From Sketch to Blueprint: supporting the creative design process.

	Beryl Plimmer	**&**	**Mark Apperley**
	University of Auckland		**University of Waikato**
	New Zealand		**New Zealand**
Email	**beryl@cs.auckland.ac.nz**		**m.apperley@cs.waikato.ac.nz**

Abstract

Imagine a computer-based design environment where you could work from first scribbling to finished design; moving freely between hand-drawn sketch and formal diagram. Designs are best started with pen and a blank canvas because ambiguity and informality are desirable attributes during initial design. However, finished diagrams need to be formal and rule compliant so that no ambiguity remains. Early, low-cost prototyping is also an effective design strategy for exploring and refining functional requirements. We show that with a suitable computer interface and supporting software it is possible to achieve two significant goals; first, to provide varying levels of diagram formality and second, to provide semi-functional informal prototypes. In this paper we begin by describing the role of visual props in the design process and the requirements of a computer tool to better support the design process. We then describe a prototype we have built and evaluated to explore and refine these issues.

Keywords

Design tools, sketching, pen-based computing

Introduction

Sketching is an integral part of early design (Tversky 1999) both for a person working alone and for designers working in groups. The sketch sets up a visual dialogue. When working alone the design provides “backtalk” (Goldschmidt 1992) that the designer uses to generate more ideas. In a group situation, some claim that the discussion and process is as valuable as the artefact (Bly and Minneman 1990) because as the design is created, the group develops a shared understanding of the problem space (Bekker 1993).

At some point the informal sketch must be formalized into a blueprint or design. Formal diagrams use regular rectangles, circles, symbols and lines and adhere to domain specific rules. Current best practice sees most designers using low-fidelity tools (pen and paper/whiteboard) for initial informal diagrams and hi-fidelity tools (computer drawing environments) for formal diagrams. The move from low to high fidelity tools creates a discontinuity in the design process (see Figure 1). This is usually a one-time, one-way process because of the effort required to convert the diagram, and for many, there is an inclination to make this move early during the design lifecycle to avoid rework. Most inexperienced designers do not recognize the adverse affect that this has on the design process (Black 1990); once a design is presented formally, people are disinclined to make significant change or revert to the informal: the on-going design effort concentrates on refinement and beautification.

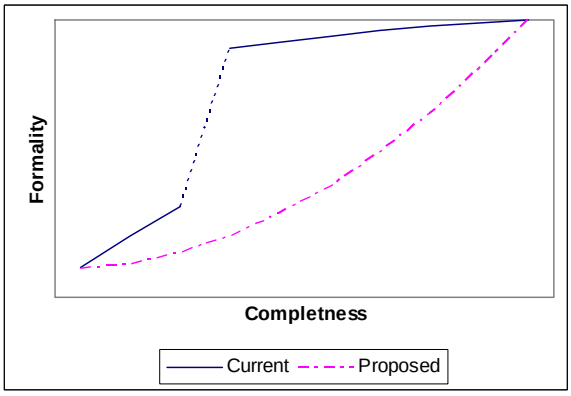

Figure 1: Design Process Support

Picture a computer-based environment that supports sketching and also automatically provides a formal representation of the design. Imagine being able to prototype a design while it is just a sketch; wander through a virtual building, run a piece of machinery or interact with a user interface. Imagine being able to work with an informal sketch until the design is complete and then being able to re-render the design as a formal diagram with out rework.

In this paper we describe a design friendly environment that provides such continuous computer support to the creative design process, where the designer can easily translate from informal to formal designs without significant effort or loss of data. Our example application domain is that of designing computer interface forms in the Visual Basic™ interactive programming environment. We first establish the requirements for this computer-based tool so that it will provide continuous support throughout the design process. We then describe the implementation and evaluation of a prototype system. The final section of the paper draws together relevant literature on sketching tools in the design process, and provides a commentary on the achievements and shortcomings of the tool described.

The intended target, and the evaluation audience for our prototype, has been students of introductory programming courses. Our experience with beginning students of programming is that it is difficult to convince them that low-fidelity tools are quicker for initial design, and that their finished user interface will be superior if they work first with an informal prototype. Our goal has been to provide an environment that is conducive to informal design, has all the inherent advantages of computer support, allows designs to flow from informal to formal visualisations, and supports interactive prototyping with the informal design.

