Intelligent Mind-mapping

Vincent Chun Hei Chik

Under supervision of Dr Beryl Plimmer

A thesis submitted in fulfilment of the requirements
for the degree of Master of Engineering in Software Engineering,
The University of Auckland, 2008

Acknowledgements

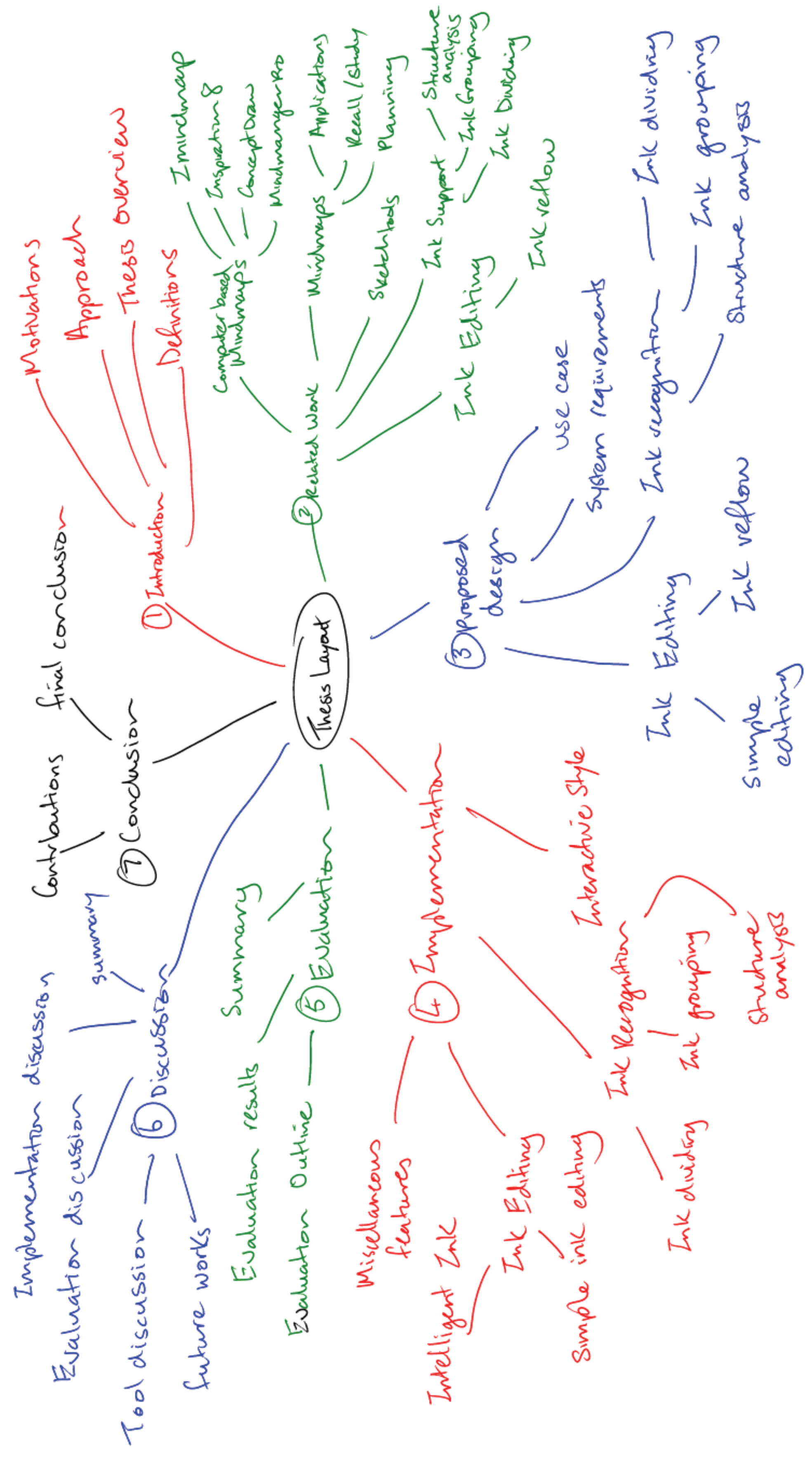
I would like to acknowledge first and foremost, my supervisor, Dr Beryl Plimmer, for providing helpful advice during tough times and guiding me in the right direction throughout the year. Also, I would like to thank Teresa & George, my mum and dad for providing support and my brother Desmond for being on hand whenever I needed help. I would also like to thank my friends especially Alistair, Stephen and Koushi for sharing ideas and pushing me on during the write up of the thesis. And lastly I would like to thank the participants of my study for the time they have sacrificed and the suggestions they offered to improve my tool.

Publications

Publications for this research are:

Chik, V., B. Plimmer, et al. (2007). Intelligent mind-mapping. Proceedings of the 2007 conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artifacts and environments. Adelaide, Australia, ACM.

Thesis Mind-map



Abstract

Mind-mapping is a brainstorming technique that has many applications. It is a technique for fast idea generation and is often used in planning, critical thinking, studying and note taking. Traditionally, mind-mapping is performed on pen and paper but the need to store these mind-maps as digital documents brought forth widget based computer tools. There is existing research that shows widget based environments being harmful to the interface design process as the user's focus is shifted from design to arrangement. However, these interruptions are minimised with sketch based tools. We hypothesise that the same applies for mind-mapping as mind-mapping has many similar characteristics to interface design.

We have constructed a sketch based mind-mapping tool, Intelligent Mind Mapper (IMM). Using a tablet PC to mimic the physical nature of a clipboard we provide a sketching environment for the user to create their own mind-maps in the same way mind-maps are traditionally constructed. Also, IMM provides the user with editing features of computer based tools, allowing users to edit individual ink properties, move keywords or even relocate an entire branch of the mind-map. Our tool differs from either environments and yet acquires the benefits from both. Our tool mimics the interaction style of the pen and paper environment yet can perform ink editing manipulation that pen and paper cannot. The IMM is also similar to the widget based software in regards to the digital support they provide. However, our system removes the distractions present in the widget based tools, allowing the continuous sketching of the mind-map. The novel techniques we have implemented results in a mind-mapping application that shows signs of potential from users who have found the interaction intuitive and enjoyable.

The findings from our research are presented here in this thesis. We commence with an overview of mind-mapping, defining its constituents and its uses. Then we progress into reviewing the computer based mind-mapping software. Afterwards we examined literature on sketch tools and how they are better at interface design than their widget based counterparts. There are many difficulties associated with sketch tools and we have identified three challenges, ink recognition, structure analysis and ink reflow. With the background knowledge gained form the literature, we design our mind-map based on personas, use case scenarios and informal study on how users construct mind-maps. Then we illustrate our tool implementation and evaluation process and lastly discuss the future works for this tool.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A mind-map is a sketchily structured visual representation of one's thoughts which may lead to a train of related ideas. It is based on radiant thinking, a concept which describes how the human brain processes ideas and information, whereby different ideas are associated to each other through relationship hooks (Buzan and Buzan 2000). A mind-map consists of a central idea or theme and related ideas branch out, connected together via "relationship hooks". Subsequent ideas are linked together, forming a hierarchical map of the user's ideas. Mind-maps have a variety of applications. For instance, Mind-mapping is an effective way of generating ideas in that each idea that branches out increases the range of possible associations (Buzan and Buzan 2000). This results in an idea generation system that is almost limitless.

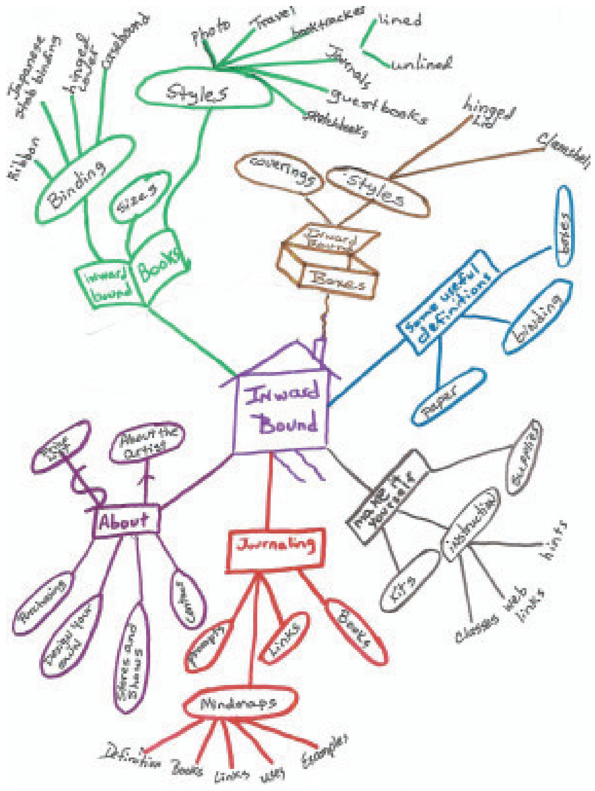


Figure 1 A hand drawn mind-map

In addition, they are used for critical thinking tasks such as strategic planning as less time is required to record and review information. Mind-maps are an effective way of rapidly jotting down and arranging information, affording reinforced association of ideas and recall, hence are advantageous for group contributions as well as a quick and effective method to study.

Mind-maps are traditionally hand drawn and with the digital age, whereby data are stored electronically, there has been development of mind-mapping software that allows the construction of a digital mind-map. These applications provide the normal components of a traditional mind-map with the addition of other functions such as export, editing and replication. It has been documented in user interface and media design, that users are distracted by tool operations such as finding and arranging widgets. In contrast, this adverse effect of the computer is minimal with sketch-based computer tools (Bailey and Konstan 2003; Plimmer and Apperley 2003).

We predict that this kind of distraction is present in mind-mapping tools and that this shift in focus from brainstorming to tool management interrupts the rapid brainstorming process that mind-maps are intended to support. They are an effective way of rapidly jotting down and arranging information, affording reinforced association of ideas and recall. We want to determine if providing a tool that removes these distractions would have a beneficial effect in the idea generation process and to do this, we have implemented a system on the tablet PC to mimic the traditional pen and paper environment whilst keeping the benefits of a computer tool.

1.1 Motivations

Our main motivation is to develop a natural but powerful computer based environment for the user when constructing a mind-map. We must create a disruption free environment to encourage creative user thinking. Pen and paper does facilitate this environment in that the user is only concerned with annotating on the paper. However, this lacks the benefits that are present with current mind-mapping software. The fact that a computer based tool enables mind-maps to be easily stored, replicated, exported and distributed weighs heavily in favour of a computer oriented application. There are a number of computer applications for mind-mapping. However these are widget-based tools that require the user to select an appropriate node widget or connector before they can enter data.

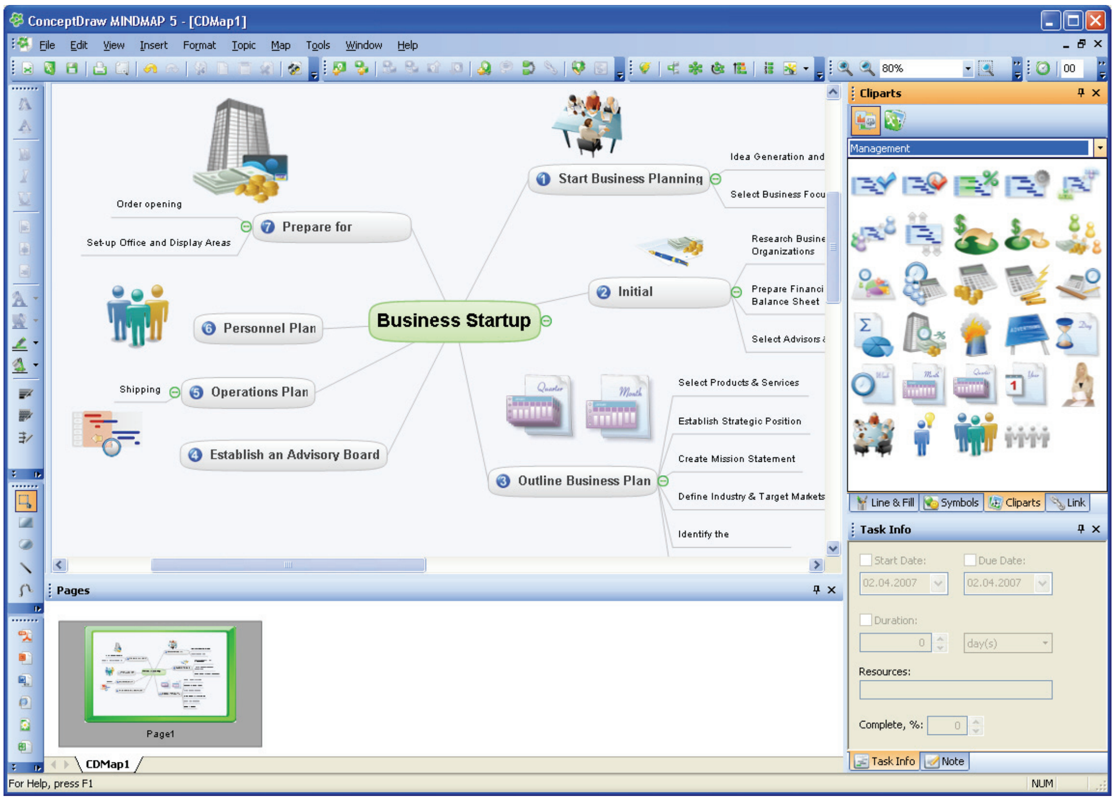


Figure 2 A mind-mapping tool

In this example, the user clicks on the canvas to create new nodes and type in keywords. To move nodes, the user has to select a particular node and drag it onto another node in order to change its location on the mind-map. We hypothesise that these tools will adversely affect the idea generation process in the same way as widget-based design tools have been shown to adversely affect the design process (Goel 1995). The distractions the user has may increase the time taken to form a map and as mind-mapping is a rapid process, there may be a decrease in the number of ideas generated using software tools.
Our goal is to construct a pen-based mind-mapping tool that utilises the benefits of both environments. This entails taking advantage of the non-intrusive nature of the Tablet PC stylus that is similar to pen and paper and as well as providing digital support. The system will support intelligent interpretation, editing and transformation of ink to facilitate the creative idea generation process. Doing so removes disruption to the construction of the mind-map and allows one to evaluate its efficacy.

1.2 Our Approach

Our approach to this problem is to firstly review the literature regarding mind-maps, sketch tools and current mind-mapping tools. The mind-mapping literature will give insight to the essential components present in a mind-map. The available literature in the field of sketch tools has also been reviewed in particular topics such as ink recognition and ink reflow and other identifiable challenges our mind-mapping tool may encounter. Exploration into available computer-based mind-mapping tools enables us to identify the functionalities that they offer and more importantly, the general constraints when constructing a map. This background material provides a good understanding of the problems that ink annotation tools face and the possible approaches we can utilise. Solutions to possible inking problems found in the literature review are critiqued and compared to determine if they can be incorporated into our system. A system requirements specify a list of features that need to be present in the tool. As part of the requirements, the sketch-based design tool must also provide the usual computer editing and archiving support while being able to intelligently classify, group and reflow the ink strokes. The requirements are then implemented to form our prototype system. This system is then evaluated using usability testing to identify the usability problems that were missed and the appropriate corrections made.

The contributions made to this project lies in ink management and providing intelligent digital support for mind-mapping. Contributions in ink management mainly focus on producing novel techniques in ink stroke grouping, context grouping and ink reflow.

1.3 Thesis Overview

Chapter 2 - Related work

The mind-map is explained more in depth, outlining its features and uses before examining the pros and cons of existing widget based mind-mapping applications and sketch tools. Furthermore, existing works surrounding extracting information from ink annotations for recognition, problems and solutions to ink reflow, and structuring problems are compared and discussed.

Chapter 3 - Proposed implementation

This chapter covers the reasoning behind our proposed implementation of our mind-mapping software, drawing upon the literature from related works. From our personas and use case scenarios we create a systems requirement for our tool.

Chapter 4 - Implementation

This chapter describes the novel ideas used in the implementation of IMM, our mind-mapping software. In addition, we discuss the implications that are present in our tool and the solutions used to overcome them.

Chapter 5 - Evaluation

The usability evaluation performed on our first prototype and the documented results are discussed in this chapter.

Chapter 6 - Discussion

Discussion on the tool and implementation. This chapter covers the benefits our tool and highlights the difficulties encountered such as the mind-map limitations and constraints.

Chapter 7 - Conclusion

Concludes the thesis with the contributions this project has towards ink annotation recognition and ink editing while assessing the potential future work for this project.

1.4 Definitions

The following definitions are used throughout the entire document

Central Node	The first and main idea of a mind-map
Node	A group of ink strokes that represent a coherent idea
Connector	An ink stroke that connects two nodes that are related
Stroke	A mouse down, mouse up movement
Himetric	Ink space coordinates (1 Himetric unit = 0.01 mm)
Grouper	A module that intelligently groups relevant ink strokes together
Divider	A module that classifies ink strokes into text or drawing by their features.

Chapter 2

Related Work

In this chapter, we discuss the literature surrounding mind-maps and digital ink tools, techniques and challenges. Section 2.1 describes a mind-map in greater detail, identifying the key components and how a map is constructed. Section 2.2 samples a selection of existing computer based mind-mapping tools and investigates the key functionalities and constraints present. Section 2.3 examines the benefits of sketch tools in design over widget based computer tools. Section 2.4 and 2.5 delves into the technical challenges of digital ink, describing possible approaches for our system.

2.1 Mind-maps

Tony Buzan (Buzan and Buzan 2000) first proposed mind-mapping as a fast creative thinking technique. To create a mind-map, a user first writes down on paper the main idea such as a "house". Once this is written down, thoughts relating to this main idea come to mind and the user writes down parts of the house such as "roof", "kitchen", "bedroom". A line is drawn to connect these ideas to "house" to show a relationship between them. As each new keyword is written down, the range of possible ideas increases and this process continues iteratively. Ideas that are formulated are drawn in a radial manner and its purpose is to invoke both the creative and logical sides of the brain (Vidal 2004) and the stimulation of the whole cortex promotes idea generation. Buzan (Buzan and Buzan 2000) commented that linear note taking "cuts off" ideas that come before or after it and the idea growth of the user becomes stunted. A mind-map however, is the opposite as it maintains an open structure that encourages the user to insert more ideas.

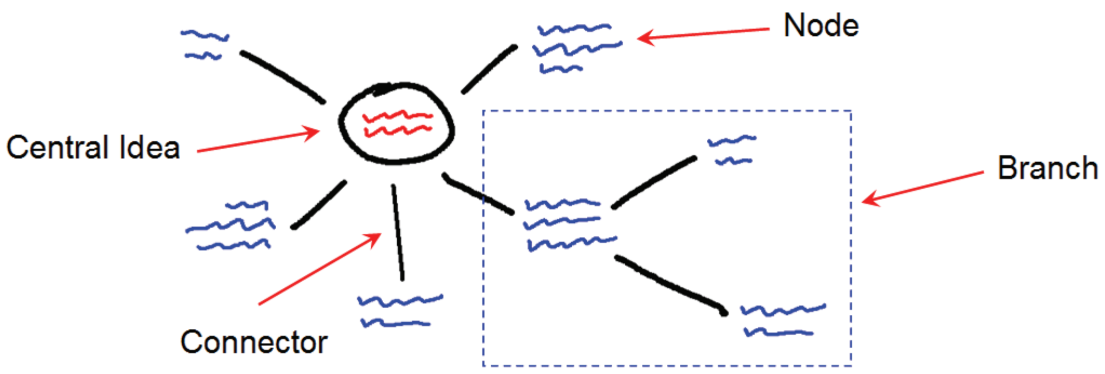


Figure 3 Components of a mind-map

The four main features of a mind-map are as follows:

1. Each mind-map has a starting location, the centre node that contains the main theme or idea.
2. The ideas of the mind-map "radiate" from the central node as branches with sub-nodes connected to each other in parent-child relationships.
3. The final structure of the mind-map becomes a hierarchy of linked nodes.
4. Each connector/branch has keywords or an image associated with it.

Since then, applications for mind-mapping have been found in (Buzan and Buzan 2000; Vidal 2004):

- Note taking
- Learning/Studying
- Problem solving
- Planning
- Teaching

Compared with normal note taking or brain-storming, mind-maps have several advantages (Buzan and Buzan 2000). For instance, time is saved by only noting down relevant key words. Associations between key points are highlighted while passively creating a hierarchy of ideas. Reviewing a mind-map takes considerably less time than to overview a set of written notes as the mind-map is effective in displaying the relevant keywords associated with a particular topic. By providing a visually stimulating environment, the retention of information by the brain is made easier.

Uses of mind-mapping for learning have been examined in the context of studying notes (Farrand, Hussain et al. 2002), non-linear & critical thinking (Buzan and Buzan 2000;

Mueller, Johnston et al. 2002), planning (Mento, Martinelli et al. 1999), teaching (Sivathasan and Ho 2005).

A study by Farrand, Hussain et al. (2002) conducted on 50 medical students at the London School of Medicine and Dentistry investigated the efficacy of mind-mapping as a tool for studying versus conventional methods. Students were given a 600 word article from Scientific American and were given five minutes to read the selected text and make notes. All participants were required sit a test from a set of questions. Half the participants were then exposed to 30 minutes of mind-mapping technique session where they were taught on how to construct a mind-map. All the participants were allowed to review the text and answered another set of questions. The students were told to return and answer the last set of questions one week later with no further exposure to the text during that period. From the results of the question sets, it was determined that the students that used the mind-mapping as a form of note-taking and study performed slightly better in the short term by approximately 10% and a 24% improvement in correct long term recall than the other students.

Mueller, Johnston et al. (2002) discusses the use of mind-mapping by nursing students to enhance their critical thinking. As the nursing process is governed by care plans, nurses are restricted to linear thinking for patient diagnosis. Although this method is effective in theory, it becomes troublesome in practice as each nurse has a different methodology of analysing the patient's overall problems. By integrating mind-maps into their care plans, it encourages the student nurse's critical thinking and link information such as patient data and diagnoses together. This technique shows such effectiveness that the nursing students from the Front Range Community College have encompassed mind-maps into its nursing care plans since 1997.

