Sketch Tool Usability: Allowing the user to disengage

Beryl Plimmer, Gene Tang and Mark Young
University of Auckland,
Private bag 92019, Auckland, New Zealand
beryl@cs.auckland.ac.nz, mark@educk.co.nz, gene_tang@hotmail.com

The key to a successful early design environment is an unobtrusive user experience. Yet there has been little focus on the usability of sketch tools. Here we describe two iterations of prototype and usability evaluation of InkKit, a domain independent sketch-tool for diagramming. We found that reducing constraints, careful use of metaphor and pairing input and output devices improved the user experience.

Sketch tools usability, pen-based interaction, metaphor.

1. INTRODUCTION

Diagrams are used in many disciplines to visually describe ideas. When first creating a diagram, the ability to quickly express ideas ambiguously and incompletely is important [4]. As this is not possible with most computer-based tools, designers do not use them until their design is almost complete. Instead they favour traditional tools, paper, whiteboards, pens and erasers. Studies of computer widget-based design environments suggest that the functions that interfere with the design process are: the requirement to select a specific widget; distractions cause by tidying the artefact while designing (aligning and sizing); and the formal appearance of the diagram – which implies completeness [4, 10]. To address these issues computer-based sketch tools closely emulate traditional tools in that they afford pen input of informal diagrams but add value by providing computational support for editing and archiving of artefacts. Additional benefits can ensue with intelligent sketch recognition for example automatic beautification, execution and translation of the hand-drawn sketches. Yet, if this functionality becomes obtrusive the benefits of an informal environment are lost. The challenge is to provide additional functionality within the sketch-tool's user interface while minimising its cogitative demands so that the designer can disengage from operating the computer and therefore fully engage in the design process.

2. BACKGROUND

The central function of a sketch tool is to support digital inking on to a digital canvas. Normal computer editing is also standard: we experimented with not providing editing (except erase) with Freeform [11], however our first usability test showed users expect and value computer support for editing. Most sketch tools also include recognition engines to enable execution and translation of the sketch. However many restrict recognition to drawing shapes – thus requiring keyboard input of text [6, 7], this is a distraction for the user [5]. Those that support hand-writing recognition are very constrained [8] or require the user to change modes [12].

Two approaches to displaying sketches are evident in the literature; an infinitely large canvas or many small canvases. A large canvas presents an interaction challenge as only a portion of the drawing can be viewed at normal resolution at any one time. To support navigation these tools have either a radar window [3] or zooming [9], another, unexplored possibility is a fish-eye view. Small canvases are restricted in size so that they can be viewed and edited at full resolution. They are often accompanied by a storyboard [1, 7, 12], where the collection of canvases can be shown as thumbnails. The storyboard is a linear sequence of sketches where the user can establish relationships between the sketches by either placing them in a particular order or establishing links between the sketches.

The functionality that makes computer-based sketch tools useful also complicates the user interaction. In particular mode and input device changes may be necessary to move between drawing and writing, inking and editing and storyboard and sketch. These mode changes are different to those in a tactile environment where one may change tools or physically move objects. Problems arise when it is not clear to the user what mode they are in and what actions are available. The drawing/writing mode changes in FreeForm [12] cause these types of interaction problems. Like other drawing environments, most sketch tools require a mode change for editing. Once in this mode ink is selected and then changed either via icons and grab-handles or functional gestures. Menus and icons provide certainty of operation, whereas functional gestures are less reliable and, if recognition fails, create more work for the user.

Pen or stylus input devices are used for sketch tools. Digital whiteboards are useful when a large space is desirable for collaborative work. However the input data that digital whiteboards provide is of a lower quality than that of digitizer tablets, this makes accurate recognition of ink more difficult. Earlier systems used input digitizers

such as Wacom™ tablets. These provide accurate position, time and pressure data but are disconnected from the output visualization which is usually on an adjacent display. Tablet PCs and newer Wacom tablets provide similar input accuracy with the advantage of the input and output surfaces being one and the same. In the following section we describe our experiences designing, building and evaluating the user interface for InkKit, a generic diagramming environment.

