Beautifying sketching-based design tool content: issues and experiences

Beryl Plimmer[1] and John Grundy[1, 2]

[1]Department of Computer Science and [2]Department of Electrical and Computer Engineering
University of Auckland, Private Bag 92019, Auckland, New Zealand

`{beryl, john-g}@cs.auckland.ac.nz`

Abstract

With the advent of the Tablet PC and stylus-based PDAs, sketching-based user interfaces for design tools have become popular. However, a major challenge with such interfaces is the need for appropriate "beautification" of the sketches. This includes both interactive beautification as content is sketched and post-design conversion of sketches to formalised, computer-drawn diagrams. We discuss a number of beautification issues and requirements for sketching-based design tools, illustrating these with examples from two quite different sketching-based applications. We illustrate ways of supporting beautification, user interface design and implementation challenges, and results from preliminary evaluations of such interfaces.

Keywords: sketching-based user interfaces, sketch beautification.

1. Introduction

The emergence of the Tablet PC, large Electronic Whiteboards and PDAs using stylus-based input mechanisms have led to demand for sketching-based interfaces in diverse software applications (Damm et al 2000; Landay, 1997; Pomm and Werlen, 2004). Such interfaces may allow users to sketch content, typically a design of some sort, which is then progressively or in a single operation converted into a formalised, computer-rendered diagram or specification. Common applications for sketching include early-phase software design (Damm & Hansen, 2004; Chen et al, 2003); user interface design (Plimmer & Apperley, 2003, 2004; Newman et al, 2003) and CAD applications (Trinder, 1999).

A major challenge encountered in such environments is the need to "beautify" sketched content, both during sketching-based input and during conversion to computer-rendered form (Plimmer & Apperley, 2003). Consider the sketching-based user interface design tool in Figure 1. In this example, the designer has drawn a rough sketch of a Visual Basic user interface form design. When finished they want this converted into a Visual Basic form design using computer-rendered components shown below.

During the sketching of this form design, the user may want the sketched content modified incrementally. For example, to group radio buttons they may want them repositioned; when writing labels and text boxes they may want them to be moved so as to not overlap; and they may want to resize sketched content by drag-and-drop but have the resized items still look fully "sketched". Similarly, when having the formal design at the bottom of Figure 1 generated, they may want similar items sized the same e.g. check boxes, text fields and radio buttons, despite these being different sizes in the sketch; they may want items aligned to grids and items grouped and repositioned, and may want text in certain styles and fonts.

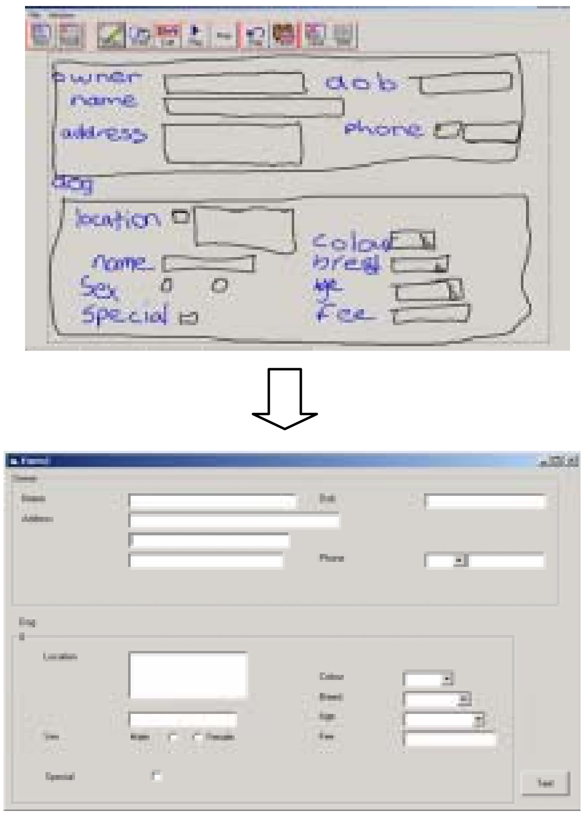


Figure 1 FreeForm Sketch to Visual Basic Form

Developing sketching-based design tools that provide appropriate sketch-time and formalisation-time beautification techniques is not straightforward. From our experiences in developing such tools we have identified different kinds of sketching-based applications that require different kinds of beautification techniques. In addition, we have identified a range of beautification techniques which can be deployed at different times and which are suitable for some applications and not others. We have evaluated the usability of a number of these techniques in two quite different sketching-based software applications. We hope that our experiences will be of help to others developing sketching-based design tool user interfaces.