Requirements

From the literature on design the requirements for a computer tool to support designers are: an informal drawing space; the capacity to translate between informal sketch and formal diagram; and the facilitation of informal prototyping (Goldschmidt 1992; Goel 1995; Tversky 1999). Most of these requirements are common across disciplines, but here, we focus on computer interface design, the target domain of our prototype application.

Informal Drawing Space

In the informal drawing space, sketching must be rapid and unconstrained. This suggests a pen applying (digital) ink directly onto the output surface analogous to pen and paper or pen and whiteboard. For individual use, devices such as tablet PCs provide such an interface, while for groups, digital whiteboards are available. While traditional media do not have built-in copy, paste and undo functionality, editing support is one of the intrinsic bonuses of computer environments and should be provided.

An advantage of computer-based drawing is that it typically provides a virtual canvas that can be of any size. This is also a disadvantage, given that the viewing window often represents a quite small portion of this canvas. Various techniques can be employed to lessen this problem, such as zooming and the use of radar windows. The potential provision of multiple linked canvases can also be an advantage.

Translation Between Sketch and Diagram

The second requirement is the ability to translate the sketch to a formal diagram. To achieve this, the system needs to be able to both recognize and appropriately beautify (tidy) the informal diagram(Plimmer and Apperley 2003). The formalized diagram must adhere to the specific notation and rules of the domain.

Recognition of sketches is technically quite challenging. It is simplified if some rules are imposed on the drawer, but a balance is required. If the user is overly constrained it is likely to impact their design work, in much the same way as current widget-based design tools distract from the design task. On the other hand, the best recognition techniques give very poor results unless there are some drawing rules. Constraining recognition to the specific notations of a particular domain is a reasonable compromise. This is not to say that users may use only that specific notation, but that unrecognized elements are dealt with either by omission or substitution on a ‘best guess’ basis.

Beautification is the process of standardising the formal diagram so that it looks appropriate and adheres to the rules of a particular notation. Continual beautification is possible (Arvo and Novins 2000), but this defeats the goal of maintaining a sketchy informality to the design. There are two different approaches that can be taken; either recognition is applied to the sketch to generate a formal presentation, or digital ink is beautified independently of the recognition. Which approach is best may depend on the specific domain.

With the recognition first approach, library versions of the recognized elements are used for the generation of the formal diagram. The attributes of, and relationships between, elements can be defined to beautify the diagram. For example user interface design rules may include ‘all radio buttons will be of height x and width y’ and ‘if two or more radio buttons are adjacent and more-or-less aligned, align and space them evenly’.

Beautifying ink independent of recognition is a multi-part process. First, freehand strokes that seem to represent straight lines, curves and regular angles (90, 45, 30 degrees as appropriate) are converted into regular straight lines, curves and angles. Then lines that appear to be intended to join or intersect in a particular way are, in fact joined or made to intersect in a regular fashion. Finally the relationship between parts of the diagram are standardized so that a row or column of elements of approximately the same size and position are made the same size, aligned and spaced (Pavlidis and VanWyk 1985).

Best attempts at automatic recognition and beautification do not result in a perfect formal representation. Failures must be acknowledged and dealt with in the system design. One approach could be for the software to continually maintain parallel representations of the diagram and alert the user whenever there is a failure with the recognition or beautification. Another approach is, regardless of whether parallel representations are maintained, to notify the user of problems only when they change views. We prefer this approach as it is less likely to interfere with the design process. Simple interaction strategies are required for correcting recognition or beautification.

Interactive Prototypes

The third requirement is to facilitate interaction with the informal prototype. The sophistication of the prototype that can be generated depends on the accuracy of the recognition and the functionality of the final product. For example Stahovich (1995) generated multiple models of simple mechanical drawings. Complex functionality is more difficult to prototype. However, a highly functional prototype may not be necessary. User interface designers assert the value of very simple prototypes that emulate screen behaviour and wizard of oz techniques where the prototype user is left to imagine the system functionality (Wagner 1990; Wong 1992).

To briefly summarize the requirements. A computer environment sympathetic to designers provides a freehand direct pen input/display space; support to move between informal and formal representation via recognition and beautification algorithms; and a way to interact with the informal design as a low-fidelity prototype.

