Tahuti: A Sketch Recognition System for UML Class Diagrams

Tracy Hammond and **Randy Davis**
AI Lab, MIT
200 Technology Square
Cambridge, MA 02139
hammond, davis@ai.mit.edu

Introduction

Sketching is a natural and integral part of software design, aiding in the brainstorming of ideas, visualizing of programm organization, and understanding of requirements. Within UML diagrams(Unified Modeling Language) (Booch, Rumbaugh, & Jacobson 1998), a de facto standard for software design, class diagrams play a central role in describing program structure.

Paper sketches offer users the freedom to sketch as they would naturally, but because they are static and uninterpreted, they lack computer editing features, requiring users to completely erase and redraw an object to move it. We have created and tested Tahuti[1], a dual-view, multi-stroke sketch recognition environment for class diagrams in UML, combining the sketching freedom provided by paper sketches and the ease of editing available in an interpreted diagram. The system recognizes objects by their geometrical properties, rather than requiring the user draw the objects in a pre-defined manner.

Previous Work

A Wizard of Oz experiment showed that users prefer a single-stroke sketch-based user interface to a mouse-and-palette based tool for UML design. (Hse *et al.* 1999) Users, though happy with the single-stroke version, requested more sketching flexibility, such as the ability to draw with multiple strokes.

Ideogramic UML^{TM}(Damm, Hansen, & Thomsen 2000), a graffiti based diagramming tool, requires users to draw each single-stroke gesture in the style specified in the user manual. A consequence of the single stroke limit is that some of the gestures drawn only loosely resemble the output glyph. For example, φ is used to indicate an actor, drawn by the system as a stick figure.

Edward Lank et al. built a UML recognition system that uses a distance metric (Lank, Thorley, & Chen 2000), classifying strokes based on the total stroke length compared to the perimeter of its bounding box. This algorithm can cause many false positives. (For example, the letter M can be detected as a box.)

[1]Tahuti, also known as Thoth, is the Eqyptian god of wisdom. He always carried a pen and scrolls upon which he recorded all things.

System

Our system uses a multi-layer framework for sketch recognition. At the most basic level, strokes (Figure 1a) drawn by the user are interpreted as line segments (Figure 1b) using algorithms for stroke processing developed in our group (Sezgin, Stahovich, & Davis 2001). A collection of spatially and temporally close strokes is chosen, and the line segments contained in the collection of strokes are then recognized (Figure 1c) as either an editing command or a viewable object (Figure 1d). Further description of the recognition stage is in the Recognition section.

The system currently recognizes seven viewable objects: a general class, an interface class, an inheritance association, an aggregation association, a dependency association, an interface association, or a collection of unrecognized strokes. An editing command is a collection of strokes indicating deletion or movement of a viewable object.

The system is non-modal: users can edit or draw without having to give any explicit advance notification. While sketching, the user can seamlessly switch between two views: the interpreted view (Figure 2a) or the drawn view (Figure 2b). Figure 2c shows the results after moving classes in Figure 2a. The drawn view is shown in Figure 2d. Editing commands operate identically in the two views, with the drawn view allowing users to view and edit their original strokes. When a class is dragged, the strokes of an attached association must be stretched, translated, and rotated in order for it to remain attached and the strokes faithful to those originally drawn.

Recognition

The third stage in the recognition framework (Figure 1c) is the recognition of a collection of strokes that have been segmented into line-segments (Figure 1b). The system attempts to correctly interpret the strokes based on several recognition algorithms. We present the algorithm for arrow recognition here. (See Figure 1c for the references to A, B, C, D, and E.)

1. Locate the arrow shaft by locating the two points furthest from each other (points A and B).

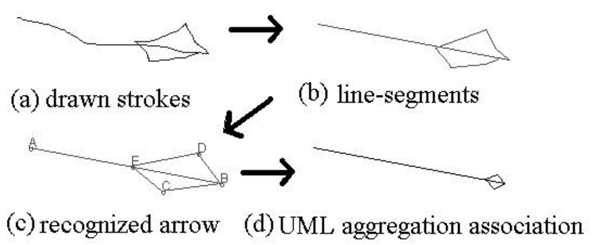

Figure 1: Multi-layer framework of recognition

2. Locate the arrow head ends by locating points furthest from arrow shaft on either side (points C and D).
3. Let point E be the point on line AB that is twice the distance from B as the intersection point of lines CD and AB.
4. Classify each of the line segments as part of the arrow shaft, an arrow head section, or unclassified (AB, BC, BD, CD, CE, DE, or unclassified) based on the line's bounding box, slope, and y-intercept
5. Based on the results of the line-segment classification, classify the arrow type as dependency, inheritance, aggregation, or leave the strokes unclassified.