In the following section we present two design tool applications which are the primary motivation for this work, comparing and contrasting their sketch beautification needs. We then describe a number of beautification techniques we and other researchers have identified and summarise their different behaviours and constraints. We illustrate many of these techniques using the two exemplar design tools and comment on challenges in designing and implementing such interface techniques. We review work done by other researchers in this area, summarise evaluations of our beautification techniques and outline key areas for future research on sketching-based user interface beautification.

2. Motivation

We have been developing two quite different sketching-based design tools – SUMLOW (Sketching UML On Whiteboard) (Chen et al, 2003), and FreeForm (Plimmer & Apperley, 2003, 2004). Both were designed to be used on a large-screen Electronic Whiteboard and to provide a shared, early-phase design environment. SUMLOW is for software designers using the UML (Unified Modelling Language). An example of using SUMLOW is illustrated in Figure 2(a & b). In the left-hand screen dump, a UML design sketch has been drawn, consisting of a mix of UML use case (ovals and stick figure shapes), classes (rectangle shapes with lines and text), and various relationships (lines between shapes).

SUMLOW allows various UML diagrams to be sketched and it incrementally formalises the diagram, recognising UML notational symbols as they are drawn. A multi-stroke input algorithm recognises complex shapes and a single-stoke algorithm is used for text recognition. The user manipulates the sketch, and after a time requests the sketch be converted into a formalised UML design diagram, as shown at the bottom of Figure 2 (b). Some information may be lost in this conversion. This can be due to both the sketching interface allowing non-UML sketched content to act as secondary notation, and also as some mixing of sketched UML notational symbols may be invalid in the formalised model.

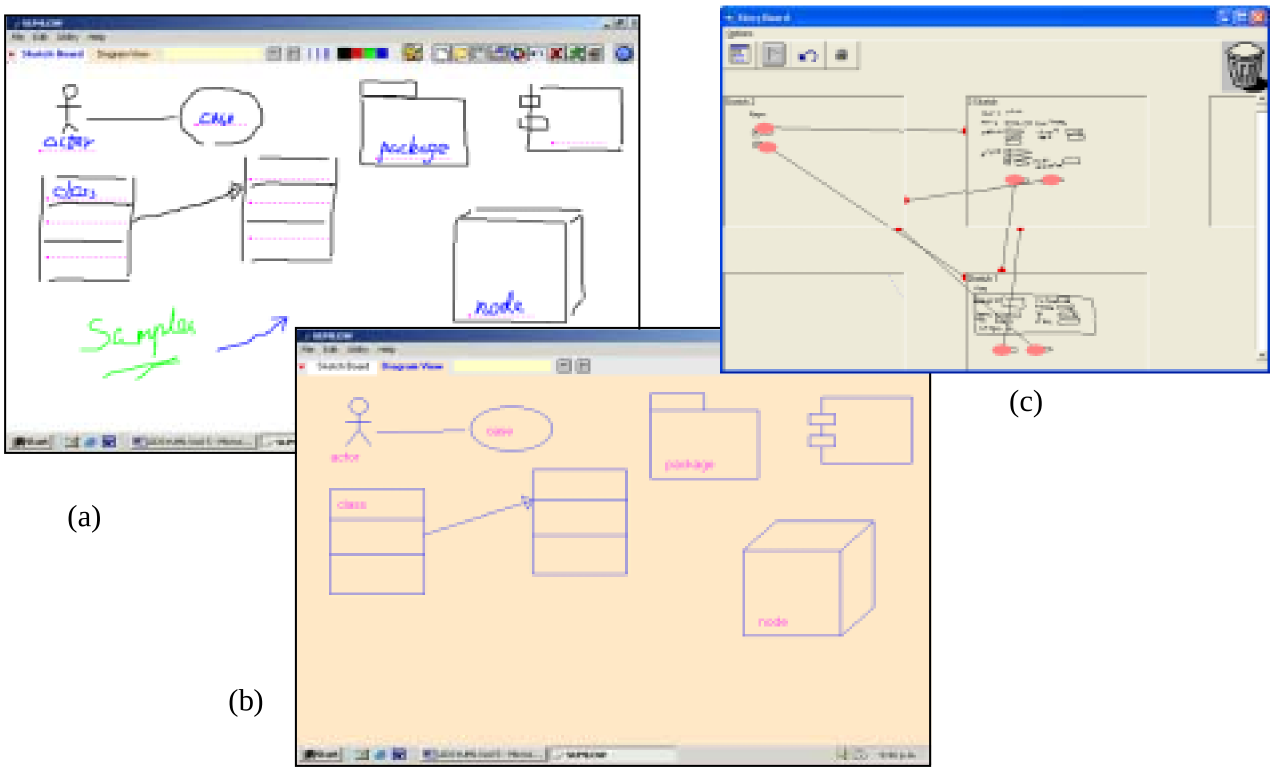


Figure 2. Examples of (a & b) SUMLOW UML design tool and (c) FreeForm storyboard.