In a study by Thorsten, Carsten et al. (2002) for designing a computer tool that supported cooperative work, the mind-map was compared with a whiteboard tool as a method for group problem solving. 45 students were split into groups of three and had to generate ideas on creative tasks such as improving airline safety using the mind-mapping and the whiteboard tool. More ideas were formed using the mind-mapping software. The cause was the hierarchical structure that was enforced in the mind-map. On the whiteboard, there was no defined structure or the grouping of ideas whereas on the mind-map, students were observed to have written headings or categories even before related ideas were written.

Similar to the merge of mind-maps and health care plans, the concept of applying creativity and non-linear thinking into a linear field was implemented with success in the mathematics department at the University of Duisburg, Germany. There has been positive feedback where “students who were not good in mathematics benefited from mind mapping” (Brinkman 2003) and that the user of mind-maps had aided them in organising information, allowing students to “see the structure of the respective mathematical knowledge”.

Planning is also another application that mind-mapping has excelled in. It has been incorporated by Mento, Martinelli et al. (1999) for their executive MBA program (EMBA) at Loyola College in Baltimore, Maryland. This is because mind-mapping allows creative thinking for business cases where students have to come up with innovative ideas and solutions to business problems. In particular, mind-mapping in the form of planning has been used for team management, a key component in the program. Students create action plans for team management. An action plan defines the goals, responsibilities and roles of each member of the team and the mind-map assists in the action planning process. By using a mind-map to manage and plan, it enables effective and organised teamwork.

From these selected text, it is clear that mind-mapping is an effective technique for harnessing the power of both sides of the human brain to foster studying, problem solving, critical thinking and memory recall. Traditionally performed with pen and paper, mind-maps have begun to migrate into electronic media with the computer age which is discussed in the following section.

2.2 Computer Based Mind-mapping tools

We have taken four of the popular mind-mapping tools (iMindMap, ConceptDraw, Inspiration 8, Mind Manager Pro) and have examined their methodology for creating and editing mind-maps. We then summarised the general characteristics of these tools and compared the pros and cons with the traditional style of mind-map construction.

iMindMap (2007)

The application starts off with a dialog where users can choose to open a saved file or choose an image for the central theme. Then the user is forced to add text to this image. The application then creates a blank canvas with the main idea in the middle. iMindMap creates mind-maps using a series of connectors linked together. The keyword representing the idea sits parallel with the connector that is drawn. When the user hovers over the main idea, a red dot appears and when the user clicks on the red dot, they can drag out a connector. By selecting on a connector, the user can then type in text. The keywords are then placed beside the connector in parallel as shown in Figure 4.

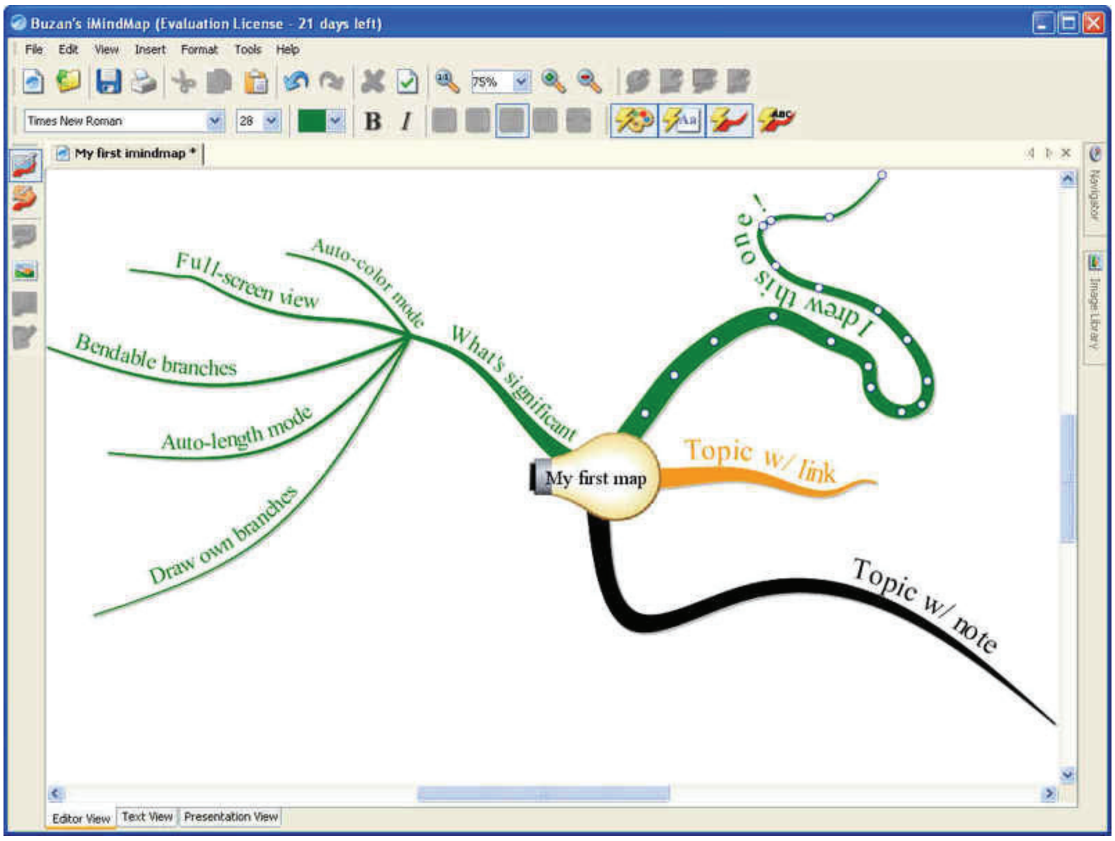


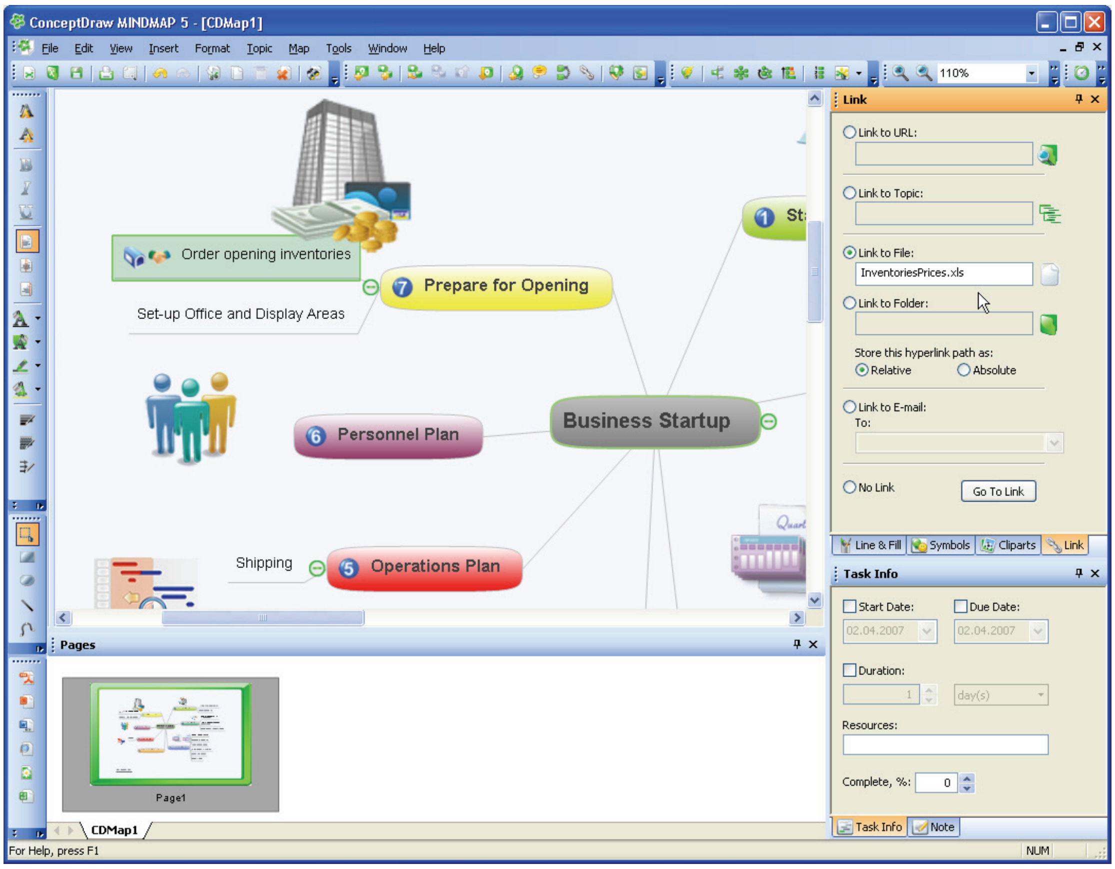
Figure 4 iMindMap

To make other connections, the user must hover over the tip of the connector, select the red circle and drag out another connection. All sub connectors branch out from this end point. To rename the connections, the user has to double click on the connector itself and not the word which is not intuitive. A small text box then appears that allow the user to make adjustments. If the text is longer than its connector, the connector automatically lengthens to fit the text.

To adjust the shape of the connector, the connector must first be selected, and there are two points to use to change the shape. To change the length of the connector, the user must hover over the end tip of the connector and select the blue circle that appears and drag the connector to another location. A connector can also be relocated to another branch in the mind-map by selecting the connector and dragging it on top of another. Subsequent connectors under the relocated connector are also shifted. However, connectors cannot be moved back under the main idea.

ConceptDraw (2007)

Upon creating a new mind-map, the user is presented with a blank canvas with a central node where the user has to select and enter the main keywords. The mind-map that ConceptDraw creates contains nodes with keywords linked together via connectors. Subsequent nodes are made by first selecting the node and pressing the "Subtopic" button located on the side bar. Keywords are inserted into nodes by selecting the node and typing in the text. To add a node with the same parent as the selected node, the "Topic" button is selected.



Link to URL:
Link to Topic:
(●) Link to File: InventoriesPrices.xls
Link to Folder:
Store this hyperlink path as: [✓ Relative] Absolute
Link to E-mail:
To:
No Link

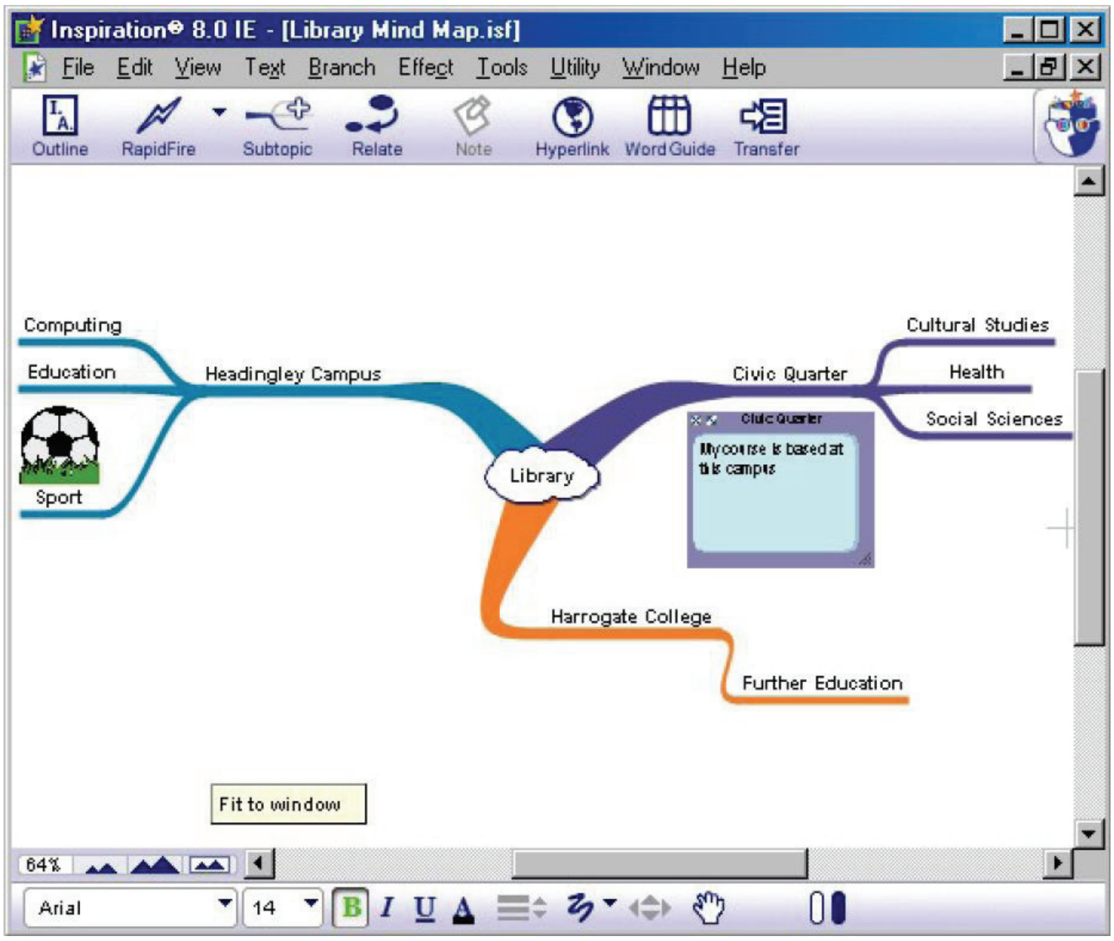
Task Info
Start Date: 02.04.2007
Due Date: 02.04.2007
Duration: 1 day(s)
Resources:
Complete, %: 0

Figure 5 ConceptDraw

Connections between nodes can be formed by dragging the node over another. More than one relationship between nodes can be made and requires the user to select the "Relationship" button and select two different nodes. Nodes can be moved to any location by dragging and dropping and its children will be moved accordingly. Also, a node can be moved up and down the map hierarchy by pressing the "Indent" / "Out dent" widget. When an object is selected, a properties window appears on the right hand side which displays all the configurable properties of the selected object. Things such as colour, line width, style transparency are all available for adjustment.

Inspiration 8 (2007)

The canvas is created with a premade central idea. The user can then change the text by selecting a node and inserting text. Similar to iMindMap, the keywords flow beside the connector rather than being situated at the end. Since the text is situated along the connector, should the text be longer than the connector, it is lengthened to accommodate the text. To add a connector, the user has to select the appropriate node and select the "+" button at the tip of the connector.



Inspiration® 8.0 IE - [Library Mind Map.isf]
File Edit View Text Branch Effect Tools Utility Window Help
Outline RapidFire Subtopic Relate Note Hyperlink Word Guide Transfer
Computing
Education
Sport
Headingley Campus
Library
Civic Quarter
Cultural Studies
Health
Social Sciences
Civic Quarter
My course is based at this campus
Harrogate College
Further Education
Fit to window
64%
Arial
14

Figure 6 Inspiration 8

Relationships between nodes can be added via selecting the “Relate” widget. To move the connector, the user just selects the appropriate connector and drag/drop. To relocate the connector under a new branch in the mind-map, the connector must be dragged over the new parent connector.

Mind-Manager Pro (2007)

The canvas is blank with the central idea situated in the middle. To create a node, the user clicks on the map or clicks on the topic/subtopic button and a blank node is created. A keyword can then be entered via the keyboard. To edit the keywords, the node must be selected and new text added which overrides the previous keyword. To form connections, the user must drag a node on top of another and a connection will be automatically made. However, the locations of the nodes are restricted to a set area such that a node cannot be placed too far from its parent.

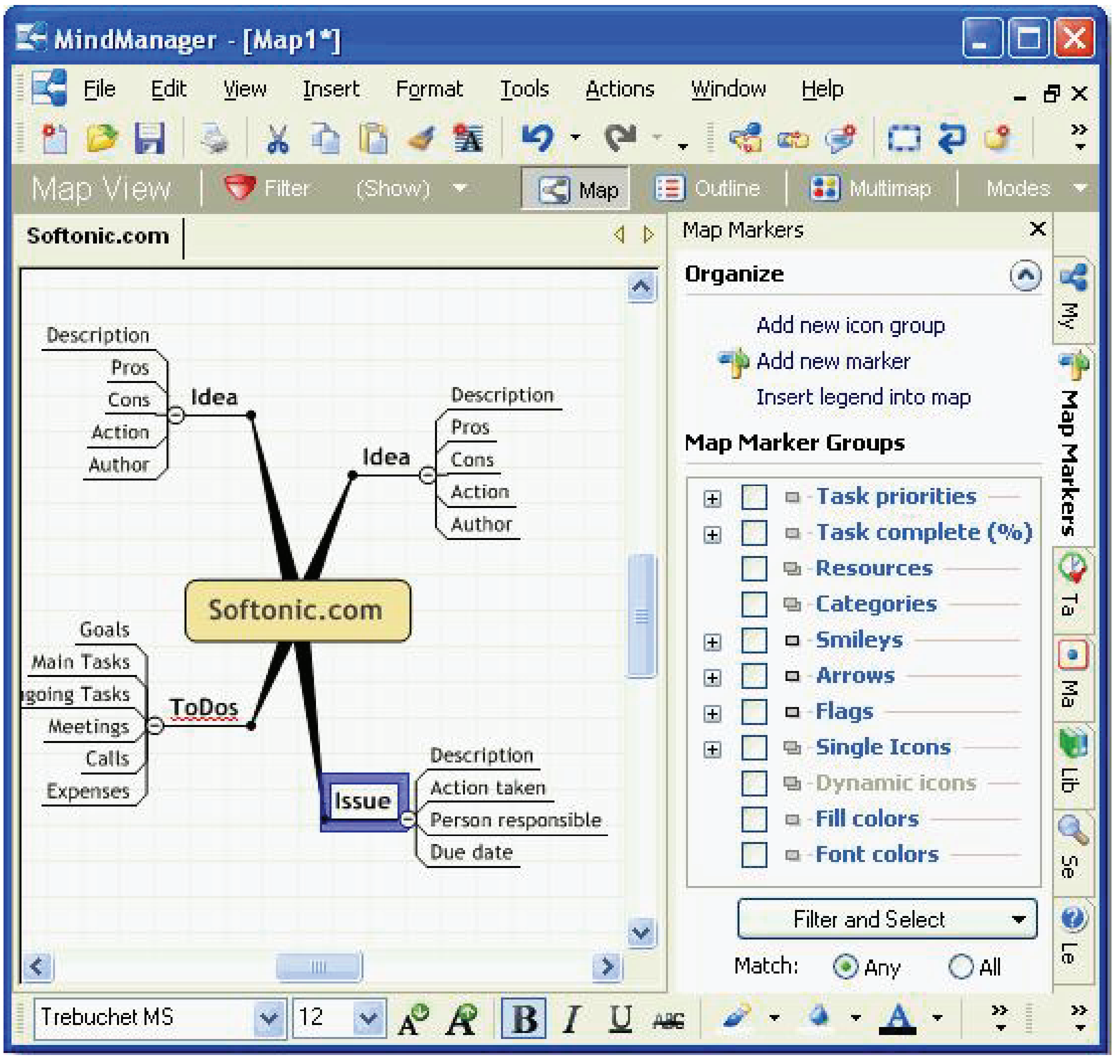


Figure 7 Mind-Manager Pro

To move the nodes, the user has to select a node and drag and drop. The position of the node is not always where the user wants it to be as there is a reflow algorithm that rearranges the other nodes. Relationships between nodes are made by clicking on the relate widget.

Summary of Tools

From these few popular examples of computer based mind-mapping tools, there are many similarities that they all share. For example the mind-mapping tools all possess the usual computer functionalities such as load, save, print, cut, copy, paste, undo, redo, spell check and search. In addition, they all allow a different view of the map, whereby users can view the map in a beautified tree structure. Lastly these applications all feature mind-map export in various formats (power point, PDF, word document, webpage and as an image).

Drawing a mind-map is a very similar process for all the mind-mapping tools. They involve pressing buttons to create nodes and sub nodes. These buttons are usually to the top or the left hand side of the display and are generally small to maximise the mind-mapping canvas. Inserting keywords require selecting nodes before using the keyboard to add text. Editing a map typically involves selecting a node or connector and dragging it to another location, although actions that shift branches up and down the mind-map hierarchy require pressing buttons or on top of other nodes.

It was observed that some of the applications constrain the user in some way or form. For example, in Mind-manager Pro, the layout of the nodes on the canvas were constrained such that nodes can only be placed three set distances away from the central node and are all aligned horizontally. Additionally, all sub nodes are attached right next to the node with no ability to move around the canvas. This grid layout has not been useful to the understanding of node and edge graphs (Ware, Purchase et al. 2002). For Inspiration 8 and iMindMap, the longer the keyword is, the bigger the connector becomes. This may affect the placement of the connector and cause overlaps that require relocation. Also, connectors that link nodes together in ConceptDraw, Inspiration 8 and Mind-Manager are fixed by the application and do not support user adjustment. This can create overlaps between connector/connector and connector/nodes which forces the user to be wary of their node placements during mind-map construction. In the case of Inspiration 8, the sub branches are all connected to the tip of the connector, which passively constrains the user to a particular area on the canvas as dictated by the location of the connector as the connector that branches off would definitely overlap if placed on the other side of the mind-map. In ConceptDraw, the presence of a properties sidebar for nodes and connectors interrupts the user from their thoughts and encourages the user to spend time in configuring the aesthetics of the components instead.

In all of these applications, there is this interchange between typing text on the keyboard and using the mouse. This occurs when creating new nodes, adding text, selecting widgets or editing the mind-map. Also, with the amount of options available to change for nodes and connectors such as in ConceptDraw, users can become distracted by arranging and beautifying their mind-map rather than constructing the map itself. It is these constraints and distractions that limit the potential of mind-mapping as mind-mapping is a fast idea generation process (Buzan and Buzan 2000) and much time is spent on the layout of the mind-map.

Yet despite all these shortcomings, one would think that the traditional method of mind-mapping would be used in preference over these computer based tools. However, the benefits of editing support and digital copy may outweigh the distractions that the human computer interaction with the mind-mapping tool creates. Sketch tools remove some of the distractions such as the interchange between a keyboard and the mouse. They have received positive feedback in the design disciplines and are investigated in the following section.

2.3 Sketch Tools

Sketch tools are applications which allow the user to freehand draw digital ink onto a canvas. They are generally low fidelity, representing concepts and ideas in a rough way. DEMAIS (Figure 8), a sketch tool for multi-media design and DENIM (Figure 9) a sketch tool for user interface design, are typical sketch tools. They all share the similarity of the need to replace paper yet allow the affordances of the computer.

