Final Thesis

Prototyping Tools for the Early Stages of Web Design

by

Irene Anggreeni

LITH-IDA-EX--06/030--SE

2006-05-03

Avdelning, institution
Division, department

Institutionen för datavetenskap

Department of Computer
and Information Science

Datum
Date

2006-05-03

Linköpings universitet

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English

☐ _____

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport

_____

ISBN —

ISRN LITH-IDA-EX--06/030--SE

Serietitel och serienummer
Title of series, numbering

ISSN —

URL för elektronisk version

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-6487

Titel
Title

Prototyping Tools for the Early Stages of Web Design

Författare
Author

Irene Anggreeni

Sammanfattning
Abstract

There is a gap between low-fidelity prototyping using paper and high-fidelity prototyping using computers in web design. Both serve well in different stages of web design, but are not well integrated. Prior studies have examined the practice of web designers. The studies resulted in a number of alternative prototyping tools, which focus on informal representation and try to prolong sketching in the design process.

The thesis proposes a design of a prototyping tool that makes use of existing paper sketches. In paper prototyping, a human who acts as the "computer" makes the sketches interactive. In the prototyping tool put forward in the thesis, the interactivity of the sketches is instead created on the computer. The novel prototyping tool needs to support the interactions and behaviours used in web design, and it must be easy to use so that the web designers do not have to invest too much time learning it.

The prototype of the tool is a sketch-and-scan interface, thus allowing the use of paper the way it is. The functionality supports both documentation and computer interactivity. Usability tests and expert reviews were conducted, involving students, lecturers and researchers in human-computer interaction.

The results elaborate previous research on prototyping practice, and a designers' wish list was formulated. A prototyping tool is expected to support communication between users, designers and developers; as well as to reduce a designer's need to change his work practice when using the tool.

Nyckelord
Keywords

paper prototyping, prototyping tool, web design, scenario-based design, interaction design, sketch-and-scan

Linköpings universitet
Department of Computer and Information Science

Final Thesis

Prototyping Tools for the Early Stages of Web Design

by

Irene Anggreeni

LITH-IDA-EX--06/030--SE

2006-05-03

Supervisor & Examiner: Mattias Arvola

Prototyping Tools for the Early Stages of Web Design

Irene Anggreeni
Linköping University, Sweden

Abstract

There is a gap between low-fidelity prototyping using paper and high-fidelity prototyping using computers in web design. Both serve well in different stages of web design, but are not well integrated. Prior studies have examined the practice of web designers. The studies resulted in a number of alternative prototyping tools, which focus on informal representation and try to prolong sketching in the design process.

The thesis proposes a design of a prototyping tool that makes use of existing paper sketches. In paper prototyping, a human who acts as the "computer" makes the sketches interactive. In the prototyping tool put forward in the thesis, the interactivity of the sketches is instead created on the computer. The novel prototyping tool needs to support the interactions and behaviours used in web design, and it must be easy to use so that the web designers do not have to invest too much time learning it.

The prototype of the tool is a sketch-and-scan interface, thus allowing the use of paper the way it is. The functionality supports both documentation and computer interactivity. Usability tests and expert reviews were conducted, involving students, lecturers and researchers in human-computer interaction.

The results elaborate previous research on prototyping practice, and a designers' wish list was formulated. A prototyping tool is expected to support communication between users, designers and developers; as well as to reduce a designer's need to change his work practice when using the tool.

Keywords:
paper prototyping, prototyping tool, web design, scenario-based design, interaction design, sketch-and-scan.

Acknowledgement

I would like to express my sincere thanks to Mattias Arvola who has been very patient and supportive in advising me and giving pointers to useful materials throughout this project. I gratefully acknowledge the contribution of several researchers and practitioners who participated in the usability testing and discussed ideas for further improvement: Thomas Abrahamsson, Per Sökjer, Mikael Kindborg, Magnus Ingmarsson, Stefan Holmlid, and Martin Wiman.

My co-students in Communication and Interactivity programme were the ones whom I showed my very first prototype, and who have given me positive suggestions: Liu Ke, Chen Bi, Zhang Pei, Kejing Zhang, and Jean-Sebastien Susset.

I am also grateful to STINT foundation that has supported my master studies at Linköping University.

My family and friends, wherever you are, thank you for your kind support and friendship.

Irene Anggreeni
Linköping, 4 May 2006

Table of Contents

1 Introduction

Within application development, there is an everlasting dilemma faced by the development team. Early usability evaluation is good for receiving feedbacks before too much effort is employed. However, designers cannot evaluate the design until it is built. Moreover, changes are often difficult to perform after the building. The solution is to simulate the design in a low-cost manner gaining a balance between cost and ability to find usability problems. Paper prototyping is a technique to show the users how the application should work by using paper sketches. Snyder (2003) explained the use of paper prototyping to achieve the maximum feedbacks with minimum effort.

Paper prototyping, to some extend, is successful to make paper sketches interactive. In usability evaluation, it is not about displaying how the product will look like, but rather actually showing how it should work (Spool, 2005). The key role in paper prototyping is a person acting "computer" who manipulates the paper version of the interfaces. This technique definitely has limitations, mainly related with its nature of not employing computer. In theory, when the design is no longer subject to major changes – usually in later stages, high-fidelity prototype is built to represent the look-and-feel and behavior that resemble the eventual product. In reality, designers might build hi-fi prototypes early – to cope with deadlines, and then patch up the prototype with more functionality in later stages.

There are limited number of tools for design purpose. Current interactive UI construction tools give more attention to widgets and graphical details. The prototype produced can show how the interface will look like in high fidelity with the final product, but not so much about what it will do. In most case, the designer is required to program or script to specify the interactions, meaning more investment is expected to learn using the tool (Newman, Lin, Hong, and Landay, 2003). Furthermore, formal representation of product may distract attention from "the big picture" into small details like fonts, colors, etc, which makes low-fidelity prototyping preferable in the early stages. This is the reason why DENIM[1] project proposed a prototyping tool that uses *sketching* to create *informal representation* of the web design.

There is a gap between low- and high-fidelity prototyping. A question to answer is; what do designers do to the papers after the sketching process

[1] http://dub.washington.edu/denim/

is done? Maintaining a pile of paper sketches is not interesting with a deadline coming soon. A more probable thing is building high-fidelity prototypes from scratch. By mentioning "from scratch", I intend to express that it is a one-to-one translation from paper to computer screen, and thus it is not an integrated process.

1.1 Delimitation

Instead of taking a broad scope, I examine the prototyping process of website design in particular. Website design is far simpler than the designs of other applications. However, the finding from this research is extendible and to some extent, applicable towards the design of prototyping tool for more general purposes.

1.2 The Problem

Most prototyping tools seem to have less interest in transferring the existing paper sketches into their systems, even though most people feel that drawing and/or writing on paper is more natural and practical. With paper, there is less expected than doing a similar thing on computer. On the other hand, when we draw a sketch on computer, we may need a stylus and an electric pad, a particular software and adequate knowledge of the software.

A common case is that a web designer has many ideas drawn on paper sketches. Is it necessary to recreate his designs using a specific tool if he wishes to conduct usability testing? Eventually, he will build hi-fidelity prototype when the design is mature enough. But in the early stages, what a designer has are merely raw ideas, which are not stabile and can change anytime. Paper prototyping is useful, but it might grow tedious along with the number of testing sessions planned. A transitional integration from paper sketches to computer prototypes could address some issues, namely how to make paper sketches interactive with less effort.

The project aims to gain understanding of how the designers perceive the existing prototyping concepts and which parts of them that they find useful or need changes. To keep the focus throughout the work, two problem statements have been formulated.

- *What is the designers' attitude towards paper prototyping?*
- *How can a computer-based prototyping tool be designed to support integration with existing paper sketches?*

The thesis will investigate a new design of prototyping tool that involve sketches, specifically the ones that are used in paper prototyping.

1.3 Thesis Overview

Review of Prototyping and Prototypes explains about prototyping and related issues, such as the problems, the practice, along with some analysis.

In *Survey of Existing Tools* analysis of some systems which can be relevant as the basis to develop a new design of prototyping tool. The crucial part is finding what is good and relevant to the concept of prototyping.

Method describes the steps of gathering ideas, designing a usable prototyping tool, and evaluating the design. After following the "recipe" in *Method*, *Result* gives detail about the findings that are actually the core of the research.

Finally, as a complement to the work, *Discussion* is composed to support the *Conclusion*.

2 Review of Prototyping and Prototypes

2.1 What is Prototyping?

According to Lantz (1986), "*Software prototyping is an information system development methodology based on building and using a model of a system for designing, implementing, testing and installing the system.*"

From an informal source Wikipedia[2], "*Prototyping is the process of quickly putting together a working model (a **prototype**) in order to test various aspects of the design, illustrate ideas or features and gather early user feedback.*"

Prototyping is a process, with which we may estimate efforts for the project and thus reducing project risk and cost. Prototyping is building a model of a system. The model later on will be the vehicle for designing the final version of the system. (Lantz, 1986)

2.2 What is a Prototype?

The term "prototype" is ambiguous, and it leads into a gap of interpretation within a project team consisting of different roles. A programmer considers a piece of software as a prototype. While a user studies expert may call a storyboard, which shows a scenario of something being used, a prototype. It leads to a situation where organizations develop their own "prototyping cultures", causing them to consider only certain kinds of prototypes to be valid (Schrage, 1996; in Houde and Hill, 1997).

What is significant about prototypes is not the media or tools used to create them, but *how they are used by a designer* to explore or demonstrate some aspects of the future artifact (Houde and Hill, 1997). A prototype should allow stakeholders to interact with the future product, to gain experience and imaginable uses of it in a realistic context. In a straightforward manner; Preece, Rogers, and Sharp (2002) define a prototype as "*a limited representation of a design that allows users to interact with it and to explore its suitability."*

A prototype is often talked about by their attributes; such as which tool was used to create it ("Director" and "paper" prototypes), how finished-looking or -behaving it is ("high-fidelity" and "low-fidelity" prototypes). The term "fidelity" means closeness to the final design. The

[2] http://en.wikipedia.org/wiki/Prototyping

characterization, though, may not always indicate the level of finish. The degree of visual and behavioral refinement of a prototype does not necessarily correspond to the solidity of the design, or to a particular stage in the process. (Houde and Hill, 1997)

Hong, Li, Lin, and Landay (2001) use another term derived from the formality of the prototypes. A *formal representation* refers to *high-fidelity prototype*, whereas *informal* to *low-fidelity*.

2.3 Paper Prototyping

A designer cannot evaluate his design until it is built, but changes are often difficult to perform after the building. Therefore, a creative way to simulate the design in a low-cost manner is promoted. A common technique is paper prototyping.

Snyder (2003) describes paper prototyping as "a variation of usability testing where representative users perform realistic tasks by interacting with **a paper version of the interface** that is manipulated by **a person "playing computer"**, who doesn't explain how the interface is intended to work."

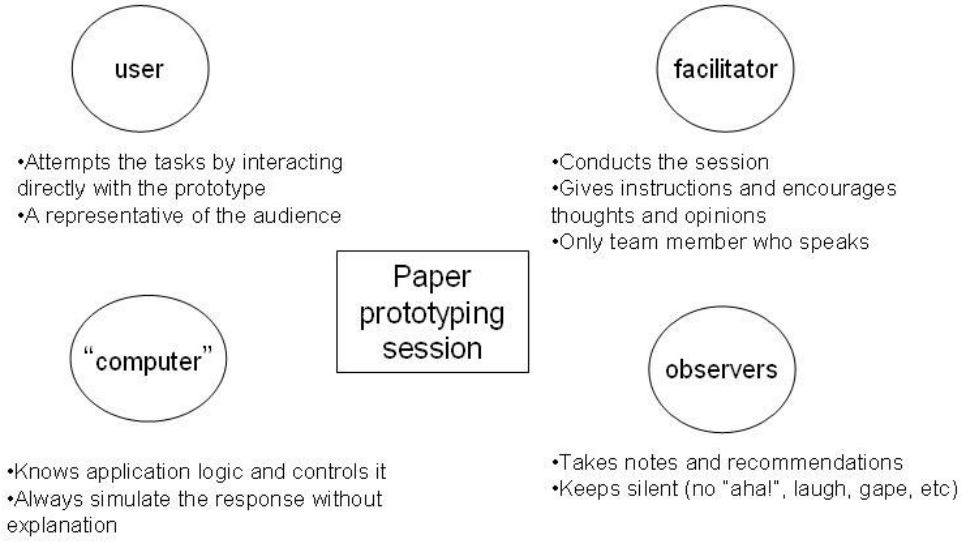

Figure 1 A paper prototyping session

There are many terms to refer other design artifacts. Snyder (2003) describes some of them to avoid misunderstanding between paper prototyping and other techniques. The artifacts below are not paper prototyping.

- *Compositions* are visual representations – usually of a web site – that show the look of the interface, including the graphical details.

Some use nonsense words to represent text and links, and primarily used in internal discussions of a site's visual design.

- *Wireframe* defines the page layout and navigation of a Web site, showing which content goes where.
- *Storyboard* is a flowchart; it is a series of drawings showing how the interface should be used to accomplish a task.

However, all of them can be used in paper prototyping as long as they contain realistic content to support a task scenario. The realistic content is needed to represent the effect of interactions (between user and "computer") within paper prototyping. (Snyder, 2003)

In comparison with technology that tends to change rapidly, paper prototyping will never be obsolete as long as people still draw and write on paper. Still, there are problems which arise along with the employment of paper prototyping.

2.4 The Problems with Paper Prototyping

What are the problems with paper prototyping? Among all other problems, it is worth mentioning the first that *paper prototyping is slow*. It takes time to prepare (drawing, cutting papers, anticipating all possible responses from the system) and to perform (human "computer" who handles a bunch of paper is not comparable with real computer).

Paper prototyping *cannot capture the realistic performance* of the designed system. For example, input and output methods are different (writing in paper prototyping, rather than typing). Another issue is the unrealistic time response due to human acting "computer". It also has different ergonomics from the eventual system. As an example, the user is most likely to use his finger to "click" rather than using a real mouse as in non-touch screen desktop application. To some extend, paper prototyping *cannot simulate certain interactions*, i.e. drag and drop, highlighting, typing, dropdown menu, etc.

There are other alternatives to gain balance between prototypes and cost. Looking into the real practice of web design can give ideas to propose an alternative in prototyping.

2.5 The Practice of Web Design

Newman et al (2003) conducted a field research during the winter of 1998-1999. The result verified a design process that the practitioners in web design follow. A generalized design process has *discovery, design exploration, design refinement*, and *production* phase.

1. *Discovery* is intended to determine and clarify the scope of the project and the requirements of users.
2. In *design exploration* phase, possible solutions to the problems identified in the discovery phase are generated and explored. Such solution can be in the form of information design, navigation design, or rough graphic design.
3. To continue to *design refinement* phase, it is required that a design idea be selected prior to the refinement. The design is iteratively refined and detailed.
4. *Production* starts when the design has reached a satisfactory level of detail -- or in a worse case when the deadlines and budget dictate so. Designers prepare the design for hand-off to the implementers (a client or a software development team). The hand-off may include interactive prototypes, written descriptions, guidelines, and specifications.

The findings about designers' practice from the study conducted by Newman et al (2003), particularly in relevance with the thesis, are summarized in brief. The texts in italic contain reflection towards the project.

Every designer sketches at least once on paper.
In practice, some designers draw all the diagrams on paper and switch to computer only for writing documentation using a wording tool. The sketching activity is common in the early stage, where design exploration phase takes place.
From this point, the situation becomes a good ground to conduct a research on a prototyping tool that supports integration with existing paper sketches.

Paper sketches are eventually abandoned.
Later on in the development stages, the sketches on paper are converted into electronic form by recreating from scratch. Some applications to accomplish the task are Adobe Illustrator or Photoshop. In almost all cases, paper would be abandoned once the designer had converted it into an electronic format.
Paper sketches alone are static, and require verbal and gesture explanation as a prototype. A particular technique (paper prototyping) needs to be employed to show how the design works. Furthermore, one cannot stay with paper throughout the whole project. A tool that can build interaction on top of static paper sketches might reduce the conversion, so that the process does not have to start from scratch.

Paper sketches are considered not appropriate to be shown to clients.
Adapting some comments from interviewees from the study, sketches on paper are considered of "poor quality" and "really rough". Some designers were reluctant to show their sketches. Everything presented to clients must look professional, which meant at a minimum a high-resolution mock-up, color printed or on a computer. Early deadlines also force designers to switch from paper sketching to electronic media earlier in the design process.
Paper sketches are the expression of personal thought from the designer. In exploring design ideas, a designer makes annotations and corrections on paper sketches. Designers feels the need to "clean up" at one point. Usually, the conversion to electronic form answers the need.