FreeForm (Plimmer & Apperley, 2003) is software for designing user interface forms. In addition to the form sketch-space and ability to convert sketched designs as shown in Figure 1, FreeForm also includes a storyboard view Figure 2(c) and supports interactive checking of the designs. The first phase of recognition is carried out immediately a user stroke is completed so that functional gestures can be actioned and overdrawing can remove the underlying items.

Studies have shown benefits in working with informal sketched designs in preference to formal diagrams (Goel, 1995; Plimmer & Apperley, 2004), therefore most beautification and formalisation is left until the Visual Basic form generation. FreeForm uses a single stroke recogniser and then a rule base and dictionary for combining simple strokes into Visual Basic widgets and words. The rule base also includes beautification size constraints. In addition to size, widgets are aligned on to a grid and grouped appropriately.

In both of these sketching-based design applications there is a need to "beautify" the sketched content, in some cases during sketching-based design and in other cases at the post-design phase. This need for beautification occurs due to a number of issues:

- The designers want sketched content moved, resized or otherwise modified as it is sketched in order to implement interaction and/or syntactic constraints in the application. For example, the UML Sequence Diagram support in SUMLOW repositions some elements as they are sketched to ensure a meaningful diagram results. In FreeForm if a user draws one element over another, it makes no sense in a form design, so FreeForm removes the underlying element.
- The designer manipulates diagram content which has a flow-on affect on other content. For example, in SUMLOW after a UML class icon is moved by direct manipulation (drag-and-drop), association and generalisation lines connecting the class to other classes must be moved to maintain the connections with the other classes. We want to preserve the look-and-feel of the sketched content when doing these modifications. Also, if a user resizes a UML class icon, the enclosed attribute and method lists and separator lines must be sensibly repositioned.
- When formalising a sketched design, the formalised design requires application of layout and consistency heuristics. For example, in FreeForm the form design content like text boxes, labels and radio buttons requires beautification to make these elements consistent sizes and fonts. Also, attention needs to be paid to grouping of elements to ensure sensible beautification is done. In SUMLOW, UML class icons are rendered using a consistent font style and drawn to just enclose their name, attribute list and method list. No overlap of labels and lines are permitted as in the sketches.
- When formalising a sketched design, the conversion of the sketched content to computer-rendered content may require application of syntax rules. For example, adjacent radio buttons that are more-or-less aligned either vertically or horizontally on a FreeForm sketch should be aligned and evenly spaced on a Visual Basic form design. In SUMLOW, some UML diagram elements can not be mixed in the formalised UML diagrams and must be discarded or converted into annotations during the formalisation process.
- After formalisation, further beautification may be applied to the formalised design. For example, in SUMLOW the UML diagram may be re-laid out i.e. all content repositioned by a diagram layout algorithm. In FreeForm, application of form design styles e.g. font, line thickness, colour and shading may be applied to all form elements of specific types.
- We may want to import a formal design e.g. UML diagram into SUMLOW or Visual Basic form into FreeForm, and display this as a "sketch" to encourage exploratory design (Goel, 1995; Plimmer & Apperley, 2004). This requires "de-formalising" a computer-rendered design into a sketched-like appearance.

3. Requirements

SUMLOW and FreeForm illustrate two fundamentally different kinds of sketching-based applications. In SUMLOW, an abstract design model is being constructed using multiple, overlapping sketched views. In FreeForm, a concrete design of a form-based user interface is being sketched using a forms and an associated storyboard. Within SUMLOW there are different kinds of diagrams – use case, class and deployment diagrams all using box-and-line style with any layout/positioning of items being allowed. In addition sequence and state diagrams are more constrained as layout of content has semantic meaning when these types of diagrams are formalised. These differences in concrete vs abstract design models and flexible layout vs constrained layout diagrams leads to quite different beautification techniques being suitable for deployment in each environment.

3.1. Sketch-time Beautifications

Key sketch-time beautifications we have identified include recognising a shape after initial sketching and then making modifications to the sketch to highlight or modify sketch content based on other related design content. Clustering of sketch content is also required for some applications, where multiple ink strokes are grouped into a single element for further direct manipulation e.g. move and resize. Overlap removal between sketched content is sometimes necessary, as is repositioning sketched content to ensure semantic consistency of the design. When a sketch is resized or moved, related sketched content may need to be resized/moved/redrawn e.g. connecting lines, enclosed shapes. Alignment and snapping to a grid may be useful in some applications during sketch-time.

3.2. Formalisation-time Beautifications

When formalising a sketch, almost all sketched content will be resized and aligned in some way. This will typically involve applying resizing heuristics to sketched content to ensure the formalised design content is consistent in height and width. In addition, grid lines may be used to reposition the resized, formalised content horizontally and/or vertically. Whole or substantial part-design layout algorithms may be applied to the formalised design to improve the presentation of the formalised design elements as a whole. Consistent styles e.g. colour, line thickness, design element types and so on may be applied to the formalised design elements.