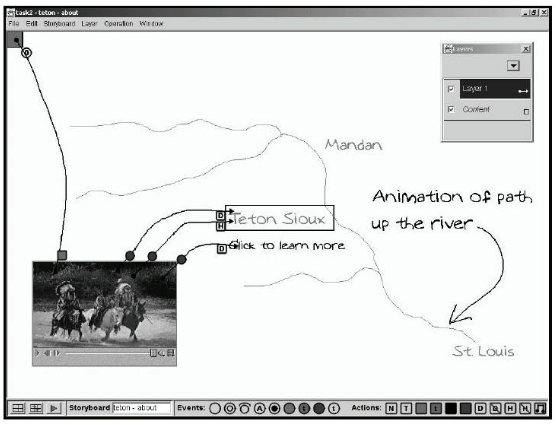


Figure 8 DEMAIS. Adapted from (Bailey and Konstan 2003)

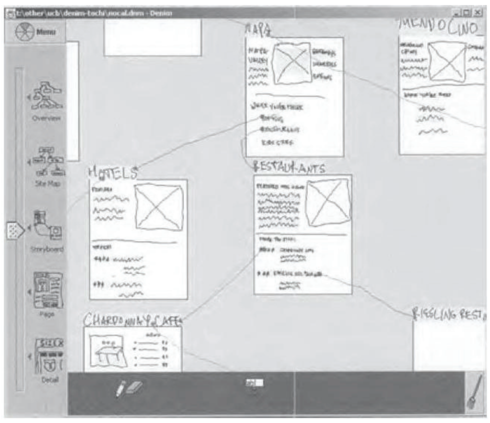


Figure 9 DENIM. Adapted from (Newman, Lin et al. 2003)

Coyette, Vanderdonckt et al. (2007) have found that designers in web design that use low fidelity tools to conceptualise ideas tend to reiterate over their plans and explore all the possibilities in their design phases. This is the opposite for computer based tools whereby users focus on a particular idea and refine it.

In the field of design, sketch tools are regarded favourably as a way to formulate rough ideas. Plimmer and Apperley (2003) performed a comparative study between their sketch based tools over conventional keyboard and mouse application. Plimmer and Apperley (2003) developed a digital whiteboard add-on to the Visual Basic IDE called Freeform. The whiteboard took in pen input and allowed the conversion from digital ink to VB forms. Twenty students were asked to create forms for two scenarios, one involving a book catalogue form, and the other, a dog registration form. They were given some sample scenarios to test their design. Results from the study reveal that on average, the students using freeform made more changes than the usual way of constructing forms with the mean number of modifications being 3.5 and 1.12 respectively. The rough sketch of Freeform provides quick feedback to the users and gives the impression of an unfinished product, which promote more changes to be made. There was positive feedback from the questionnaires with the participants enjoying Freeform more than the conventional design method. Of the twenty participants, seventeen would rather use Freeform as their design environment.

Bailey and Konstan (2003) compared DEMAIS to other forms of design tools for early multimedia design. As low fidelity tools lacked the behavioural aspects of design and high fidelity tools require more effort on the user’s behalf, DEMAIS was designed to bridge the gap between low and high fidelity tools. It enabled the user to sketch ideas while allowing user interaction by converting the sketches to a low fidelity prototype. This tool was then compared with a low (pen and paper) and high (Macromedia Authorware) fidelity product. Participants where split into 2 groups, designers and clients. The designers were given a multimedia application to construct and the clients were asked to critique the final product. From the results, it was documented that the pen and paper was ranked the most creative environment to work with. DEMAIS was close behind and that Authorware was ranked the least creative. This may be due to the fact that the complexity of the conventional computer tool did not properly express the designer's intentions. Also, DEMAIS was ranked the highest in exploring the behaviour of the application. The Authorware was deemed the lowest in that the designer spent a substantial amount of time configuring the tool as opposed to actual idea generation.

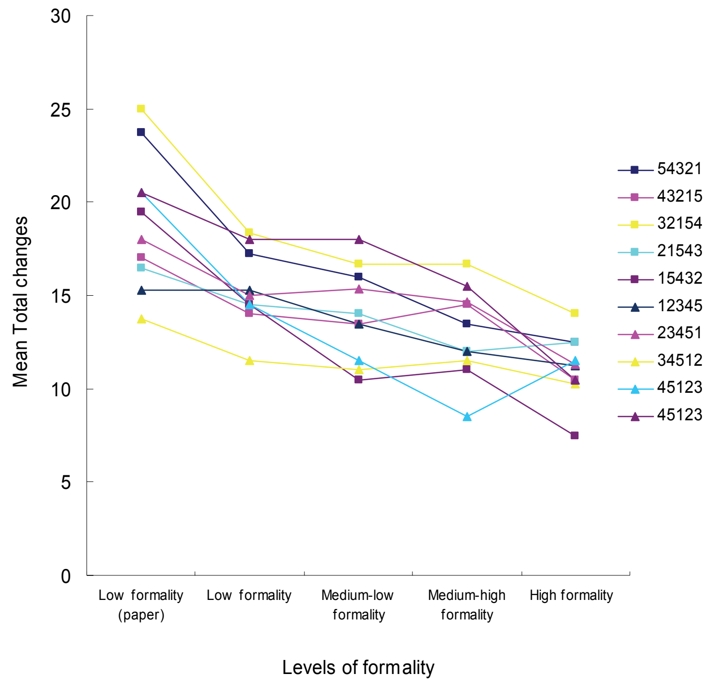


Figure 10 Graph depicting the decrease in the number of changes as formality increases. Adapted from Yeung (2007)

Yeung (2007) discusses the beautification process of computer based sketches and how their evaluation findings discovered that different levels of formality of the design in the early stages of form design resulted in varying amounts of modifications made. Participants were asked to make changes to an existing web interface with deliberate problems and the number modifications made were recorded for varying levels of formality. Again, the trends are similar to the findings of Bailey and Konstan (2003). As the formality of the web form increased, the mean number of changes made decreased as shown in Figure 10. Having a high formality look to the design suggests a prematurely completed plan and this is a problem that has been observed in web design (Newman, Lin et al. 2003). Designers and clients alike when presented with a high formality design are observed to focus more on the colour and overall look of the application as opposed to the behaviour and main concepts.

Mind-mapping is about idea generation, rather than design yet it shares some essential similarities with design, in that it is a fast idea generation process and visual representation. Hence, we hypothesise it would be more advantageous for the user to be focused on conceptualisation of ideas as opposed to being interrupted by the properties and aesthetics of the mind-map components like in early multimedia design. Informal sketch tools do provide some of the flexibility of pen and paper with the addition to maintaining the affordances of computer technology. Obviously there are disadvantages to sketch tools as discussed previously and studies have shown that it is not as good as paper (Yeung 2007). However, the advantages have significant weighting for our purpose as it may remove the disturbances that interrupt the brainstorming process. Hence the following section discusses the possible problems and challenges that a sketch tool may encounter.

2.4 Ink Support

In essence, Basic Inking is simplistic to implement for our purpose. This is due to the existing APIs already implemented that collects ink from a control with some method calls. However, it is the process after collecting the ink that is difficult, with many technical challenges. In the scope of our project, in order to support export and intelligent editing, the ink strokes must be correctly identified and categorised as discussed in 2.4.1. This is a multi part process, with the ink being separated into text and shapes. Note that the divided ink strokes by themselves lack contextual information. For example, letters of a word are not informative if treated as singletons, but do become meaningful when grouped together. Hence, there needs to be the grouping of ink in order to make sense of the user's input and section 2.4.2 investigates the problems regarding ink grouping. Finally the structure of the mind map can be extracted, as described in Section 2.4.3 which explores the technical problems of ink structure.

2.4.1 Dividing

When a user writes with a sketch tool, ink objects are generated for each stroke written. These strokes are just pure data as mentioned previously. In order to give these ink strokes meaning they must first be classified. Inkkit (Chung, Mirica et al. 2005) efficiently uses some good dividing techniques. Inkkit currently is implementing a divider engine created by Patel (2007). Many different ink attributes are analysed and incorporated into their ink divider. By using heuristics as well as existing works from Rubine's algorithm (Rubine 1991), specific values for attributes such as spatial location, ink intersections, temporal data, ink total angle etc are obtained to form their ink divider. In addition, the divider also takes into account the ink strokes that comes before and after for its recognition. The success rate at which the ink divider correctly classify drawing strokes and text strokes has been experimented to fall around 89.2% and 91.2% correctly identified strokes respectively.

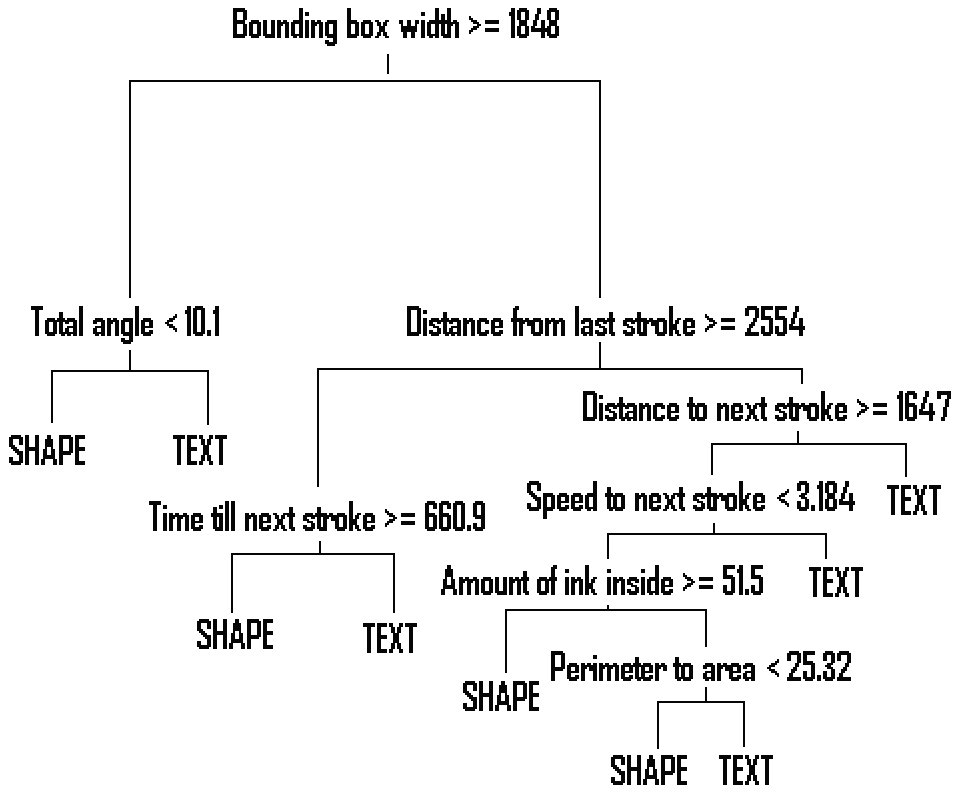

Bounding box width >= 1848
Total angle < 10.1
Distance from last stroke >= 2554
Distance to next stroke >= 1647
SHAPE
TEXT
Time till next stroke >= 660.9
Speed to next stroke < 3.184
TEXT
Amount of ink inside >= 51.5
TEXT
SHAPE
TEXT
SHAPE
Perimeter to area < 25.32
SHAPE
TEXT

Figure 11 Binary tree partitioning diagram. Adapted from (Patel 2007)

As shown in Figure 11, the strokes are sorted using a binary tree partitioning method. The attributes at the beginning of the tree are very general and sorts out the majority of the strokes. As the tree progresses downward, more specific measures are used such as ink speed and the amount of ink inside the stroke.

How Lank, Thorley et al. (2000) have done is to classify strokes using their sizes. They state that for UML components, the difference in size between them and the text that the user writes is significant enough to be used as a method of ink classification. Their methodology involves creating a histogram of the largest side of each stroke's bounding boxes. They have discovered that the histogram is bimodal, with the two maxima caused by the text and UML components. Hence their method of classification is a simple size check. Values that fall within a standard deviation or two away from the text maxima are classified as a text stroke. They claim that with this method, text have not yet been misclassified as a UML component. However, this cannot be hold true vice versa as the disadvantage to this methodology is that the smaller UML components become misclassified as text.

A paper by Sezgin, Stahovich et al. (2001) mainly discusses using two ink attributes to recognise ink. They focus on two attributes, the stroke speed and ink curvature of a stroke to identify simple shapes. The stroke speed is used to detect vertexes in the stroke as the stroke speed slows down on the corners of a shape. A combination of ink curvature and stroke speed is used to identify vertices, since shapes and stroke speed have difficulty with small shapes as the pen stroke may not achieve enough speed to register as a vertex. For curve strokes, they use Bezier curves to create a curve with the same start and end points as the stroke and using least squares to determine how well the stroke fits the Bezier curve. Although their methodology is effective in discerning shapes, the system may have some difficulty dealing with a combination of words and shapes which a mind-map will most definitely contain.

Alvarado and Davis (2004) have implemented an ink recognition engine that can be used for many domains. They have incorporated their engine into the two domains, family trees and electric circuit sketches. They firstly defined all the possible components present using a shape description language and the relationships between components beforehand. Then they used Bayesian networks to predict the strokes that the user is going to make or has made using these relationships. On their evaluation study, their system was able to recognise 77% of all symbols in the family tree domain and correctly identifying 62% of all objects in the electric circuit domain. However, they only deal with sketches of shapes in their context which may run into problems with a domain that contains a combination of text and shapes.

Sezgin and Davis (2005) uses Hidden Markov models to predict the possible outcomes of a group of strokes based on the ordering of strokes of a particular component. By using the stroke ordering as a means to identify objects and with training, it allows the user to freely write in his/her style. This ensures that the user is not restricted and distracted by having to follow a set of drawing strokes in order to describe a component. Also, the advantage of using such a system is that it is not as computationally intensive as other engines. Their methodology enabled them to recognise ink in polynomial time as opposed to exponential time needed by conventional methods which is a great advantage with an increasing number of objects. However, training is involved and a good training data set must be used.

Real-time recognition is one of our intended goals for this project. Junfeng, Xiwen et al. (2005) have completed work surrounding real-time recognition. They claim that real-time recognition engines currently available either are too simplistic or require large amounts of computational power. They proposed an incremental recognition system that has a high efficiency in sketch recognition. They take inputs from the stylus and extract a set of points that represents the user’s stroke. Once a certain number of points are obtained, the system uses these points in an attempt to identify and classify the stroke. As the user continues sketching, the system repeats this process but takes into account the previously recognised set of points and using probability to determine which shape the stroke fits into. The range of shapes is limited however, to clockwise/anticlockwise elliptical lines, straight lines and multiple lines.

Priest and Plimmer (2006) implemented an IDE add-on which allowed ink annotations on code. The system contains 2 components, linkers and freeform text. Linkers are ink strokes that points to a particular region of the code such as an arrow or highlights a code region such as an ellipse and are anchored onto a particular line in the code. In the context of code annotations, it is assumed that a particular region of the code would be linked before text is written, so the system has been designed with two modes, a linking mode and ink drawing mode. This effectively classifies the ink strokes into shapes and text automatically with no errors. Since the linkers are used to highlight a particular section of text, whereas lines are used to point out a particular line, there needs to be a differentiation between them in so that the comments point to the correct location in the code. Priest and Plimmer (2006) have come up with some interesting heuristics such as the start points and end points touching the left and right bounding box of the stroke, to categorise the strokes.

A technique that was employed by Gross (1994), based the concept of stroke ordering, uses grids to identify ink strokes. When a stroke is recognised, a 3x3 grid is overlaid on top and the order in which the stroke appears on each grid is used to compare with a dataset of possible shapes and text. Again the system must be trained beforehand with the possible shapes and text that may be used. This method of recognising was deemed a “low level recogniser” as only very distinct outlines can be differentiated. Similar shapes or letters such as ‘V’ and ‘U’ may have the same sequence and are outputted as potential items. Similar shapes could be recognised by the system’s recogniser if the grid size was increased (such as 4x4) but would require a substantially larger training set to properly recognise the ink. Gross (1994) also used this “low level recogniser” in conjunction with a “high level recogniser” enables similar strokes to be classified. An example would be the letter “O” and a circle. The low level recognisers cannot differentiate the difference but by parsing this information to a higher level recogniser, more complex comparisons can be made, such as if the stroke contained other strokes would most likely make it a circle instead of text.

Ink dividing can only classify individual ink strokes to a certain extent. This is because some ink strokes can have multiple meanings. For example, when a user draws a vertical line, it could possibly represent a connector, the lowercase letter ‘L’ or the numeral ‘1’. An “O” drawn by the user can stand for a bullet point, an empty radio button, the letter ‘O’ or a circle. The only way to differentiate these differences is by the context they are in. This requires examining surrounding strokes in the spatial and temporal sense which is covered in the following section.

2.4.2 Ink Grouping

There are two obvious drawing attributes that can be used for grouping ink; temporal and spatial. The assumptions are (based on the observations of people drawing) that people draw and write ideas or letters that belong together at the same time and in the same general area of a diagram. Of course people break both of these rules on occasions, either going back to a section later to review it or drawing a related piece of information further away from the first part. These two types of attributes have been explored in different projects.

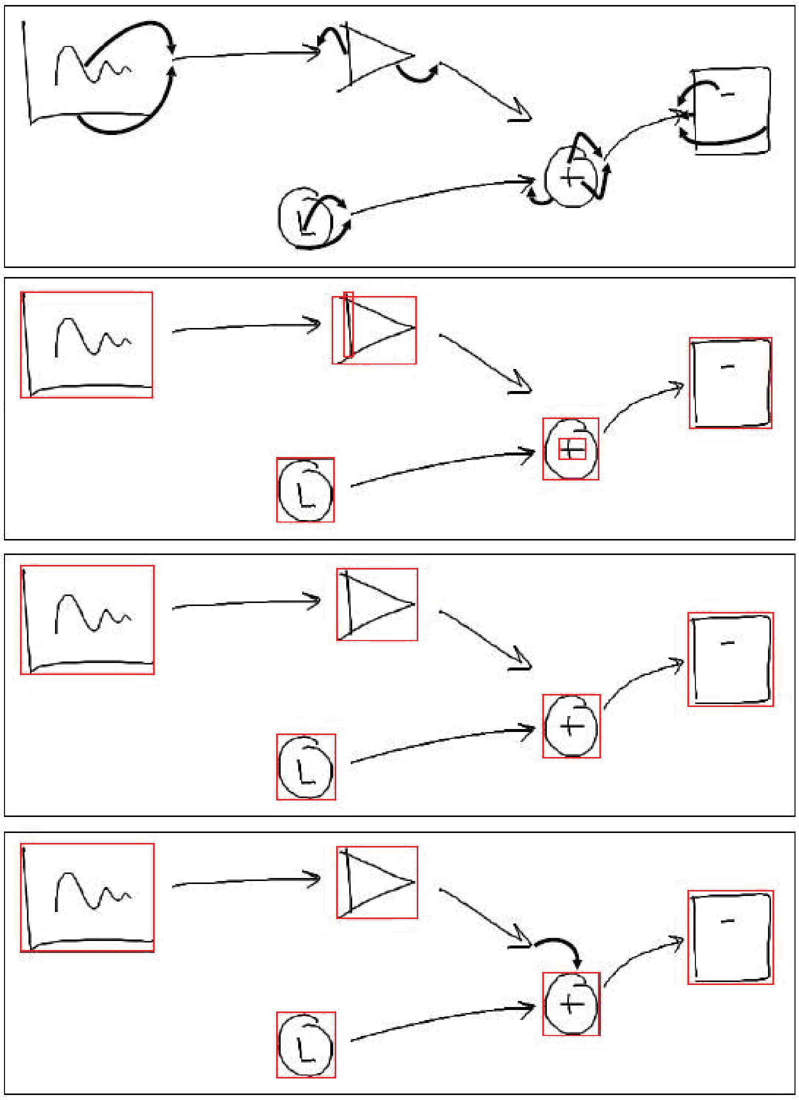


Figure 12 Example of a Simulink model. Adapted from Kara and Stahovich (2004)

Kara and Stahovich (2004) uses a spatial technique to group strokes. They have developed a system that recognises feedback control and dynamic systems for a Matlab package called Simulink. Their diagrams consist of components and arrows or links connecting to each other. They proposed a simple yet effective technique which involves the distance between connectors and the other ink strokes. As their grouping method is "on demand" grouping, when the user desires to group the ink, the system finds the closest connector for every ink stroke. The ink strokes are firstly grouped into clusters according to the closest connector. After that clusters with overlapping bounding boxes are merged together as a component may be linked via several connectors. However, problems identified included components being drawn too close together which ended up being recognised as a single entity.

In their context where all the components seem to be encased by an ink stroke, this method works flawlessly. There may be problems if the ink strokes of the same component gets split into separate clusters yet the bounding boxes don't overlap, resulting in two separate entities which should be one. Such a problem would happen in mind-mapping as text may not always be encased by an ink stroke. This may be useful for ink grouping on demand as the mind-map would contain complete components. However, with eager recognition (ink grouping on the fly), it may be too inefficient as all the ink strokes are computed and grouped over and over.

Chiu and Wilcox (1998) suggested using a agglomerate clustering technique. This is where each stroke is a cluster and the distance between each stroke is calculated and the closest two strokes forms a single cluster. The way the ink strokes are grouped is by spatial and temporal stroke data. For each stroke, the stroke timings and the distance with another stroke is compared. These values are then piped into a function and the two strokes that return the minimum value are grouped together. Each time a cluster is merged, a new level is defined. This process continues until there is only one cluster. The user can then select the level of clustering to choose the appropriate grouping of strokes.

Landay and Myers (1995), implemented SILK, a sketching tool for user interface design. The main grouping technique was a spatial process. They checked whether a component was situated inside or encompasses another component i.e a drop down combo box. They also checked for surrounding components which may be linked such as a tick box with text. Lastly they test for consecutive components of the same type such as a set of radio buttons or a bullet point containing a list of words. These methods reinforce the need for context grouping as not all ink strokes should be grouped together just because they are close to each other. We must look to the context of the sketch, with the domain in mind, to make specific context grouping decisions.