3 OUR APPROACH

InkKit is a domain independent, extensible diagramming environment. The goal is to provide a context free environment that imposes no additional cognitive load on the user: basic designing must be as simple as pen and paper. Yet InkKit is extensible so that it can incorporate sophisticated support for beautification, execution and translation of diagrams. InkKit runs on the Microsoft Tablet OS utilizing the tablet hardware and OS character recognition. As with any computer-based drawing tool, display space is a constraint. This is exasperated with the tablet OS as digital whiteboards are not yet supported as input devices. Apart from the user interface the other major part of InkKit is the recognition engine. InkKit is the first sketch tool to provide comprehensive, modeless drawing and writing recognition. Adaptation for specific types of diagrams is achieved by providing example components to the recognition engine. Extensibility allows user written modules to export recognized diagrams into different formats. We have implemented a variety of domains including user interface diagrams, organization charts and UML class diagrams. Output formats include html, java, Microsoft Word drawing objects and a variety of picture formats. A report has been published on the development of the first prototype [2]. Here we discuss the user interface of that prototype, the usability evaluation of it, and the subsequent reengineering and retesting

The design of the first prototype was based on our experience with FreeForm [12] and the experiences of other researchers. With the user interface we experimented with a single view comprising of multiple MDI forms (figure 1), removing the constraint of linearity and fixed sized storyboard elements evident in most sketch tools. There are two modes; storyboard and form, to change mode the user taps an icon on the tool bar. To visually differentiate modes, in storyboard mode the ink is greyed, in form mode the ink on the active form is black. In storyboard mode the forms are manipulated as entities, they can be positioned anywhere on the storyboard and resizing zooms the sketch. Relationships can be established between forms by creating a link by first tapping on the start-point of the link and then on the end-point. The meaning of the link is domain dependent, for example a user interface form with a link between a dropdown control and a form with words on it indicates that the list should be filled with the words. With organization charts the links may indicate different departments. In form mode the sketch is a computer supported drawing space. Unlike other sketch tools InkKit provides a modeless drawing/writing sketch space; this means, that users can write and draw on the form much as they would on a piece of paper. To edit ink users must first enter edit mode by tapping the lasso icon on the toolbar, this changes the cursor and stylus contact with the surface produces a dotted line. Selected ink is surrounded by a bounding box, dragging this boxes handles resizes the ink. Undo and redo are activated by tapping the appropriate icon. Resizing a form in form-mode changes the page size. We have not incorporated functional gestures into InkKit as our experiences with FreeForm suggested that they are not intuitive.

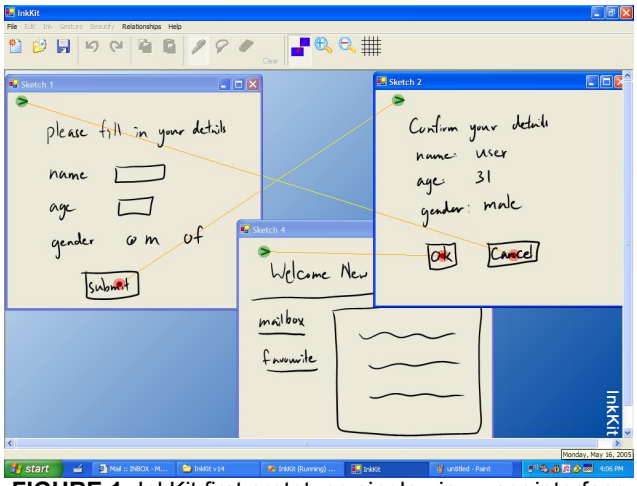

FIGURE 1: InkKit first prototype single-view user interface

FIGURE 2: InkKit second prototype two display, dual-view user interface

3.1 Usability Evaluation

InkKit is the first sketch tool specifically designed as a domain independent environment. This usability evaluation checked the fundamental design premises, such as the interface metaphor and map between the system model and user model, and also detailed interaction techniques.

The terminology used to describe an interface is usually metaphorical and is a powerful suggestion of the system model. This first prototype of InkKit adopted a mixture of computer and design jargon commonly used for sketch tools. As the goal is to provide a domain independent sketch tool we re-examined the terminology and compiled a

short list of possible metaphors and associated words including: parent form, storyboard and portfolio and child form, sketch and canvas. We then showed InkKit to a variety of people and asked them which they thought best described it. We found that parent and child form were too computer specific, storyboard was not widely understood and that it implied a linear sequence of sketches (as used in film storyboarding), and canvas is more closely associated with the fabric than an artist's canvas. The best metaphor is a *portfolio* full of *sketches*; hence we changed the terminology to portfolio and sketch.