4. Examples

In this section we illustrate some of the sketch-time and formalisation-time beautifications we have implemented in FreeForm and SUMLOW.

4.1. Draw-and-Change

When sketching a UML use case, actor, class or object icon in SUMLOW, the tool adds "text entry" annotations to the sketched content after recognition of the shape type, as shown in Figure 3.

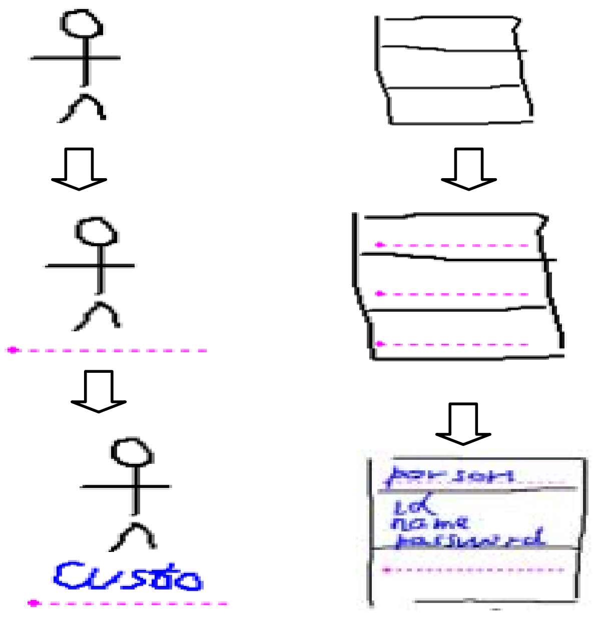


Figure 3 SUMLOW Use Case Sketch Beautification

For example, when sketching an actor shape, SUMLOW adds a label text area, and moves this label area down as the user writes the name. When recognising a UML class shape, SUMLOW adds three text entry areas for class name, attributes list and methods list. If necessary SUMLOW resizes the sketched class icon to fit the text areas.

4.2. Remove and Replace

In FreeForm an erase gesture (Figure 4a) will remove the underlying ink. Also if an ink stroke is drawn over another (Figure 4b) the underlying stroke is removed as layered ink makes no sense in a form design.

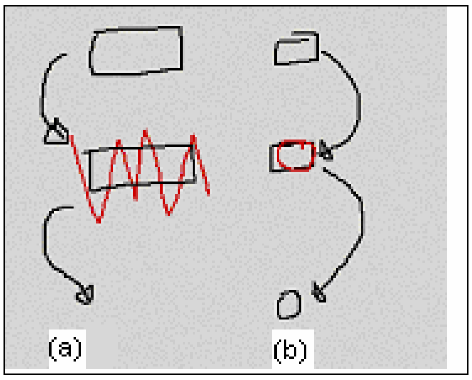


Figure 4 FreeForm Removing and Replacing Ink

In SUMLOW, a draw over-and-replace algorithm is used to resize existing sketched content. For example, to resize a class icon the user pushes the pen down inside the existing shape, then draws a replacement boundary for the class, and then SUMLOW resizes the class icon boundary and internal lines and text (Figure 5).

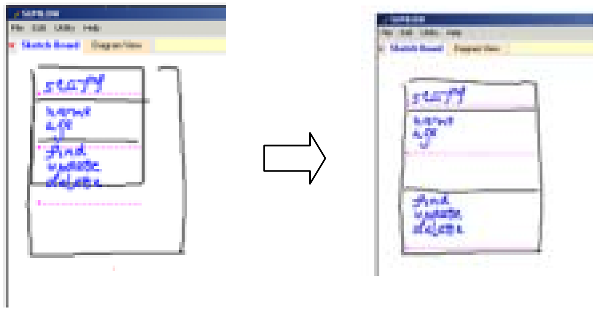


Figure 5 SUMLOW Resize

4.3. Move and Resize Element

To move a shape in SUMLOW, the user holds the pen down inside the shape boundary. They then drag the pen to the new position they want the shape and release it. When a shape is moved, the various sketched content making up the shape is moved with it. Lines connecting the shape to others are perturbed to try and preserve existing sketched connectors. For example, connectors between Actor and Use Case shapes in a UML sequence diagram are repositioned in this way (Figure 6).