Learning a new tool is compromised.
It is worth mentioning that in practice the designers have tendency to stick to only one tool they know best for doing everything, even if the tool has limitation to other tasks. A subject within the research admitted that she uses Microsoft Word for reporting, drawing layout, and diagrams - including site maps and schematics. It seems like the promise of potential gain from learning a tool that is made specific for a task outweighs the inconvenience of having to learn another tool.
With tight deadlines in a project, a designer is less keen on learning new software, because it takes time and effort to get familiar with it. It is a clear requirement that whatever tools used in web design must be easy-to-use.

Designers annotate printouts.
Some practitioners use paper sketches to communicate ideas with others and gain valuable feedbacks early. Even when the electronic version is already built, a group of designers may pass around the printed version to colleagues for comments.
Writing and drawing on paper is natural and quicker. It is preferable by designers that the activity is prolonged within the design process.

2.6 Comparing Low- and High-Fidelity Prototypes

Hong et al (2001) ran an experiment to verify that the comments generated by end-users evaluating a web site design in an informal, sketched representation will be of greater value to designers and lead to a higher quality end product than a formal, cleaned-up representation. They held a task-based usability evaluation of two web site designs in both formal and informal representation in the same medium – computer. DENIM was used to develop the informal representation of the web site. The result did not confirm the expectations; moreover it confronted the

hypothesis. Majority of suggestions for the formal representations were structural and navigational, whereas for the informal representations they were visual. A conclusion was drawn that conveying informal representations using an electronic medium raises end-users' expectations of the design specifically the details at the formal level, despite the explanation that the designs were at the early stages. (Hong et al, 2001)

A related later research conducted by Walker, Takayama, and Landay (2002) confirmed the finding from a previous research by Virzi, Sokolov, and Karis (1996); that there was a significant degree of overlap between the usability problems found using a low-fidelity (i.e., paper) prototype and a working prototype. The study by Walker et al (2002) also proved that low- and high- fidelity prototypes are equally good uncovering usability issues and that despite the difference in interaction style, the findings of usability testing is independent of medium – computer or paper. The findings strengthened the idea of using low-fidelity prototypes for design and testing. A good prototyping technique is one that finds the maximum number of real usability problems during user testing, while being both inexpensive and flexible for designers. (Walker et al, 2002)

2.7 Further Issues about Prototyping Tools

Traditional UI design tools focus on creation of high-fidelity prototypes. When we observe Microsoft Frontpage, Adobe GoLive, Macromedia Dreamweaver or NetObject Fusion, all of them are concerned more about page layouting rather than web site architecture. The tools produce *high-fidelity* prototypes, and ***they are not appropriate in every situation***, for example in the early stages of design. Some of the tools are well tailored to serve a particular task. Macromedia Director is specific for multimedia authoring and useful to create storyboards. While Visio, as a general-purpose diagramming tool, can work well in creating high-level information architecture. (Newman et al, 2003)

In June 2002, GUUUI[3] conducted a survey on web prototyping tools usage (Olsen, 2002). The purpose of the survey was to find out what tools are used for prototyping, what requirements interaction designers have for their tools, and how happy they are with the tools they are using. To make it concise, I only present the result which is relevant with this thesis, namely about the preference of tools for prototyping.

[3] GUUUI is a site for people engaged in the various fields of making the web a better experience for the users. The site contains weekly postings and quarterly articles about interaction design, information architecture, usability, visual design and the like.

Visual or text based HTML tool (e.g. Frontpage, Dreamweaver or Homesite) was the most used tool for web design, while non-computerized tool including paper prototyping, use of whiteboards, Post-It notes, or overheads was less preferred. The use of presentation tool such as Microsoft PowerPoint fell in a slightly higher position than the use of non-computerized tool. A reason for the popularity of HTML tool is related with tight deadlines that encourage designers to get a jump on the eventual product (a website).

In today practice, sketching is seen as the alternative. It supports ambiguous ideas and focus on the basic structural issues by leaving unimportant details. An example of the contrary, when creating a canvas in such a tool like Adobe Photoshop the designer is confronted to decide the canvas size, the background color, the resolution before he even draws anything on the canvas. Rapid prototyping technique should allow the designers to explore ideas freely without committing too much time and effort. Sketching encourages more iterations. While traditional methods take too long, i.e. *sketches* > *prototype* > *evaluate* > *iterate*, the use of low-fidelity prototyping can simulate *sketches* > *evaluate* > *iterate* (Landay, 1999). In the latter, the sketches immediately play as prototypes, with the help of a human who plays as "computer". This method has limitations, as discussed previously, which I intend to address in the thesis.

3 Survey of Existing Tools

Early design practice employs informal representation to gain valuable structural and non low-level-detail feedbacks. Paper, along with other physical tools such as whiteboard, Post-it notes, supports informality more than computer tools. Paper is an integral part of early design process. For informal tools to realize the maximum potential, they must provide similar benefits. (Cook and Bailey, 2005)

As a complement to the research conducted to find out the current web design practice (Newman et al, 2003), a recent research focused on how and why designers utilize physical tools in early design practice has been conducted. The research aimed to seek recommendation on how informal tool could better accommodate the needs. Paper support some common design tasks better, whereas computer tools might cause difficulties to designers. Nevertheless, besides the positive characteristic of using paper (quick, easy, direct, portable, and sociable) the designers desired the benefits of computer tools as well (design execution, access to design history, and remote collaboration). (Cook and Bailey, 2005)

3.1 Integration Mechanism

Cook and Bailey (2005) analyzed three mechanisms for connecting physical (paper) and informal tools and argued that digitalized ink interface is the most effective to complement the electronic interface of an existing informal tool. The discussion about each mechanism is summarized and the pros and cons are also explained below.

Sketch and scan interface allows the use of paper the way it is, and after scanned, some behavior can be added to the electronic interface, making it possible to execute the design and share with remote clients. It could grow tedious along with the size of the design and it is difficult to extract the semantics because the stroke sequence is not preserved.

A ***tangible interface*** extends the functionality of sketch and scan interface with integrated components, such as digitizing pad, stylus, or electronic display. Sketching on paper on digitizing pad can acquire both physical ink stroke and electronic stroke to the informal tool. Although it is real-time and can be quickly analyzed, the mechanism does not support portability and requires specific structure for interacting with the paper.

In a ***digital ink interface***, designers can sketch on paper and meanwhile having the stroke information stored in the pen itself. This is made possible by using a tablet or a digital pen, such as Anoto's (Anoto, 2004).

The digital pen works as a normal pen, but once placed in its cradle, it would upload the stroke information to the informal tool.

3.2 Related Work

The following discussion covers existing design tools in a variety of domains, focusing on their main purpose and what features they have to support the needs of designers in the domain.

3.2.1 SILK and DENIM[4]

Designers, in any domain, most likely sketch initial design ideas on paper. To complement the shortfall of paper, some computerized design tools have been developed. An electronic sketching tool interprets a designer's ink stroke and extracts a semantic representation out of it. The semantic representation acts as the input and as the base on which rules and computation can be applied. (Cook and Bailey, 2005)

SILK is an interactive tool that allows designers to sketch an interface quickly using an electronic pad and stylus. SILK tries to recognize the widgets as they are drawn by the designer, yet it is unintrusive. The behavior lies among the interface elements in the sketch, and also sequencing between the screens in a storyboard. Much of it may need to be specified by the visual language developed in SILK. (Landay and Myers, 1995a)

The visual language consists of *screens* and *arrows,* and is considered similar to the kinds of notations that would be made on a whiteboard or a paper when designing an interface. Each screen represents an interface in a particular state. Arrows connect objects contained in one screen with a second screen, so that when the object in the first screen is clicked, the second screen will be displayed. This static representation is natural and easy to use, unlike the use of scripting by other systems, such as HyperCard. *Programming-by-demonstration* (Cypher, 1993) was also considered for specifying the behavior, but it lacks of a static representation that can be understood and edited by the users – interface designers. (Landay and Myers, 1995b)

[4] http://dub.washington.edu/projects/denim/

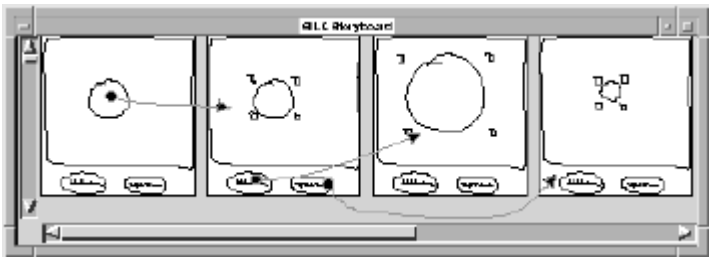

Figure 2 Illustration of a sequence, in which the user can select a circle by clicking it. The circle's size can be increased or decreased by clicking on the buttons at the bottom of the screen.

DENIM is an outgrowth of SILK, but unlike SILK which is a general user interface design tool, DENIM is specific to support early web interface design. Therefore in some ways, DENIM is similar to SILK, especially in the way of using gestures and specifying behavior. DENIM uses a zoomable canvas to integrate viewing of commonly used representations when designing Web sites, i.e. site map, storyboard, and individual page.

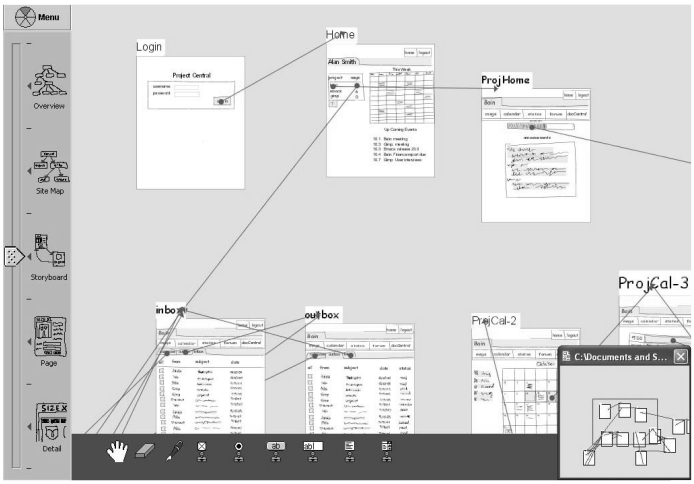

Figure 3 Storyboard view. Multiple pages and the links between them are visible. It is possible to indicate the links within a page to reach another page. Creating new pages combines both ways in "site map" and "page" view, either by writing a label or typing text, or by drawing a rectangle in the size and the shape of a page.

Gestures and pie menus have been used widely to execute commands. Other functions, which are not easily mapped to gestures, are accessible via pie menus (Callahan, Hopkins, Weiser, and Shneiderman, 1988). Pie menus also provide redundant access to certain commands.

To represent a relation between pages, DENIM provides *navigational* and *organizational* arrows. Navigational arrows represent hyperlinks in HTML, while organizational arrows represent a conceptual relationship between two pages but without details on how it can be done. The designer can use organizational arrows when he wants to skip filling the details at this time.

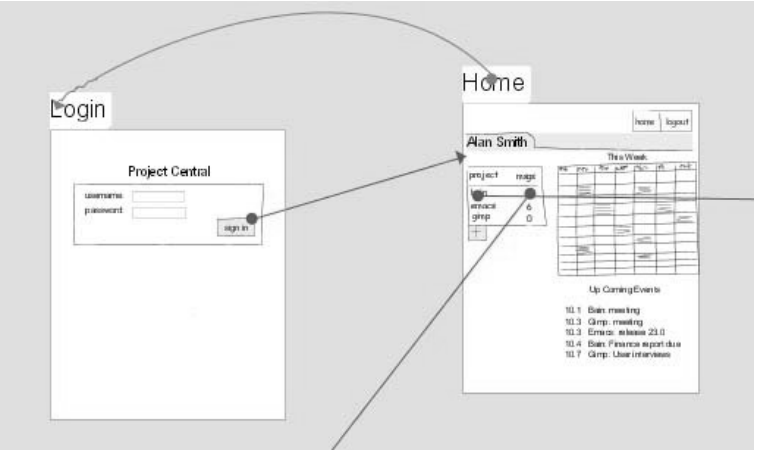

Figure 4 Arrows. Organizational arrows, in gray color, originate from a page label or an empty region of the page indicate that there is a relationship between two pages but there is no specification on how to navigate from one to the other. While navigational arrows, in green color, originate from a word or drawing inside a page and specify which link in the source page should be click to reach the target page.

In run mode, a simple "browser" window appears on the screen and acts like a real Web browser. With it, designers can experience the interaction and gain feedbacks without having to create a full-fledged prototype. DENIM also allows designers to export the designs to HTML, which results in one HTML page for each individual page, containing one GIF image and image maps to render areas for navigating to other pages.

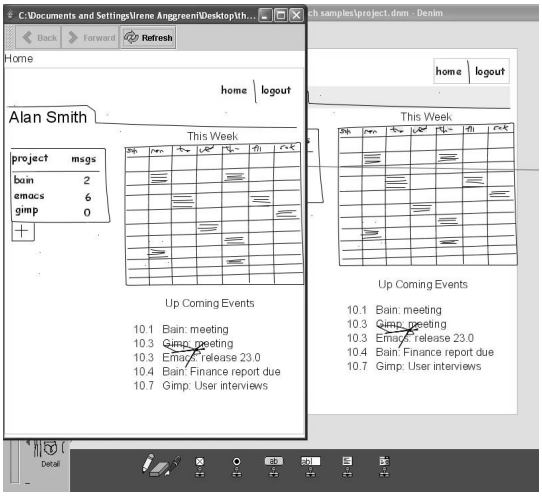

Figure 5 Run mode. A simple browser enables users to interact with the "sketchy" designs. The links are shown in blue color. They can be clicked to open another page within the same simplified DENIM browser.

3.2.2 CrossWeaver[5]

CrossWeaver enables designers prototype multimodal, multidevice user interfaces. The project set forth with the concern that the end-user's tool of the future will not only be a solitary PC, but as a diverse set of devices, ranging from laptops to PDA's to tablet computers. It implies that the input mode would vary from keyboard and mouse to pen or speech input. Informal design tools need to support the interface for the interaction.

The sketched designs in CrossWeaver form an executable multimodal storyboard. Transitions between scenes are represented by icons of different user input modes along with their respectable modifiers. The storyboarding technique is inspired by those used in SILK and DENIM. CrossWeaver allows multiple types of media as the input to build each scene.

In creating reusable operations, each designer typically makes sketches a sequence representing an operation, and declares that the sequence applies to many scenes. This process is called programming by illustration. It tries to match the visual language used by the designers and not to make much inference to the users. The process is made possible when there is enough information to specify the operation in the sketches, the sequencing and the designer's annotations. (Sinha and Landay, 2003)

[5] http://guir.berkeley.edu/projects/crossweaver/

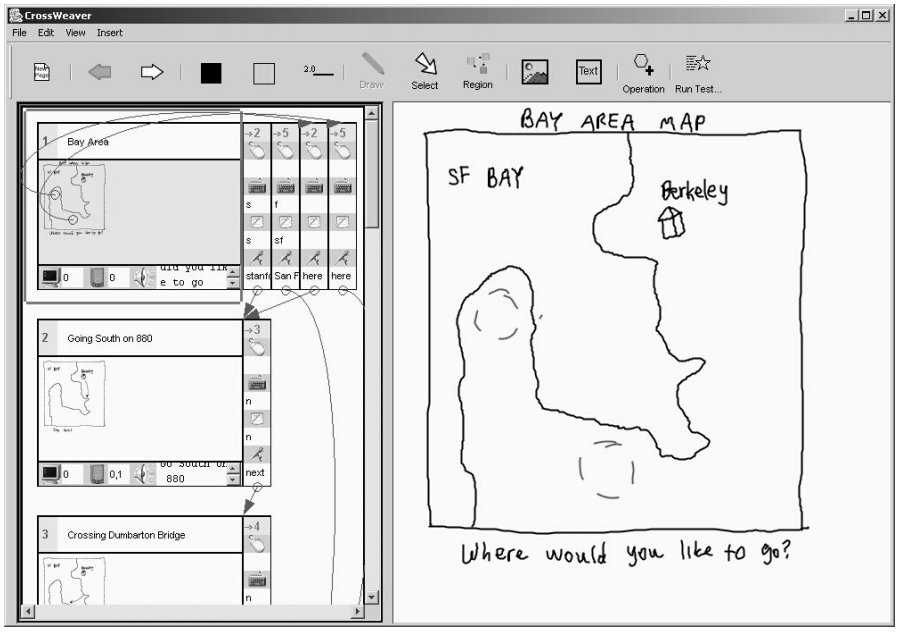

Figure 6 Overview of CrossWeaver. The left pane contains scenes with the interaction modes and selection of output target, while the right pane serves as a drawing area for the selected scene.