We also conducted a formal usability test of this prototype on at Toshiba M200 tablet. The six test participants were 4th year software engineering students. As the participants were unfamiliar with the tablet PC and InkKit a short training session was given first. Then each was asked to create a three-form user interface for joining a sports team supporter's website. The sessions were observed and recorded using Morae™ usability testing software. The sketch space and mode was well received by the users. The main difficulty we identified with our previous tool Freeform was the mode change required for drawing and writing; we have eliminated this in InkKit with a more sophisticated approach to recognition. Editing support for ink was commented on as an advantage of digital sketch tools. And although this requires a mode change, the change of cursor, and appearance of the stylus path were sufficiently clear to users for this not to cause difficulties.

However the storyboard mode was not as successful. Moving between form editing and storyboarding required a mode change. Just as we had found with inking modes with Freeform, this created real difficulties for the users. Although InkKit greyed the ink on the forms to indicate storyboard mode users frequently attempted form editing functions in storyboard mode and storyboard functions in editing mode. We also identified an interaction problem when creating links between forms (taps to position start and finish points); most users tried to drag a link. In addition to these problems, the small display size of the tablets means that a multi-form design either has a large percentage of the forms obscured or the forms zoomed very small.

3.2 Reengineering

To recap: we found that the terminology we had adopted from previous sketch tools was poorly understood. We also identified three major usability problems; the mode changes between storyboard and form, limited display space and the link creation method.

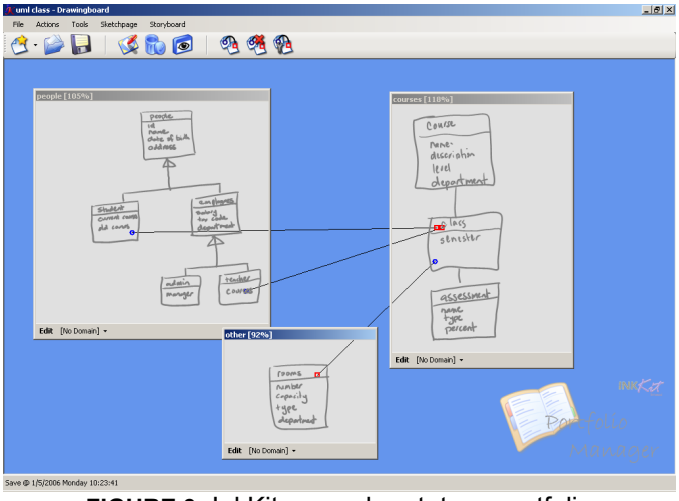

FIGURE 3: InkKit second prototype portfolio

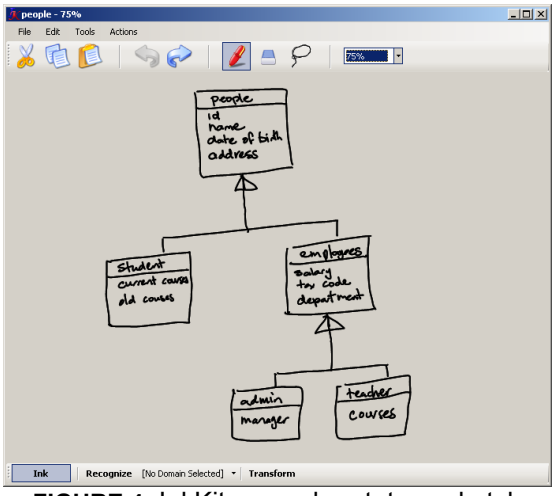

FIGURE 4: InkKit second prototype sketch

In principle the concept of a single view is appealing, however two of the usability problems, modality and display size are linked to this approach. We considered various approaches to reducing the impact of modality however none were likely to alleviate it completely. The display size limitations are solvable by networking tablet computers or using e-whiteboards. However networking computers adds a layer of complexity and is quite clumsy and e-whiteboards are not yet of sufficient quality to support accurate recognition. We chose to adopt a two monitor, two view interface; portfolio and sketch (Figure 2).

The portfolio (Figure 3), (the storyboard mode of the previous prototype), is displayed on an auxiliary monitor; each MDI form (sketch) has an edit button added to the bottom left of the form and the ink is always greyed. The link creation method was altered so that the user drags a line from the start-point to end-point. To edit a sketch the user opens the MDI form as a separate window (Figure 4). The sketches are shown on the tablet screen (where the stylus can be used for input). A number of architectural changes to the software were required so that the two views are synchronized. If no auxiliary display is available the portfolio is shown on the tablet display as an independent window.