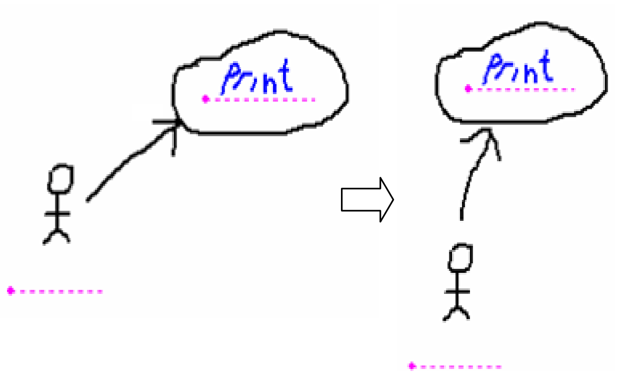


Figure 6 SUMLOW Connection Between Elements is Maintained When One is Moved

4.4. Move and Resize Group

In FreeForm the designer may decide to reorganise the form, the designer changes to edit mode, selects the ink to be move and drags and resizes.

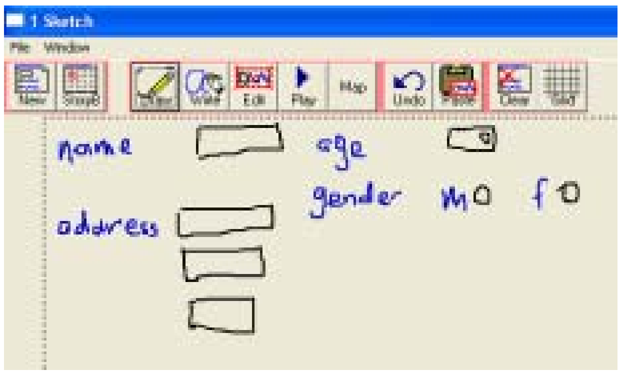


Figure 7 FreeForm Initial Diagram

The sketchy appearance of the elements is maintained during this process. Figure 7 shows a initial sketch, in Figure 8 the address group and age/gender group have been swapped and the address group has been resized

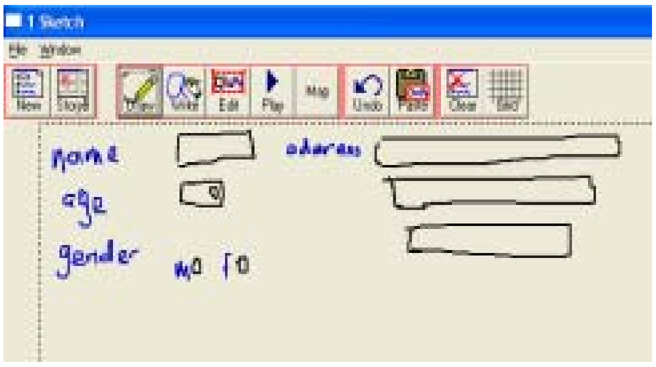


Figure 8 FreeForm Moving and Resizing

4.5. Alignment

In FreeForm, in preparation for conversion to a formalised Visual Basic form design the sketch elements are aligned onto a grid. The algorithm parses the sketch three times during this process. First each identified widget is positioned onto the grid by moving its top-left corner to the closest intersection point: the ink strokes of widgets that consist of more than one stroke, such as words or the dropdown list in Figure 9, are together. Sometimes two glyphs may gravitate to the same grid position, the beautification process moves one of the overlapping elements. Last, the algorithm parses the sketch aligning groups vertically and horizontally.

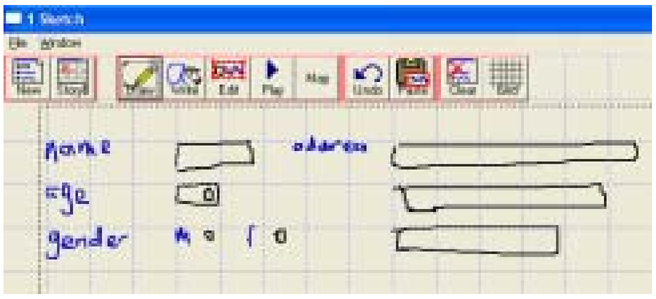


Figure 9 Aligned FreeForm Widgets

Sequence diagrams in SUMLOW require special layout as they are drawn, to ensure the semantics of the sequence diagram are adhered to (all objects at top; Liveness bars for method Invocations; staggered method invocation lines). As sequence diagram content is sketched, SUMLOW interactively moves objects to the top of the diagram, pushes existing method invocation lines down, and resizes method Liveness bars.

For example, when a user sketches an Actor or Object shape in a sequence diagram view, these are automatically moved above a line at the top of the view (Figure 10). As method Liveness bars (rectangular shapes between Actor and Object shapes) are added and connected by method Invocation lines (arrowed lines between Liveness bars), Liveness bars and Invocation lines are automatically resized and moved down to admit new ones.