3.2.3 Damask[6]

Damask is a similar project to CrossWeaver, a tool to support the early-stage design of user interfaces targeted at multiple devices such as PCs, PDAs, and cell phones. To design a user interface for one device, the designer sketches the design and then specifies which *design patterns* the interface uses. The patterns will help Damask generate user interfaces optimized for the other devices targeted by the designer. In case of common patterns, designers do not need to sketch everything from scratch, but instead can make use of the patterns built in Damask. Reusability is applied on the patterns, on which the designer can create their own design patterns and share them with other designers. (Lin and Landay, 2002)

[6] http://guir.berkeley.edu/projects/damask/

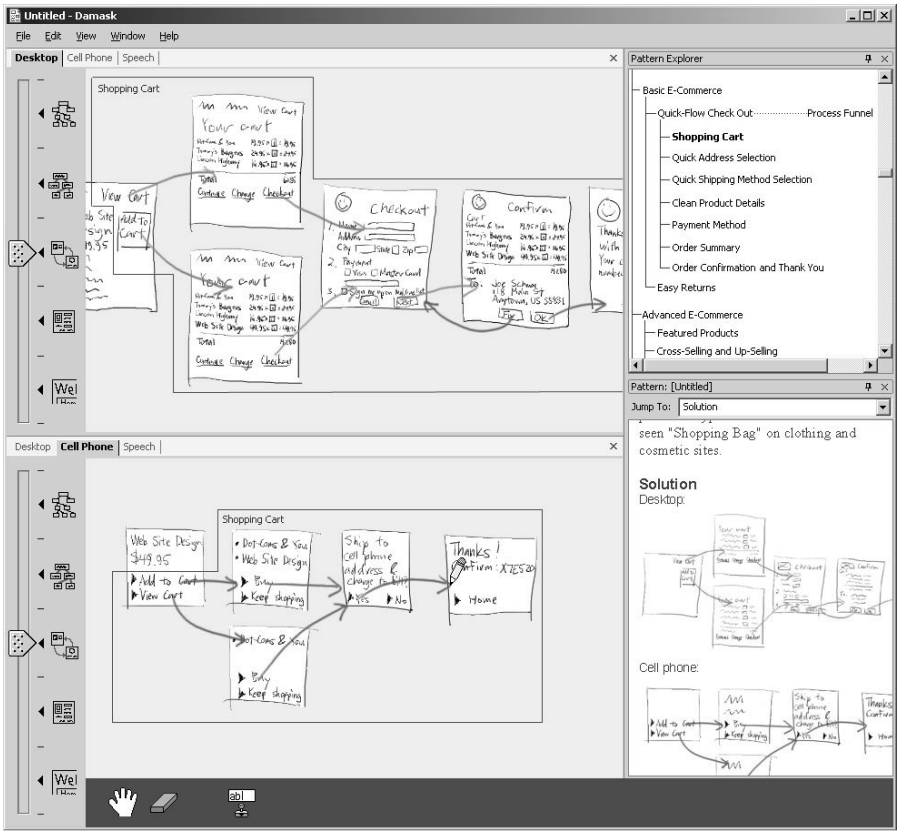

Figure 7 Shopping Cart pattern as targeted in different output devices. The split canvas shows both patterns for Shopping Cart module in desktop and cell phone. The right pane loads the built-in *pattern explorer* displayed in hierarchical tree.

3.2.4 DEMAIS[7]

Rising from the lack of computer tools and techniques to support early behavior exploration and to communicate behavioral design ideas, Bailey, Konstan, and Carlis (2001) conducted a survey on multimedia design process. The research resulted in a sketch-based, interactive multimedia design storyboard tool, which employs a designer's ink strokes and annotations as an input for the design.

DEMAIS is a multimedia design tool which helps multimedia designers to explore and communicate their design ideas. Enabling collaboration with other designers, DEMAIS supports incremental sketching and annotation. The interface consists of storyboard, voice script, and multi-view editors. An ink stroke that connects rectangular strokes, text objects, or imported media is interpreted as a behavioral stroke. A source event and destination action of the behavior is represented by icons near the ends of the stroke. (Bailey et al, 2001)

[7] http://orchid.cs.uiuc.edu/projects/demais/

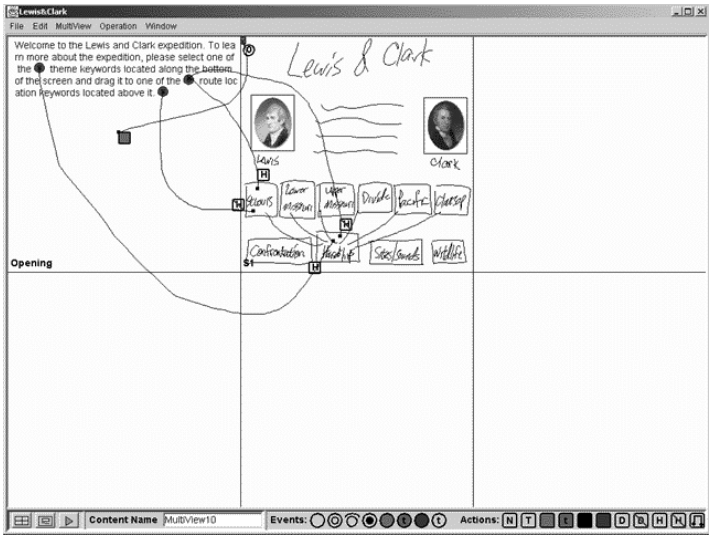

Figure 8 A multi-view editor showing the behavioral strokes. The narration text contains synchronization markers which highlight objects in the next pane.

3.2.5 PatchWork

Rapid prototyping requires fast implementation of screen mockups. A popular technique is paper prototyping, which is described as similar as a slide show with a personnel act as "computer" to give back "system responses". Either computer based or paper-and-pencil based works well. (Kant, Wilson, Bekker, Johnson, and Johnson, 1998)

PatchWork, designed by Kant et al (1998) is a prototyping tool, complying with the concept of slide show. Central to Patchwork is patches, the easily modifiable and rough looking building blocks. Patches are simply bitmaps and can be in any shape. It is possible to import sketches into patches by means of digital photography.

The realization is a simple hypergraphics mechanism. Clickable slides map into a state transition diagram representing the application concept. There is no complex design, because PatchWork is intended only as basic means for mask design, such as writing, basic "paint" functionality, and bitmap import. (Kant et al, 1998)

3.2.6 ComiKit

ComiKit is not a tool for prototyping, but it is included here since it has a number of interesting features and qualities. It is actually a toolkit for children that uses comic strips to program the behavior of graphical characters. The system aims to enable children to author interactive media and to create their own interactive games and toys, rather than being passive consumers of ready-made game software. As a frame of

reference of ComiKit, Kindborg (2003) has chosen Stagecast Creator (Smith, Cypher, and Tesler, 2000), ToonTalk (Kahn, 1996), and Squeak eToys. All of the chosen tools support programming with animated characters and represent many years of research. Both Stagecast Creator and ToonTalk approach on making programming easier using *programming by demonstration* or *programming by example* (Lieberman (2001).

The difficulty with programming is that there is a mental gap between program representation (the source code) and the resulting runtime representation. When program representation and runtime representation use different kind of signs, the programmer will have to bridge the mental gap himself. If the gap is too wide, the relation between two representations grows to be not understandable (Smith et al, 2000, p. 77; in Kindborg, 2003). The gap is the source of all bugs and programmers' difficulties while working. Thus, making programming easier is a task of reducing the mental gap, by using source codes that look similar to what is seen on the runtime display (Smith et al, 2000).

Kindborg (2003) explained that a comic, just like a program, is a static representation of something dynamic. A static representation is straightforward to edit, which is essential to programming, and independent of the real-time flow of a dynamic representation. The medium of comics gives a direct impression of the action going on in the story, as its nature of visual representation.

In ComiKit, a program is created by drawing pictures of characters and making even strips for their actions. The tool has a play area, where the runtime result of a program can be seen, and an editor for drawing pictures and programming comic strips events. Graphical objects depict figurative characters and items. Each object can have one or many pictures to represent different state of it. Each may also have one or more comic strip events attached to it. The program is run by putting the characters on a play area, and causing events to trigger.

A new notion, conditional strips, was introduced to accommodate events in a program. Such strips describe non-linear and potentially concurrent events, rather than the sequential flow of a story in comics. Two subsequent panels represent the preconditions and actions. The precondition panel has an arrow shaped to help communicate the status. The preconditions implemented are picture matching, touching another object, keyboard pressed, time interval, and random time interval. Meanwhile, the actions currently implemented are changing the picture of

an object, moving an object, delete an object, and creating a new object instance. Some examples are given in Figure 9 and Figure 10. (Kindborg & McGee, 2005)

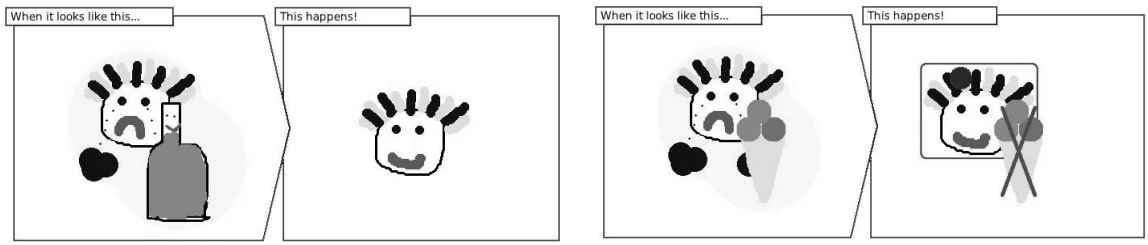

Figure 9 Conditional event strips. In the first event strip, when the sad character touches the medicine, his picture is changed into a happy one. In the second strip, the actions of ice cream deletion and changing of picture occur when the sad character touches the ice cream.

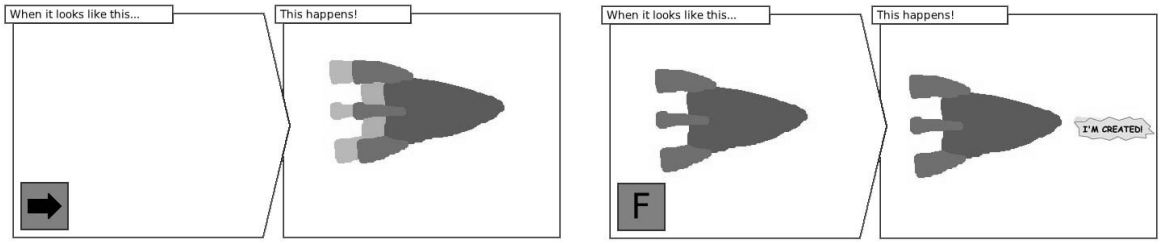

Figure 10 Conditional event strips. Both strips show precondition of keyboard pressing. When the right arrow key is pressed, the spaceship moves to the right. To express the action clearly, a ghost image is shown as the trail. The second strip shows that when the key "F(ire)" is pressed, the spaceship produces a yellow laser beam.

3.2.7 What else?

Lantz, Artman, and Ramberg (2005) conducted an interview study and a series of workshops in Sweden to find out the practice of interaction designers. Most designers need to create order from the chaos of ideas and thoughts, and to get them externally represented by sketching on paper. The initial sketches are normally kept personal, while it is always the electronic version to be shown to others. A result from the survey is that Microsoft PowerPoint is widely used to make a digitalized version of the sketches, accompanying scenarios and personas, to represent ideas. The main reason is PowerPoint supports templates, which makes it easier to make copies. (Lantz et al, 2005)

From a similar research conducted by the counterpart in USA (Landay and Myers, 1995a), HyperCard and Macromedia's Director are said to be the most popular prototyping tools used by designers. However, these tools fall short either in the early design stages or for producing final product interface.

HyperCard employs the metaphor *card-stack,* and the programming is based on sequencing different cards. Cards hold data and use layout engine similar in concept to a "form" in most Rapid Application Development environment, such as Delphi or Visual Basic. The background of a stack refers to elements that appear constantly on all cards within that stack. Hypercard, like other traditional user interface builders, requires too much design detail and often it must be extended with a scripting language, HyperTalk. Due to its poor performance, in some cases the Hypercard prototype is discarded after testing. This is more likely to happen when the applications require complex computation, portability to different environment, or efficient performance. (Schrage, 1996)

Director excels in combining video, animation, audio, pictures, and text. Together with its scripting language, Lingo, Director is powerful for developing a final product for specialized applications, such as a multimedia information kiosk. The metaphor of Director is that of a stage or of an animated film, and that the designer plays the role of the director. A Director application centered around a *score*, which represents a carefully coordinated and timed sequence of activities by a *cast of characters*. The basic programming structure is similar to the kind used in producing movie cartoons, which is frame-by-frame animation. (Landay and Myers, 1995a; Schrage, 1996)

No matter how powerful it is, Director is not appropriate as a general interface design tool. It lacks support for creating standard widgets and specifying their behavior directly. Flash is related to Director and it includes standard widget libraries. However for both of them, the scripting language is not easy to understand or to learn by non-programmers. Visual Basic also has the same drawback, even though it is becoming popular as a prototyping tool due to its rich widgets and third-party supports. (Landay and Myers, 1995a)

Particular to web site prototyping, the survey from GUUUI showed that Visual or text based HTML tool, such as FrontPage, Dreamweaver or HomeSite, was the most widely used, while almost all respondents using a diagramming tool depend on Visio. The next popular tools are graphic

design tool, such as Photoshop or Illustrator, and presentation software, such as PowerPoint.

Most designers in the survey were not content by the tool they were using. The major problem with HTML tools is concerning site wide revision. While diagramming tool did not score high on any criteria, it was considered particularly poor at supporting site wide revisions and simulating advanced functionality. Consistent to the finding by Lantz et al (2005), presentation tool (PowerPoint) users found that their tools are very easy to use and suitable for presentation and setting up pages.

3.3 Further Issues

Almost all of the tools discussed in this chapter use additional hardware, such as PC tablet or stylus pen. It is sensible that some tools support gestures and pie menus because frequent users can benefit – in term of learning how to use the tool efficiently – by getting used to the motion of gestures and pie menus selection. Considering the advances of recognition techniques, we are no longer just hopeful for a pen which can record its own stroke, such as Anoto's pen (Anoto, 2004), to fully accomplish tasks in a digital ink interface.

In the other hand, adding hardware adds an extra layer in the user's interaction. An evaluation on DENIM found that it was fairly easy to learn and understand, but not particularly easy to use. It is believed that some of the blame lies with the hardware. Users who were unfamiliar with the hand printing on tablet felt awkward in using DENIM. When the users are required to make a lot of adjustment using the tool, instead of the other way around, the tool would not likely to be successful.

All of the tools are supported with thorough background research, and also evaluations on the developed tools. Taken for an example, DENIM has been going on numerous improvements due to positive users reactions. In the evaluation, the Web designers being interviewed ranked the usefulness fairly high (9.0 out of 10) indicating that even with the existence of some shortcomings on the implementation used in the evaluation, they felt that the basic concepts were on target.

Sketchy appearance is still considerably not presentable to clients. It excels in communicating the ideas with other designers, but lacks the ability to produce "cleaned-up" version of sketches. Looking at DENIM, all users gave it relatively low marks in terms of ability to communicate with client, which correlates with its inability to produce sketches nice enough to present to clients.

A vote from the companion website for the book "Interaction Design" by Preece et al (2002)[8] surprisingly showed that designers (and users) who responded still prefer paper-based prototypes to prototypes created in DENIM. I must mention that the vote consists of only two questions and a small number of responses, and therefore we cannot draw a conclusion from it; but it does give a hint on what designers and users prefer.