Freeform Prototype

To operationalize and further explore these requirements we have developed Freeform, a software tool for designing Visual Basic™ (VB) forms. Freeform is imbedded in the VB integrated development environment (IDE) as an add-in. It is designed for use on a LIDS digital whiteboard (Apperley, Dahlberg et al. 2001) that uses a Mimio™ digitizer bar to accept input from Mimio pens, and provides a shared workspace for small groups. However, Freeform can be used on any computer that will run VB and accept mouse input, including tablet PCs. Freeform has completed two cycles of requirements specification, development and usability testing. There are four parts of the tool: sketch spaces; storyboard and run mode; recognition and beautification; and creation of the VB form.

Sketch Space

In a sketch space (Figure 2) the user can draw much as they would a traditional whiteboard with basic computer-editing functionality. The pen input is in three modes: two inking modes, drawing and writing; and edit mode. The division of ink types is to improve the automatic recognition. The inking modes generate digital ink output strokes that are displayed on the pen path. One editing gesture is recognized in ink mode, the delete gesture (Figure 3). In edit mode ink strokes can be cut, copied, moved, resized and re-typed (drawing to writing or visa versa) either individually or in groups. The system can have any number of sketches open at one time; they can be saved individually or together as a project.

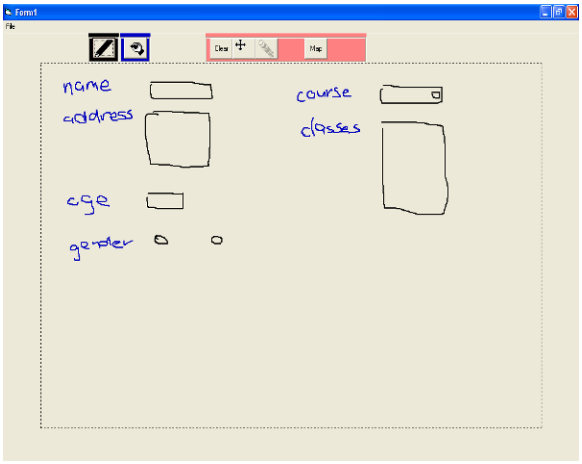

Figure 2: Sketch Space

Figure 3: Delete Gesture

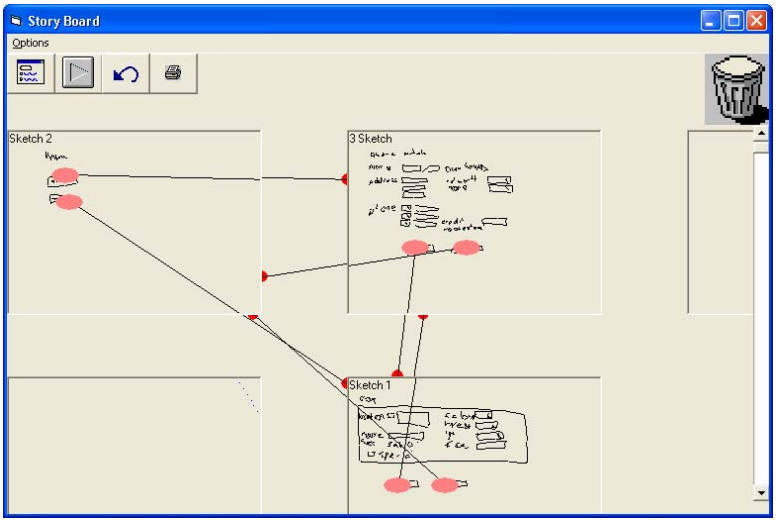

Figure 4: Storyboard

Interactive Prototype

Screen-level interaction emulation can be achieved by linking components on a screen with new (changed) renditions of the screen in a linked storyboard. In the storyboard (Figure) each sketch in the project is shown as a zoomed-out picture. Navigation paths can be created between the sketches. By drawing a line from one sketch to another a link is established that is active in run mode. The pictures can be rearranged on the storyboard to better depict the relationships between the sketches.

Run mode provides an interactive prototype. Users can ink anywhere on a sketch and use the navigation hotspots to move between sketches. The prototype is, apart from the hotspots, non-functional. More functionality could have been added by invoking recognition before the user entered run mode. However, we chose to deemphasize recognition as we did not want users concerns about recognition to overshadow the design process.