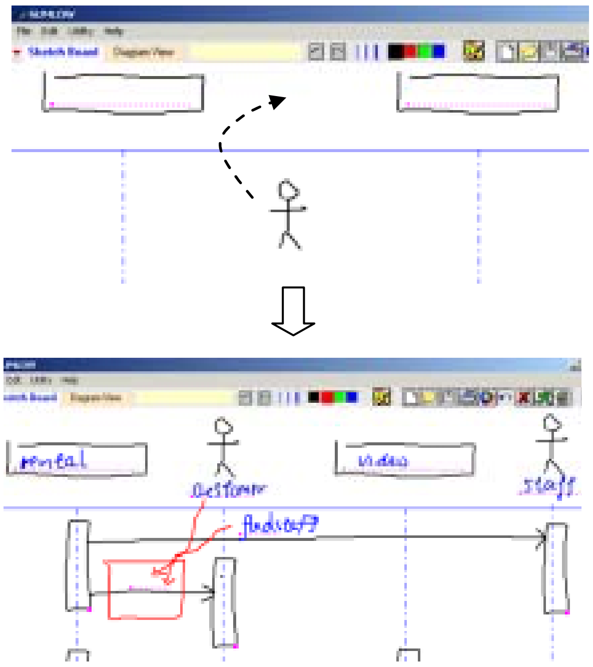
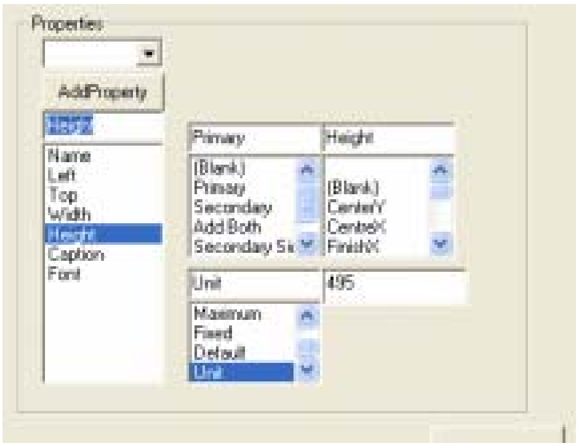


Figure 10 SUMLOW Automatic Layout of New Sketch Content in Sequence Diagram

4.6. Resize Formalise

FreeForm sizes and sets other attributes of generated Visual Basic widgets by applying user-defined beautification rules. In the options pane (Figure 11) the user defines how a widget's properties will be derived from the sketch glyph. For example the height of a text box may be set to be a unit of n pixels and this height can be related to the height of either the primary or secondary ink stroke. FreeForm calculates the height as being the integer number of units in the height of the sketched glyph. Other properties may be set with fixed, minimum and maximum values. Where text is associated with the Visual Basic widget, for example radio buttons, the text can be used in the generation of the widget name.


Properties

AddProperty

Name
Left
Top
Width
Height
Caption
Font

Primary	Height
(Blank)	(Blank)
Primary	CenterY
Secondary	CenterX
Add Both	FinishX
Secondary Si	

Unit	495
Maximum	
Fixed	
Default	
Unit	

Figure 11 FreeForm Properties Pane

The result of this beautification process is that the sketch Figure 8 creates the form in Figure 12 where all the widgets are aligned and of standard sizes.

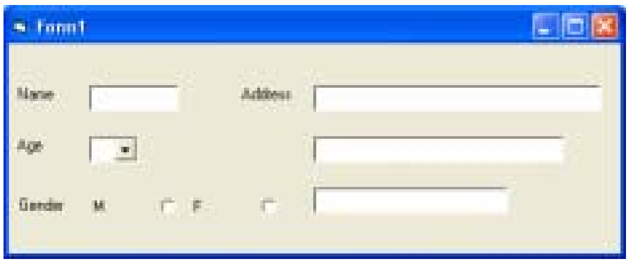


Figure 12 FreeForm Beautified Visual Basic Form

When SUMLOW diagrams are formalised, shapes are rendered in a standard style – line thickness, colour, size enclosing text and so on. Diagrams can be laid out using automatic layout algorithms, as the designs are abstract rather than concrete as in FreeForm. For example, Figure 13 shows a formalised version of the bottom sequence diagram from Figure 10. This is displaying Objects and Actors using a fixed size, has repositioned the method Liveness bars and connectors, and displays all text using the same font size and style.

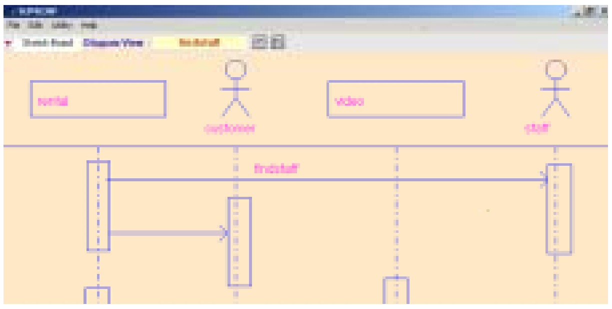


Figure 13. Formalised SUMLOW UML Sequence Diagram.