Ye, Sutanto et al. (2005) presents a novel way to group words, using a neighbourhood weighted graph and a cost function. The distances between each word is calculated, and the distance between the heights of the text strokes are used to give an indication of the probability of the text being on the same line. If these values fall within a range, they are grouped together. Next, a neighbourhood graph of these groups is determined. Two groups are deemed neighbours by having no ink strokes and a certain minimum distance between them. These data are then parsed into the cost function to determine if the neighbours belong to the same group. This process reiterates until the cost function no longer groups anymore. Tests carried out have shown that the algorithm converges rapidly which is a significant benefit for real-time processing.

2.4.3 Structure

Only when the said above recognition is complete can a structure of the mind-map be constructed. With ink dividing and ink recognition, there is no relationship between ink objects. In order to recreate the mind-map hierarchy the user has drawn, there needs to be a relationship between such objects. Such relationships are made by an overlap of a connector over a node as explained by Freeman and Plimmer (2007). Using this method, the relationships between all nodes can be recreated, which leads onto the construction of the mind-map hierarchy. There are many types of diagramming graphs such as undirected/directed graphs and organisation charts. As mind-mapping has a tree like hierarchy with ideas linked as a sub node to another idea, organisation charts have the closest similarity in structure to mind-maps. Of course, they differ from mind-maps as there are lone ideas with no relationships whatsoever with other entities and many parents to one child relationship. These instances present in organisation charts need to be filtered out to meet the needs of our mind-mapping tool. Freeman and Plimmer (2007) also discuss implementing a list to determine the diagram structure by making a pass through all its connections, and record each node with its relationship with other nodes. In essence the structure of a mind-map is similar to the organisation chart mentioned. They both have a root or starting location. The organisation chart, however, begins at the top of the document and traverses down the hierarchy with its children at the bottom. A mind-map begins from the centre of the document and radiates outward to the edges of the canvas. Since a mind-map does not have an ordering after the root (ie there is no order in the tiers of ideas), a list described by Freeman and Plimmer (2007) possibly meets the needs of our mind-mapping tool.

Only when the digital mind-map structure is recognised and a proper hierarchy constructed can complex and intelligent ink editing be supported. The following section probes in finer detail, the techniques of ink alteration.

2.5 Ink Editing

From a computer based tool, people expect the generic functionalities that are present in normal applications to be standard (such as copy, paste, undo, redo, load, save). This can be seen in Plimmer's (2004) work where a system for user interface design was built for Visual Studio. Upon the first evaluation of the tool, it was observed that the participants expected that a computer based tool would have standard editing functions. In particular, they commented that they would have liked an undo/redo functionality incorporated into the system. In a second prototype, the undo function was added and the second evaluation found that most of the users did in fact utilise this function quite a bit.

2.5.1 Ink Reflow

The low-fidelity environment that a sketch tool provides makes it an effective tool. Facilitating easy revisions during the design process helps create a better design. Since nothing is finalised, people can move ideas around the drawing space during this phase. This is similar for annotations on digital documents (Golovchinsky and Denoue (2002) described a technique that may be applicable to mind-mapping. They constructed Xlibris, an annotation tool for digital documents. When the user has circled a block of text and the digital document changes, the annotations should reflect those changes. An example is when part of the circled text gets deleted or more text is inserted. In their system, they find the new bounding box of the circled text and scale the old annotation's bounding box to this new value, effectively stretching annotation to fit the context. Such simple mechanisms may be used in the mind-mapping tool.

When nodes on a mind-map get moved, the connectors that link ideas together must also be moved with the node. It is important that the overall appearance of the mind-map is preserved. For instance, the shapes of the ink should be unchanged to maintain a sketch feel. Arvo and Novins (2005) discusses this type of ink reflow, where connectors are lengthened and shortened while maintaining the user's look and feel of the mind-map. They deal with directed graphs where arrows point to circles which are nodes. This has a large similarity with mind-maps but instead of circles the mind-map has unenclosed text and the links are directed but there are no arrows to show the direction as ideas are assumed to radiate from the centre. When a connector gets elongated, a set of points are taken from the original ink stroke and is interpolated in a linear fashion to its location on the same ink stroke that has been fully stretched.

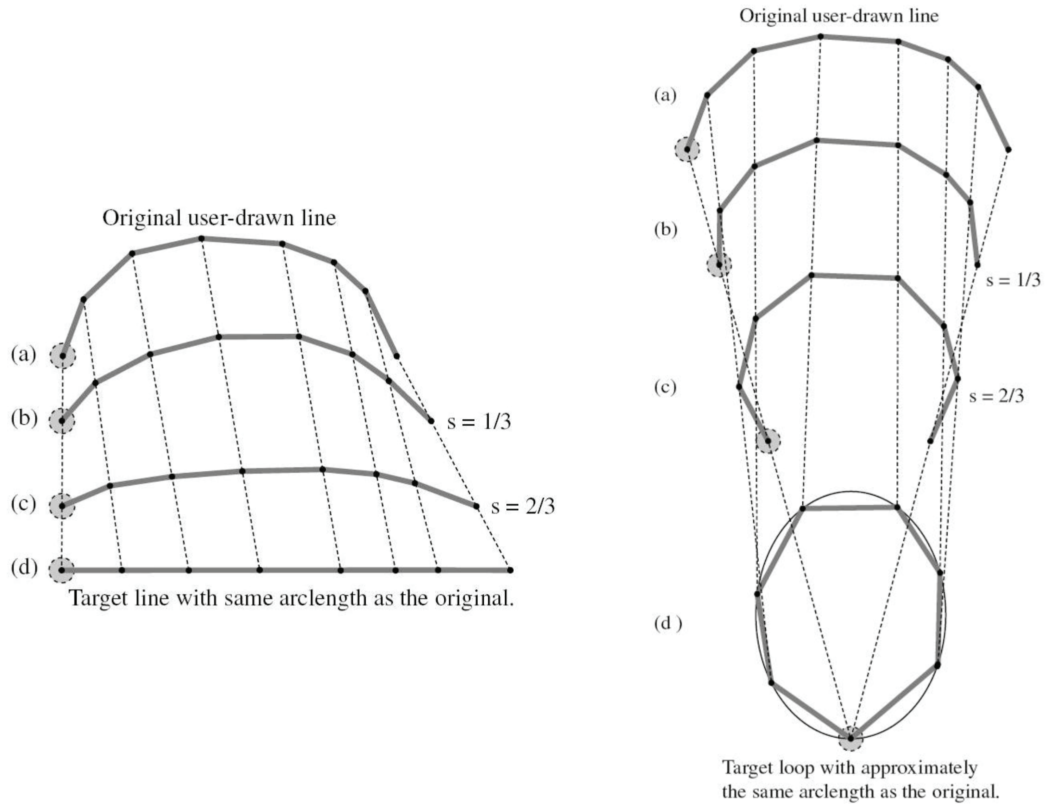

Original user-drawn line
(a)
(b)
(c)
(d)
s = 1/3
s = 2/3
Target line with same arclength as the original.
Original user-drawn line
(a)
(b)
s = 1/3
(c)
s = 2/3
(d)
Target loop with approximately the same arclength as the original.

Figure 13 Connector stretching & compression. Adapted from Arvo and Novins (2005)

Compression of connectors is handled in a similar manner, by linear interpolation to an elliptical arc as shown in Figure 13. However, by using the elliptical arc for interpolation, the ends of the connector no longer points into the centre of the node. Hence, attachment points were created to solve this problem. These attachment points are found by the intersection between an augmented line and the boundary of the ink. The connector is then resized relative to these points. Therefore, the connectors are allowed movement around the node and the connector continues to point into the node.

Reid, Hallett-Hook et al. (2007) implemented Arvo & Novins' (2005) ideas for ink reflow for undirected hand drawn graphs. However, there were some problems that they identified while building their undirected graph system. Such a problem is the overstretching of ink. This is where the distance between the nodes is larger than the ink stroke itself. Arvo and Novins (2005) had encountered this problem and their solution was to turn the connector into a straight line and extend it. However, when the connector was compressed again, it had lost all its hand drawn features and was now a perfectly smooth elliptic arc. Another problem encountered was the compression of connectors. When the connector becomes very compressed, the shape of the connector becomes a loop and this look makes it look artificial and out of place. Their solution was to transform the connector in the direction its being moved. A connector is first rotated so it becomes parallel to the x-axis, scaled in the x direction and rotated back. This method ensures the connector maintains its hand drawn appearance no matter how over extended the connector becomes. This technique is also removes the loop appearance that is shown when connectors become compressed by a large amount.

2.6 Summary

We have surveyed the literature on mind-maps, identified its key components and the benefits of using mind-mapping in areas that require fast idea generation such as brainstorming. Four computer based mind-mapping tools are surveyed. Their pros and cons are examined and the key trends present in these software tools are revealed. We then examine the benefits of sketch tools in design in comparison with widget based computer tools. And lastly, we have investigated ink support and editing strategies that have been implemented and are of some value for our application. From the related works, it can be seen that ink support in a sketch tool is a step by step process. Ink division is required to classify ink strokes and ink grouping to merge ink strokes into coherent entities. All this must be completed before a structure can be formed. In addition, sketch tools must provide the editing support users expect of computer applications. Ink reflow is a difficult process as the hand drawn appearance must be kept. The problem base and suggestions presented here will be factored in the design aspects of our system that is discussed in the following section.

Chapter 3

Proposed Design

This chapter discusses the design requirements of a sketch tool to support mind mapping. It first involves conceptualising the necessary features that are required for our system to be viable. To accomplish this, two personas and use case scenarios were constructed. Also in section 3.1, describes the informal study undertaken to learn more about how mind-maps are drawn. Section 3.2 describes the derivation of a list of functions that a mind-mapping tool can have from the mock scenarios. Section 3.3 onwards covers the technical challenges of the project: the ink support side of the system. Here the methods of ink dividing, grouping are discussed. The possible solutions already implemented and reviewed in the previous chapter are critiqued and approach selected for the proposed design. Section 3.4 delves into the challenges of ink editing and our design decisions to the challenges discussed in chapter 2.5.

3.1 Use Case

In order to fully comprehend the mind-mapping tool, we first constructed personas to simulate a user's perspective and model out how they would use the tool and what they would expect from using it. Here we have defined two likely candidates that would possibly use a mind-mapping tool:

Personas

John Smith is a 22 year old student studying in the School of Computer Science at university. He is in his final year of study. He has chosen to take a course in the field of computer graphics. At the moment he is having difficulty for this particular course. He is finding the theory side of this course rather challenging and is looking for a good way to learn his notes. He has tried all sorts of note taking techniques and highlighting lecture notes. He has heard about mind-mapping being a great technique for note taking and self study. He decides to give this technique a try.

Mary Jane is the head of department in the software engineering faculty at university. She has been recently informed that the extension to the engineering building is completed and that her department will to move into this new area within the next two months. As the head of department, she has the large task of relocating her department by that date. As relocation requires a substantial amount of planning, Mary has decided to plan out the various procedures using a mind-map.

Scenarios

In his lecture, John is using the mind-mapping tool to take his notes. He starts up the mind-mapping application. The mind-mapping application is ready and in ink collection mode as soon as it is opened, allowing John to immediately begin taking notes. As it is the 8th lecture, he writes down this down as his main idea and draws a circle around it. As the lecture progresses, he draws connectors and writes down the keywords like the name of an important algorithm. He proceeds to write a short note of what it does and links it with a connector. He thinks he should look up this particular algorithm for more information as he is still confused after the lecturer explained so he draws another connector and writes down the keyword "look in library" to remind himself to do some extra reading. Also, when he hears the lecturer warn everyone that this lecture would be examined in a test, John immediately writes down the keyword "testable material" and links it with a particular algorithm. As soon as he's written this down, John realises that he had linked this keyword with the wrong topic and so he changes the mode to ink select mode and chooses the connector and moves this branch to link up with the correct topic in his mind-map. To make his note stand out more, John switches to select mode and selects his note and then selects a red colour. Halfway through the lecture, he finds that he has run out of writing room on his screen. With a quick toggle of the zoom button, john zooms out of the canvas and revealing more blank space to take notes. He then chooses an area to zoom back in and continues note taking. At the end of the lecture he makes another note, reminding himself that there are online notes available for this lecture. He misspells a word and using the end of his stylus; he erases the misspelt word and writes in the correct spelling. Then he goes and saves the ink file.

Once John gets home from university, he wants to make a start on his report. So he begins to plan his report structure with his mind-mapping tool. He first writes the topic of his report as his main idea. Next he sketches the keyword "Introduction" in the top right corner links this with his main topic. Then he jots down what he's going to talk about in his introduction and writes more keywords and links them to his introduction keyword with more connectors. John continues methodically through all the topics he plans to discuss. Soon enough he has made an overall plan of his report. Satisfied with this, he saves his digital ink on his computer. As John also wants to base his report on his plan, he prints a copy out for reference. Also, he uses the tools export function, exporting the mind-map into a word document with the ideas on his mind-map converted into document headings and subheadings.

After hearing news of the department relocation, Mary immediately set about the task of planning to move her whole department over to the new engineering building extension. She first takes out her tablet PC and boots up her mind-mapping tool. Once opened, she writes in the centre of the canvas with the main keyword "Department Relocation" and circles it. Once completed, she begins to jot down key components of such a task, quickly writing the key words "Staff", "Furniture Movers", "Office Allocation", and "Administration". She then links each of these keywords with a line to her main idea. As soon as she is done, ideas start coming forth. Looking at a furniture moving business card, Mary scribbles down a phone number of the office furniture moving company and links this with "furniture movers". She remembers she left out the name so she writes the name under the phone number. Moving on, Mary then draws out a connector from "office allocation" and writes down "floor plans" to remind her to get hold of a copy so she can assign her staff rooms. Under the keyword "staff", she draws another connector and links it with another keyword "notify staff" to inform all staff of the relocation. On the idea of notifications, Mary realises she needs to inform the students also and so she goes back and makes the keyword "notify Students" and links it with "notify Staff". Mary quickly reviews what she has written so far and thinks that all the notifications belong together. She then proceeds to write the keyword "notifications" and links it with the central idea. Then she relocates the keywords "notify staff", "notify students" and "inform technicians", with their connectors pointing to "notifications". And as theses keywords are relocated, she expects the tool to prevent overlaps in ink. She continues to add more ideas to her mind-map and pretty soon the space on her screen fills up with many keywords. Mary then zooms out from the mind-map and zooms in on a blank area on the canvas and continues brainstorming. At the end of her brainstorming session, she emails a copy to the department, saves her mind-map and prints out a copy to stick on her wall.

In addition to these use case scenarios, we have decided to discover how people draw mind-maps by undertaking an informal study to determine how a mind-map is created from another perspective. Several participants are given an ink painting program and are asked to classify a group of dogs. A set of pictures and names of the dogs are given. The following figures are samples of the type of maps drawn by the participants.

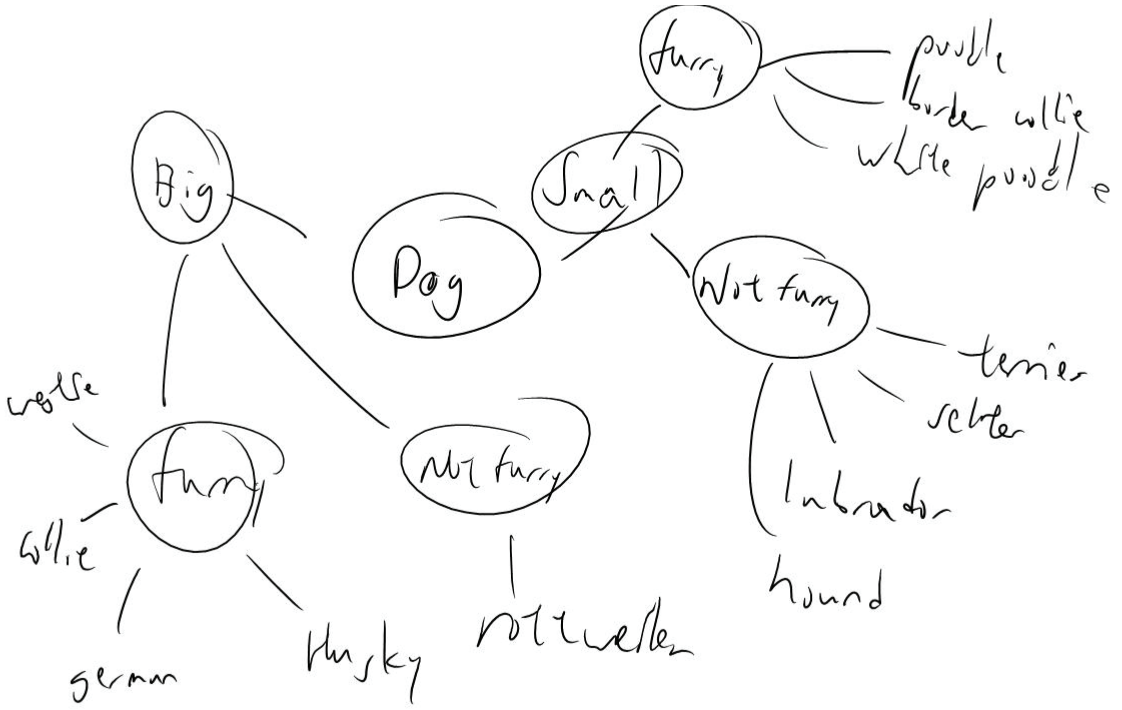


Figure 14 A sample mind-maps of dogs.

Observation of the user’s actions during the construction of the mind-maps in Figure 14 shows that the user does not always finish a train of thought (ie finish brainstorming all the ideas of a particular branch) and more often than not, they jump onto the next idea or keyword that is on their mind before coming back later. And that some users sometimes cross out words as there was no ink erase support for the stylus. Another important point is that some users used more than one stroke to draw a shape. Others use a stroke to visually make corrections/adjustments. For example, if a mouse up occurs accidentally before drawing a complete circle, the user uses another stroke to connect the ends of the incomplete circle. Also, connectors that the user draws may not necessarily be very long if keywords are clumped together and must be taken into account

The above scenarios describe the actions a mind-map user would carry out. From these examples, we have identified some requirements that must exist in our system. The use case diagram below shows the building blocks and use cases, for a working mind-mapping application.

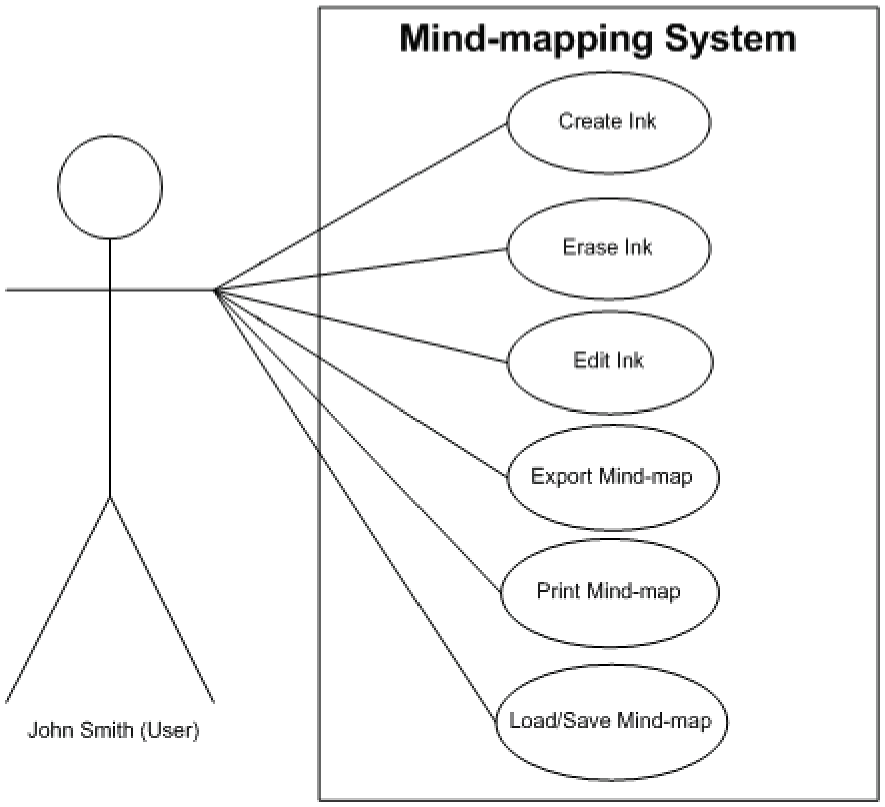


Figure 15 Use Case diagram of our mind-mapping software

3.2 System Requirements

From the use case scenarios and the informal data gathering for mind-map generation, we have expanded on the use case diagram, describing more in depth the utility that our mind-map tool should provide.

General Layout	
Interaction Style	Simple and low modality when possible. This is to preserve the informal and simple nature of the traditional pen and paper environment. This can be carried out by using a Tablet PC which takes in the inputs from a stylus.
Interaction Drawing Space	The drawing space should mimic a paper metaphor, with a substantial area of canvas space for the user to sketch. Also, to overcome the restriction of paper, the amount of drawing space should be very large.

Ink Recognition	
Ink Input	The system must be able to obtain ink strokes input data from a control (ie a drawing canvas). This data is then used for ink recognition and classification.
Ink Recognition	The system must be able to correctly classify the ink strokes the user writes given the ink data collected.
Eager Recognition	The system should be able to recognise the ink strokes as the user is writing. This is vital as it enables editing to be continuously supported.
Ink Grouping	The system should group all ink strokes correctly so to provide coherent and correct ink editing.
Structure Analysis	Given the classified and grouped ink strokes, the system must be able to use this information to determine the hierarchical structure of the mind-map.
Ink Properties	The system should enable the user to draw and edit ink sketches with varying colours and thickness. The use of colours and ink stroke thickness in a mind-map enhances memory recall, highlighting keywords of importance.
Ink Editing	
Resize	The system should allow any ink stroke to be resized.
Undo / Redo	The system should enable the user to undo and redo any editing/addition done to the mind-map.
Manual Ink Classification	Ink recognition may not be correct all the time. If recognition errors do occur, the system should allow the user to manually change the type of component an ink/group of ink represents.
Editing (Move + Reflow)	The user must be able to select an ink stroke or a group of strokes and be able to move it around the canvas.
Branch Collision Detection	The system should intelligently detect if ink objects overlap each other when moved and move them out of the way.