Recognition and Beautification

Recognition and beautification are necessary in order to create a formal VB form. The first step in recognition is classifying the individual ink strokes. Freeform implements a modified version of Rubine's (1991) algorithm, a statistical, single-stroke recognizer that matches input strokes against a library. To improve success rates we split ink into two types, drawing and writing, and added two features to Rubine's feature set, the x and y position that represent the point of balance of a stroke. Rubine's algorithm gave satisfactory results for drawing.

Recognising writing is much more difficult. Existing text recognition components such as Microsoft's Text Services™ were difficult to integrate into the project. Our first prototype used only Rubine's algorithm for character recognition. Although character recognition rates were reasonably high, the success rate for a word was close to zero, so during our first usability study we hid the character recognition. Participants commented that there was little point writing if the writing wasn't recognized and used greeking in place of words – a technique that, while quick is not very meaningful.

To improve recognition in the second prototype we restricted the character set to lower case characters, implemented a rule base to combine strokes for characters that are commonly formed using two strokes (f, t, i and j) and then matched likely letter combinations to a vocabulary list. We also made it easy to correct recognition errors by selecting a word from the vocabulary list (see Figure 5). While recognition rates are still low, the combination of recognition and easy correction were sufficient to encourage users in a subsequent usability study to write whole words.

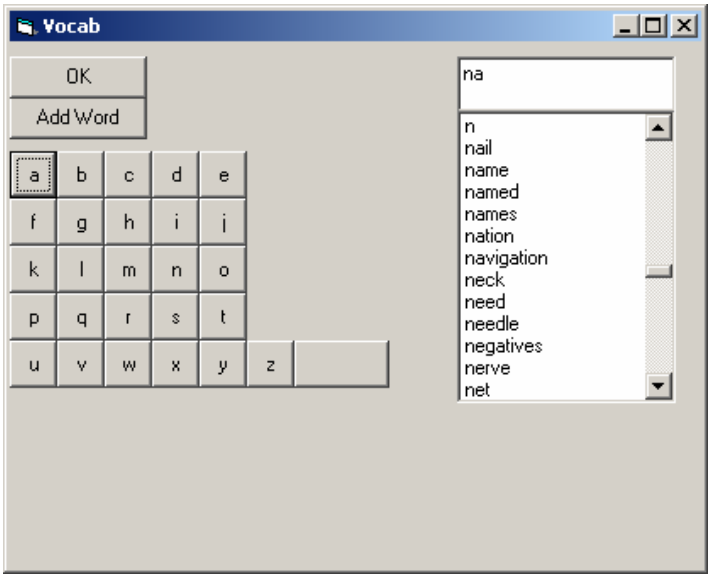

Figure 4: Vocabulary form

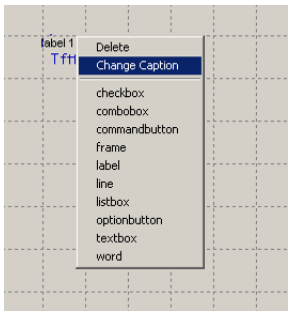

Figure 5: Glyph Menu

We had two concerns about the constraints that recognition placed on user interaction. The first was the need for users to form shapes such as rectangles with a single stroke; this proved not to be a problem, with users adapting very quickly. The second was the need for two separate inking modes, drawing and writing. This did prove to be an issue.

A second rule-base is used to combine drawing strokes and words into glyphs. The rules for each glyph are defined in terms of a primary element (stroke or word) and, optionally, a secondary element. Secondary elements have a spatial relationship (inside, beside, left, right, above, below) to the primary element. For example a textbox is a medium sized long rectangle with no secondary element. Buttons and dropdown lists are the same primary element (medium rectangle) with a secondary element; a command button contains a word, and a dropdown list contains a small triangle. During one of the usability studies, the success rate for recognising glyphs was 86% with two of the six participants achieving 100%, the worst score 62% (Plimmer 2004). To correct miss-recognition the user taps on the recognition data (**Figure 5**) and chooses from the list of alternative glyph types.

To create the formal diagram, each glyph is transformed into a VB widget and placed onto a form in the VB Form Designer. Freeform runs as a VB add-in, so the user can move between the VB Form Designer window and Freeform with a single tap. We found that beautification is important when transforming a sketch to a formal diagram. Our first prototype did not include beautification; participants in the first usability test were almost unanimous in their dissatisfaction with the formal representation of their design. A sketch that looks quite tidy (Figure 6), without beautification transforms to something that makes us grimace (Figure 7)!