"Is it easier to produce an initial storyboard using a system like denim than using paper?"
Yes (34.00%) 17 votes No (66.00%) 33 votes

"Did users prefer to evaluate a paper-based prototype rather than one produced in DENIM?"
Yes (77.27%) 17 votes No (22.73%) 5 votes

ComiKit opens a new dimension for using comic strips in programming. The visual language of comics offers the potential of being expressive and flexible to represent event-based visual programs that directly maps to the runtime representation. It is a strong example of *programming by demonstration*. Particularly in comparison with *graphical rewrite rules* or *before-after rules* (Smith et al, 2000), in comics there is no restriction that the before part must be the same size and location as the after part and no limitation of having to have objects confined in grid world (Kindborg & McGee, 2005). Other informal tools which support sketching also need to be unlimited in space and free of grid world. Bailey et al (2001) coined the term *programming by illustration* which is an interesting counterpart of programming by demonstration. Unfortunately there is no other literature discussing about programming by illustration in details by the time of this writing.

3.4 Approaches to Programming for Designers as End-users

Expert end users sometimes desire to create their own custom commands, and get away from repetitive activities. "End user programming" tries to achieve effects with less programming to the end user. The approaches can be categorized into *Preferences, Scripting Languages, Macro Recorders,* and *Programming by Demonstration (PBD).* (Cypher, 1993)

Preferences are pre-defined alternatives supplied by the application designer to accommodate the varying needs of end users. End user can get the application work appropriate to his working style. Since the

[8] http://www.id-book.com/interactive_denim.htm. Retrieved on 4 May 2006.

application designer cannot foresee the real situation of end users, preferences are general and of fixed alternatives. They are too specific to be considered a kind of programming.

A ***scripting language*** is simple programming language whose vocabulary is adapted to the objects and actions of a particular application domain. While it is a popular approach to end-user programming, it has major drawbacks, as users still have to learn the syntax and vocabulary conventions of the language, and the standard concepts of variables, loops and conditionals in programming. The reward of learning them does not outweigh the effort from end users.

Macro recorders are a basic implementation of "Watch what I do". They record the user's actions literally. It is common to have built-in macro recorders in spreadsheet programs. The main drawback is that macro recorders are too rudimentary for complex operations. It takes the record of user's action sequentially and literally, not allowing any generalizations to be brought in.

Programming by Demonstration is an elaborate idea from macro recorders. The user instructs the system to "watch what I do"; but the system, not just records the whole sequence, it creates a generalized program. Programming by Demonstration is "Programming in the User Interface" (Halbert, 1993). There is no mental gap between the source code of program and the run time result, because the users are programming in the same environment in which they perform the actions.

Some PBD systems take their inspiration from macro recorders and extend the approach to work with high-level events and also to generalize actions to create programs. The approach rise from the ease of use of macro recorders, where even end users do not realize that they are creating programs. Another approach, inspired by the teaching metaphor, is more explicit about the fact that the systems are creating program and making generalizations. The end user is put on the role as a teacher, encouraging him to provide clear and informative examples for the system. (Cypher, 1993)

A closely related subject is visual programming language (VPL). It exploits icons to construct a visual language, with which the user can avoid scripting while achieving the effects he wants. Technically speaking, a visual programming language is specified by a triple (*ID,G*0*,B*), where *ID* is the icon dictionary, *G*0 is a grammar, and *B* is a

domain-specific knowledge base. (Tortora, 1990; in Boshernitsan & Downes, 2004)

I mentioned that ComiKit is a good example of PBD. In general, *before-after rules* or *graphical-rewrite rules* serves a good basic for programming by demonstration. The visual programming language in game authoring tools, such ComiKit, StageCast Creator, AgenSheet, or Squeak eToys, provides a clear hint for users on how to proceed. The users are getting familiar with the rules creation approach because the rules are defined in the same run-time environment, allowing no gap to occur.

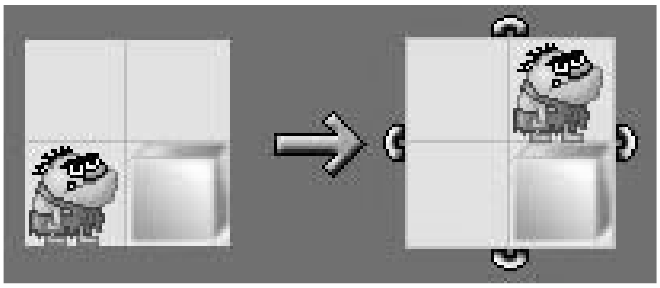

Figure 11 An example of graphical-rewrite rule in StageCast Creator.

4 Method

The issue of prototyping culture has attracted many researchers to contribute to the field. Many previous studies have successfully gathered valuable information regarding the practice of prototyping. They took forms as ethnographic study (Newman et al, 2003), interview study and workshops (Lantz et al, 2005), and survey (Olsen, 2002). The previous researches contribute a rich insightful background into this project, by giving ideas about web design practice, which is valuable in this project.

As the result of the studies, many alternatives of prototyping tool have been developed. The similar characteristics of the tools are the support of sketching, which results in informal appearance of the prototypes. Most of the tools are designed to be operated with extra hardware, namely stylus, PC table, or a special pen (i.e. Anoto's pen). The previous chapter analyzed and briefly discussed these tools.

The existing tools do not accommodate the use of paper as it is; instead they add a new layer for interaction. The use of hardware makes designers conform to the specific behavior of the hardware. There is a tendency to make things become less simple and the needs for anticipation grow. Paper prototyping, as the simplest of all prototyping technique, is also evaluated. Along with the practice of sketching on paper, I consider to "fill in the gap" of prototyping by creating a tool that makes use of paper sketches.

4.1 Personas and Context Scenarios

Based on the literature research and survey of existing tools, *personas* are created to represent the goals and behaviors to the prototyping domain. According to Cooper & Reimann (2003), *personas* are design tools which are based on research and represented as individuals, while at the same time they also represent a class of users in context. The motivations of the potential users (i.e. real web designers) are captured within personas. Later on, *context scenarios* are developed to express goals and activities of personas in narrative story. Context scenarios represent goal-directed product instead of system behaviors. The scenarios translate goals into design. (Cooper & Reimann, 2003)

The following is an example of the context scenarios which have been prepared to elicit user requirements. The complete scenarios can be found in Appendix.

Anna is a freelance web designer. She has educational background in art, and learns web design autodidact. Her main passion is traveling, and blending-in with the culture of the places she visits. Her last trip to Spain has earned her a project.

Right now, Anna is designing a website for a youth hostel which is located in southern Spain. The website, in addition to describing the hostel itself, also provides a lot of information on the culture, the events going on in Spain, traveling advices, suggestion for tours, etc. Conforming to the spirit of young travelers and their nature loving of pop culture, she pays attention to the artistic aspect of the site and experiments with icons and symbols to express ideas.

……

While working in the design, Anna always draws on paper, creating sketches on the icons look and the layout within the website. She thinks it is more practical to work on the raw ideas on paper, and later on she will refine the appearance of the design using a graphical design tool. The main reason is that she feels more at ease when drawing on paper, where she can express her potential artistic skill without interruption. This way, she can also work when she is traveling by train or airplane.

4.2 Defining Interaction Framework

While brainstorming ideas for defining interaction framework, one approach – among many- was to apply paper prototyping technique. I followed up the approach because of the promising simplicity and easiness to understand.

A construction of paper prototyping involves *drawing* (on papers), *cutting* (the papers to fit it in), and later on *placing the layers* on a proper place during the test with users. Paper prototyping utilizes a main activity, i.e. *moving objects* (taking objects away and putting objects somewhere). For example, a mouse-over on an image displays a description of the image below it. A "computer" will put a piece of paper containing the description when the user points the image; and take the piece of paper away when the user points somewhere else.

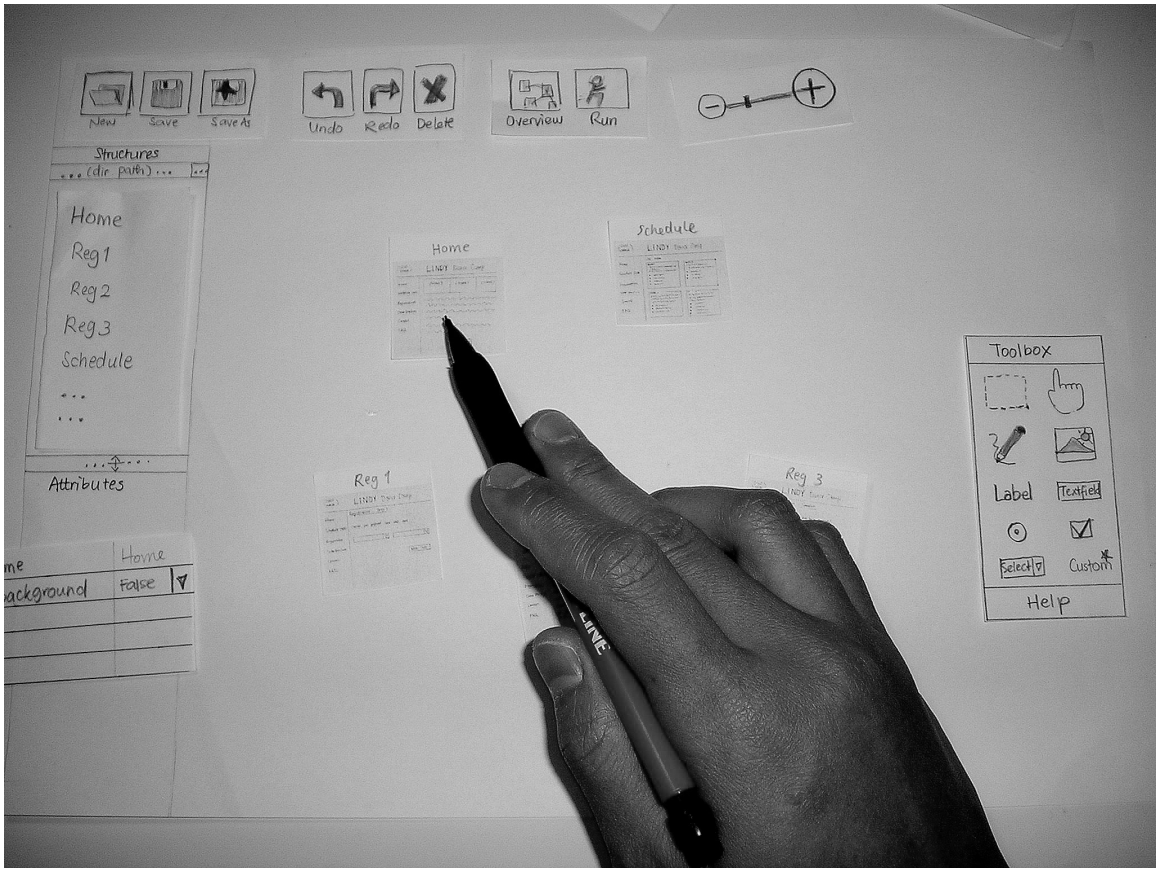

Figure 12 A snapshot of paper prototyping. The user interacts by simulating mouse operations with hand gesture/tap, while another person playing "computer" moving pieces of paper to simulate the response.

The result of ***mapping out the functionality in paper prototyping*** is surprising to see how simple things can be. The tool could have been build with fewer components; but I decided to implement full support for standard widgets for the web. Most computer users are already familiar with the standard widgets. Breaking them up into more simple pieces may cause confusions to the users, because then they will have to think in a different metaphor.

At this point, I started drawing ***sketches of the interaction framework***, iterating them with other "experts" in this research. We also brainstormed the likely interactions in web site to be anticipated. I created a key path scenario to depict the interaction within our envisioned tool. Key path scenario is more task-oriented, unlike the goal-oriented context scenarios; they focus on task details broadly described and hinted at in the context scenarios (Kuutti, 1995; in Cooper & Reimann, 2003). A key path scenario used for usability testing can be found in the Appendix.

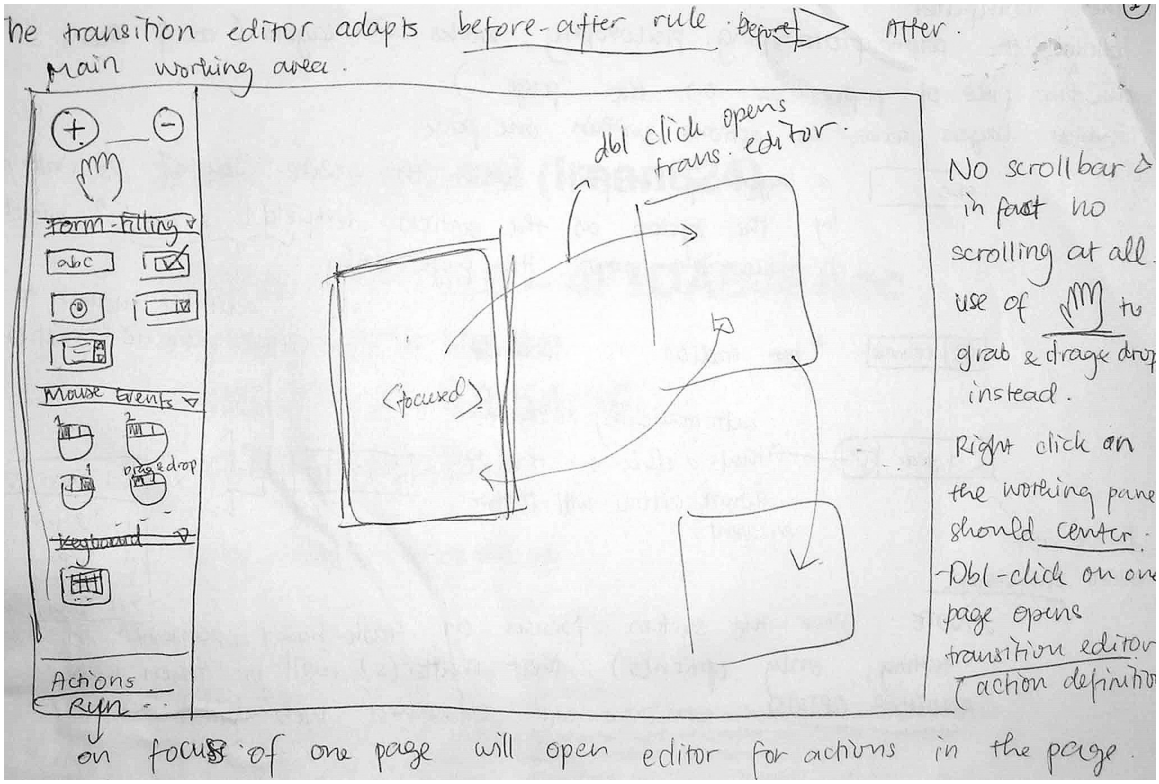

Figure 13 A rough design of *overview*, annotated with envisioned functionality.

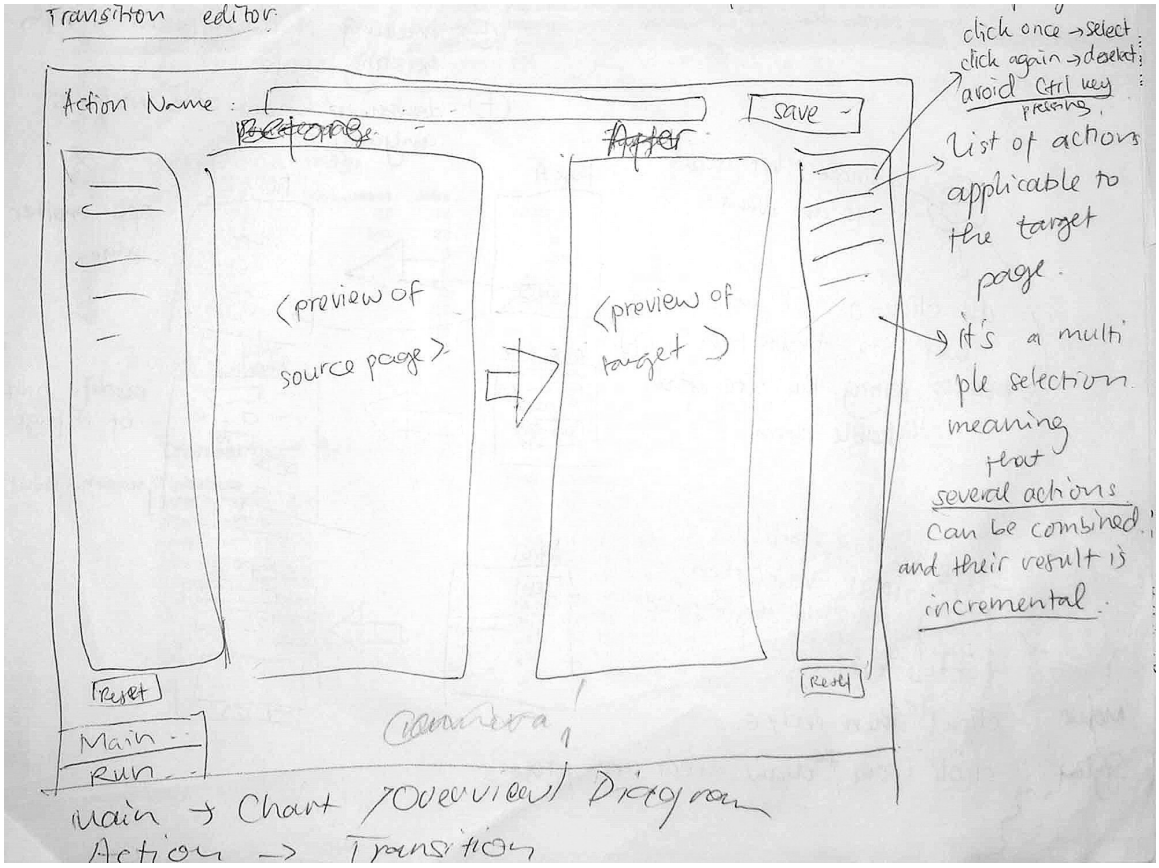

Figure 14 An early sketch of transition editor. Notice that the initial idea of transition was based on action name, an action triggers several actions to be executed. Because the approach limits the interaction, this idea was not implemented.