Miscellaneous	
Beautification	This allows the user to create a tidy version of their mind-map.
Zoom	The system should allow the user to zoom in and zoom out of the canvas.
Print	Should be able to print a hard copy of the mind-map.
Load & Save	The system should be able to save and reload an existing mind-map.
Export Capability	The system should allow the user to export the mind-map into another format.

Table 1 Table of mind-mapping tool requirements

With these general requirements, we now delve deeper into the technical design of our mind-map. The significant design areas of the tool are ink recognition and ink editing. The following discusses the finer requirements of each in detail.

3.3 Ink Recognition

Ink recognition is a vital part of the whole intelligent mind-map concept. By identifying each stroke, we provide the base for intelligent ink manipulation. To support a system that provides a smooth flow from creating ink to intelligently reflow ink, there must be "eager" recognition. "Eager" recognition is an ongoing recognition process where strokes are recognised as they are drawn. This is different to "on demand" recognition where the ink is only recognised when the user activates a widget. Kara and Stahovich (2004) have used "on demand" recognition in their system as the results of the recognition may interrupt the user during the design process. However, all the strokes must be sorted beforehand before intelligent ink editing can be accomplished. Therefore the "eager" recognition must be as unobtrusive as possible. On demand recognition should also be accessible to the user. Section 3.3.1 discusses the classification of ink strokes. The required components of a mind-map that needs to be recognised are defined. Section 3.3.2 discusses the importance of ink grouping and what classified strokes are required to be grouped together. And lastly, section 3.3.3 briefly goes over how the hierarchy of the mind-map is then found.

3.3.1 Ink Dividing

The components of a mind-map consist of nodes and connectors. The connectors are just single lines with no arrows. Keywords or ideas are known as nodes. These can be enclosed nodes or open nodes. Enclosed nodes are keywords surrounded by a shape such as a rectangle or circle while an open node is just made up of text. With these components defined, it is clear that our mind-mapping system should be able to differentiate shapes from text that the user draws. Moreover the mind-map application must further classify shapes into sub categories as different shapes represent different components. When a user sketches on the canvas, the system must first classify the ink into text and shapes (Patel 2007). Should the ink stroke be classified as a shape, it must be further categorised into connector or node shape (Priest and Plimmer 2006). In essence if the shape is a line, then the ink stroke is a connector. Otherwise, the ink stroke forms a node.

3.3.2 Ink Grouping

As mentioned previously, an individual ink stroke may not have any meaning, especially a text stroke. Hence we must group text strokes together to form coherent keywords. The system should be able to group text strokes and assign them to a node. As keywords may span lengthwise and downwards in the form of paragraphs, the system should intelligently group these effectively. Also, as a node may be unconstrained or enclosed, shape strokes that are part of a node must also be grouped together with the text strokes of the node. As mentioned above, the user does not necessarily sketch in a predefined order. They may switch back and forth topics and subtopics so we must not only just rely on temporal data to group strokes.

3.3.3 Ink Structure

In a mind-map, the structure is defined by the connectors that link nodes together. So there needs to be a method of storing the nodes that the connector links together. The ordering of the keywords in regards to the connector is also significant. We need to know which keyword links to which end of the connector. This is due to the hierarchical nature of the mind-map. When this is successfully accomplished, by systematically going through all these connectors and obtaining this data, an overall structure to the mind-map can be obtained (Freeman and Plimmer 2007).

3.4 Ink Editing

As mentioned above, ink recognition is required for intelligent editing. Only when all the ink strokes are properly recognised can correct ink editing occur. In section 3.4.1, the requirements for ink reflow are reviewed in detail. Miscellaneous editing functions are then defined briefly in 3.4.2

3.4.1 Intelligent Move

Ink move is defined as the movement of a single ink stroke or a set of ink strokes from one position on the canvas to another. Intelligent move is defined as the movement of an ink stroke or ink strokes with the context of the mind-map in mind. There are 3 types of ink reflow identified: ink stroke move, node move and branch move. Ink move is simply relocating a single ink stroke or a group of strokes. There is no intelligent reflow present as it allows users to move ink in an unconstrained manner. Examples include moving the horizontal stroke of the letter “T” to intersect with the vertical stroke or moving the dot of the letter “i” closer to the vertical stroke. Node move is the relocation of all the ink strokes belonging to a particular node (text and shape) on the canvas whilst the mind-map structure remains unchanged. This occurrence is when the user only wants to move a node to another location without affecting the positioning of other ink strokes. Lastly branch move is the relocation of a node and all the nodes children under another location in the mind-map’s hierarchy. There is intelligent reflow for both node and branch relocation. In both cases, the connectors that link the relocated node/s with its parent/children must also be relocated to maintain the mind-map hierarchy.

3.4.2 Simple Ink Editing Features

Our mind-mapping application should contain simple ink editing options as they enhance the mind-map experience. Stroke colour and thickness adjustments should be present as it allows the user to highlight specific portions of the mind-map. Stroke resizing should also be supported. Of course, the user must be able to undo and redo changes made to the canvas, whether after creating, deleting or editing ink.

3.5 Miscellaneous Features

As mentioned previously, users want standard computer based functions. From the list of functions that are applicable in our mind-mapping software, an important feature would be load and save to enable users to store digital copies of the mind-maps. To simulate a pen and paper metaphor, we require an erase feature present. It is preferable to have such a feature on the reverse end of a stylus as using the end of a pencil to erase is intuitive to everyone. A zoom feature was also deemed a necessity. A user's mind-map may span several screens worth of ideas and a zoom feature allows the user to review the complete map at the same time. Printing is also a necessity in the mind-mapping software in order to duplicate the mind-map on hard copies. Our tool must also be able to export the mind-map into other useful formats for manipulation. In addition, users may want a beautified version of their mind-map to be exported or printed so there must be an option to beautify the existing mind-map.

3.6 System Architecture

The following is a flow diagram of the back end of the system. Each ink stroke drawn by the user is first parsed to the ink divider.

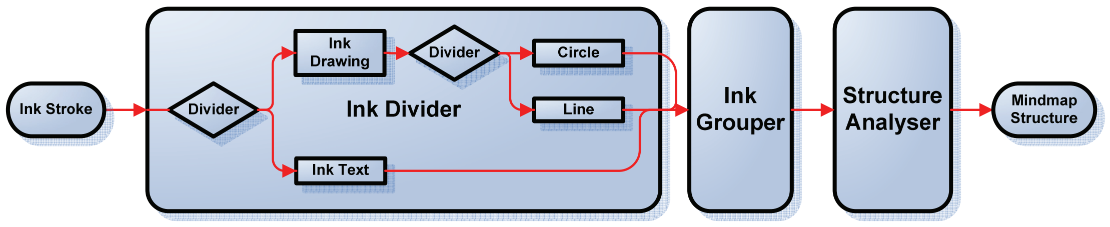


Figure 16 Mind-mapping tool system architecture

The ink is firstly divided into 2 categories, text and shapes. Upon stroke division, the ink classified as ink is further categorised as lines or circles. Then the ink stroke is then passed into the ink grouper which takes strokes that are close to each other and groups them together as a node. After grouping, a hierarchical order of the nodes is established. As new ink strokes are added, the software may alter the categorization of previously entered strokes.

3.7 Summary

In this chapter, we have simulated the actions of what a mind-map user would make in our software by creating a persona and creating a use case scenario. From this scenario and examples of mind-map drawn by users, we are able to determine the features that are necessary for a working prototype. We then progress deeper into the architecture of the mind-mapping tool, stating the necessary conditions that are present in ink division, grouping, reflow and structure. Lastly, we have looked briefly over the minor functions that should be included. With these design concepts in mind, we now proceed onto the implementation process of our mind-mapping system.

Chapter 4

Implementation

Having laid out the foundations of the mind-mapping tool in the previous chapter, we now go in depth into the implementation of our prototype mind-mapping tool, the Intelligent Mind Mapper (IMM). The tool is programmed in C# using Visual Studio. The reason for this is that the .NET languages have digital ink support libraries and C# has a well established knowledge base and support. Every aspect of the IMM is discussed from unsuccessful ideas to the techniques used in the prototype. Figure 17 shows the user interface of our prototype mind-mapping application.

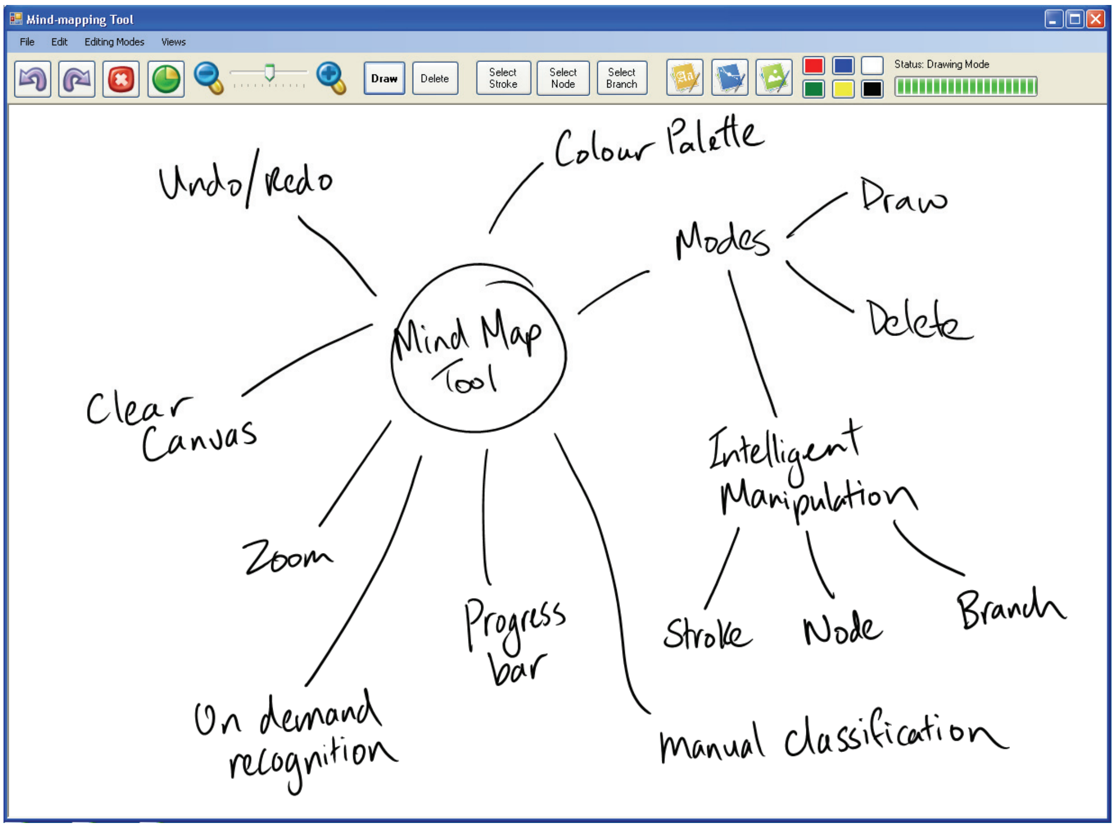


Figure 17 Our mind-mapping software functions

It consists of a large white canvas for users to sketch and a small menu strip across the top of the program. From left to right, there are the undo and redo buttons, a canvas clear feature that wipes the canvas clean. There is the divide function to allow on-demand recognition. The user can zoom in/out on the canvas via clicking on the magnifying glass icons or dragging the bars for varying degrees of zoom. The software has three modes: Draw, erase and three different selection types for intelligent manipulation of the mind-map. The user's cursor changes into an eraser, pen or select tool depending on the mode. Lastly there is a progress bar that indicates when the program is busy with the recognising the ink.

↶ ↷	Undo / Redo
✖	Clear Canvas
◔	On Demand Recognition
🔍− ▭ 🔍+	Zoom
Draw Delete	Draw / Erase
Select Stroke, Select Node, Select Branch	Stroke Mode, Node Mode, Branch Mode
Aa ✎ ✎	Manual Classification (text, connector, circle)
🟥 🟦 ⬜ 🟩 🟨 ⬛	Colour Palette
Status: Ready ▬▬▬	Progress Bar

Table 2 Table of user interface functions

Section 4.1 examines the considerations that are made in regards to the tool layout. Section 4.2 reviews the backbone of the system, the ink recognition engine implemented to successfully categorise the ink strokes. Our novel ink reflow techniques that allow intelligent ink manipulation is discussed in detail in section 4.3. Then the miscellaneous functions are explained in section 4.4.

4.1 Interactive drawing space

As mentioned in section 3.2 of the design chapter, we require a large interactive space for the user to make sketches. Hence for our prototype, there is only the standard menu and a

control strip at the top of the application as shown in Figure 17. The rest of the tool remains widget free, presenting the user with maximum canvas space to sketch.
An effort has been made to imitate a pen and paper environment: to keep the interface simple and with as few modes as possible. We use a Tablet PC which enables the user to pick up a stylus and sketch on the mind-mapping interface just like a pen. We also make sure that all the actions performed on our application can be made solely with the stylus. This enables us to remove the keyboard and mouse inputs, effectively cancelling the interruptions caused by switching between keyboard, mouse and stylus input. In addition, it allows the tablet PC to be folded up like a clipboard, imitating our pen and paper metaphor as shown in Figure 18.

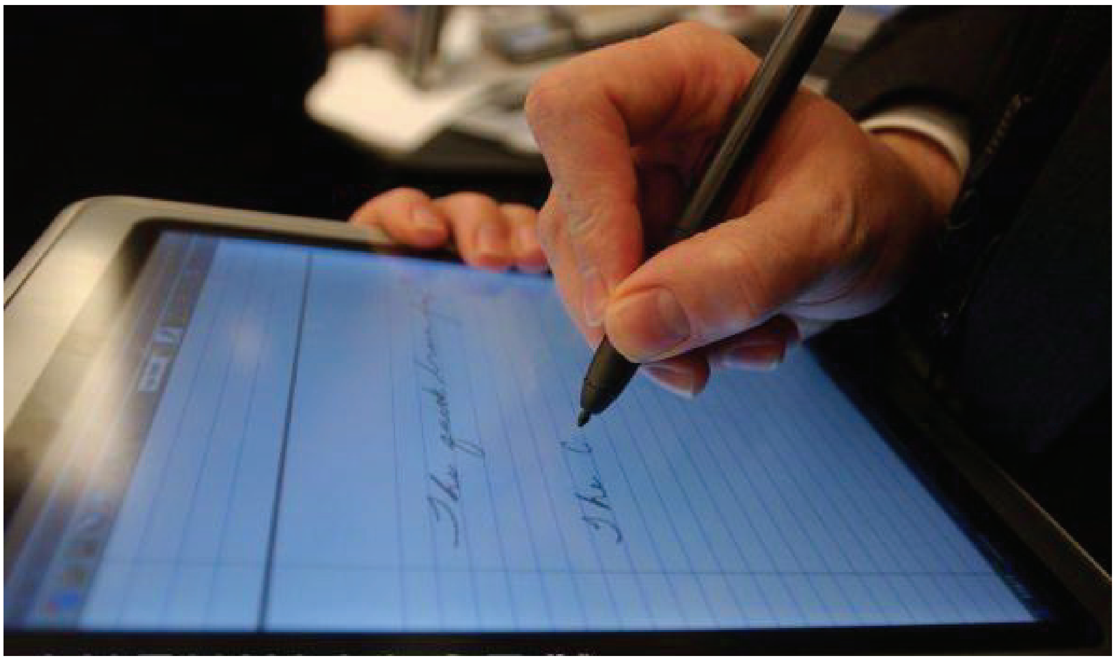


Figure 18 Tablet PC in clipboard configuration

The user uses the stylus and sketches on the screen of our IMM tool. As the user writes on the canvas, the ink is being recognised on the fly in preparation for ink editing. To edit the ink, the user changes to one of the available modes. The user enables the "Select Stroke" mode and selects one or more strokes. The user can then move and change the stroke properties (such as ink thickness, ink colour, scale). There is no intelligent editing involved in this mode so the user is unconstrained. To move a single node around while keeping the other nodes intact, the user changes to "Select Node" option, clicks on a node and is able to move the node to a new location. This mode has intelligent ink editing and connectors that are linked to the selected node are reflowed. If the user desired to relocate a whole branch in the mind-map hierarchy, the "Select Branch" mode would be used. The user selects the top most node to relocate and the system automatically includes its children. There is a line which indicates where the branch will latch onto as the user moves the node. When the user is satisfied and releases the node, the system will then reflow the node in an outward direction.

Gestures could have been used for specific commands such as undo and redo. However, we have chosen to not use gestures for editing as it would be too confusing for users and would require additional training for our study.

4.2 Ink Recognition of drawing ink

Ink recognition is one of the main constituents of our IMM tool. A robust ink recognition system is the key to effective intelligent ink editing. Section 4.2.1 describes the ink dividing system used to categorise a user’s ink strokes. Section 4.2.2 then progresses into ink clustering and the techniques used to generate good groups. Lastly, Section 4.2.3 concludes the ink recognition section with the identification of ink structure after ink classification.

4.2.1 Dividing

Dividing the ink into text and drawing is the first process that must be looked at when the user sketches on the canvas. There are endless possibilities as to the representation of each stroke but in the domain of mind-mapping, only writing and two shapes are required so we have restricted the scope to three types of components: a circle that represents an enclosed node, a line to represent a connector and text to represent an open node.

Patel (2007) has implemented an effective ink stroke divider that divides ink into text and drawing strokes with a higher rate of success than the Microsoft divider. We have acquired this divider and implemented it for our IMM tool. The divider uses a tree based partitioning system where a stroke is compared against specific ink features. This divider looks at the features of each stroke including time and distance between it and the previous and next stroke. The implemented divider has been tested for its effectiveness on mind-maps and we have indentified a number of cases where it has failed for our particular type of diagram. We have extended the divider to add shape differentiation for connectors and circles and to operate in eager mode (as the sketch is being created), to better differentiate connectors from similar letters. Further refinements to the recognition are discussed in the grouping section 4.2.2 when the context of the ink is better understood.

As mentioned in the system requirements, 'eager' recognition is vital for our mind-mapping application. This is because of the intelligent editing that takes place as the mind-map is being created. A stroke event will trigger the system which pipes the last stroke in its list to the ink divider for ink recognition. Recognising each stroke as it is written allows the user to make intelligent adjustments without having to manually execute the ink recognition feature. Also, to recognise the complete map at once would take time. This incremental recognition gives the impression of zero waiting time when the user wants to intelligently edit the ink.

Patel's (2007) divider assumed that the sketch is complete when recognising ink. For any stroke, the divider takes the preceding stroke and the stroke following it and uses them for classification. For the first and last stroke, the divider only takes the stroke before or after it respectively. We have observed that the first and last strokes are often not recognised properly. This is because when dividing these strokes, there is less temporal and spatial data to work with (i.e data from the previous or next stroke). This is not the divider's fault as it does divide the ink effectively given the limited amount of information. For eager recognition, the most recent stroke only has the previous stroke's data for categorisation. To maintain the same level of correct categorisations in eager and on demand recognition each stroke (except the final stroke) is classified twice, once when it is received and again when the next stroke is received as at this point the classification tree has all necessary information. On demand recognition is supported for system testing and is in essence identical to eager recognition. The system merely cycles through its list of all the ink strokes on the canvas and passes each through to the divider to be re-categorised.

As the ink divider that we implemented is a generic diagramming ink divider, strokes classified as drawing strokes must be further split into lines and circles. We have extended the divider, continuing on from where it ends and inserting our own heuristics for shape classification. Once the ink stroke has been indentified as a shape stroke, it is further analysed and divided. Our first implementation of shape division is loosely based on Priest and Plimmer (2006). A feature that is first examined is the proximity of the start and end points of any stroke. A circle is formed by connecting the starting point and end point together unlike a line whose start and end points would generally be far apart. Hence, to sort a line from a circle, the starting point and the end point of the stroke is extracted and the distance between them is determined. If this distance is very small (i.e the points are close together) then the stroke has a high chance of being a circle. Conversely, if the distance is large, the stroke is likely to be a line. After some testing, the threshold value of 950 Himetric units is used to sort the lines from the circles. We used the ratio of the distance between the start and end points over the largest side of the stroke bounding box in addition to the measure above to separate the shape strokes. A line would generally have a ratio close to one and a circle would have a smaller ratio.

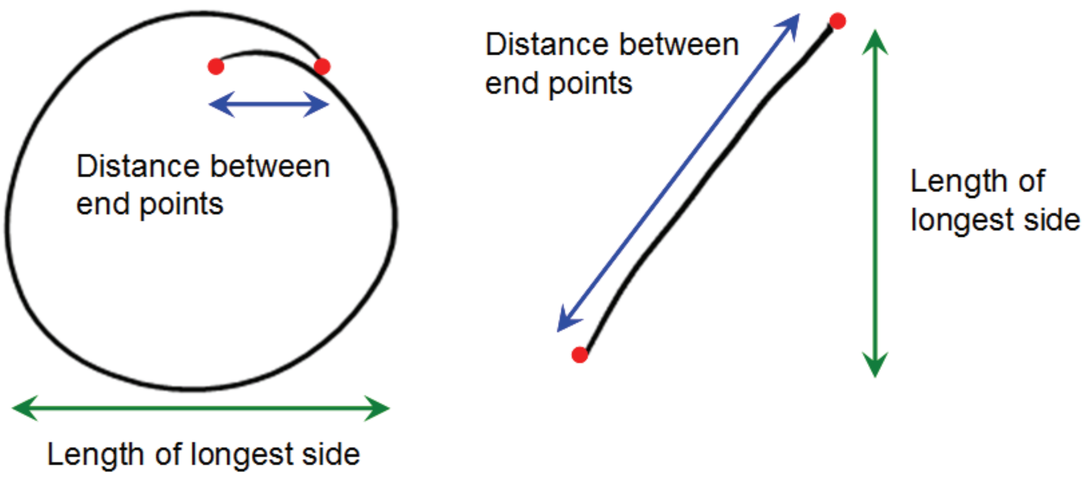
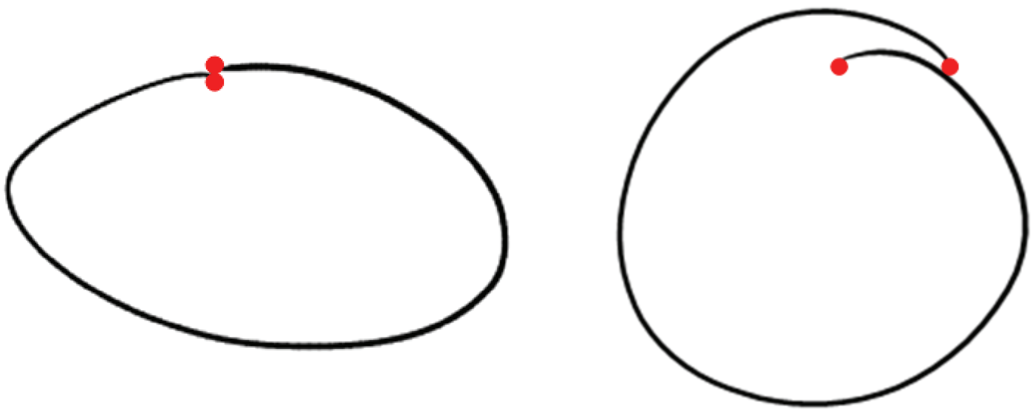


Figure 19 Shape dividing heuristics

This method was a naïve method of separating the ink. We assumed that the circles were drawn fairly accurately. Flaws in the method began to show when a circle was drawn quickly. A hastily drawn circle resulted in the start and end points of the circle being further away from each other, passing the threshold. A compromise was then made on the threshold but this caused the lines to be classified as circles more often.