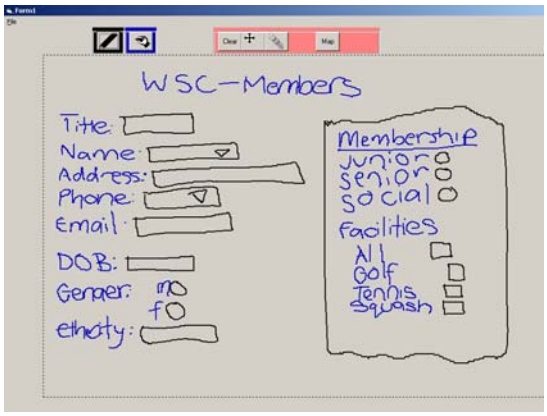

Figure 6: Sketched Design

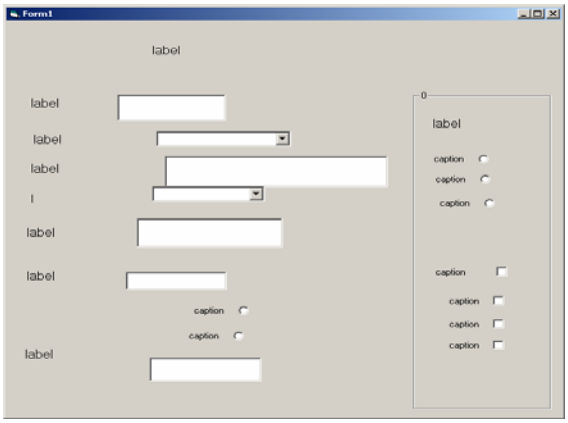

Figure 7: Form from Sketch in Figure 6

In order to beautify the recognized diagram, size attributes were added to each VB widget definition and a placement grid utilized. The dimension rules set fixed, minimum, maximum or unit values for the height and width of each widget. Widgets such as radio buttons have a fixed height, while textboxes are calculated from the sketch element and set to a multiple of the unit height (say 50 pixels). To align the elements, each is adjusted on the nearest grid intersection point. With beautification, the forms in a subsequent usability study received positive comments from users (see Figures 9 and 10).

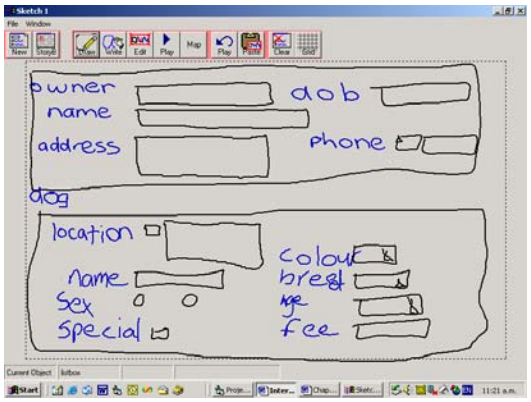

Figure 8: Sketched design

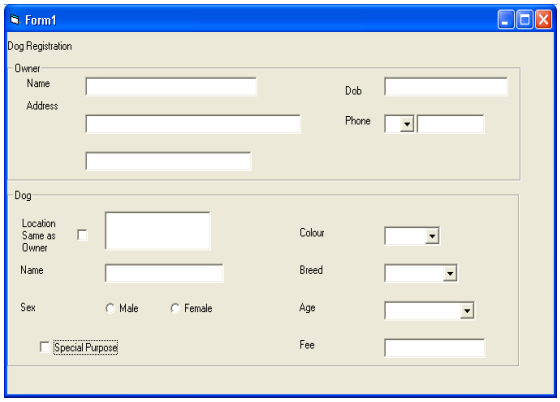

Figure 9: Form from Sketch in Figure 8

Evaluation

Two comparative evaluations have been completed. For both studies the subjects were beginning student programmers. In the first study Freeform was compared with a whiteboard as a design environment with respect to: the designs produced; the design process; the participants' understanding of the design problem; and their attitude to informal prototyping. We observed during this study that the participants made many more changes to their design when using the Freeform environment. This lead us to conduct a second study where participants were asked to review two designs, one presented as a formal diagram in the VB Form Designer, and the other an informal diagram in Freeform. Again, we found that the use of Freeform resulted in more changes during the design process, and better design outcomes. Details of these two studies, and a more comprehensive description of the outcomes, are described elsewhere (Plimmer and Apperley, 2004).