We constantly reviewed the design as the experts in this study. After the refinement of interaction developed into a design with which we are confident, I continued with the creation of a high-fidelity prototype.

4.3 Refining the Form and Behavior

After defining the interaction framework, it is necessary to build a high-fidelity prototype with realization of look-and-feel and behavior. I chose Java Swing to develop the prototype, mainly because I am already familiar with the programming language. Another reason is that there are many open-source Java projects from which I can utilize in my project. I adapted the graphical module (drawing, coloring, moving objects) from an open-source project, DrawSWF[9]. It results in a shorter period of prototype development. Otherwise, I could have spent a few months alone to learn and write codes for graphical functions which I am not familiar with.

[9] http://drawswf.sourceforge.net/

I visualized the three kinds of views for the tool, namely *overview*, *page detail*, and *transition editor*. In *overview*, a user perceives the view of all pages he is currently working with. The pages are represented as small thumbnails. The user can move the position of the pages so that he can put aside pages he will be working on later, or create such arrangement that he can easily recall the location of a page. From overview, the user can zoom in to see the pages closer.

Page detail displays a single page in the screen, enabling the user to add components and define areas – which are the source of interaction. The interaction between and within the page(s) is created in *transition editor*. A transition contains the triggering event on a source object which causes an action to be executed on an object.

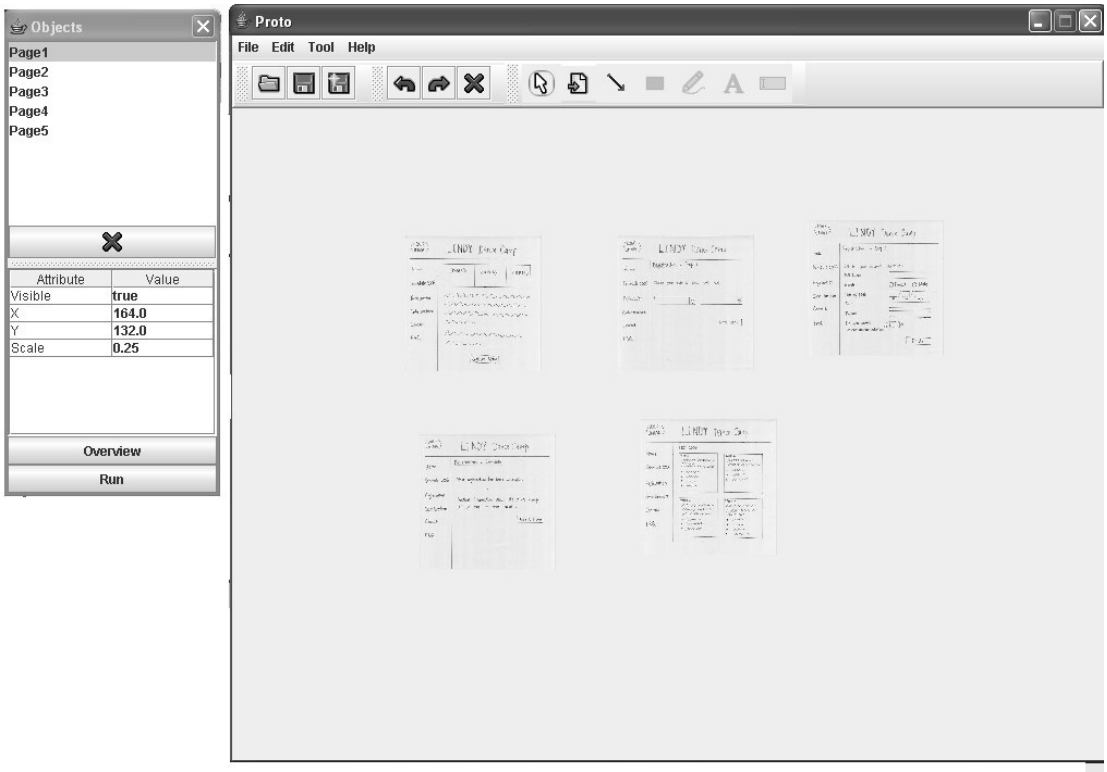

Figure 15 Overview. The earlier prototypes have implemented parts of the eventual design. Pages are represented as thumbnails, and the object structure and attributes are displayed on the left pane. Object naming and other small details (such as nametags on thumbnails, moveable toolbox, etc) were not implemented at this stage.

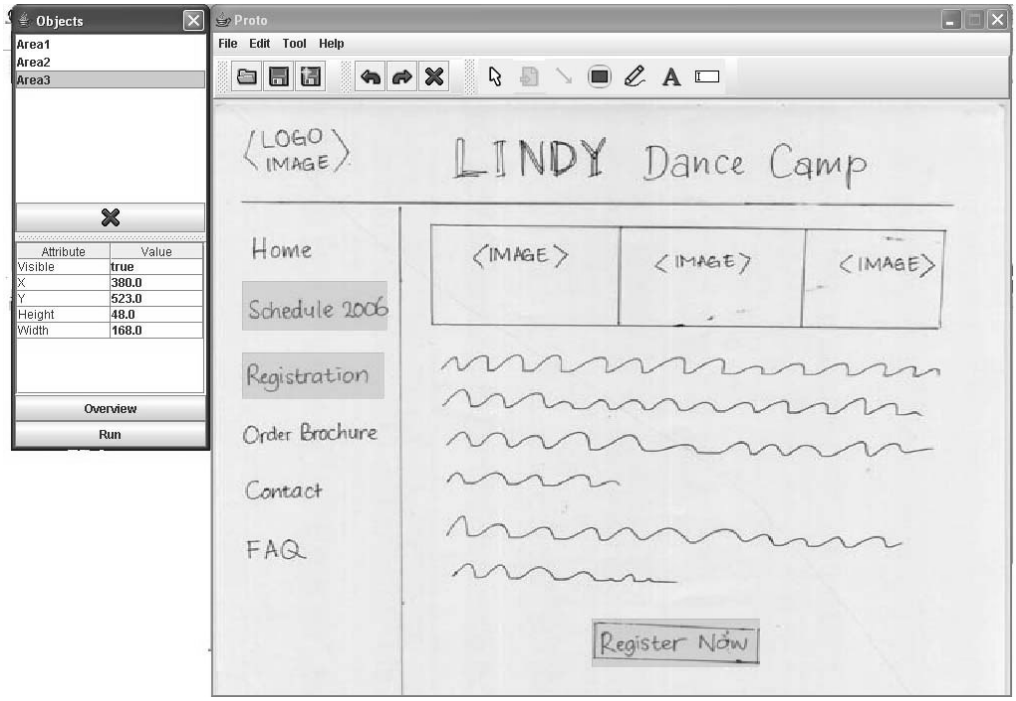

Figure 16 Page detail shows a page in full size. Three areas, which serve as base of hyperlink, are listed on the left pane in a flat non-tree structure. The active icons on the toolbar can be used to draw objects on the page.

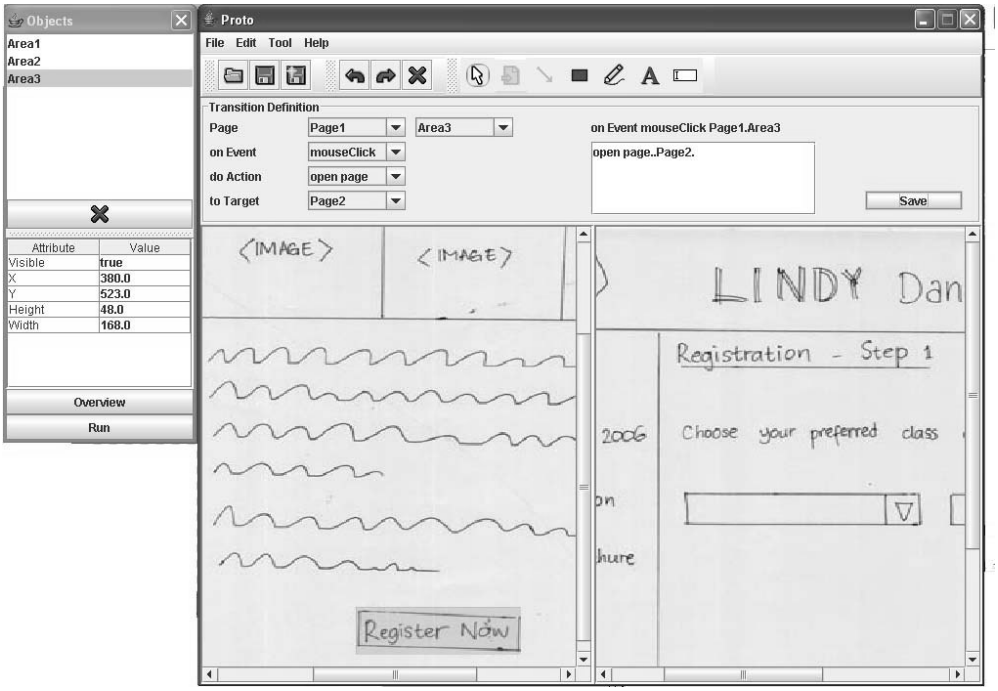

Figure 17 Transition editor is opened when the user double-clicked an area in page detail. The transition definition is structured into requiring user to select available values for the source event and the target operation.

4.4 Usability Testing

The usability testing involved several human-computer interaction students and researchers. The main idea is not on evaluating the usability score of the prototype; but to figure out – from interview, discussion, and task-based scenario – how to improve the prototyping tool to fit in with their practice of profession. To describe usability testing; some potential real users were asked to accomplish several tasks using the system being tested. There are many variations of usability testing depending on the budget, time of the project, but an essential part is that the users talk about the tasks as they work through them (how difficult or easy they are), which will be the base to detect usability problems. (Starling, 2002)

4.4.1 Iteration 1

I conducted the testing with my co-students in the master programme of *Communication and Interactivity*[10]. There were five participants, each of whom has taken usability and human-computer interaction modules, and has been involved in interaction design project within the related courses. The participants in iteration 1 are generally good at programming, but they have little experience of using prototyping tool. Nevertheless, they have all applied paper prototyping during the course project, so that they already have their own independent opinion about it.

A version of the early hi-fi prototype was shown to the users in informal sessions. The users were asked to try out the prototype after being shown paper prototyping technique. Think aloud was essential as I valued any early feedback at this iteration. I did expect numerous feedbacks on the interface of the tool. Based on the feedbacks, I refined the interface and continued to the second iteration.

4.4.2 Iteration 2

Within this iteration, I conducted the usability testing with researchers and practitioners in human-computer interaction. In total, there were six participants, two from the industry, two from academia, and the last two have both industry and academia background. They all have mature experience in interaction design. They have used physical surface (paper, whiteboard, etc) to communicate design ideas with users or colleagues, either independently or as a designer in a development team.

A testing session was structured as an interview combined with task based scenario. It was much easier to practice think aloud because the format was already an interview. The interview procedure is as followed.

[10] http://www.liu.se/en/education/study/master/?id=97

1. Get basic information about the test participant
2. Introduction about the project
3. Show context scenarios
   There might be feedback already after the showing of context scenario, which was of course gladly invited.
4. Demonstrate the prototype *(and receive feedback)*
5. Task based scenario with think aloud
6. Questionnaire regarding the usability of the prototype of the tool
7. Discussion about the project in its relevance with prototyping
   Within the discussion, some ideas for re-design and the future work of the prototyping tool were raised.

All written materials used in the usability testing can be found in Appendix.

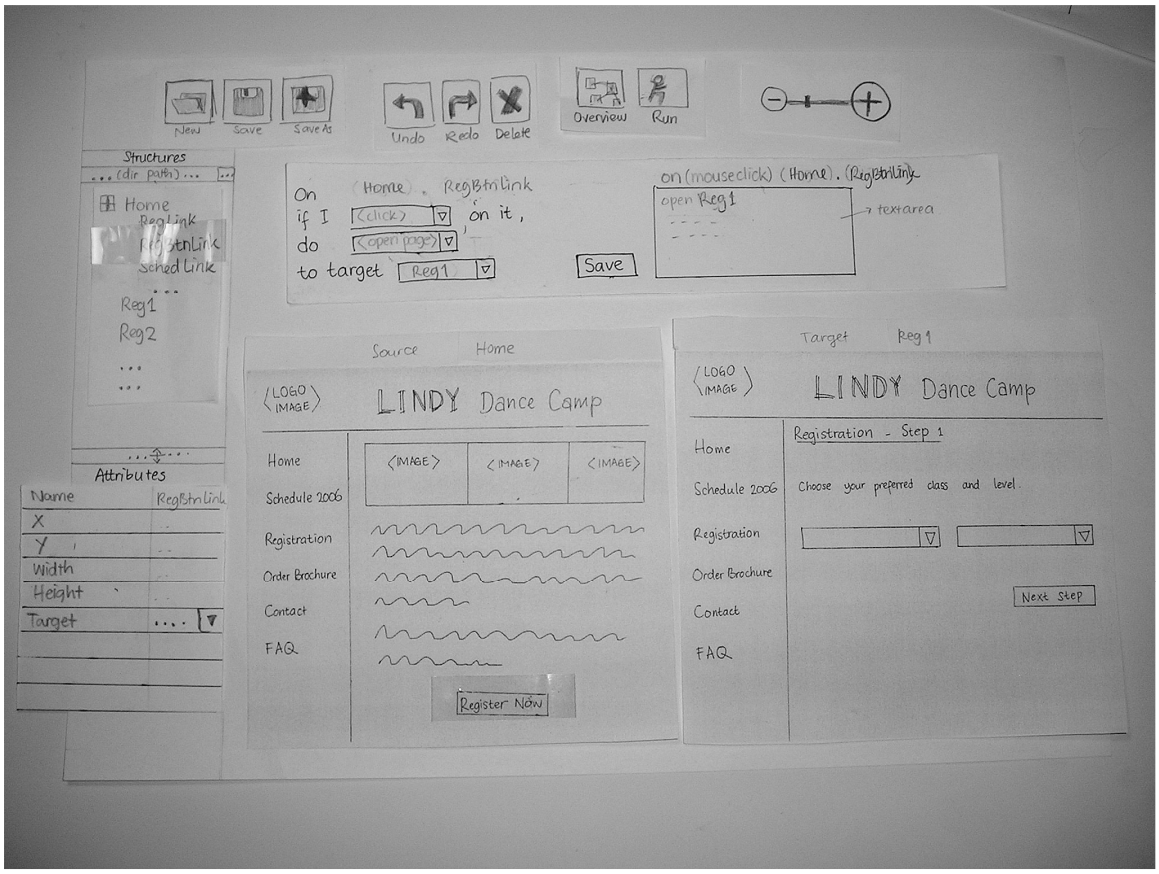

Figure 18 Paper prototyping was used during usability testing. The figure shows a transition creation, where both source and target of the transition are shown side by side, like in before-after rules (graphical rewrite rules).

I redesigned the prototyping tool based on the usability problems found and suggestions from the participants. I present the design ideas in the following chapter for further improvement to the prototype.