Ideal Circle with endpoints close together Circle endpoints too far apart

Figure 20 Problems with heuristics

Hence on our second attempt we have tried using different heuristics. Instead of using the stroke start and end points, the total stroke length is calculated and compared with the diagonal of the bounding box. As a circle is defined by its diameter multiplied by PI, the distance is always longer than twice the length of the stroke bounding box diagonal. This is shown in Figure 21. This method neatly fixes the quickly drawn circle problem, preventing their misclassification into lines.

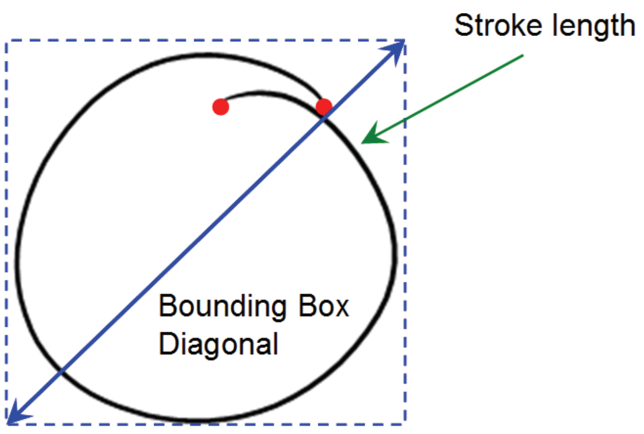


Figure 21 New heuristics for circle differentiation.

We have found that there are a number of problems with differentiating connectors (which are typically straight lines) from some similar letter strokes (‘L’, ‘F’, ‘T’ and the numeral ‘1’). Firstly connectors are classified as letters because the bounding box width of the ink stroke is used at the very top of the decision tree. In generic diagrams, this measure is an effective mean of sorting out the shape strokes as most text strokes do not have a very wide bounding box. However, this proves to be a problem in the domain of mind-maps. As the mind-map connectors radiate outwards, connectors that are close to vertical, pointing upwards or downwards, have a small bounding box width. This causes the strokes to progress down the left path of the decision tree and become misclassified as letters. To rectify this problem, the diagonal of the stroke bounding box is used instead of the width. This enabled vertical connector lines to pass the threshold and advance down the right path of the decision tree.

As a consequence of this change, text strokes resembling a straight line become categorised as connectors. The culprit strokes that get misclassified are the letters ‘L’, ‘F’, ‘T’ and the numeral ‘1’ shown in Figure 22.

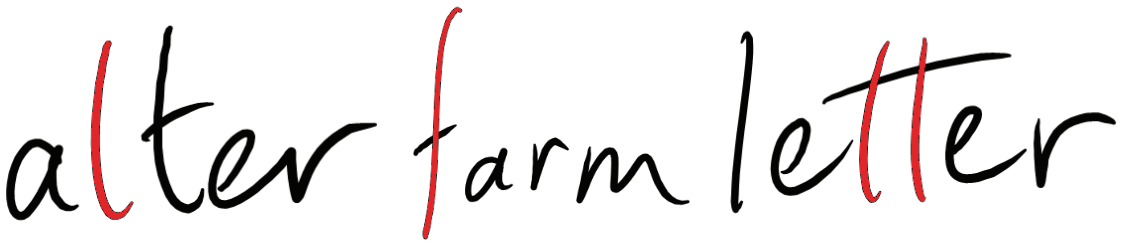


Figure 22 Problems with the divider (Red strokes identified as connectors)

We introduced two new rules to reduce these errors. Slightly curved strokes such the vertical of the letter 'f' are discriminated from connectors by their ratio of the stroke length over the distance between the start and end points of the stroke. If the ratio is less that 1.2 (meaning the line is relatively straight), then the stroke is classified as a line stroke else it is text. However, depending on the user's hand writing, the lines and letters sometimes cannot be differentiated. For example, a line and the lower case 'l' cannot be differentiated. Hence we have created another rule that takes into account the size of the stroke. If the stroke is of considerable length, then it is most likely a connector. The IMM tool checks for the maximum side of the stroke and if it exceeds 2500 Himetric units, it is classified as a connector else it is a text stroke.

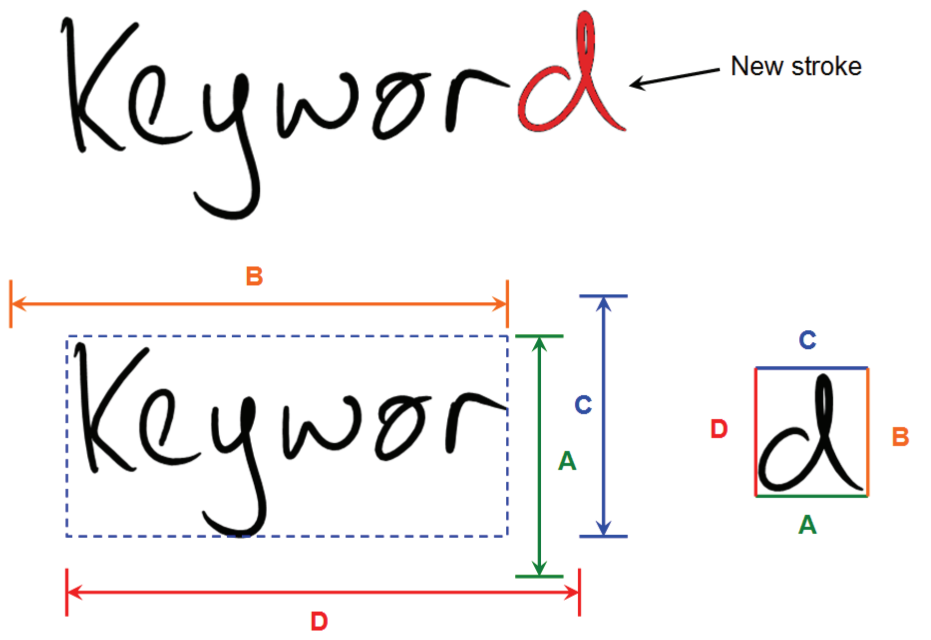
Another problem that arose is the letter 'O' becoming commonly misinterpreted as a circle shape since it fits all the requirements of a circle. When a user writes a heading with a larger size, the 'O' is often than not be interpreted as a circle instead. As the letter and the shape share the same properties, the only distinguishing feature between the two is if there is ink inside the stroke. A circle that represents an enclosed node will ultimately contain ink strokes while a text stroke will generally not. However, since a circle may contain no text inside when it is drawn, this feature can only partially distinguish these two types of strokes. A solution to this was to comprehend the context of the stroke in relation to the other strokes on the canvas and is covered later in context grouping.

Even with our measures, it is still difficult to understand the intentions of the users when they sketch on the canvas. The measures are not 100% perfect and strokes are still misclassified. The ink strokes can be better understood if we know the context they are in. This is where the context grouping plays a vital part in correcting these mistakes and is discussed in the next section.

4.2.2 Ink Grouping

Ink grouping is the clustering of related ink strokes together to form a coherent entity of strokes that correctly represents the user's intentions on the mind-map. Grouping is divided into two parts: the grouping of ink strokes to form coherent words and the grouping of words and shapes to form nodes.

In ink grouping, once a stroke has been identified, it is grouped straightaway. If there are no nodes, the system automatically creates a new node and stores the text stroke in the object. If there are nodes present, the IMM tool first determines if the stroke resides inside a circle (i.e an enclosed node). If it does then the stroke is added to that node. If not, the system looks for the closest node on the canvas and compares the distance between them. If the stroke is not close to any of the nodes, then it has a high chance to be a separate keyword and thus a new node is created for that stroke. The grouping algorithm used to group the strokes contains two main parts. Firstly, the system determines if the node lies on the same plane as the ink stroke. This is carried out by comparing the y coordinate of both the stroke and node bounding box top and bottom sides. A stroke is considered on the same level as the node if the y coordinates of the top side of the ink stroke lies between the node's bottom side and top side – 200 Himetric units as shown in Figure 23. A similar rule applies for the bottom sides, if the y coordinate of the ink stroke bottom lies within the node's top side and the bottom side + 200 Himetric units. If the either the tops or bottoms of the stroke's bounding box lie within these ranges, it can be concluded that the stroke and node lie on the same plane. The reason for taking either the top of bottom value but not both is due to the letters in the alphabet. Letters such as "h" and "d" have a small top bounding box value and letters "p" and "y" have large bottom bounding box value (as the co-ordinate system begins in the top left corner). If the top and bottom bounding box values were both utilised, then these strokes would not be classified as being on the same plane as the node. Once the stroke is confirmed to lie on the same level, it is determined if it lies within a certain distance in the horizontal plane to the node in question: 1000 Himetric units to the left or right side of the node. If both these conditions are true then the stroke belongs to that particular node and they are merged together.



The sides of the stroke (represented by a letter) are checked to see if they lie within the limits of the corresponding area of the node. A & C check if the stroke is on the same plane and D & B checks if it is on either side of the node.

Figure 23 Grouping algorithm diagram

Informal testing suggested that the preliminary values for the algorithm's top and bottom threshold were a little too strict. The variation of the positioning of letters in a word (as there were no underlines to guide the user to write on the same level) meant that some ink strokes were not grouped properly. Increasing the leeway to 400 Himetric units meant that more strokes were successfully grouped. This was the reverse for the unit threshold for the second measure, the proximity to the sides of the node. The 1000 Himetric unit threshold value was excessively large. Therefore the ink strokes that were drawn too close to the node would become incorrectly incorporated. This value had to be reduced to 500 Himetric units resulting in much improvement to the grouping result.

Also, it was found that by comparing the ink stroke with the latest created node, the grouping process becomes more efficient. This is due to the fact that the most of the consecutive strokes that the user makes are strokes of a keyword that all belong to the same node. By comparing the stroke with the latest node first, the majority of the time the system computational time is saved from comparing with the other nodes on the canvas. However, for new nodes it does not increase performance as all nodes must be checked in that instance.

In the previous section, the ink divider misinterpreted a selection of text strokes as a shape strokes due to similar ink features. But by knowing the context of the stroke we can confidently identify its true type and apply corrections. The misclassifications of strokes are mainly: text strokes that are categorised as a connector (‘l’) or circles (‘o’).

There are three occasions we have identified where a text stroke becomes identified incorrectly as a connector. Incorrect classifications of letters that have been classified as connectors occur with the ‘connector’ residing within a node, partially inside a node and outside the node’s bounding box shown in Figure 24.

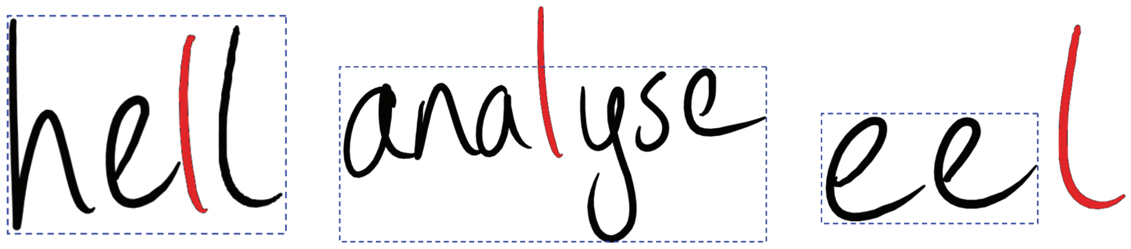


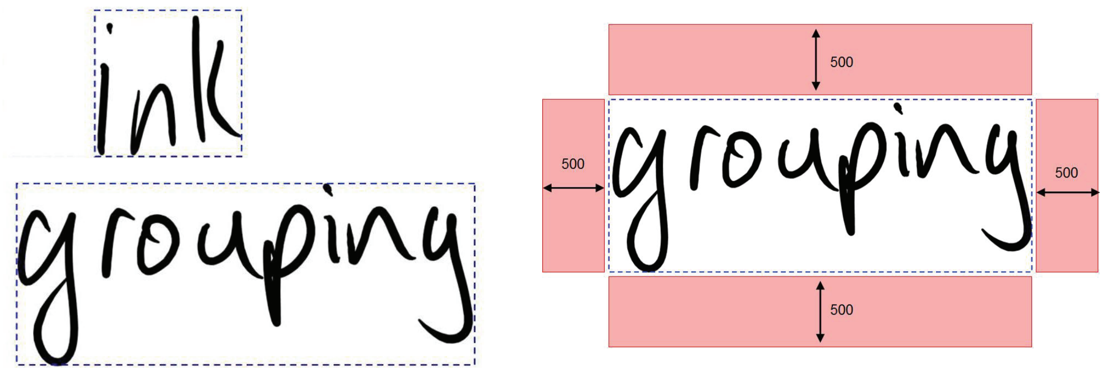
Figure 24 Instances of reclassification (Red strokes identified as connectors)

To remedy these misclassifications, the system first checks if the connector is completely inside a node. As mentioned earlier, all ink strokes that reside within a node are considered text. If the ink is not completely inside the node bounding box, the connector is made certain that it lies on the same plane of the node and then a % hit percentage test is used to determine the percentage of ink in the node bounding box. For our purposes, a 30% hit threshold was used. All strokes with more than 30% of their ink inside that node are listed. If the target connector is one of the strokes inside the list, it indicates a misclassified text stroke and is corrected accordingly.

If the connector is neither of the two instances, the system considers the last instance of the misclassified stroke: a connector that lies outside the bounds of a node. The system first determines if the connector is level with the node. The orientation of the connector is then checked as we only want to correct the miscategorised straight vertical strokes. If the height and width ratio of the stroke exceeds 0.4, the connector is not vertical enough to be classified as text stroke. Otherwise, the system then proceeds to check its proximity (500 Himetric units) with the node and merges them together when that condition is met.

Another significant problem is text being misclassified as circles. If a text stroke is large enough and its stroke length is greater than twice the bounding box diagonal, then it is mistakenly classed as a circle. Trials have shown that this rule does not accurately split certain strokes such as the capital letters “M”, “W” and “E”. As capital letters are generally written bigger, it exceeds our premade threshold and these letters have a considerable stroke length hence they become misclassified. To remedy this incorrect categorisation, the system checks for strokes that reside in the bounding box of the ‘circle’. If there are strokes within, then the possibility of the stroke being an actual circle is high and is disregarded. Otherwise the system checks if there is another node that encompasses this stroke. If there is, then the ink stroke is actually text and is adjusted. And lastly, a size measure is used. If the smallest side of the stroke is less than 1550 Himetric units, it is deemed a text stroke. This last measure corrects the strokes that are narrow such as the numeral ‘8’. If the ink stroke has been re-classified, the ink stroke property is changed to text and is added to the appropriate node.

The second part of ink grouping is the grouping of nodes. The first stage of grouping mentioned above only deals with grouping ink strokes that lie on the same horizontal plane. It does not, however, manage ink strokes that are further apart on the same plane or written below each other. Figure 25 shows a sample keyword that is written on different lines.



Keywords not grouped together

Node intersection check area for node grouping

Figure 25 Node grouping

To supplement the above grouping, we have incorporated node grouping into the IMM tool. This is the grouping of nodes that are not linked to any connector. The system first checks the node to see if a node is linked to any connector. If it is not, then the node is marked as eligible for node grouping. A node that is within 500 Himetric units of the standalone node is merged together. To implement this rule, four rectangles surrounding the four sides of the node are the areas that are checked for the presence of other nodes as shown in Figure 25. Other nodes that intersect with the any of the four sectors around the box are then merged together.

With the ink strokes grouped together and the misclassified strokes corrected, the system can progress onto the formulation of the mind-map hierarchy.

4.2.3 Structure

With the types of ink strokes identified, the information extracted needs to be catalogued. To store these classifications, we created two custom classes, a connector class and node class. The connector class creates a connector object for each connector that has been identified and associates its stroke data with the object. The IMM tool creates a node object for each circle identified. Node objects are not created for text strokes upon identification as they need to be grouped first. A node object stores the circle (if the node is enclosed) and the text strokes that make up the user's idea. Each node object is assigned a unique ID upon creation and is used to differentiate nodes as more are created. This section explains how the relationships between nodes are constructed and how the overall mind-map structure is determined.
The method for discovering the mind-map structure was based upon (Freeman and Plimmer 2007)'s method of creating a list of parent and child relationships. Our connector object allows the system to store the connector stroke's relationships with other nodes. We assumed the user draws a well formed mind-map (i.e a mind-map whereby the connectors all link an node with another and there are no isolated nodes).

Before the mind-map hierarchy can be formed, the individual relationships between the nodes must be found. A strategy that we considered was finding the nodes which the connectors intersected. Connectors that do not reach a node were extended until it intersected with a node. This method was hastily abandoned as it is difficult to determine the direction of extension as the connector can have varying degrees of curvature. The last few points of the connector could not be used as a means to extend the connector because if users accidentally flick the end of the stroke, the extension would be interpolated in the wrong direction.

We have opted to implement a proximity check, where connectors can be assigned quickly to the appropriate node. This is carried out by acquiring the end points of the connector and finding the node with the smallest distance to each endpoint's co-ordinates. These values are then examined and if the distance between the endpoint and the node is less than 1000 Himetric units, then the node's ID is recorded for that end of the connector. Should the distance exceed the threshold, then it is assumed that relationship is incomplete as the user may draw the connector first before writing the keyword.

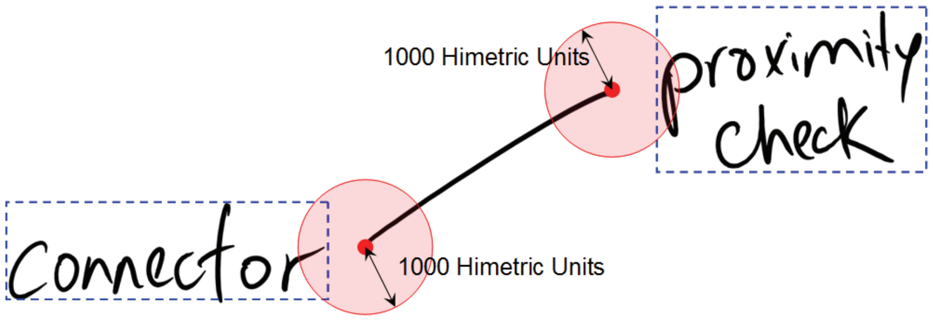


Figure 26 Connector proximity check

All incomplete relationships are assigned a default dummy node ID until another node is sketched close enough to complete the relationship. There is another check that ensures the endpoints do not have identical nodes stored so a connector cannot link a node to itself.

With the relationships between nodes confirmed, the overall structure of the mind-map can be found. As the mind-map is a hierarchical structure, the central node is the starting point from which relationships expand outward. Typically, the first idea that the user sketches is the central node and we have assumed this in our application. As there is no ordering in the mind-map structure after the main idea, this gives us flexibility as to our approach. We have opted for a depth first search approach to cycle through all the connectors. First a connector with one end linked with the main idea is selected. The node ID on the other end is extracted out. This node ID is then used to obtain the next connector. This process recursively progresses through the list of connectors until all the connectors have been analysed. As the connectors are cycled though, the structure is recorded (i.e which nodes are interconnected) for other functions such as mind-map export. Also, the connectors that have been traversed through is tracked with a list as a safety measure in case the user links two branches together. Without the list, the searching algorithm would continue in an endless loop if two branches are connected. The final step is to pass the ink strokes that are classified as text to Microsoft Vista's ink recogniser so that the text can be exported into a standard format if required. The completion of the mind-mapping structure paves the way for intelligent editing which is examined next.

4.3 Ink Editing

Mind-mapping is idea generation technique and we expect the users to iterate through their mind-map and refine their ideas. We need to have a good foundation of ink editing features to support this. The ink recognition above has paved the way for intelligent editing in our IMM tool. The following sample scenario demonstrates the type of editing functions that are at the disposal to the user.

A mind-map user has just created a mind-map and wants to make adjustments. He wants to erase a keyword and by simply flipping around the stylus, the pen input immediately becomes an eraser and lets him rub out ink just like using a pencil. The user now wants to change the colour of some ink to green. Using the stylus, the user enables the "Select Stroke" mode and highlights some ink strokes and then clicks on the green colour in the colour palette. Next the user wants to move a node to make room for another idea. So after changing the mode to "Select Node", a keyword is selected and is repositioned. The system automatically reflows the connectors linked to the keyword. With the extra room, the user now wishes to make one last adjustment, relocating an entire branch of the mind-map. Selecting the "Select Branch" mode, the user chooses the most central node they want to move and the system automatically includes its children in the selection. As the user drags the selection to its new location, the system displays the closest node that the selection would latch onto. Once the user lets go of the selection, the node and all its children are relocated.

We have implemented three modes that encompass the editing functions of the system. They range in different levels of complexity from simple editing features discussed in 4.3.1 to intelligent ink reflow covered in section 4.3.2.

