Sketch Recognition in Software Design

Tracy Hammond, Krzysztof Gajos, Randall Davis, Howard Shrobe

The Problem: Sketching is a natural and integral part of software design. Software developers use sketching to aid in the brainstorming of ideas, visualizing programming organization, and understanding of requirements. Unfortunately, when it comes to coding the system, the drawings are left behind. We see sketch recognition as a way to bridge that gap.

In addition to the vast amount of information given by a sketch, a plethora of other design information may be voiced during a software design meeting. We can capture the spoken and visual software design meeting information by videotaping the meeting and any white-boards used. By indexing these videos, we make it easy to retrieve the videotaped information without watching the entire video from start to finish.

Motivation: We want to allow software design meetings to continue as they are, with software designers discussing the design and drawing free-hand sketches of these designs on a white-board. Using our system, designers can sketch naturally, as we place few requirements on the sketcher. We recognize and interpret these diagrams using sketch recognition. Because the diagrams are interpreted, we provide natural editing capabilities to the designers, allowing the users to edit their original strokes in an intuitive way. For instance, the designer can drag their drawn class from the center and move all of the strokes used to draw the class as well as stretch and skew the strokes used to create an attached arrow. The interpreted diagrams are used to automatically generate stub code using a software engineering tool. Software design meetings are videotaped to capture visual and spoken design information unobtrusively. When drawn items are interpreted, we use these understood sketch events to index the videotape of the software design meeting.

We decided to design our application as a Metaglue agent since the Metaglue agent architecture provides support for multi-modal interactions through speech, gesture, and graphical user interfaces[2]. The Metaglue agent architecture also provides mechanisms for resource discovery and management which allows us to use available video agents or screen capture agents in a Metaglue supported room.

We have selected UML-type diagrams because they are a de facto standard for depicting software applications. Within UML [1] we focused on class diagrams, first because of their central role in describing program structure, and second because many of the symbols used in class diagrams are quite similar, and hence, offer an interesting challenge for sketch recognition. We added several symbols for agent-design since many of the applications created in the Intelligent Room [6] of the MIT AI Lab are agent-based.

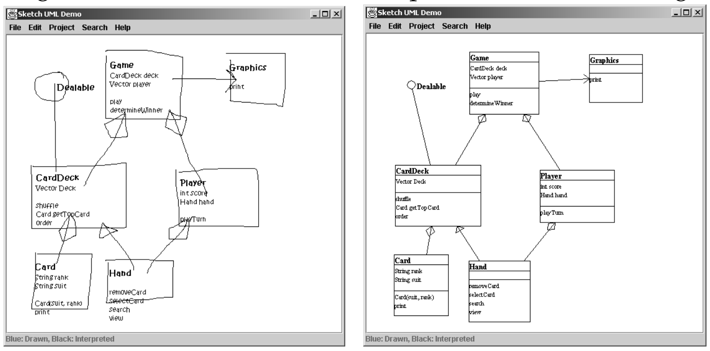
Previous Work: Work at Berkeley by Hse [7] has shown that users prefer a single-stroke sketch-based user interface to a mouse-and-palette based tool for UML design.

One company [3] has developed a gesture based diagramming tool, Ideogramic UML,™which allows users to sketch UML diagrams. The tool is based on a graffiti-like implementation and requires users to draw each gesture in one stroke, in the direction and style as specified by the user manual. As a consequence, some of the gestures drawn only loosely resemble the output glyph. For example, φ is the stroke used to indicate an actor, drawn by the system as a stick figure.

Work at Queen's University has developed a system to recognize sketches of UML diagrams using a distance metric [8]. Each glyph (square, circle, or line) is classified based on the total stroke length compared to the perimeter of its bounding box (e.g., if the stroke length is approximately equal to the perimeter of the bounding box, it is classified as a square). The shape of the stroke is not considered.