5 Result

The usability testing was a significant step to formulate the result of the project. As what I have developed is merely a prototype, I did expect the testers to have problems during the test. It turned out that the hi-fi prototype in Java was not acceptable by the testing participants, so that I switched to paper prototyping in the middle of the usability testing. From the problems found and the suggestions, I summarize some important findings and ideas for redesign.

5.1 The Practice

Some designers might sketch very little, or even not at all. It subjects to the background and the responsibility of the designer. As an example, a designer with programming skill and tight deadline would skip sketching on paper and "jump straight" into the codes, even though he wishes to sketch more. An participant shared his experience conducting participatory design on a whiteboard with a team of designers and developers. Later on, the documentation is written in a report by wording tool based on the design on the whiteboard. No paper sketches were involved in this situation.

On the other hand, some testers do sketch on papers a lot. This case happens to testers who happen to work individually or not in a big team. The paper sketches could have several levels of refinement and spread out in several parts. For example, the screen content is depicted on a separate paper with higher details than the general screen layout.

A participant has been meeting users at their home and discussing ideas for the design of product to be used in family environment. Later on, he would report the ideas to his affiliated company to be followed up with development of the product. He made the following remark about how he uses paper sketches:

> *"Many of my sketches work as prototypes. It depends a bit on how you define prototypes. I mean some of them work as sketches…..I have used my sketches together with the target group of end-users just to get their thoughts."*
> *"Do you explain to your clients verbally: if you click here, then you will have this page, and so on?"*
> *"Yes"*

Another participant who works in a more formal setting expressed a similar situation. He travels to his clients' workplace and conduct the

discussions there to elicit design requirements. He mentioned that he shows both paper sketches and computer screen shots, depending on the situation:

> *"The most when I use paper sketches is in the office. And then I make some screen shots if I want to show to users. But sometimes I show them paper sketches as well. It's good to show them paper sketches because they know that it's not ready."*

Meanwhile, a participant who works independently, i.e. not affiliated with a team, has also used paper sketches to sustain the design ideas for himself in a later time:

> *"I collect them [paper sketches]. ..and then I have them when I program."*

Regarding the feasibility of conducting paper prototyping, the testers also have different experience depending on the nature of their work. Some of them easily get by with paper prototyping, but others also mentioned about difficulties faced in using paper sketches with the clients.

A participant from the industry is responsible for the design of a large application. As the only information architect in the team, he has to design the interface and interaction in many parts of the product, and puts them together as a whole application. He compared his experience with users in using paper and computer prototypes:

> *"When you do it on paper, everybody agrees. But when you do it on screen, everybody starts to ask questions…..If you just want to discuss a small part of the interface, then it's easier to do it on paper. Because then they realize that you're just talking about the part that you draw. But if you put them on the computer, they will think that you're talking about the whole interface and they start commenting on other things, which they think should be there."*

Another participant who has been dealing with programming tools for children, addressed the problem of ergonomic difference when he used paper prototypes:

> *"It didn't work as expected on paper. I mean when kids tried out on paper they had difficulties understanding how it will work on the computer....it's difficult for them to imagine how*

it will work and understand it. But on the computer they get feedback immediately and they see it."

The same participant continued with the following remark:

> *"The computer kind of restricts what is possible. On paper, everything is possible, so I had to explain and so on and they don't really understand that. But this restriction is good because they can see what happens, what they can do and what they can't."*

A practice worth mentioning here is that an interaction designer within a development team could function as a "bridge" between users and developers. Interaction designers meet up with clients and design the application based on the requirements, then move on with creating specification for the developers. Some of the test participants do exactly this in their work. As an example, an information architect works with clients using either paper sketches or computer prototypes, and later on conducts a meeting with developers to present the functional specification using presentation tool.

5.2 The (Additional) Designer's Wish-List

Researchers and professionals in interaction design who were interviewed have expressed their expectations of a good prototyping tool. The expectations are based on the context and the environment they are working in. To some extends, we also discussed about the prototyping tools they currently use and remarked the problems and limitations with the tools. Below is a summary I make about the qualities designers would love to have in the prototyping tool.

Enable designers to work efficiently and totally against any tedious operations. The issue is related with avoidance of repetitive tasks. As an example, a designer may wish that his own created component can be used in different screens (pages) without having to copy and paste the code or making a lot of adjustment. The same goes for consistent part of web pages, let's say the navigational menu links in a website is commonly consistent throughout the pages. The interactions/behaviors of the menu links should be defined only once and be existed in all the pages that use menu links.

A "card and stack" metaphor is favorable, because it gives reusability of a background "card" (screen). However, it also has limitation that slightly different backgrounds must be implemented in different stacks. If

something must change within these slightly different backgrounds, then all backgrounds in the relevant stacks must be modified.

Upon discussion with one participant, another favorable solution is by providing *objects support* within the tool. A possibility to refer to an object can be more flexible and powerful. The designer can reuse his own created object component and apply changes to the object easily.

Integration with other tools (i.e. programming tools) which should support an easy –if not automatic- layouting. Designers, who also write programs, would love not to create layout from scratch again when moving to the real programming environment. It is highly preferable that the prototyping tool is capable of exporting the design into other formats relevant with other tools. For example, the pages, objects and interactions are exported into XML and later on be imported to another system. There are several open source projects which enables UI building using XML. Taken as an example is an open source project, Luxor XUL[11]. The long list of similar projects can be found in Java-source.net[12]. This idea is an envisioned future of the prototyping tool and is completely unimplemented in the current prototype.

The following is a remark from one participant who is working independently (both as designer and developer in his project) about the mental gap between the resulting interface and the programming codes:

> *"When I work with the sketches on paper and try implementing it, it's always different when I implement it. Things don't look the way I expected it and then I have to do lots of adjustment."*

A designer in a medium-sized company also realizes the amount of time that can be saved as the development team progress faster by integrating paper sketches and the interface on computer:

> *"That would be nice that something that I actually use could end up being translated into Swing components or whatever they [programmers] use because it is very time-consuming just to build interface."*

Support communication between designers and developers/programmers, in addition to with users. Designers

[11] http://luxor-xul.sourceforge.net/
[12] http://java-source.net/open-source/xml-user-interface-toolkits

experience tedious work of creating extensive specification which will be the base for developers to work on. Specification is not merely screenshots of how the interface looks like, but also a thorough explanation on how it behaves. If the designer can generate behavior specification of the interface by using the prototyping tool, it will be a great time saving and avoidance of misunderstanding between designers and developers.

A problem which is common in a software company, is shared by one participant. As the number of programmers is far bigger in comparison with the number of people who understand what the users need (and also the business logic), he experienced that his design is sometimes not implemented how it should have been:

> *"It's important to be able to communicate with programmers in an efficient way. If there's something that they [programmers] don't understand about my design, they will most likely try to solve the interface problem themselves without consulting me."*

Direct manipulation to the objects, extensive use of drag and drop mouse operation. This is regarded by the designers as the most convenient and quickest way of working with any tool. Not all operations can be done efficiently by direct manipulation, but keeping it in mind during redesign stage has a good point. Thinking that "the interface is magic" at this point is better than limited oneself into what can be done or what cannot. The features that cannot be implemented will at least contribute to the future work of this project.

Run a scenario to show the flow of the designed system. As an example, a designer wants to show how to perform a specific operation step-by-step to the users and the developers. Even though a user or a developer can figure out how the system works by trying out, still there is a risk of missing an important detail. A built scenario can demonstrate what the designer thinks is important to the other stakeholders.

5.3 User Interface and Interaction Framework

An interactive system is considered successful when it gives positive feelings to the users. The basic idea, referred to as *direct-manipulation* interface, is visibility of objects and actions of interests. A more concrete way to describe it, is: *rapid*, *reversible*, *incremental actions*; and *applying actions to object representation* (grab, hold, click, etc.) instead of typing commands. (Shneiderman & Plaisant, 2005).

Following the idea of direct manipulation, three principles can be summarized.

1. Continuous representation of objects and actions of interest using metaphors
2. Physical actions, instead of complex syntax
3. Effects of actions on the objects of interest are visible and reversible immediately

When appropriate direct-manipulation strategies cannot be designed, menus and forms can be good alternatives. Menus provide users with a choice that can be a choice of command or a choice of options related to a command. Grouping is essential when using menus.

Command language design is a method of interaction primarily used by expert users. It encompasses qualities such as efficiency and speed. Its minimized interface allows for quick interaction. However, for a novice user, this type of interface proves to be fruitless since he cannot possibly know the necessary commands. Hence, it would take an extended period of time to become familiar with the system.

The interface of the tool I put forward is a combination of the three interfaces. With preference over direct-manipulation, other types of interfaces serve to make certain specific operations more efficient. Direct-manipulation can grow tedious when there are many objects involved. Others can make a balance between easiness and efficiency.

Some screenshots were taken and described as seen in the figures below. The presentation starts with the components that make the system, followed by the overall of the system where the components are put altogether.

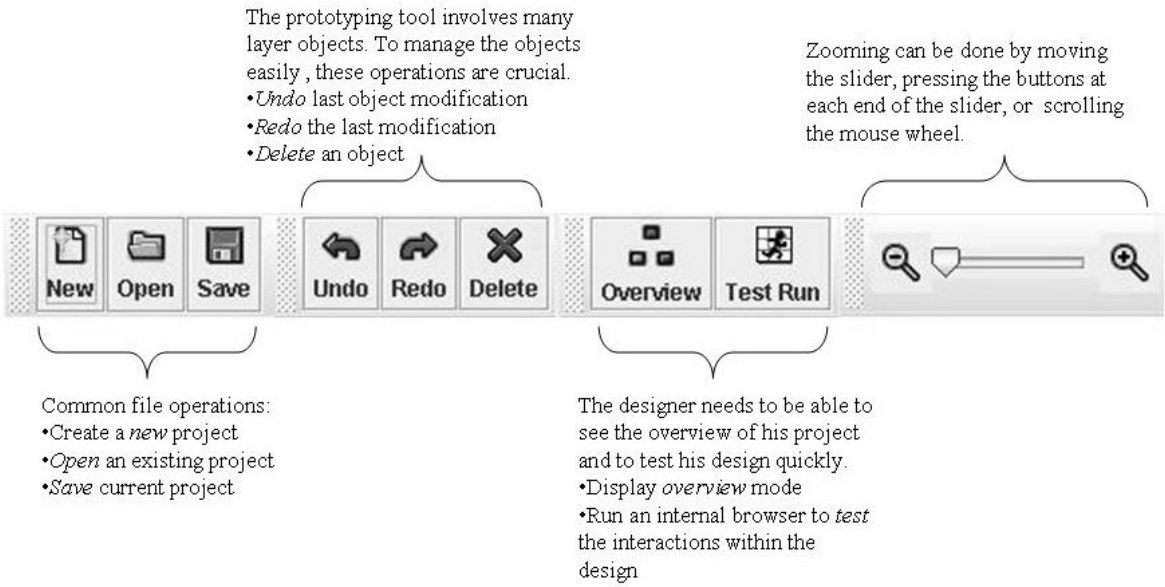

Figure 19 The most common operations are grouped on a toolbar. A shortcut to go to the overview is re-located to the main toolbar; prior it was in the tree structure but then the users had problem finding it. Another particular shortcut is "Test Run", which is most likely to be used often to test the design.

As the result of the usability testing, I relocated "Overview" and "Test Run" buttons on the main toolbar because they are important. Users tend to look on main toolbar first to find important functions. Before, they were at the left-bottom corner on a separate pane under the attributes panel. The new place is also strategic for mouse operation, as it is closer with the main pane and easily reachable. Textual description is provided on each tool, making it clear to the designer.

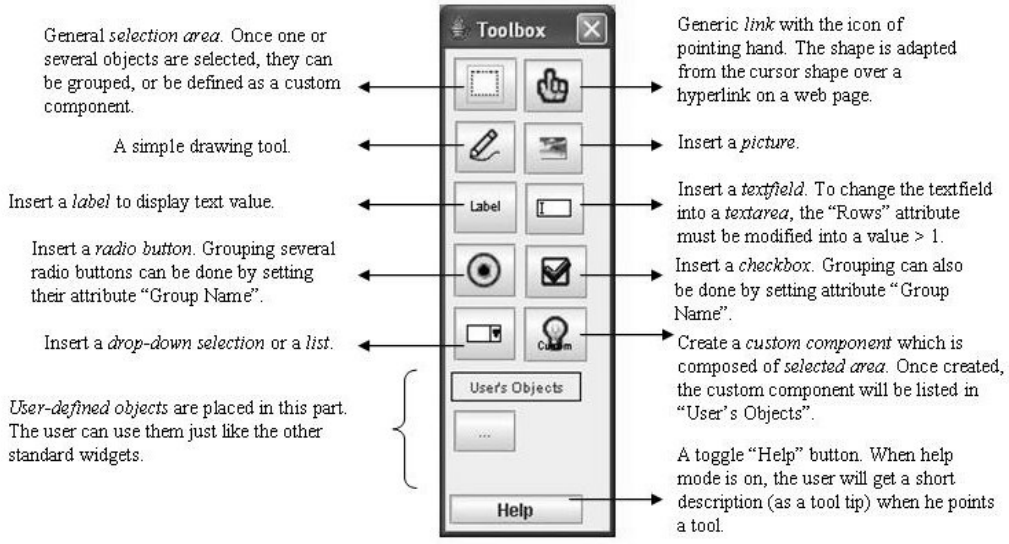

Figure 20 The toolbox contains standard layer objects and possibly some user-defined objects. Each tool is represented as a toggle button so that there is a clear feedback about which tool is currently active.

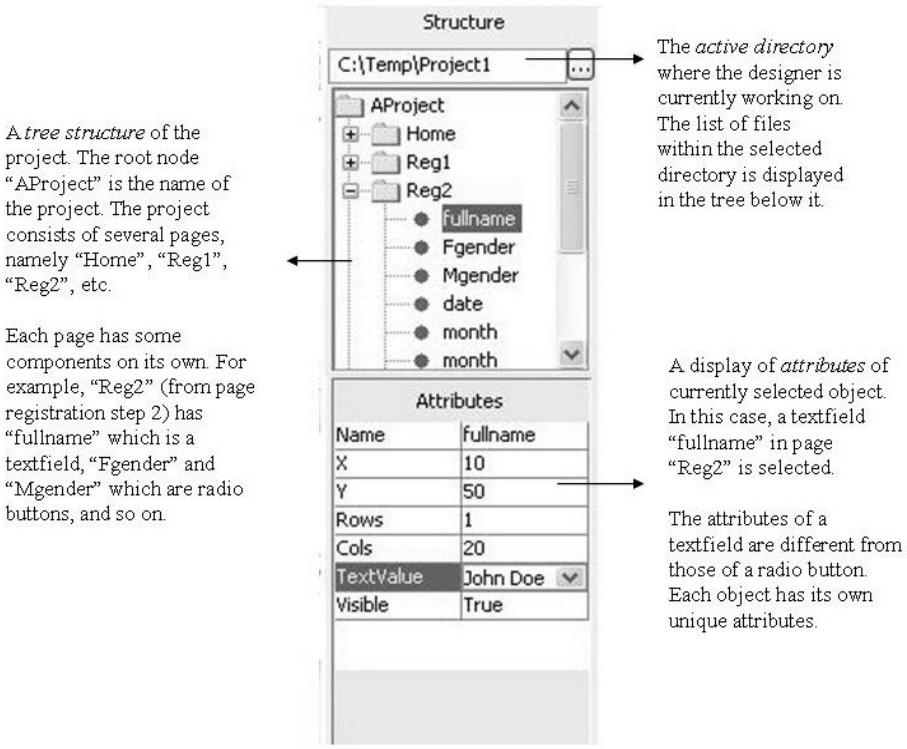

Figure 21 The tree structure displays the list of the pages along with their respective objects. Double-clicking on a page or an object should open up the appropriate mode in the main pane; selecting an object on the main pane also moves the focus of the tree structure into the right object. In this figure, the attributes are those of a textfield "fullname" in page "Reg2".

Each object has a set of attributes. The selected object's attributes are listed in the attribute pane below the tree structure. A complete list of attributes for object components can be found in Appendix 1.

To put them all together, the next figure is depicted to show a whole screenshot of overview. Overview mode is the base of interaction within the prototyping tool.

