A Visual Language for a Sketch-Based UI Prototyping Tool

James Lin
Group for User Interface Research, Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776 USA
+1 510 643-7354
jimlin@cs.berkeley.edu

ABSTRACT
We describe the design of the visual language for SILK 2.0, a sketch-based tool for prototyping user interfaces. The new SILK visual language has been designed to allow user interface designers to quickly prototype the behavior of a user interface. This includes behavior that depends on the state of certain UI elements and the ability to create "sketchy" components to be reused in other sketches.

Keywords
User interfaces, design, prototyping, sketching, visual language, SILK, informal interfaces

INTRODUCTION
SILK [1] is a sketch-based user interface design tool that combines the traditional strengths of the computer with the speed of paper-based sketching. The designer sketches various states of the user interface, to specify the behavior of the interface. The sketches are assembled together into a *storyboard* [2]. By drawing a transition arrow from a user interface object in one sketch to another sketch, the designer says that, in the user interface, if the user clicks on the object, the program will transition to the screen shown at the end of the arrow (see Figure 1).

At any time, designers can switch SILK into Run Mode. At this point they can interact with their sketches, like a real program, in another window separate from the storyboard window. The sketch behaves as specified by the storyboard.

The storyboarding mechanism constitutes a visual language, which is too simple for sophisticated programs. The only type of interaction supported is single clicking the left mouse button. There are no conditional transitions; a transition cannot depend on the state of other UI elements in the panel. It is difficult to reuse parts of storyboards in other storyboards, and SILK does not allow designers to design their own widgets, a desire that was expressed in earlier studies [1].

We have extended SILK's visual language to address these issues, while keeping the language accessible to non-programmers and maintaining the rapid prototyping qualities that SILK's sketch-based interface makes possible.

INTERVIEWS WITH UI DESIGNERS
To help us decide how to design the extensions, we interviewed eight designers of graphical user interfaces in industry. We discussed the design cycle and if and how the designers use sketches. If time permitted, we also presented our design ideas for SILK to get feedback.

All of the designers used sketches to design basic screen layout. Often, all of the sketches on a sheet of paper dealt with only one particular screen layout. Navigation was described either through callouts from individual sketches or by a *walk-through*, where the designer assembles a sequence of sheets, each with a sketch of a screen, that shows what the user would see as he or she does particular tasks. Sketching the user interaction with SILK-like storyboarding, i.e. drawing arrows between the sketches, was only done if the interaction was particularly complex.

Two designers used graphs or trees to show how different parts of the user interface were related to each other. The nodes of the graph were simply phrases or very rough sketches describing the screen configuration. Only overall structure, as opposed to all possible navigation, was diagrammed. On the other hand, two other designers did not diagram any interaction because they said it would take too much time.

Due to time constraints, we were able to show several design variations for SILK to only five of the eight designers. Four of them were enthusiastic about the basic idea of SILK, and their feedback helped us decide on the designs we describe in the next section. One designer did almost no sketching with pen and paper; he used a vector-based drawing tool instead. He did not believe SILK would be an improvement over his existing design process.

EXTENDING THE VISUAL LANGUAGE
Based on feedback from the designers, we have introduced the following elements into SILK's visual language.

Support for More Event Types
SILK has been extended to support certain types of events other than single clicking the left mouse button, such as timer, simple animation, and double clicking. Designers choose what type of event they want to specify in a tool palette before drawing the storyboard arrow. Figure 1

shows an early prototype of this tool palette. Sound events could also be specified in this manner.

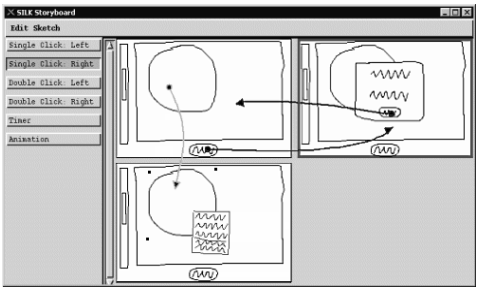

Figure 1. A storyboard window in SILK with more event types. The arrows pointing left and right represent left single click, and the arrow pointing down represents right single click. In the prototype, the arrows are different colors.