5. Discussion

Our first version of FreeForm included none of the sketch-to-form beautification described here. The Visual Basic form represented a direct translation of the sketch, even if the sketch appeared to be very tidy with glyphs aligned and of similar size the resulting formalised Visual Basic form looked untidy. The users in our first usability study suggested a tidier Visual Basic form as one of the highest priorities for system enhancements. In subsequent studies users were satisfied with the beautified Visual Basic form.

The current version of FreeForm moves the sketch glyphs onto a grid as a precursor to creating the Visual Basic form. In retrospect we are not sure that this is appropriate as it disturbs the appearance of the sketch. We plan to experiment with overlaying the sketch with regular shapes that indicate the position of the Visual Basic widget and allowing these shapes to be repositioned without moving the underlying sketch glyphs. Further studies will be required to assess the affect of this both on the retention of the sketch-feel, that is so important during early design, and the sketch-to-formal diagram transformation.

Beautifying handwritten words that are part of a diagrammatic design is problematic. Handwriting is usually larger than the equivalent computer font both in height and width, particularly on a digital whiteboard. Deciding on appropriate font sizes from the handwritten letters has been an ongoing problem in FreeForm, currently FreeForm generates all fonts at the default font size. We are aware that the Visual Basic forms generated by FreeForm are stretched horizontally because of the difference in size between handwritten and computer generated fonts.

SUMLOW included the described beautification techniques when we performed a usability study with the tool for experienced UML designers. In general they found the on-the-fly beautifications made by the tool took some getting used to. For example, users had to learn to pause while SUMLOW adjusted the size of a shape to include one or more added text areas after initial recognition of the shape. The draw over and replace metaphor was found to be appropriate by most users, though only after some time using the tool. Many users assumed a more traditional CASE-style resize operation was supported by SUMLOW instead of the more novel draw over and replace to affect a resize. We used the replace metaphor as we thought this was more like a real whiteboard, whereas some users perceive the tool more as a sketching-based CASE tool than E-whiteboard.

Moving, resizing and deleting shapes are all problematic for existing sketch content in SUMLOW. We wanted to preserve the sketch look-and-feel and hence we have a simple algorithm for shifting and redisplaying existing content like connectors between shapes. However some users found this can produce ugly results and preferred a real whiteboard-like metaphor of leaving existing content for manual update by users.

The need for layout constraints in some diagram types e.g. UML sequence diagrams, imposes order on some SUMLOW sketches, whereas others have no such layout and automatic resizing. Some users found this imposition of constraints disconcerting at first. However in general they accepted the additional editing constraints were necessary to ensure at least a partially-sensible design would result when the diagrams are formalised.

Currently SUMLOW imposes a default size and layout of shapes when formalising diagrams e.g. UML class icons are only rendered big enough to just enclose all of their attribute and method text. Users expressed a desire to sometimes be able to retain some diagram layout e.g. overall shape size, in the formalised diagrams. For example, to keep class icon size roughly the same as in the sketch, as they were using size as a secondary notation to denote importance. As with FreeForm, deducing formalised diagram font style and size, along with line thickness, might be useful in future to preserve user-defined annotations of sketch content.

SUMLOW does not impose an automatic layout algorithm on formalised diagrams apart from normalising shape and text size. However, in general it would be possible to do this as the diagrams represent abstract software design rather than concrete form layout as in FreeForm. Users did not request such a facility though this may be useful if larger designs were constructed, to enhance overall readability in the formalised version.

6. Related work

A number of systems have been developed to support approaches to beautifying sketched diagrammatic content. Pavlidis and Van Wyk (1985) describe a process of inferring from the original diagram appropriate constraints and then impose these constraints on the beautified version. Their work focused on rectilinear drawings such as hierarchy charts, it constrains the line segments by angle, checking parallel, joining and intersecting lines. They also discuss techniques for locating and standardising clusters and evenly spacing co-located items. AssistenzComputer (Bolz, 1993) analyses diagrams that consist of straight line segments using a user-defined knowledge base. This program looks for gaps and miss-alignments, using magnitudes of deviation to recognise defects. These are smoothed over in the beautified diagram's content. Igarashi et al. (1997) transform pen strokes into straight line segments, applying constraints so that lines lie at fixed angles and connections and intersections are constrained. When the user's intention is ambiguous the system presents multiple alternatives from which the user can then choose.