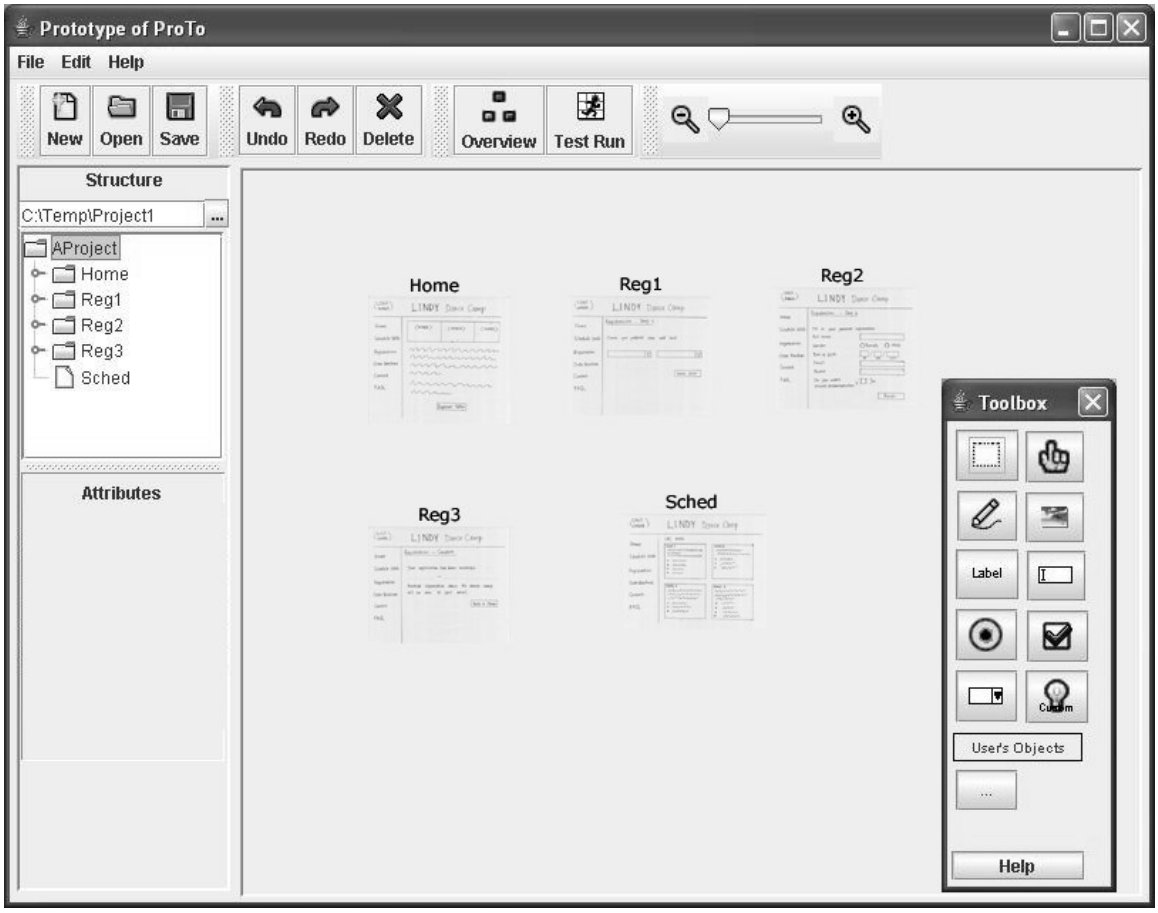

Figure 22 An overview with five pages loaded. The pages are represented as thumbnails which are movable within the pane by dragging. Notice that the level of zoom is currently at the minimum.

The thumbnail can be opened to display the page detail by double-clicking or by zooming. Notice that the root node "AProject" in structure is automatically selected to give feedback to the user that he is working at the most general mode - overview mode.

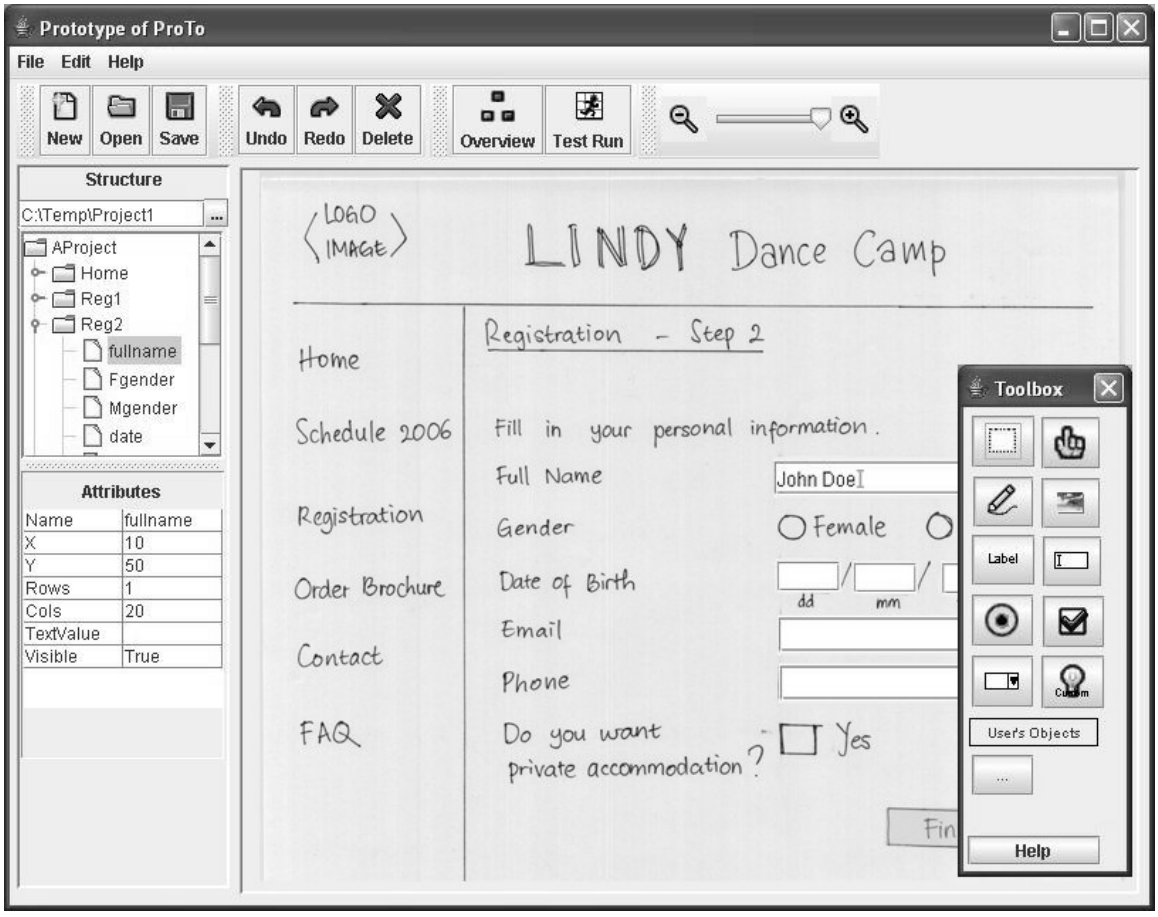

Figure 23 In page detail mode, the designer can add components to the relevant page. Some specific components can serve to be the source of interaction within the website being prototyped. The interaction itself is defined in transition mode.

The interaction within the designed web is defined as a transition. A transition contains several elements, namely the source object, the triggering event, the action/operation and the target object. As depicted in the figure below, the transition is composed by selecting the element values from the dropdowns. The easy language is intended to help novice users. A more familiar designer may choose to work by typing commands and comments directly in the script area. The complete combination of possible events, actions and target is listed in Appendix 2.

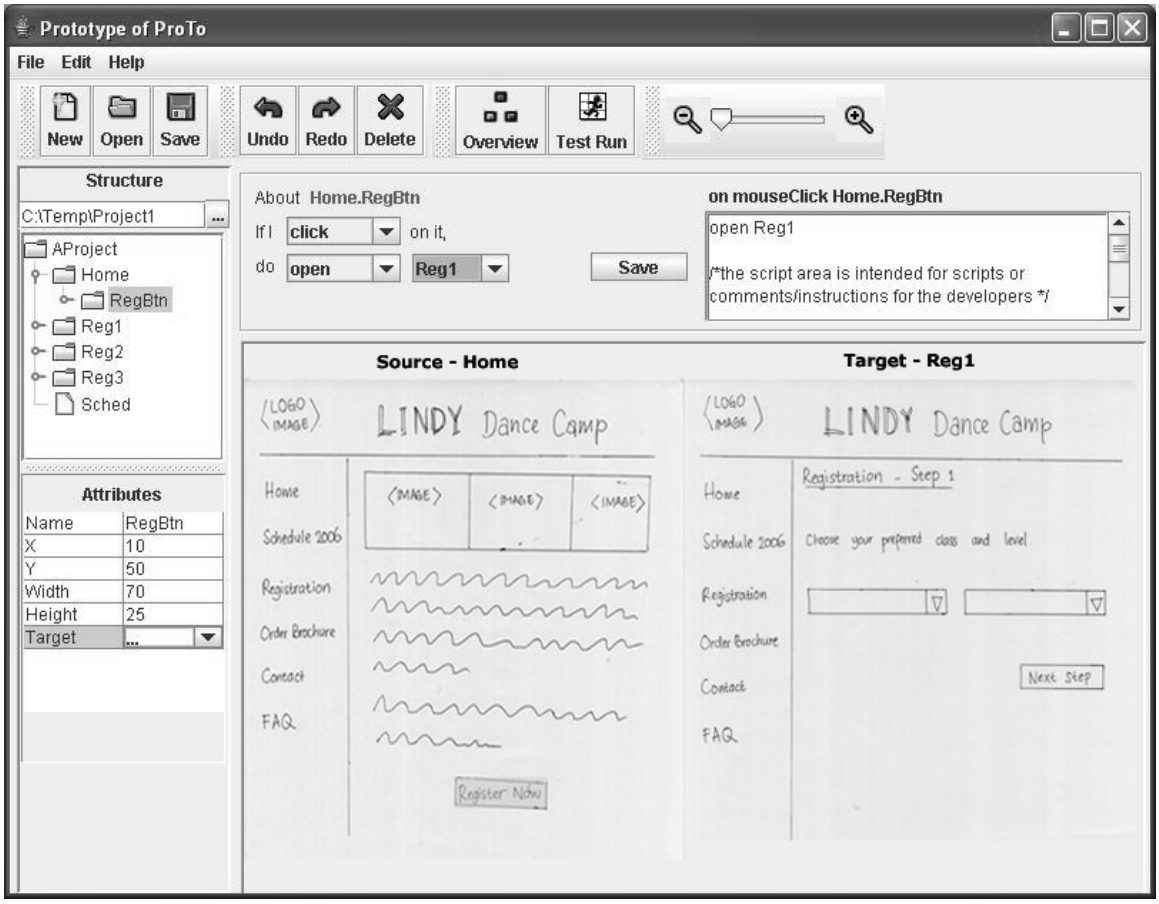

Figure 24 A transition is defined in transition editor. A double-click on "RegBtn" element in the tree structure or an attempt to set value of attribute "Target" brings up transition editor mode. Once in this mode, a transition is created by selecting values of triggering event, operation/action and target, which will automatically update the content of command area. Commands and/or comments can also be typed directly in the command area.

The system constantly gives feedback to user in which mode he is. The selected element in the tree structure is a clear clue on which object the designer is currently working on. Putting tool tips for the interface is a more explicit way, which is worthy to be implemented in the eventual tool.

For some specific functions, form and menu based interface is effective to provide structuring of the interaction design. An example is the interaction in defining transition by selecting values from available dropdowns or by typing commands directly. Such approach can be applied to other interactions to duplicate the flow so that both novice and expert users can choose their own preference.

6 Discussion

Designers produce sketches as a design process, either personally or with colleagues. Some difficulties of using paper in prototyping are rooted from the difference between paper prototypes with the eventual product, which causes the users to misunderstand.

Designers have high expectations towards prototyping tools. It has to fit into their practice, be easy to operate with natural interaction (direct manipulation), supports communication between designers, users and developers, and be integrated with other development tools, i.e. programming tools.

The design of prototyping tool put forward in this thesis adapts other design tools in preserving sketchy or informal appearance. It applies end-user programming concept, such as *before-after rules* (Smith et al, 2000).

6.1 The Practical Issues of Prototyping

Paper prototyping as described by Snyder (2003) has crucial limitations: inability to capture the realistic performance and to simulate specific interactions. The limitations are related with ergonomic difference between interaction with paper and with computer.

From the discussion with testing participants, I found out that while working with paper sketches, some designers apply "divide and conquer", showing just a small part of the entire system to make the users focus on that part only. Thus, the users they are communicating with see "zoomed-in" pictures of small parts of the system. This might lead to misunderstanding, as users are unaware of the "big picture" (i.e. how the whole system should work). The misunderstanding also appears when designers put too much detail in the design, as the users become distracted and start commenting non-relevant parts (Newman et al, 2003).

6.2 About Prototyping Tools

The participants in usability testing have not practiced using design tools, as mentioned in Chapter 3, to sketch on computer in their work. The reason could be that there is a necessity to learn about the tool, no matter how easy it is. In addition, the designers have to see how the tool fits in his practice, and analyze if it will make the work more efficient.

The discussion with participants in usability testing confirmed that Microsoft PowerPoint is widely used because it is easy to use and familiar in the sense that people generally know how to use it. To some

degrees, it is similar with using paper as one presentation page is comparable with one piece of paper. The prototyping culture still has not changed; designers use easy and available tools instead of complicated prototyping tools. (Landay and Myers, 1995a; Lantz et al, 2005; Newman et al, 2003; Olsen, 2002)

6.3 The Design of Prototyping Tool

The prototyping culture is a dynamic issue, and will likely to change with new trend of tools or new emerging disciplines. Each designer might have his own particular practice, influenced by his organization/team culture. An approach to make use of existing paper sketches is accepted by the designers, as expressed by a testing participant:

> *"..when you can directly overlay [the sketches] with computer, it's good. I get a bit closer [to the final product]. It's not such a wide gap between the sketch and the implementation."*

The prototyping tool put forward in this thesis attempts to keep the prototypes in informal appearance by using sketch-and-scan interface, as other surveyed tools also support sketching on tablet PC to keep the informal appearance to the users (Landay and Myers, 1995a; Kant et al, 1998; Bailey et al, 2001; Lin and Landay, 2002; Sinha and Landay, 2003).

Designers are the end users of the prototyping tool discussed in this research. The design of the tool should be easy as no one expects the end-users to learn it devotedly. An approach to achieve the wanted effects with less programming to end users, i.e. designers, is *programming by demonstration* (Cypher, 1993). For example, the transition creation applies *before-after rules* (Smith et al, 2000) as a form of PBD. Two screens in transition editor serve as the source and the target of interaction. The testing participants barely said anything about it, which is a good sign because they find it natural and have nothing to complain about it.

7 Conclusion

The conclusion gives answers to the research questions mentioned in Chapter 1. Some future works are proposed to envisage what could be extended from this work.

- *What is the designers' attitude towards paper prototyping?*

A finding that is consistent with previous research (Newman et al, 2003), is that designers sketch on paper at least once. In other case, they also draw their design on whiteboard. Regarding the use of paper sketches, some designers use them in informal discussion, just to show to users to get their thought. It is done using narration with open-ended discussion. Some other designers display the operational functionality by simulating responses on the paper as described by Snyder (2003). The effectiveness of the two approaches is depending on the environment. The development of ordinary computer application tends to use less paper prototyping because the users are generally familiar with the interface; while designing a new metaphor or interface will benefit from using paper prototyping to familiarize the concept to target users.

- *How can a computer-based prototyping tool be designed to support integration with existing paper sketches?*

The major drawback of prototyping tools is because they do not fit into practice. The tools tend to be complex and extend more functionality rather than usability. Meanwhile, the prototyping practice is diverse among designers as individuals or in development teams. The reality matters greatly into the problem of gathering designers' requirements. This research has proposed a new design of prototyping tool, which does not overlay designers' practice with an extra layer of hardware or software. Sketch-and-scan interface allows designers to sketch freely on paper. To integrate with the eventual product, a proposed work for the future is to implement transferability from the prototyping tool to programming tools, for example to generate layout or source code framework.

A further study on artifacts and how the designers produce and use them is suggested to improve this piece of work. Another important issue is to investigate the effectiveness of using prototyping tools. A research on effectiveness investigation of mobility in developing prototypes is taking place, using PROTEUS – a tool to examine artifacts creation in web design (Mohamedally, Zaphiris, and Petrie, 2005).