We usability tested InkKit again using the same test setup described above with different test participants. Users commented that it was clear that inking was not available in the portfolio view and that it was easy to swap between windows. Creation of links by dragging also solved that problem. However the auxiliary display created a

different problem: a natural split of input devices is to use the stylus for the tablet and mouse for the auxiliary display. However, because the OS links the mouse and tablet cursors, each time a user moves focus to the other display both cursors follow. This causes a particular problem with the auxiliary display as manually moving the mouse cursor back to the display is slow. We are modifying InkKit so that when it is running with two displays the cursors and pointing device input events are split – the stylus cursor and events are handled by the active window on the tablet and the mouse cursor and event are handled by the active window on the auxiliary display.

4 DISCUSSION AND CONCLUSION

Early design environments must allow quick unconstrained recording of ideas. To provide this in a computer-based environment the computer interaction must be context free and minimize cognitive load so the designer can fully engage in the design process. Yet the value of computer-based sketch tools is derived from their added functionality. The initial design of the InkKit user interface was based on our experience with Freeform and a survey of other sketch tools and hardware options. Our first prototype was implemented with a single view interface requiring mode changes to indicate whether actions were to the storyboard or an individual sketch. The evaluation of this prototype suggested that different terminology provided a better metaphor and user testing showed that the single view modal interface and limited display space was likely to cause ongoing interaction problems.

We re-engineered the interface, changing the terminology and splitting the interaction between two views, the portfolio, and individual sketches. The portfolio is displayed on an auxiliary monitor, thus maximizing viewing space. Retesting showed that the new interface solved the modality problems while increasing the viewable area. Our original approach of a single viewable space remains closer to a tactile environment such as a tabletop or whiteboard and may be worthy of re-examination as large displays with sufficient input accuracy emerge. InkKit, in its current form, provides and excellent, easy to use, context free sketching interface that is supported unobtrusively by a powerful recognition engine.

An easy-to-use sketch tool allows the designer to engage in the design process rather than the computer interaction. Ensuring that sketch tools are intuitive requires ongoing careful design and evaluation as new hardware and more intelligent support continues to change the user environment.

REFERENCES

[1] Bailey, B. P. and Konstan, J. A., (2003) *Are Informal Tools Better? Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design*. Proceedings *CHI 2003*. Ft Lauderdale: ACM.

[2] Chung, R., Mirica, P. and Plimmer, B., (2005) *InkKit: A Generic Design Tool for the Tablet PC*. Proceedings *CHINZ 05*. Auckland: ACM.

[3] Damm, C. H., Hansen, K. M. and Thomsen, M., (2000) *Tool support for cooperative object-oriented design: Gesture based modelling on and electronic whiteboard*. Proceedings *Chi 2000*: ACM.

[4] Goel, V., (1995) *Sketches of thought*. The MIT Press: Cambridge, Massachusetts.

[5] Jarrett, R. and Su, P., (2003) *Building Tablet PC applications*. Microsoft Press: Redmond.

[6] Kara, L. B. and Stahovich, T. F., (2004) *Sim-U-Sketch: a sketch-based interface for SimuLink*. Proceedings *of the working conference on advanced visual interfaces*. Gallipoli, Italy: ACM Press.

[7] Landay, J. A., (1996) *Interactive sketching for the early stages of user interface design*. Carnegie Mellon University: Pittsburgh, PA.

[8] LaViola Jr., J. J. and Zeleznik, R. C., (2004) *MathPad2: a system for the creation and exploration of mathematical sketches*, in *ACM Trans. Graph.* p. 432-440.

[9] Lin, J., Newman, M. W., Hong, J. I. and Landay, J. A., (2000) *Denim: Finding a tighter fit between tools and practice for web design*. Proceedings *Chi 2000*: ACM.

[10] Plimmer, B. and Apperley, M., (2005) *From Sketch to Blueprint: Supporting the creative design process*. Proceedings *Workshop on improving and assessing pen-based input techniques*. Edinburgh.

[11] Plimmer, B. E. and Apperley, M., (2002) *Computer-aided sketching to capture preliminary design*. Proceedings *Australasian User Interface Conference*. Melbourne.

[12] Plimmer, B. E. and Apperley, M., (2003) *Freeform: A Tool for Sketching Form Designs*. Proceedings *BHCI*. Bath.