4.3.1 Simple Ink Editing

Ink Properties

We have implemented an interaction mode for making simple ink stroke adjustments. Enabling the "Select Stroke" mode enables the user to make minor modifications without intelligent ink support. This is to provide user flexibility with their editing without putting constraints on their actions which is present in intelligent editing. The following are ink properties that are deemed important: ink colour, ink stroke thickness and resizing ink. These actions can be performed by first selecting an ink stroke or a set of strokes. Colours can be changed via selecting the appropriate colour in the colour palette. The same applies for the stroke thickness. The user can also resize the selected strokes by dragging on the corner tabs of the selection box. These four features were implemented using the appropriate functions within the Microsoft ink library.

Erase

The erase function is created based on a pencil metaphor. As the stylus is used as input, we have enabled an erase function at the end of the stylus. When the user flips the stylus around, the ink erase mode is activated. This is made possible with a checker that detects when the stylus is in range of the screen and confirms which side of the stylus is being used. This then determines the editing mode of the stylus. This pencil metaphor is intuitive as all users have been accustomed to utilising the opposite end of a pencil for erasing. An issue that is quickly identified is that the mode switching is initially set to ink drawing mode and ink erasing mode. However, when a user changes to ink select mode and then proceeded to erase some ink, they were automatically defaulted to ink drawing mode when they switched back which proved frustrating. This has been corrected by storing the current state of the stylus and reloading the state when the user flips the stylus back. Also, to allow support for other input mediums, we have placed an erase widget which allows keyboard and mouse users to also access this functionality.

Manual Ink Classification

Although our ink divider correctly identifies most ink strokes, there are occasions where misclassifications occur. As incorrect categorisation of strokes have a detrimental effect on determining the structure of the mind-map, we have implemented a manual classification tool for the user to change the type of component an ink stroke has been classified as. The user can set an ink stroke to the three types of mind-map components we are currently using: a circle, line or text. All the strokes that have been manually changed have their stroke ID stored in a list. During ink division, the strokes which have been manually configured bypass the ink classification process and move directly to ink grouping.

4.3.2 Intelligent Move

For intelligent move support, we have identified 3 types of possible ink reflow when constructing a mind-map. They include moving ink strokes (Figure 27), a single node (Figure 29) and a mind-map branch (Figure 28).

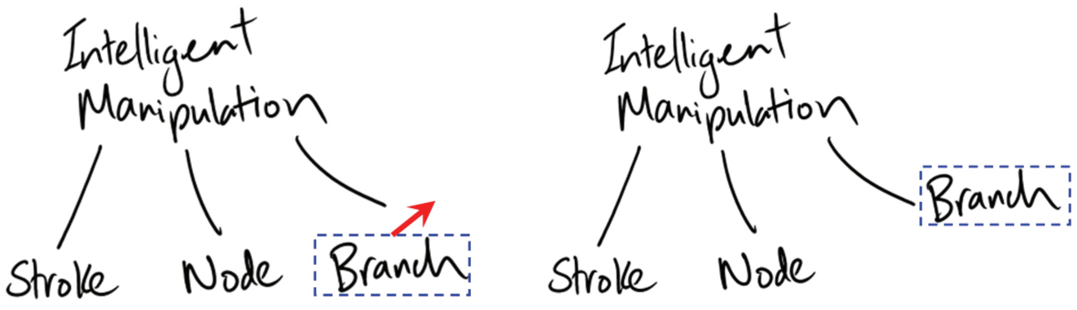


Figure 27 Simple editing (No intelligent reflow)

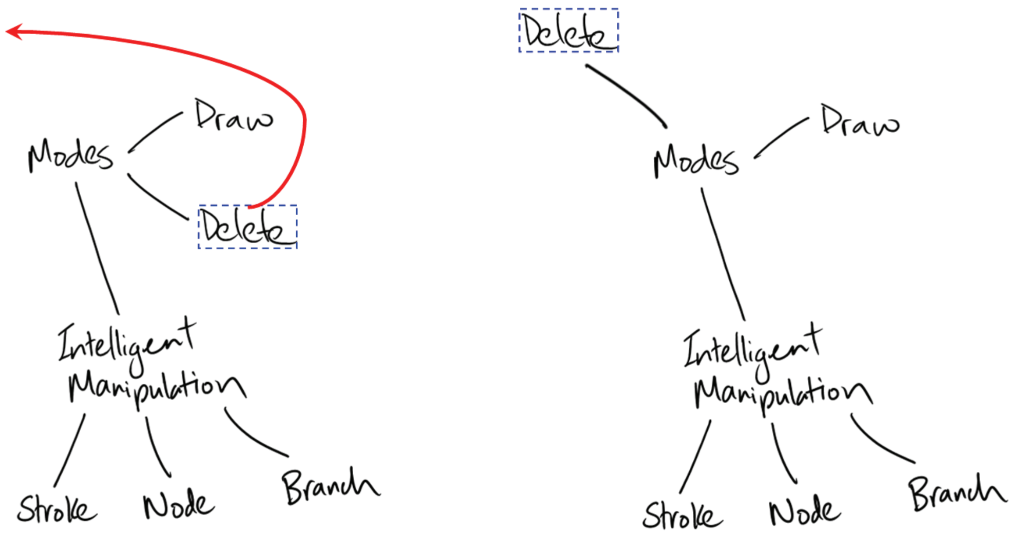


Figure 28 Intelligent node move

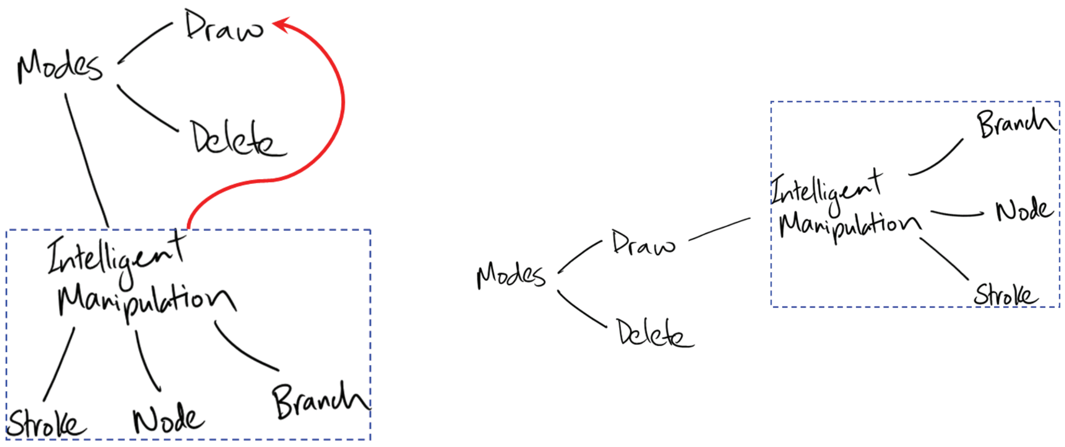


Figure 29 Intelligent branch move

From our own observations, a single editing mode was not sufficient to cater for all three types of ink move as the system cannot automatically differentiate a simple move function from a more complicated task such as repositioning a branch. Having a single editing mode leads to other issues as the intelligent editing requires restrictions (such as allowing users to only select nodes and when users select a stroke of a node, the whole node is selected). Because of these restrictions, there has to exist separate modes for such intelligent ink support. We have created additional 'branch' and 'node' modes specifically to relocate a branch of the mind-map or a single node. Each of these modes is discussed later in the section.

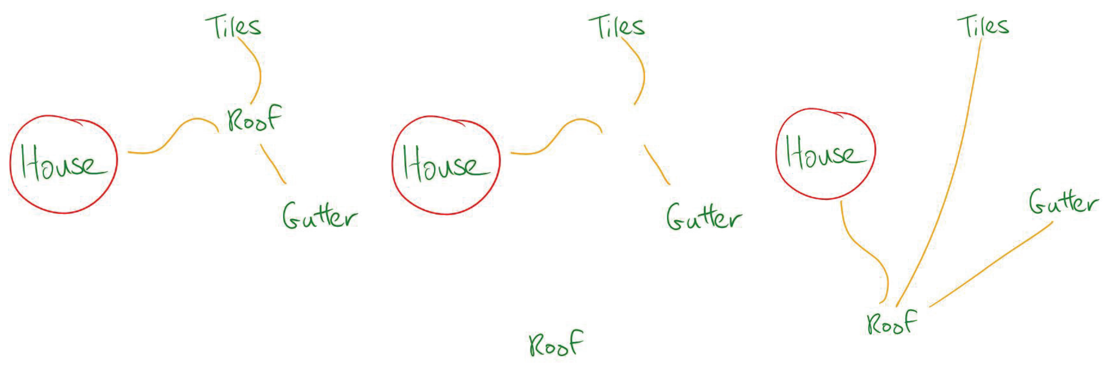
Simple move

Simple move is where the user freely moves any number of ink strokes around the canvas. This may be due to revisions and corrections to the mind-map such as moving a letter closer to a keyword. To allow unrestricted interaction for the user, the system provides no intelligent support. Our IMM tool uses inbuilt functions present in the Microsoft Ink library to enable users to move ink around the canvas.

Node move

A node is defined as the total amount the ink strokes that make up an idea. Node reflow is the movement of a single node to another location while preserving the positions of the other nodes on the canvas. Firstly, when the user desires to relocate a node, all the appropriate strokes must be moved. Our system has been implemented with the intention that the user needs to only select an ink stroke and the appropriate node becomes selected. This is relatively a simple task, by obtaining the stroke ID of the stroke selected by the user and comparing this with each node. When the system finds the node containing a matching stroke ID, all the strokes in the corresponding node are added to the selection. This allows the user to quickly choose the node they desire as opposed to meticulously encircling all the strokes belonging to that node. The system is also programmed to ignore the selected stroke if it is a connector.

With the complete node selected, the user can then move this selection to its new position. The mind-map structure is held together by the connector's proximity measures. When a user relocates a keyword, the mind-map structure becomes broken as shown in Figure 30. In order to maintain the structure of the mind-map, we must also transform these connectors.



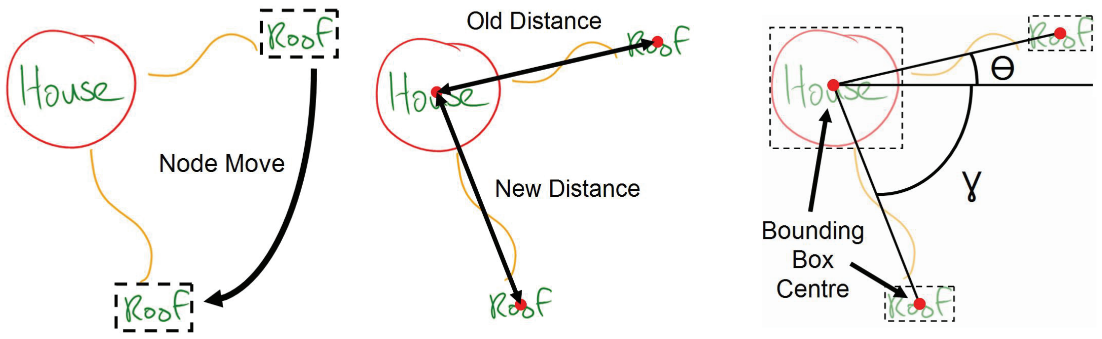
Original mind-map | Moving "Roof" without intelligent reflow | Moving "Roof" with intelligent reflow

Figure 30 Before & after connector reflow implementation

There has been research into connectors in other domains and ideas have been drawn from Arvo and Novins (2005) and Reid, Hallett-Hook et al. (2007) for implementing connector reflow in our mind-map. Arvo and Novins have a promising algorithm to interpolate the connector for reflow. However, there are disadvantages such as the overstretching of a connector would result in a straight line. Also, should the connector later become compressed, the connector remains straight. This is detrimental for a mind-map as the reflowed connectors would lose its hand drawn appearance if overstretched. Hence we have adopted the strategy of Reid, Hallett-Hook et al. (2007) as their scaling technique addresses these problems effectively.

Once the user has finalised the node move, the 'pen up' event triggers the connector reflow process. The system cycles through the list of connectors and if a connector contains the reflowed node ID, then it is eligible for reflow. Firstly, the stationary node that is linked to the connector is determined as it provides one of the coordinates used for the connector reflow. Then we must determine whether the connector is being compressed or extended and the new orientation of the node. The amount of transformation performed on the connector is found by finding the old distance between the two nodes and the new distance after moving. Using the x and y coordinates of the centre of the bounding boxes of the two nodes, two distance values are calculated. The scaling value is calculated by ratio of the new distance value over the old distance value. If the new location of the reflowed node is closer to the stationary node, then the scale ratio would be less than 1.0 and greater than 1.0 if the reflowed node was moved further away.

Scaling is performed by rotating the connector so that it is parallel to the x-axis and then scaling the ink stroke in the positive x direction. This is done by finding the coordinates of the end points of the node and finding its angle relative to the horizontal axis (i.e angle Θ) as shown in Figure 31.



Node being reflowed Step 1: Finding Scale Step 2: finding angles Θ & γ

Figure 31 Connector reflow process

Once the angle of rotation is found, the stroke is rotated about the stationary end of the connector by angle Θ. Now the stroke should lie horizontally and is then scaled by the scale value calculated previously. The stroke now needs to be rotated to its new position and this can be found in a similar way by calculating the angle between the x-axis and the new position as indicated in the diagram as angle γ.

There were some problems that surfaced during testing. One of the issues that was quick to reveal itself was the centre of rotation for the connector. On our first attempt, the connector was rotated about its stationary end point. A problem was reflowing the connector when the node has moved to the opposite side of the stationary node. The resulting reflowed connecter is shown in Figure 32 where the connector overlaps the node. This was fixed by taking the centre of rotation about the stationary node instead, resulting in a nicely reflowed connector no matter where the node was moved.

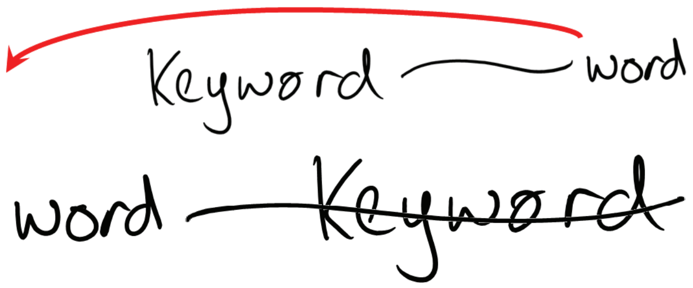


Figure 32 Connector reflow problem: Centre of rotation

Another problem that was identified was the scaling methodology for the connectors. The original algorithm scaled the connector according to the distance between the centre points of the two nodes. However, since the distances between the node centre points are different to the actual length of the connector, the amount of scaling was sometimes inadequate. If the nodes were large, then the scale factor for reflow becomes marginally smaller as the change in distance is less significant.

Below is an example situation,

The radius of two nodes is 5 units. If the original distance between the node's bounding box is 10 units and the new distance is 20 units, then the ratio for scaling the connector would be 30/20 = 1.5. If the nodes are larger, with a radius of 10 units and the distances remained the same would result in a ratio of 40/30 = 1.33.

This smaller scale indicates that the stroke gets scaled less whenever the nodes are moved. This scaling is very noticeable when the nodes distance changes by a large margin. The connectors do not expand enough when the nodes are pulled further apart nor do they condense enough when compressed. The result is a connector being ridiculously short when expanded or too long and protruding into the nodes when compressed.

The first solution to this problem was to calculate the closest distance between the stationary end point of the connector and the mobile node. This effectively solved the problem of under/over scaling when the nodes are pulled a fair distance apart. However, if the node is moved to the opposite side of the stationary node, the calculation becomes erroneous again as there is extra distance that is taken into account as shown in Figure 33.

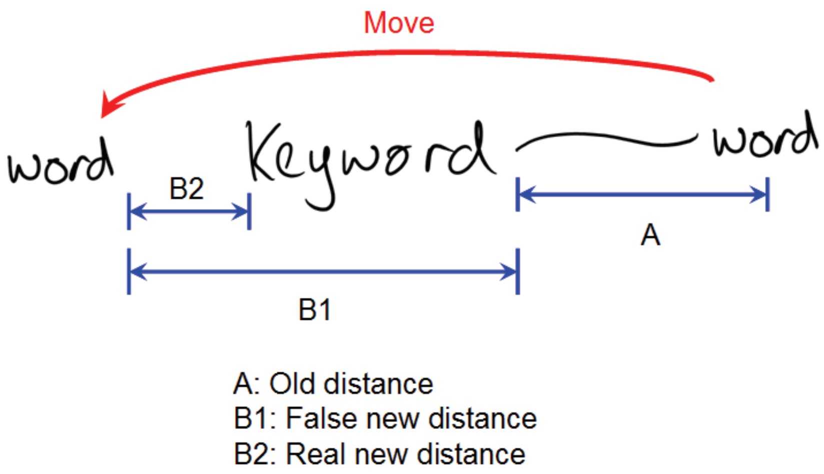


Figure 33 Attempts to solve problem

Our final attempt to correct this problem was changing the ratio formula to the distance between the closest distance between the bounding boxes of the two nodes was used instead. Since the minimum distance is used, no matter where the node is positioned, it ensured that the connector never too long or too short when compressed or expanded.

The rotation of the connector also provided problems albeit minor. The centre point of the node was used to rotate the connector so the connectors start point rotates around the node as opposed to a fixed position. One flaw with this is the fact that the node may not always be square and if the connector is linked to a rectangular node, which is fairly common as many keywords end up being rectangular. Rotating the connector would either cause the connector to overlap the node or be pulled away from the node as shown in Figure 34.

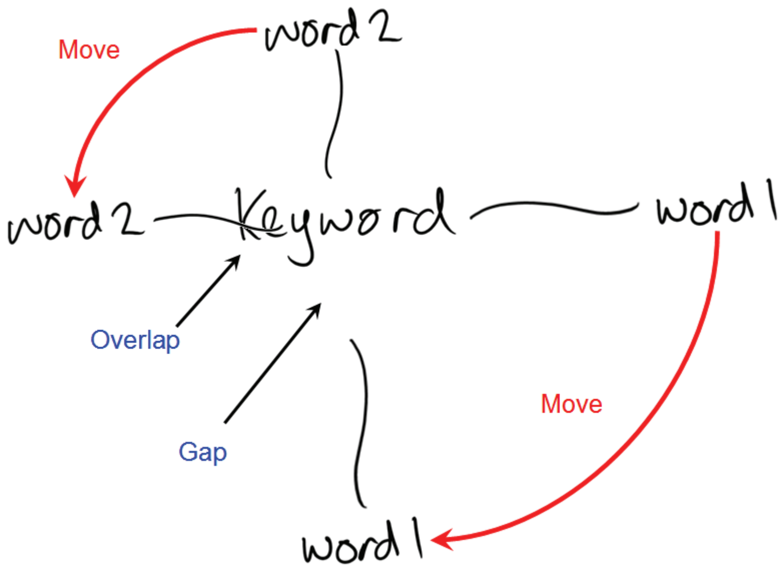


Figure 34 Rotation connector problem

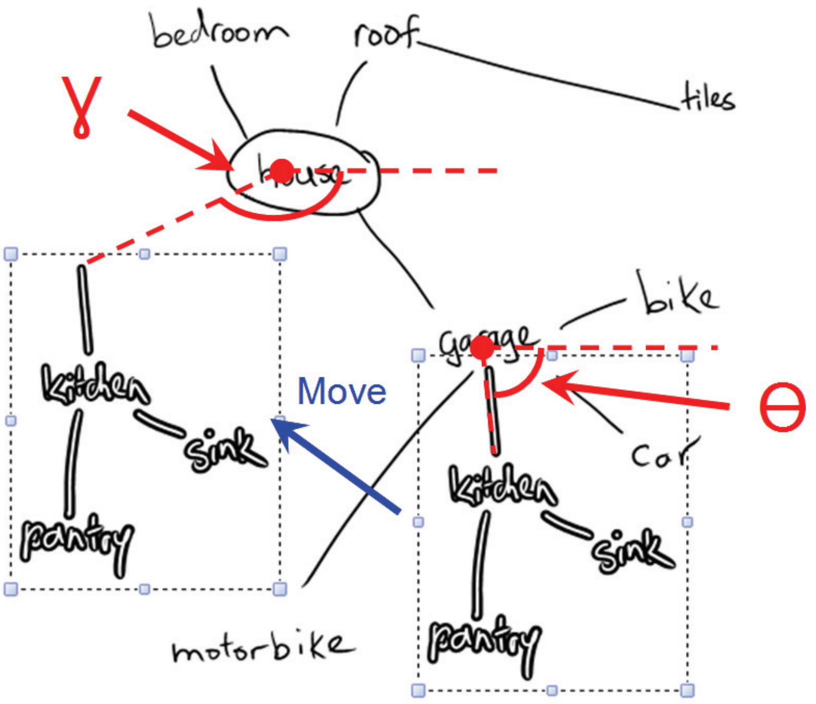
A solution to this problem is to implement a checker to keep the connector close to the bounding box of a node. If the end point of the connector resides inside the box, the minimum distance between it and any of the four sides of the node bounding box is calculated. The connector is then shifted outward by that amount. An issue with this method is that the connectors that are outside the box remain unaffected. To correct this, the closest distance between the node and connector is measured and the stroke is moved inwards towards the node by the calculated distance.