8 References

Anoto. (2004). http://www.anoto.com

Bailey, B.P., Konstan, J.A., & Carlis, J.V. (2001). DEMAIS: Designing Multimedia Applications with Interactive Storyboards. *Proceedings of ACM Multimedia,* 2001, pp. 241-250.

Boshernitsan, M. & Downes, M.S. (2004). *Visual Programming Languages: A Survey*. Technical Report, University of California, Berkeley.

Callahan, J., Hopkins, D., Weiser, M.; & Shneiderman, B. (1998). A Comparative Comparison of Pie vs. Linear Menus. *In Human Factors in Computing Systems, Proceedings of CHI,* 1988.

Cook, D.J. & Bailey, B.P. (2005). Designers' Use of Paper and the Implications for Informal Tools. *Proceedings of OZCHI*, 2005.

Cooper, Alan & Reimann, Robert. (2003). *About Face 2.0: The Essentials of Interaction Design*. Wiley Publishing, Inc.

Cypher, A. (1993). *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA.

Halbert, Daniel C. (1993). SmallStar: Programming by Demonstration in the Desktop Metaphor. In: Cypher, A., ed. *Watch What I do:Programming by Demonstration*.

Houde, S. & Hill, C. (1997). What do Prototypes Prototype? In Helander, M., Landauer, T.K.; and Prabhu, P.(eds.), *Handbook of Human-computer Interaction*. Apple Computer, Inc. California. pp.367-381.

Hong, Jason I., Li, Francis C., Lin, J., & Landay, James A. (2001). End-User Perceptions of Formal and Informal Representations of Web Sites. *In Extended Abstracts of Human Factors in Computing Systems: CHI* 2001, Seattle, WA, March 31-April 5, 2001, pp. 385-386.

Kahn, Ken. (1996). ToonTalk – An Animated Programming Environment for Children. *Journal of Visual Languages and Computing*, vol. 7, no. 2, pp. 197-217.

Kant, M., Wilson, S., Bekker, M., Johnson, H., & Johnson, P. (1998). PatchWork: A Software Tool for Early Design. *Proceedings CHI'98 Late Breaking Results*, 1998.

Kindborg, Mikael. (2003). *Concurrent Comics - programming of social agents by children*. PhD Dissertation. Linkoping University.

Kindborg, Mikael & McGee, Kevin. (2005). Comic Strip Programs: Beyond Graphical Rewrite Rules. *International Workshop on Visual Languages and Computing (VLC2005)*, Canada, September 2005.

Landay, James A. (1999). *Low-fidelity Prototyping & Storyboarding*. Lecture Notes. Computer Science Division, University of California, Berkeley.

Landay, James A. & Myers, Brad A. (1995a). Interactive Sketching for the Early Stages of User Interface Design. *In Proceedings of CHI '95: Human Factors in Computing Systems*, Denver, CO, May 1995, pp. 43-50.

Landay, James A. & Myers, Brad A. (1995b). Just Draw It! Programming by Sketching Storyboards. *Carnegie Mellon University, Human-Computer Interaction Institute Technical Report CMU-HCII-95-106 and School of Computer Science Technical Report CMU-CS-95-199*, November 1995.

Lantz, Kenneth S. (1986). *The Prototyping Methodology*. Prentice-Hall, Englewood Cliffs, NJ.

Lantz, A., Artman, H., & Ramberg, R. (2005). Interaction Design as Experienced by Practitioners. *In Proceedings of In The Making, Nordic Design Research Conference*, Copenhagen, Denmark, May 2005.

Leone, P., Gillihan, D., & Rauch, T. (2000). Web-based prototyping for user sessions: Medium-fidelity prototyping. *In Proceedings of the Society for Technical Communications 44th Annual Conference*, 2000, pp. 231-234.

Lieberman, Henry. (2001). *Your Wish is My Command: Programming by Example*. Morgan Kaufmann Publishers.

Lin, James & Landay, James A. (2002). Damask: A Tool for Early-Stage Design and Prototyping of Multi-Device User Interfaces. *In Proceedings*

of the 8th International Conference on Distributed Multimedia Systems (2002 International Workshop on Visual Computing), San Francisco, CA, September 26-28, 2002, pp. 573-580.

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

Mohamedally, D., Zaphiris, P., & Petrie, H. (2005). Incorporating Digital Inking Methods in HCI Knowledge Elicitation. *Workshop Proceedings of HCI*, 2005.

Nielsen, J. (1992). The Usability Engineering Life Cycle. *Computer*, 1992, pp. 12-22.

Olsen, Henrik. (2002). Results from a survey of web prototyping tools usage. Online resource. *The Interaction Designer's Coffee Break*. Retrieved 18 August 2005 from http://www.guuui.com/issues/01_03_02.php

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction Design: beyond human-computer interaction*. John Wiley & Sons, Inc.

Schrage, Michael. (1996). Cultures of Prototyping: HyperCard, Director, and Visual Basic. In: Winograd, Terry, ed. *Bringing Design into Software*. Addison-Wesley.

Shneiderman, Ben & Plaisant, Catherine. (2005). *Designing the user interface. 4th ed*. Boston, San Francisco, New York: Addison-Wesley.

Sinha, Anoop K. & Landay, James A. (2003). Capturing User Tests in a Multimodal, Multidevice Informal Prototyping Tool. *Fifth ACM International Conference on Multimodal Interfaces: ICMI-PUI 2003*. Vancouver, B.C., November 5-7, 2003.

Smith, D.C., Cypher, A., & Tesler, L. (2000). Novice Programming Comes of Age. *Communications of the ACM*, vol. 43 no. 3 (March), pp. 75-81.

Snyder, Caroline. (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann Publishers, Elsevier Science, SF.

Spool, Jared M. (2005). Looking Back on 16 Years of Paper Prototyping. Online resource. *User Interface Engineering*. Retrieved 18 August 2005 from
http://www.uie.com/articles/looking_back_on_paper_prototyping/

Starling, Andrew. (2002). Usability Testing in Practice. Online resource. *The Web Developer's Virtual Library*. Retrieved 2 April 2006 from
http://wdvl.internet.com/Authoring/Design/UsabilityTesting/

Tortora, G. (1990). Structure and interpretation of visual languages. In Chang, S.-K., editor, *Visual Languages and Visual Programming*, pp. 3–30. Plenum Press, New York, 1990.

Virzi, R. A., Sokolov, J .L., & Karis, D. (1996). Usability problem identification using both Low- and High-Fidelity Prototypes. *Proceedings of ACM CHI '96*, Vancouver, British Columbia, Canada, pp. 236-243.

Walker, M., Takayama, L., & Landay, James A. (2002). High-fidelity or low-fidelity, paper or computer medium? *In Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting: HFES 2002*. September 30-October 4, 2002, pp. 661-665.

9 Appendix

9.1 Objects and Attributes

OBJECT	ATTRIBUTE	TYPE	DESCRIPTION
Page	Name	Text	Name of the page
	isBackground	Boolean	True if the relevant page serves as background to other pages
Hyperlink (area)	Name	Text	Name of the hyperlink
	X	Numeric	Coordinate of x-axis
	Y	Numeric	Coordinate of y-axis
	Width	Numeric	Width of the hyperlink area
	Height	Numeric	Height of the hyperlink area
	Target	<dialog>	Opens up transition editor when modified
Drawing	Name	Text	Name of the drawing
	X	Numeric	Coordinate of x-axis
	Y	Numeric	Coordinate of y-axis
	Stroke width	Numeric	The boldness of the lines constructing the drawing
	Visible	Boolean	If true, the drawing is displayed
Image	Name	Text	Name of the image
	X	Numeric	Coordinate of x-axis
	Y	Numeric	Coordinate of y-axis
	Width	Numeric	Width of the image
	Height	Numeric	Height of the image
	Source	<dialog>	Opens up a file open dialog to browse image source file
	Visible	Boolean	If true, the image is displayed
Text	Name	Text	Name of the text
	X	Numeric	Coordinate of x-axis
	Y	Numeric	Coordinate of y-axis
	Font	<dialog>	Opens up a font configuration dialog
	Text value	<editable selection>	Text value can be typed directly or be selected from value list of text objects
	Visible	Boolean	If true, the text is displayed

Textfield/ Textarea	Name	Text	Name of the textfield/textarea
	X	Numeric	Coordinate of x-axis
	Y	Numeric	Coordinate of y-axis
	Rows	Numeric	Number of rows. Default value is 1. If value>1, the object becomes a textarea.
	Cols	Numeric	Number of characters length to be displayed. Default value is 20.
	Text value	<editable selection>	Text value can be typed directly or be selected from value list of text objects
	Visible	Boolean	If true, the textfield/textarea is displayed
Selection	Name	Text	Name of the selection
	X	Numeric	Coordinate of x-axis
	Y	Numeric	Coordinate of y-axis
	Rows	Numeric	Number of rows. A dropdown has value of 1. If value>1, the object becomes a list.
	Cols	Numeric	Number of characters length to be displayed. Default value is 20.
	Text value	<editable selection>	Text value can be typed directly or be selected from value list of text objects
	Multiple	Boolean	If true, multiple selection is allowed
	Visible	Boolean	If true, the selection is displayed
Radio button/ Checkbox	Name	Text	Name of the radio button/checkbox
	X	Numeric	Coordinate of x-axis
	Y	Numeric	Coordinate of y-axis
	Group name	<editable selection>	User can type in a new group name or select from a list of existing group
	Selected	Boolean	If true, the radio button/checkbox is selected
	Visible	Boolean	If true, the selection is displayed

9.2 Transition Elements

EVENT
Click
Double click
Hover
Drag & drop
Keypress <key value>

ACTION	VALUE	TARGET OBJECT
Open page	N/A	<page>
Set visibility	Toggle	<object which has "visibility" attribute>
	True	
	False	
Set text value	<user typed>	<text OR textfield OR textarea>
	<selection of text values from other text objects>	

9.3 Context Scenario 1

Anna is a freelance web designer. She has educational background in art, and learns web design autodidactic. Her main passion is traveling, and blending-in with the culture of the places she visits. Her last trip to Spain has earned her a project.

Right now, Anna is designing a website for a youth hostel which is located in southern Spain. The website, in addition to describing the hostel itself, also provides a lot of information on the culture, the events going on in Spain, traveling advices, suggestion for tours, etc. Conforming to the spirit of young travelers and their nature loving of pop culture, she pays attention to the artistic aspect of the site and experiments with icons and symbols to express ideas.

On the main page, she provides image links along with short description to the topics which are culturally important for Spanish people. Each link opens a new page detailing the subsequent topic. In this stage of design, Anna does not care about the content of the page so she just takes *greeked* text to specify the layout. To capture the dance culture on the main page, she depicts a figure of a *bailador* – a male Flamenco dancer. In another occasion, the eating and drinking culture is represented in a drawing of a

plate of *tapas* – typical small food items or snacks – and a glass of wine. Arising from the concept of simplicity, she intentionally creates the icons in comical representation in black and white, and with very little details.

While working in the design, Anna always draws on paper, creating sketches on the icons look and the layout within the website. She thinks it is more practical to work on the raw ideas on paper, and later on she will refine the appearance of the design using a graphical design tool. The main reason is that she feels more at ease when drawing on paper, where she can express her potential artistic skill without interruption.

Once she finishes the design of the entire website, she wants to show her design to the stakeholder – i.e. the youth hostel's owner. She feels the need to tidy up her design and define the navigation between pages to make sure there will be no misunderstanding between her and her client. She uses a scanner to make a digital format of the sketches. Now that she has done it, she feels safer because she has the digital backup of all her work. Paper is vulnerable and can break off because of rain, a splash of coffee, etc, she thinks.

She adds the image files into a prototyping tool which supports the early stages of web design. An image file is a page within the tool, and she can arrange the position of the pages to make them easier to organize. There she checks into the detail of every page, while defining clickable rectangular areas. The areas are shown as transparent green rectangles, and serve as the source of interaction. She creates simple transitions, i.e. clicking on a specific area within a page will open another page.

The tool wraps the digital sketches into HTML pages, along with the interaction the pages should have while maintaining the informal appearance. The navigation is easily reproduced by clicking on a specific area on a page. Anna emails the HTML files to her client – emphasizing that her client should comment on the structure of the website and the drawings that she uses to symbolize a topic.

9.4 Context Scenario 2

Juan is a full-time web designer and he practices paper prototyping to explain his design ideas to his colleagues and clients. It costs almost nothing, but it works to elicit valuable feedback, so why not, he thinks. In his current project – designing a kiosk application for a furniture shop – he has scribbled down the design in detail on papers. His intention is to conduct paper prototyping with some potential users. Since he wants to optimize the resource for other projects, he needs to compress the time

needed for one paper prototyping session. He thinks "computer" – human acting as computer – is slow in response to user's operation. With several sessions to be done, he wants to automate "computer" response without having to create a higher-fidelity prototype using UI builder tool or the similar.

He scans his sketches into digital images and loads them into a prototyping tool. Firstly, he defines the navigation between screens. He also needs to create the interaction within one screen. For example, a mouse over on a specific area changes an image within the same screen. A mouse click on the image causes a paragraph of text to show up. He works on such interaction by defining areas and manipulating the visibility of the layer objects.

Once he finishes working with the interactive prototype, he checks the result in the "Run" mode. This mode executes an integrated browser which loads the automatically created HTML files. The interactions and navigations can be simulated within the browser. When he eventually conducts the testing session, he or the tester can immediately put annotation by typing in a comment box at the bottom of the browser.

9.5 Key Path Scenario

Every summer hundreds of people join "Lindy Dance Camp" in Stockholm to learn dancing. The camp offers several kinds of dance classes with variety of levels. So far, registration and enquiry from prospective participants are sent via postal service or via phone. With the increasing interest from people, the camp decided to have a website where people can find detail information about the camp, registration and payment, etc. The camp asked you as an experienced web designer to develop the design and to structure the content.

You have created detailed sketches on paper. The digital format of your sketches is available – you have scanned them. You want to add interactions to the static paper sketches so that the client easily perceives your ideas without your having to explain so many little details verbally. You use a prototyping tool for this purpose. You have loaded the pages and are ready to create the interactions.

Some interactions are highlighted to demonstrate the capacity of the tool. You are requested to **think aloud** while working on the tasks below.

Navigate from "Home" to "Schedule 2006" and "Registration".

Draw areas on top of the text “Schedule 2006” and “Registration” in the left-side menu.
Draw an area on top of “Register Now” button at the bottom.
Create a transition from the area. You will need to define the source, triggering event, action and target consecutively.

If the user clicks at “Registration” menu item in “Home” page, it will open “Registration” page.
Source page is the relevant page of “Home”, source object is the relevant area which covers the menu item “Registration”.
The triggering event is on mouse click. The action is to open the page.
The target page is the relevant page of “Registration”, while target object is not applicable.

Text inputs, Checkbox and Radio Buttons in Registration Step 2
Draw textfields on top of fields full name, date/month/year of birth, email and phone.
Draw areas on top of the selection of female/male gender, the preference for private accommodation and the “Finish” button. There are 4 areas in total.
Change the default value of full name into “John Doe”.

The checkbox mechanism is done by manipulating the visibility of the tick mark. When the user clicks on the area for private accommodation, toggle the visibility of the tick mark.
Create a transition from the area. Source page is the relevant page of “Registration 2”, source object is the relevant area which covers the square next to private accommodation text.
The triggering event is on mouse click. The action is to set visibility.
The target page is the same page, “Registration 2” page. The target object is the tick mark drawing.

Displaying Values in Registration Step 3
The first empty space should display the full name of the person who just filled in “Registration Step 2”.
Draw a text on the empty space. Change the value to display the relevant full name textfield in “Registration Step 2”.

9.6 Questionnaire

Please mark the numbers which most appropriately reflect your impression after using the prototyping tool.

Tedious	1 2 3 4 5 6 7 8 9	Efficient

	10	
Frustrating	1 2 3 4 5 6 7 8 9 10	Satisfying
Difficult	1 2 3 4 5 6 7 8 9 10	Easy
Dull	1 2 3 4 5 6 7 8 9 10	Stimulating
Clueless	1 2 3 4 5 6 7 8 9 10	Intuitive

Does the prototyping tool cover necessary interaction for web design?
□ Yes
□ No, these are missing: ……

Is it easy to find the appropriate tool? 
□ Yes
□ No, I have problems with: ……

Do you find the available tools sufficient for web design?
□ Yes
□ No, I would consider adding: ……

Do you find ProTo intuitive to operate?
□ Yes
□ No, I have preference/suggestion regarding: ……