Related Work

A number of other sketch tools have been developed and described. The earliest is Sutherland's (1963) sketchpad that used a cathode-ray screen and light-pen to create geometric shapes. More recently the work of Xerox Labs (Elrod, Bruce et al. 1992) on the Liveboard project for meeting support using digital whiteboards, which has given rise to further interest and research on informal interfaces.

Sketch software has been developed for a range of different application domains. Landay and his colleagues (Landay and Myers 2001; Newman, Lin et al. 2003) have developed Silk and Denim for computer form and web page design respectively. Silk allows users to draw a form, add typed words to the form, interact with it in a run mode and export the form in two different formats. Denim extends Silk for web site design, and provides a hierarchy of five levels to show the structure of the web site.

Bailey et al (Bailey and Konstan 2003) developed an environment for prototyping multimedia applications. This environment allows users to incorporate pictures and functional video clips and sound bites into the sketch. This has been evaluated against a more formal commercial tool, and paper.

Both Knight/Ideogramic (Damm, Hansen et al. 2000; Damm and Hansen 2002) and SUMLOW (Chen, Grundy et al. 2003) are sketch-based UML tools. Each includes techniques for showing the design at different levels of formalisations. With Knight, a diagram can comprise of a mixture of formal, rule-compliant widgets and informal widgets. SUMLOW maintains two concurrent views of a diagram, one informal and the other formal.

Gross (1994) developed a sketching platform that has been used as a basis for a number of different applications, including one for an index to web pages and another for electrical diagrams. Stahovich et al. (1996; 1997; 1998) analysed mechanical engineering diagrams to synthesize new designs that met the same mechanical requirements. Trinder (1999) worked with architectural drawings and has contributed ideas on the layering of sketches.

Discussion

Appropriate computer tools can provide support for designers throughout the design lifecycle and facilitate interaction with informal prototypes. A design-friendly computer tool needs to support direct pen input into a space for informal design that does not constrain the designer, but supports them with normal editing functionality. Low-fidelity prototypes have been lauded in the past; interactive informal prototypes are an exciting extension. They offer the advantages of low commitment and interactivity. Recognition and beautification provide a path to smoothly translate informal diagrams to formal designs. Freeform, the example we have developed demonstrates that an electronic environment can produce designs that are as good as those produced using traditional media.

In order to support preliminary design a computer tool needs to facilitate inking with a pen directly onto the output surface in such a manner that it seems the same as working on paper or a whiteboard. This is now possible with both large digital whiteboards suitable for group work and tablet PCs that are more suited to individual work. It is not enough to simply provide the equivalent of a pen and whiteboard. The user expects a computer tool to facilitate ‘normal’ computer support for editing and file persistence. This alone is an advantage over traditional media.

However, a computer can add much more. First, it is not difficult, as we have illustrated with Freeform, to turn a simple sketch into a non-functional prototype demonstrating screen-level interaction. Our first evaluation studied noted that students engaged in prototyping with Freeform a manner we did not observe with the whiteboard. Our second evaluation study demonstrated that the participants were more inclined to change the informal prototype than the formal one. Taking these two findings together suggests that interactive informal prototypes may add a new dimension to the design process that is not possible with static designs.

Further, with ink recognition and beautification the translation from informal to formal can be delayed until latter in the design lifecycle, maintaining fluidity in the design, and avoiding undesirable over-early fixing of the design. There are technical challenges with both recognition and beautification.

The design space we provided in Freeform imposes a number of restrictions if the user wishes to invoke recognition. We were concerned by both the need for single-stroke shapes and the two ink modes. The single-stroke shapes caused no difficulty in either the usability studies or the evaluation studies. The existence of two inking modes remains a problem to be addressed.

Beautification is an area where there is much to be learnt. We obtained user satisfaction with the beautification we applied to our second prototype. However we can see scope for more sophisticated algorithms. The reverse process of beautification, representing a formal element as informal, perhaps to allow redesign of an existing artefact, is also an interesting area worthy of more exploration.

