agent has been tested and used by over sixty users. It has been deployed for use in teaching object oriented programming in four computer science classrooms at Columbia University. The Metaglue technology described in this paper has also been deployed for several years at several locations and is widely used by the large number of people at the MIT AI Laboratory who use the Intelligent Room daily (about 50 users per day). Thus far, we are still in the testing phase of our system and only experimental users have used the system, but we anticipate a positive reaction to the new system.

7.2 FUTURE WORK

In the future, we plan to increase the number of sketchable shapes in our domain to include those in flow chart diagrams and UML sequence diagrams. Designers would then be able to create more semantically rich sketches.

7.3 CONTRIBUTIONS

We present an agent-based system for capturing and indexing software design meetings. Design meeting history is captured using available audio, video, and screen capture services in the environment. Tahuti, a sketch recognition agent, recognizes UML-type sketches drawn during the software design meeting and produces significant events based on the sketches drawn. These events are then used to index the videos and audiotapes for fast retrieval of specific information. The system is composed of multiple agents and runs in Metaglue, a multi-agent software infrastructure that provides for seamless multi-modal interaction between the various agents of the system as well as the users.

Acknowledgements

This work is supported by Acer Inc., Delta Electronics Inc., HP Corp., NTT Inc., Nokia Research Center, and Philips Research under the MIT Project Oxygen partnership and by DARPA through the Office of Naval Research under contract number N66001-99-2-891702.

References

Bergenti, F and Poggi A: 2001, Agent-oriented Software Construction with UML, *Handbook of Software Engineering and Knowledge Engineering*, Vol 2.

Brunelli, R and Mich, O and Modena CM: 1996, A Survey on Video Indexing, *IRST Technical Report*.

Cho, SJ and Yoo, SI: 1998, Image Retrieval Using Topological Structure of User Sketch, *Proceedings of IEEE SMC98*.

Coen, M: 1998, Design Principles for Intelligent Environments. *Proceedings of AAAI'98*. Madison, WI, 1998.

Coen, M and Phillips, B and Warshawsky, N and Weisman, L and Peters, S and Finin, P: 1999, Meeting the Computational Needs of Intelligent Environments: The Metaglue System. *Proceedings of MANSE'99*, Dublin, Ireland.

Coen, M and Weisman, L and Thomas, K and Groh, M: 1999, A Context Sensitive Natural Language Modality for the Intelligent Room. *Proceedings of MANSE'99*. Dublin, Ireland.

Gajos, K: 1999, Rascal - a Resource Manager For Multi Agent Systems In Smart Spaces. *Proceedings of CEEMAS'01*, Cracow, Poland. Also available in LNAI 2296.

Hammond, TA and Davis, R: 2002, Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams, *Proceedings of the 2002 AAAI Spring Symposium on Sketch Understanding*, pp. 59-66.

Hanssens, N and Kulkarni, A and Tuchinda, R and Horton, T: 2002, Building Agent-Based Intelligent Workspaces. *Proceedings of The International Workshop on Agents for Business Automation*. Las Vegas, NV.

Jacobs, C and Finkelstein A, and Salesin D: 1995, Fast Multiresolution Image Querying, *Computer Graphics, Annual Conference Series (Siggraph '95 Proceedings)*, pp. 277-286.

Kato, T and Kurita, T and Otsu, N and Hirata, K: 1992, A Sketch Retrieval Method for Full Color Image Databases – Query by Visual Example. *11th IAPA International Conference*

on Pattern Recognition, IEEE Computer Society Press, The Hague, The Netherlands, pp. 530-533.

Louridas, P and Loucopoulos, P: 2000, A Generic Model for Reflective Design. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, New York, NY.

Moran, TP and Carroll, JM: 1996, Overview of Design Rational, *Design Rationale: Concepts, Techniques, and Use*, TP Moran and JM Carroll, Eds. LEA Computers, Cognition, and Work-Series, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1-19.

Odell, J and Parunak HVD and Bauer B: 2000, Extending UML for Agents, *AOIS Workshop at AAAI*.

Oh, A and Tuchinda, R and Wu, L: 2001, MeetingManager: A Collaborative Tool in the Intelligent Room. *Student Oxygen Workshop*, Cambridge, MA.

Peters, S: 2002, Using Semantic Networks for Knowledge Representation in an Intelligent Environment. In Submission.

Shipman, FM and McCall RJ: 1997, Integrating Different Perspectives on Design Rationale: Supporting the Emergence of Design Rationale from Design Communication, *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing (AIEDAM)*, 11(2): 141-154.

Shum, SJB and MacLean, A and Bellotti, VME and Hammond, NV: 1996, Graphical Argumentation and Design Cognition, *Human-Computer Interaction*, 12(3): 267-300.

Warshawsky, N: 1999, *Extending the Metaglue Multi Agent System.* M.Eng. Thesis. MIT, Cambridge, MA.