The following new language features are in the process of being implemented.

Conditional Transitions

Conditional transitions are represented by multiple storyboard panels. Each panel displays the condition that must be satisfied before the transition can occur. The conditional panels are surrounded by a sketched brown box. Figure 2a shows an example of a conditional transition. Inside a conditional panel, ovals surround the user interface objects whose state must match what is shown for the corresponding transition to occur.

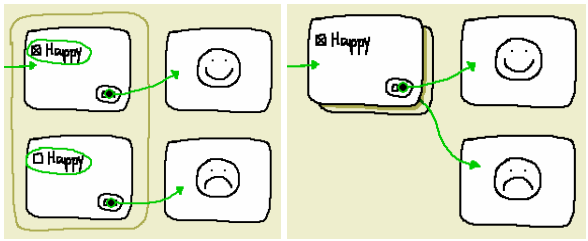

Figure 2. a) A storyboard fragment with a conditional transition. b) The conditional panel collapsed into a stack.

If there are many conditions, the panels can take up a lot of space, making the storyboard cumbersome to use. To address this problem, the conditional panels can be collapsed into a stack of panels, as shown in Figure 2b. This gives designers an overall view of the structure of the storyboard, without burdening them with unnecessary details. If they want to examine the details, they can expand the stack. They can also cycle through the stack to see each condition one at a time.

Components

Components allow designers to design their own "sketchy" widgets that can be used in other SILK designs. Internally, components are defined in the same way as normal programs—through a storyboard. Components are surrounded by a sketched blue box. The distinguishing factor is that components can have input parameters and external events. An example of a component, which encapsulates a wizard, is shown in Figure 3.

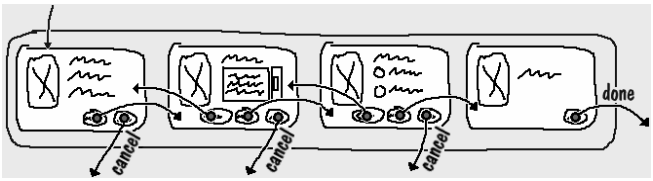

Figure 3. A component in SILK

A transition leaving the component boundary defines an external event. Components can define their own named events that are at a higher level than simple interaction. The component in Figure 3 defines two events, `cancel` and `done`. This component can now be reused in different storyboards, as shown in Figure 4.

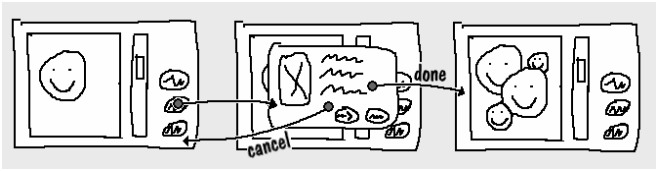

Figure 4. Using a component in SILK

FUTURE WORK

There are more outstanding issues in SILK that we would like to address. If the designer tries to completely specify a complex user interface, the number of panels in the storyboard grows exponentially, as does the number of arrows between the panels. The "exponential growth" problem is similar to that of finite-state machines.

We are looking into a number of ideas to address this problem. One idea, inspired by the interviews, is to overlay a tree structure on top of the storyboard. Panels in the tree would inherit behavior from ancestor panels. The tree would also serve to represent the overall structure of the user interface. Another idea is to generalize the storyboarding mechanism into multiple mini-storyboards. Each mini-storyboard would have a precondition that would need to be met before executing.

CONCLUSION

A user interface design tool with a sketch-based interface allows designers to rapidly brainstorm, develop, and iterate ideas for user interface designs. By adding support for reusable sketchy components, conditional transitions, and more sophisticated types of user interaction, we hope to encourage designers of user interfaces to use SILK and enjoy the benefits of sketching in all phases of UI design.

REFERENCES

1. Landay, James A. *Interactive Sketching for the Early Stages of User Interface Design*. Ph.D. dissertation, Report #CMU-CS-96-201, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December, 1996.
2. Landay, James A. and Brad A. Myers. *Sketching Storyboards to Illustrate Interface Behaviors.* In CHI 96 Conference Companion: Human Factors in Computing Systems, Vancouver, BC, Canada, April 1996