Our example is quite specific in terms of diagram type and domain. We believe that there may be some fundamental differences between diagrams that describe an artefact such as architectural drawings, engineering drawings and user interface designs and those that describe a process or virtual relationship such as UML diagrams and organisational hierarchy charts. It may be that for UML diagrams early formalisation is positive, as the notation is a powerful part of what the diagram is saying. We are also aware that there are different recognition and beautification techniques required for glyph-type diagrams, such as Freeform supports, and graph-based diagrams where there are connectors and intersections, such as UML diagrams (Plimmer and Grundy 2005).

References

Apperley, M., B. Dahlberg, et al. (2001). Lightweight capture of presentations for review. IHM-HCI, Lille, ACM.

Arvo, J. and K. Novins (2000). Fluid Sketches: Continuous recognition and morphing of simple hand-drawn shapes. UIST '00, San Deigo, ACM.

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

Bekker, M. M. (1993). Representational issues related to communication in design teams. Chi '93, ACM.

Black, A. (1990). "Visible planning on paper and on screen: The impact of working medium on decision-making by novice graphic designers." Behaviour and information technology **9**(4): 283-296.

Bly, S. A. and S. L. Minneman (1990). Commune: A shared drawing surface. Conference on Office Information Systems.

Chen, Q., J. Grundy, et al. (2003). An E-whiteboard application to support early design-stage sketching of UML diagrams. Human Centric Computer Languages and Environments, Auckland, NZ, IEEE.

Damm, C. H. and H. R. Hansen (2002). Ideogramic.

Damm, C. H., K. M. Hansen, et al. (2000). Tool support for cooperative object-oriented design: Gesture based modelling on and electronic whiteboard. Chi 2000, ACM.

Elrod, S., R. Bruce, et al. (1992). "Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration." CHI '92: 599-607.

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

Goldschmidt, G. (1992). "Serial sketching: Visual problem solving in design, The backtalk of self-generated sketches." Cybernetics and Systems **23**: 191-219.

Gross, M. (1994). Recognizing and interpreting diagrams in design. AVI 94, Bari, Italy, ACM.

Landay, J. and B. Myers (2001). "Sketching Interfaces: Toward more human interface design." Computer **34**(3): 56-64.

Newman, M. W., J. Lin, et al. (2003). "DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice." Human-Computer Interaction **18**(3): 259-324.

Pavlidis, T. and C. J. VanWyk (1985). "An automatic beautifier for drawings and illustrations." ACM SIGGRAPH Computer Graphics archive **19**(3): 225 - 234.

Plimmer, B. E. (2004). Using Shared Displays to Support Group Design; A Study of the Use of Informal User Interface Designs when Learning to Program. Computer Science. Hamilton, University of Waikato.

Plimmer, B. E. and M. Apperley (2003). Software for Students to Sketch Interface Designs. Interact, Zurich.

Plimmer, B. E. and J. Grundy (2005). Beautifying sketch-based design tool content: issues and experiences. AUIC 2005, Newcastle, Australian Computer Society.

Rubine, D. (1991). Specifying gestures by example. Proceedings of Siggraph '91, ACM.

Stahovich, T. F. (1997). Interpreting the engineer's sketch: A picture is worth a thousand constraints. AAAI Symposium on Reasoning with diagrammatic Representations II, Cambridge, Mass.

Stahovich, T. F. (1998). "The engineering sketch." IEEE Intelligent Systems(13 (3)): 17-19.

Stahovich, T. F., R. Davis, et al. (1995). Turning sketches into working geometry. ASME Design Theory and Methodology.

Stahovich, T. F., R. Davis, et al. (1996). "Generating multiple new designs from a sketch." AAAI-96: 1022-1029.

Sutherland, I. E. (1963). Sketchpad: A man-machine graphical communication system. Spring joint computer conference: American Federation Information Processing Societies, Montvale, New Jersey.

Trinder, M. (1999). The computer's role in sketch design: A transparent sketching medium,. Computers and Building, CAAD futures 99, Atlanta.

Tversky, B. (1999). What does drawing reveal about thinking. Visual and spatial reasoning in design, Cambridge, Mass.

Wagner, A. (1990). Prototyping: A day in the life of an interface designer. The art of human-computer interface design. B. Laurel. Reading MA, Addison-Wesley**:** 79-84.

Wong, Y. Y. (1992). Rough and ready prototypes: Lessons from graphic design. Human Factors in Computing Systems CHI '92, Monterey.