Approach: We have created Tahuti [4], a system for recognizing hand-drawn sketches of UML class diagrams. Users can sketch UML-type diagrams on a white board in the same way they are drawn on paper, and have the diagrams recognized by the computer. The system differs from graffiti-based approaches to this task in that it recognizes objects by how they look, not by how they are drawn. While sketching, the sketcher can seamlessly switch between the interpreted designs and the original strokes (See Figure). Editing commands operate identically in the two views.

Figure 1: The sketched view, interpreted view, and auto-generated code of a UML diagram in Tahuti




```
//Source file: c:\\cardgame\\Deck.java

public class Deck implements Dealable
{
   public Game theGame;
   public Card theCard;

   /**
    * @roseuid 3C4623D50279
    */
   public Deck()
   {
   }
}
```

The recognizable shapes in Tahuti include the components of UML class diagrams as well as agents (indicated by a double-edged rectangle), and speech grammars (indicated by a triangle) [5]. Tahuti itself has been designed as an agent in the Intelligent Room, enabling it to interact with other agents in the room, such as a videotaping agent and the Meeting Manager [9]. Tahuti labels significant events in the sketching of a software diagram to help index a video of a design meeting.

Impact: A small user study shows that users prefer drawing and editing in Tahuti over a traditional paint application and a professional UML editing tool.

Tahuti has been used at Columbia University to teach 65 students Object Oriented Programming. The system was well-received and appeared to aid both in the initial program design and in progressive program design, although a formal study was not done. Even simply having a graphical picture of the program seemed to allow the students to maintain a clear picture of the program structure throughout the coding process. Tahuti has recently been integrated into the Intelligent Room at the MIT AI Lab for use in software design meetings, including the design of agent-based systems.

Future Work: Future system enhancements include allowing the user to sketch more detail about a program. For instance, we plan to add the ability to recognize multiplicity relationships by interpreting annotations on associations (e.g., $\infty \rightarrow 1$). We would also like to recognize other software diagram types. We also hope to integrate a code editor, allowing users to alternate between diagram view and code view, using existing software to reverse engineer the code into diagrams.

Research Support: This work is supported by the MIT Project Oxygen partnership and by DARPA through the Office of Naval Research under contract number N66001-99-2-891702.

References:

[1] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, MA, 1998.

[2] Michael H. Coen, Brenton Phillips, Nimrod Warshawsky, Luke Weisman, Stephen Peters, and Peter Finin. Meeting the computational needs of intelligent environments: The metaglue system. In *Proceedings of MANSE'99*, 1999.

[3] Christian Heide Damm, Klaus Marius Hansen, and Michael Thomsen. Tool support for cooperative object-oriented design: Gesture based modeling on an electronic whiteboard. In *CHI 2000*. CHI, April 2000.

[4] Tracy Hammond and Randall Davis. Tahuti:a geometrical sketch recognition system for uml class diagrams. *AAAI Spring Symposium on Sketch Understanding*, pages 59–68, March 25-27 2002.

[5] Tracy Hammond, Krzysztof Gajos, Randall Davis, and Howard Shrobe. An agent-based system for capturing and indexing software design meetings. In *Proceedings of International Workshop on Agents In Design, WAID'02*, 2002.

[6] Nicholas Hanssens, Ajay Kulkarni, Rattapoom Tuchinda, and Tyler Horton. Building agent-based intelligent workspaces. In *Proceedings of The International Workshop on Agents for Business Automation*, 2000.

[7] Heloise Hse, Michael Shilman, A. Richard Newton, and James Landay. Sketch-based user interfaces for collaborative object-oriented modeling. Berkley CS260 Class Project, December 1999.

[8] Edward Lank, Jeb S. Thorley, and Sean Jy-Shyang Chen. An interactive system for recognizing hand drawn UML diagrams. In *Proceedings for CASCON 2000*, 2000.

[9] Alice Oh, Rattapoom Tuchinda, and Lin Wu. Meetingmanager: A collaborative tool in the intelligent room. In *Student Oxygen Workshop*, 2001.