Experiment

In a preliminary study, four subjects were asked to draw and edit a UML diagram in four different ways: A) using a paint program, B) using Rational Rose^{TM}C) using Tahuti in interpreted view D) using Tahuti in drawn view. Subjects were aided in the use of Rational Rose if they were unfamiliar with it, but little instruction was given otherwise so that subjects would draw as they do naturally.

The subjects were asked to rank the four methods on a continuous scale from zero to five (with zero being the hardest and five being the easiest) both for ease of drawing and for ease of editing. The average scores were as follows:

- **Drawing** A: 2.25, B: 1.75, **C: 4.375**, D: 3.1
- **Editing** A: 1.65, B: 1.925, **C: 4.825**, D: 2.6

The results display that subjects greatly preferred the interpreted sketch interface of Tahuti. They appreciated having the freedom to draw as they would on paper along with the editing intelligence of a computer application. Subjects said that editing was difficult in the paint program because of the large amount of resketching required for class movement. Subjects complained that Rational Rose was extremely nonintuitive and that they had difficulty performing the commands they wished to perform.

Conclusion

We have created and tested Tahuti, a dual-view, multi-stroke sketch recognition environment for class diagrams in UML, combining the sketching freedom provided by paper sketches and the ease of editing available in an interpreted diagram. The system recognizes objects by their geometrical properties, rather than requiring that the user draw the objects in a predefined manner. The experiments showed that users preferred Tahuti to a paint program and to Rational Rose^{TM}because it combined the ease of drawing found in a paint program with the ease of editing available in a computer application.

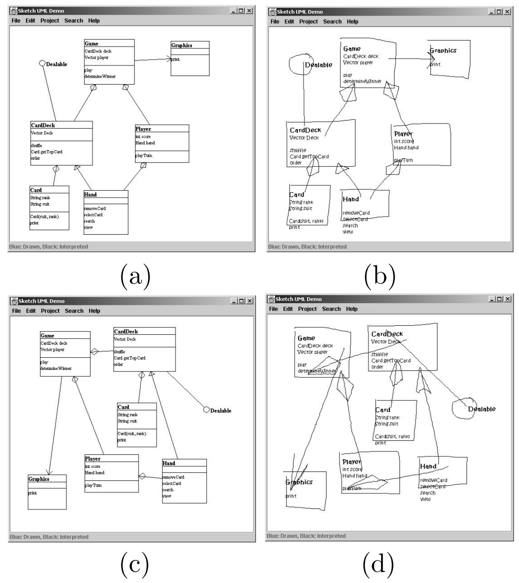

Figure 2: (a) Interpreted UML class diagram, (b) Drawn view of (a), (c) Diagram of (a) with the classes moved, (d) Drawn view of (c). (The text shown in the figures is entered via the keyboard.)

Future system enhancements include the ability to recognize multiplicity relationships and modification of recognized objects, (e.g., changing a dependency association into an inheritance association by adding a stroke). We also would like to compare Tahuti with other UML tools.

References

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

Damm, C. H.; Hansen, K. M.; and Thomsen, M. 2000. Tool support for cooperative object-oriented design: Gesture based modeling on an electronic whiteboard. In *CHI 2000*. CHI.

Hse, H.; Shilman, M.; Newton, A. R.; and Landay, J. 1999. Sketch-based user interfaces for collaborative object-oriented modeling. Berkley CS260 Class Project.

Lank, E.; Thorley, J. S.; and Chen, S. J.-S. 2000. An interactive system for recognizing hand drawn UML diagrams. In *Proceedings for CASCON 2000*.

Sezgin, T. M.; Stahovich, T.; and Davis, R. 2001. Sketch based interfaces: Early processing for sketch understanding. In *To appear in The Proceedings of 2001 Perceptive User Interfaces Workshop (PUI'01)*.