Immediate morphing of stylus input into regular geometric shapes (Arvo & Novins, 2000) and words (Pomm & Werlen, 2004) requires ongoing accurate recognition. The shapes are converted into computer-rendered diagram content as soon as enough has been drawn that the recognition algorithm can find a match. Such tools are restricted to a small set of standard geometric shapes and recognisable words.

A number of other sketch tools undertake some forms of diagram beautification. For example SILK (Landay, 1996) supports sketching of user interface designs with conversion into computer rendered content. Limited beautification is applied to the content, mainly focusing on recognising and rendering interface primitives. Demin (Newman et al., 2003) is a sketch tool for designing web sites. It immediately recognises simple symbols such as rectangles and lines. Beautification varies depending on the level the user is working at, for example in storyboard view a line drawn to indicate navigation between pages is smoothed and has a dot added to the source point and arrow to the destination point. Ideogramic (Damm & Hansen, 2002), a sketch-based case-tool, allows the user to choose the level of beautification. The ink strokes can be left unaltered or immediately recognised and transformed into a formal UML shape such as class or package widget. Formal and informal widgets can coexist on the same diagram. It also includes layout beautification.

A number of tools with sketch-based input perform beautification immediately on recognising content. Amulet (Myers, 1997) provides gesture-based construction of diagram content but replaces the sketched gesture immediately on recognition with computer-rendered content.

7. Summary

With the increase in popularity of the Tablet PC, stylus-based PDAs and Electronic whiteboards, sketching-based design tools have become more accessible, However, such applications require a range of "diagram beautifications" to improve their effectiveness and usability. A range of application domains mean some tools suit adoption of incremental beautification while others better suit post-sketch beautification of diagrams. We have built and evaluated two tools that adopt these different styles of beautification techniques. Experiences from usability studies have shown that different beautification techniques applied in appropriate ways significantly improve the acceptance of sketching-based design tools.

8. References

Arvo, J., & Novins, K. (2000). Fluid Sketches: Continuous Recognition and Morphing of Simple Hand-Drawn Shapes, Proceedings of UIST '00, San Diego, pp. 73 - 80.

Bolz, D. (1993). Some Aspects of the User Interface of Knowledge Based Beautifier for Drawings, Proceedings of Intelligent user interfaces '93, pp. 45-52.

Chen, Q., Grundy, J.C. and Hosking, J.G. (2003). An E-whiteboard Application to Support Early Design-Stage Sketching of UML Diagrams, In Proceedings of the 2003 IEEE Conference on Human-Centric Computing, Auckland, New Zealand, October 2003, IEEE CS Press.

Damm, C. H. Hansen, K. M. Thomsen, M. (2000). Tool Support for Cooperative Object-Oriented Design: Gesture Based Modeling on an Electronic Whiteboard. In Proceedings of CHI 2000 on Human factors in computer systems: the future is here, ACM Press, pp. 518-525

Damm, C. H., & Hansen, H. R. (2002). Ideogramic. http://www.ideogramic.com/ accessed 30 August 2004

Goel, V. (1995). Sketches of Thought. Cambridge, Massachusetts: The MIT Press.

Igarashi, T., Matsuoka, S., Kawachiya, S., & Tanaka, H. (1997). Interactive Beautification: A Technique for Rapid Geometric Design, Proceedings of UIST 97, Banff, pp. 105-114.

Landay, J. A. (1996) SILK: sketching interfaces like krazy. In Proceedings of CHI'96 on Human factors in computer systems: common ground, ACM Press, pp. 518-525.

Myers, B.A. (1997): The Amulet Environment: New Models for Effective User Interface Software Development, IEEE Transactions on Software Engineering, vol. 23, no. 6, 347-365, June 1997.

Newman, M. W., Lin, J., Hong, J. I., & Landay, J. A. (2003). Denim: An Informal Web Site Design Tool Inspired by Observations of Practice. Human-Computer Interaction, 18(3), pp. 259-324.

Pavlidis, T., & Vanwyk, C. J. (1985). An Automatic Beautifier for Drawings and Illustrations. ACM

SIGGRAPH Computer Graphics archive, 19(3), pp. 225 - 234.

Plimmer, B. E., & Apperley, M. (2003b). Software for Students to Sketch Interface Designs, Proceedings of Interact, Zurich, pp. 73-80.

Plimmer, B. E., & Apperley, M. (2004). Interacting with Sketched Interface Designs: An Evaluation Study., Proceedings of SigChi 2004, Vienna, pp. 1337-1340.

Pomm, C., & Werlen, S. (2004). Smooth Morphing of Handwritten Text, Proceedings of AVI '04, Gallipoli, pp. 328-335.

Trinder, M. (1999). The Computer's Role in Sketch Design: A Transparent Sketching Medium,, *Proceedings of Computers and Building*, *CAAD futures 99*, Atlanta, pp. 227-244.