Branch move

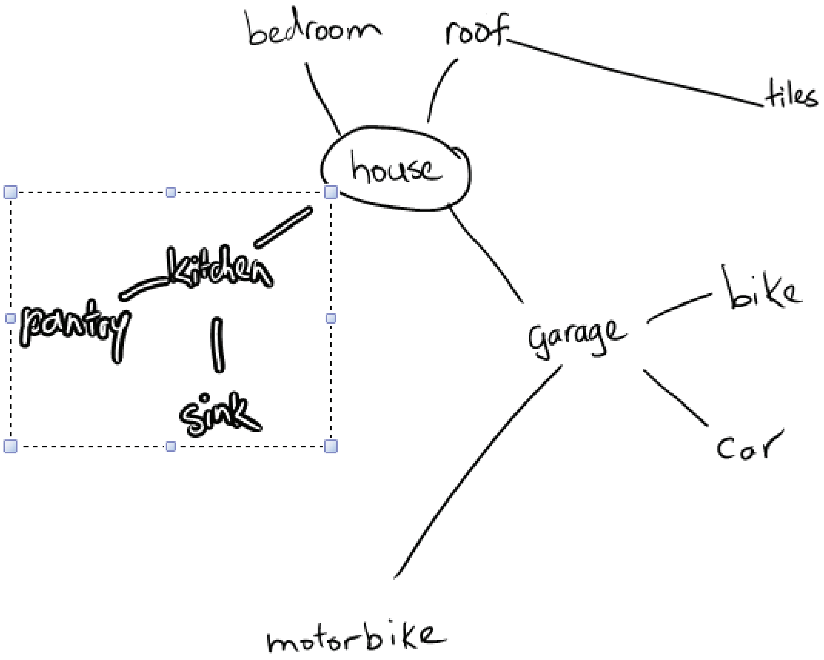
The last type of mind-map adjustment we have considered is the relocation of a whole branch within the mind-map. It is likely that after expanding the mind-map, a user may

want to modify the mind-map structure and move a group of ideas to another idea on the map. The user first enables the "Select Branch" mode and selects the centre-most node of the branch to move. The system systematically obtains the nodes belonging to this branch in the mind-map using a recursive technique. Then these nodes are inserted into the selection. We want a means of reflow that causes the least amount of ink overlap while maintaining the user's look and feel to the mind-map. A method considered is to try maximising the minimum angle between the nodes parent connector and child connections. Purchase (1997) has done this using a symmetry metaphor. Connectors are positioned so that they are as further apart as possible from each other. Its goal is to spread out the connections which possibly makes it easier to understand and provides a cleaner look. In an inking environment this method could possibly decrease the amount of ink overlap. However, this technique enforces a strict constraint; the user has no power in the arrangement of the branch whatsoever. Therefore, we decided against using such a methodology as its interaction is too restrictive, similar to the computer based software examined in Section 2.2.
We intended to implement an intersection check to reflow ink when nodes and connectors intersect as result of a move. However, this creates a chain reaction with sections of the map being adjusted. This in turn causes the loss of the user defined look of the mind-map, from minor reflow to significant changes to the map depending on the intersections. This is problematic as the whole look of the mind-map changes and is disorientating for the user. Hence we have restricted to node / node intersections only.

Having taking into account the factor of ink layout preservation, we have implemented a branch reflow where the selected branch can rotate about the new node it latches onto. Firstly, the system needs to find the angle to rotate the branch. This is split into 2 separate tasks. First the angle that the connector makes with the original node must be found. Then the angle the relocated connector makes with the new node must be found. In order to correctly rotate the node, both these angles need to use the same reference point and we have chosen the horizontal axis for this.



Mind-map branch before reflow.



Mind-map branch after reflow.

Figure 35 Branch reflow

In Figure 35, the angle Ɵ is calculated using the end point of the connector in conjunction with the centre point of the node and the horizontal line. The same applies for the angle Ɣ, using the horizontal axis and the new node centre point. The difference in the two angles is the amount of rotation required. As all the ink strokes are rotated, the keywords may not be positioned at a readable angle depending on the branch move. Users would end up having to rotate the tablet in order to read the keywords. Another significant point to note is that since the keywords are not level, the ink recognition would most likely be erroneous. To account for this, the node's ink strokes are rotated back by the amount the whole branch was rotated. This results in all the keywords remaining level.

A problem that surfaced from the branch rotation was that the connectors began overlapping the nodes due to the rotation needed to keep the node level on a horizontal plane. This problem was similar to the one discussed in node reflow section and the same connector reflow methodology was used to correct these overlaps by shortening or lengthening the connectors.

A usability issue that arose during testing was a lack of feedback during the branch reflows. It was difficult to determine which node the branch would latch on as there was no feedback from the system. Therefore we implemented visual feedback for the user, letting them know the node where the branch would latch on. To do this, a proximity check was used to determine the closest node and a temporary connector coloured line links the selection with the appropriate node. Also, the closest node has its bounding box highlighted as shown in Figure 36.

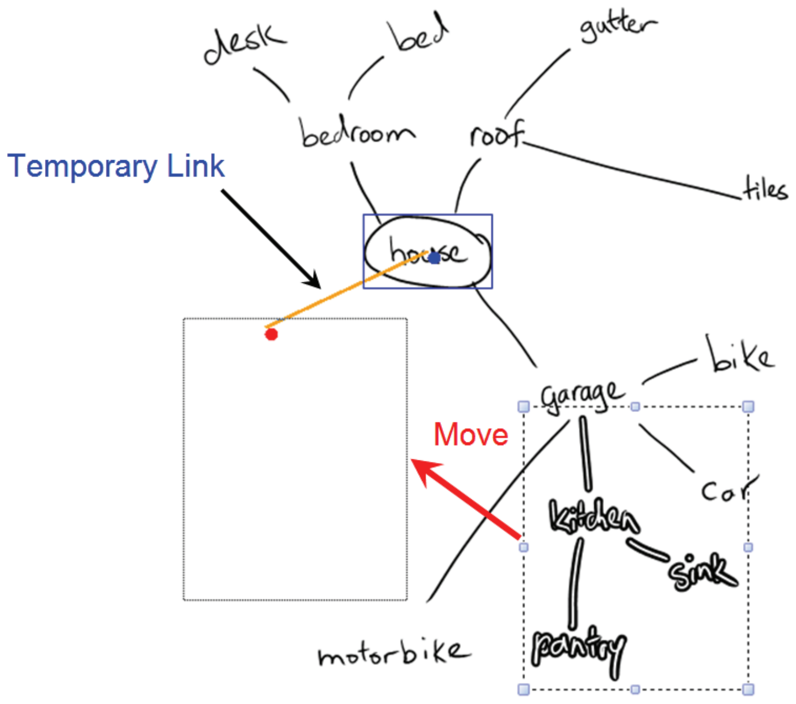


Figure 36 User feedback during branch reflow

Also, with the branch reflow, we noticed there were no restrictions as to the positioning of the node. It allowed users to rotate the branch so that it points inwards towards the main idea. This was deemed inappropriate as this goes against the mind-map convention of ideas radiating outward (Buzan and Buzan 2000) as well as resulting in a significant ink overlap. A solution was to restrict the angle which the connector pointed from the newly attached node. Firstly the connector whose child connection was the newly attach node was found and that connector end point was taken as a point of reference. The system then calculated the angle between the two connectors using the centre point of the node. A rule we have established ensures this calculated angle exceeded 100 degrees and if not, then the whole branch was rotated by the deficit amount until the angle reaches this value. This change in the branch reflow ensured that the mind-map radiated outward from the main idea on the canvas. However, if the reflowed branch was latched onto the main idea, the rule was bypassed.

4.4 Standard Functions

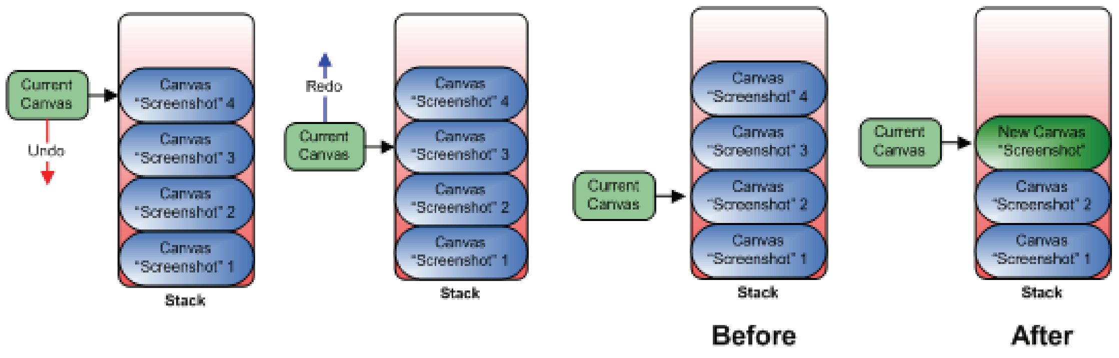
This section describes the standard functions of the mind-map that enhances the user experience and is included in our tool.

Zoom

As the computer medium provides an almost unlimited canvas space for the user to perform sketches, there must be a method to view them all. Therefore a canvas zoom feature is implemented in the IMM tool. The zoom in and zoom out feature is executed by transforming the canvas by increments. Two buttons are inserted into the mind-map interface, a zoom in and zoom out button. Each time the user selects one of them, the canvas is scaled in a positive or negative increment. There is also a slider bar that the user can drag to change the zoom factor.

Undo/Redo

The Undo/Redo function implemented is based upon a stack. By coursing back and forth through the stack, it allows the user to undo/redo everything they have done. Whenever an action is performed, such as creating/deleting/editing ink, the system takes a “snapshot” of the ink objects on the current canvas and stores this in a list. There is a pointer that keeps track of the current state of the canvas by pointing to its corresponding state on the stack. When the user wants to undo an operation, the state of the canvas changed to the state previous to the current state on the stack. The pointer is shifted down to indicate the current canvas state on the stack. When the user wishes to redo an action, the state of the canvas is shifted up to the next canvas state higher on the stack.



A.) Undo B.) Redo C.) A change in the canvas

Figure 37 Diagram of the stack used in undo/redo

Whenever a user makes an action and the pointer is not directed at the top of the stack (i.e a user has just undone a stroke and then proceeded to add a new stroke on the canvas), any stored canvas states above the pointer is erased and the new canvas state is added to the stack.

Load/Save/Print

These features are deemed standard and important in order to support the basic tool functionality. Mind-map loading is performed by streaming in the bytes of data and loading it onto the ink overlay. The reverse is done for saving ink on a canvas. The data is written into an "isf" format, an xml file in 'ink standard format'. The Print function involves converting the strokes onto a graphics object for printing.

4.5 Summary

This section has described the inner workings of our mind-mapping tool in detail. We first overviewed our mind-mapping tool with a short description of our tool. Each component of the tool is explained in greater detail, describing the various tasks performed when an ink stroke is drawn and altered. We have based some of the implementation on other existing works and have explored ways of adjusting and extending this work to meet our needs. There have been technical difficulties that surfaced during each of the implementation stages of our mind-map tool. The causes of these problems have been explained and solutions to them explored. Although many complications have been identified and dealt with, there are still minor problems that have been missed and are identified when we evaluate our tool.

Chapter 5

Evaluation

Although there was on-going testing performed on the mind-mapping tool during development, it was mostly testing the technical side of the application, such as making sure all the functions perform correctly. As we have spent a substantial amount of time developing the tool, and our goal was to optimise the usability of the application. Therefore this section describes the usability testing performed on our system in hopes of finding the difficulties that normal users may encounter. Section 5.1 outlines the tests and perquisites for conducting an evaluation for our mind-mapping tool. We then describe our methods of testing our application by using sample scenarios to guide the user into using the system's features. And lastly section 5.2 reports on the results obtained from the usability test.

5.1 Evaluation Outline

A usability evaluation is an assessment test to measure the effectiveness of the mind-map implementation as it identifies the usability problems that may have been missed. It also ensures that the usage of the mind-mapping software is a pleasant experience for the user.

Test Objectives

The objectives of our usability software can be categorised into two sections: General Inking and Ink Reflow.

General Inking are the functions that allow the user to create a mind-map using our system. These include:

- Ink Strokes – Tests if the user can use the application to draw a mind-map.
- Ink Editing – Tests if the user can alter the ink properties of the ink strokes.
- Ink Erase – Tests if the user can delete strokes that they have created.

The second category of objectives is ink reflow. The following are the tasks that we hope the user is about to accomplish with our software:

- Single ink stroke reflow – Tests if the user can select a single ink stroke and move it
- Ink node reflow – Tests if the user can select a single node and relocate it.
- Mind-map branch reflow – Tests if the user can select a branch in the mind-map and relocate it under another node.

Participants

Users who would utilise the IMM tool may come from a wide spectrum of people as mind-mapping does not require specific skills or knowledgebase. People who use mind-maps for brainstorming, group thinking and written/verbal note taking would be suitable candidates for the usability test. Eight people participated in the study. Two have had experience with the Tablet PC and were computer science students while the other rest did not have any experience with Tablet PCs and belonged to other disciplines ranging from engineering to biology.

Test Environment

The equipment required was a Tablet PC containing the mind-mapping software. It was an Intel Pentium M 1.6 GHz computer with 512 Mb RAM running Windows XP Tablet Edition with a screen resolution of 1024 x 768 pixels. Installed on the computer is a video capture software called Morae. The video recording feature of Morae unobtrusively records the display screen in the background and this allows us to observe the ordering and editing of ink strokes when performing their task. The tablet had an inbuilt speaker and thus the comments made by the user were also recorded along with the video capture. The location of the test area was in an office of the HCI laboratory. The participant was provided with a desk and a chair, with adequate lighting and airflow. Having planned the testing conditions and the target population for our application, we can move onto the design of our usability tests.

Procedure and Tasks

Participants were first introduced to what a mind-map was and its various uses. Then they were briefed on the application and what the system does. The participants were given time to read the participation sheet and sign the ethics approval form (University of Auckland human ethics approval). After accepting to undertake the usability test, the participants are first asked to fill in a short pre–questionnaire on their views and use of mind-mapping in general. As not all users are familiar with a Tablet PC, they must be trained to use the stylus and the touch screen. For the user to become accustomed to applying the right amount of pressure on the touch screen, they were invited to have a five minute play with Microsoft Ink Ball and Microsoft Paint. The participants were then trained on how to use our mind-map software before commencing the study.

The usability test consists of 2 sections: constructing a mind-map that tests the general inking objectives and editing a mind-map to test the ink reflow goals. The reason for splitting the test into two sections is to not overwhelm the user with all the tool functions at once. Each section consists of a training scenario that guides the user through using all the software's features. Before the actual test, a demonstration in using the software is carried out. A mock scenario in a similar format to the actual scenario is first used to show participants how to use the software's features. After the demonstration, Morae is enabled and the user begins to complete the actual tasks. This process is repeated again for the second part of the test.

Usability Test Part 1: Constructing a Mind-map

Training Scenario 1

The participant is given a set of 12 dog pictures and they are to classify them by their physical features into a mind-map. The central idea of this mind-map is "Dogs". The participants are shown which buttons are used for drawing strokes, erasing strokes, undo, redo, load and save. The participants are then shown how a well formed mind-map looks, with the central node encased in a circle and connectors always connecting two nodes together.

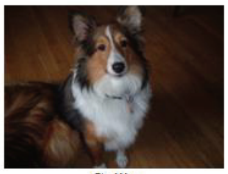
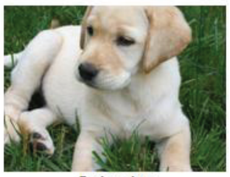
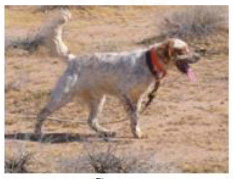
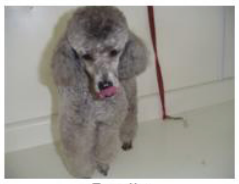
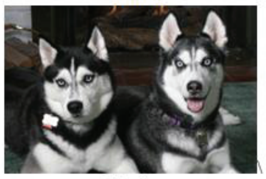
 Collie	 Labrador	Pointer
Chow Chow	Dalmatian	Greyhound
German shepherd	Great Dane	Chihuahua
 Setter	 Poodle	 Husky

Figure 38 Training scenario containing 12 dog pictures to categorise

	Description	Script
Task 1	Mind-map Construction	You are given a set of 12 dog related pictures. The task is to create a mind-map to classify these animals into categories of your choosing. The main idea of this mind-map is "Dogs".
Task 2	Changing ink Colour	After creating your map, you want to change the colour of the central idea. Change the colour of your central node to blue.
Task 3	Deleting Nodes	You find that one of your nodes is "incorrect" and would like to remove it. Delete one of the nodes you have drawn.
Task 4	Saving a Mind-map	You are satisfied with the mind-map and you would like to save it for future reference. Save the mind-map as "dog".

Table 3 Training scenario 1 task list

Test Scenario 1

The participant must now construct their own mind-map to classify the following 12 farm related pictures.

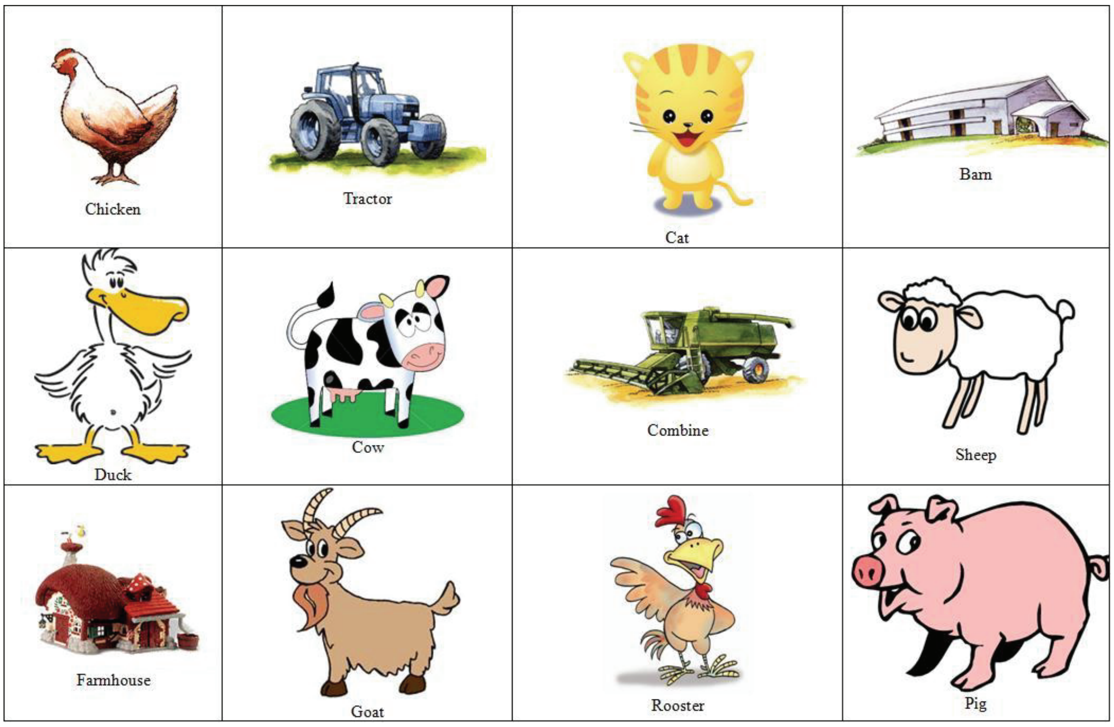


Figure 39 Test scenario containing 12 farm pictures to categorise

Test Scenario 1: Task list		
	Description	Script
Task 1	Mind-map Construction	You are given a set of 12 farm related pictures. The task is to create a mind-map to classify these into categories of your choosing. The main idea of this mind-map is “Farm”.
Task 2	Changing ink Colour	After creating your map, you want to change the colour of the central idea. Change the colour of your central node to red.
Task 3	Deleting Nodes	You find that one of your nodes is “incorrect” and would like to remove it. Delete one of the nodes you have drawn.
Task 4	Saving a Mind-map	You are satisfied with the mind-map and you would like to save it for future reference. Save the mind-map as “farm”.

Table 4 Test scenario 1 task list

Usability Test Part 2: Editing a Mind-map

Training Scenario 2

The participant is given a pre-made mind-map of a house. The participants are then shown editing features of the software, the explanation on the use of the three editing modes available: ink stroke, node and branch reflow.

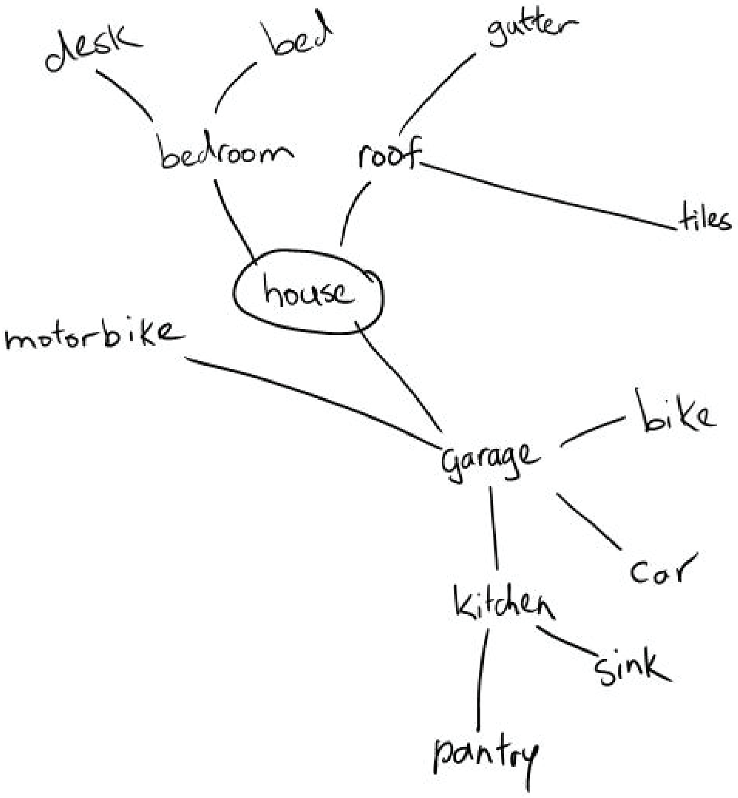


Figure 40 Training scenario premade mind-map

Training Scenario 2: Task list		
	Description	Script
Task 5	Loading a Mind-map.	You have another mind-map that you have made and want to make adjustments to. Open the mind-map named "house".
Task 6	Relocating Branches	You are given a pre-made mind-map regarding the parts of a house. However, some branches of the mind-map are not where they are supposed to be and you want to move them under the new branch. Relocate the branches with the heading "kitchen" under the main node "house". You may need to

		make room for the kitchen branch. Move the motorcycle node down to make room.
Task 7	Moving Nodes	The nodes are too packed together and you would like to space them out. Move the node "roof" so the mind-map looks spread out.
Task 8	Changing Modes + adding more nodes.	You realise that you have missed out the toilet. Create a new node with the name "toilet" and add that to the "house" node.
Task 9	Moving Branch	You review your map and you find out that the toilet can also belong in the bedroom (ie an en-suite). Move "toilet" node you have just drawn under the "bedroom" node.
Task 10	Saving a Mind-map.	Finally satisfied with the adjustments, you want to save the mind-map. Save the new mind-map as "house".

Table 5 Training scenario 2 task list

Test Scenario 2

The participant must now edit another pre-made mind-map according to the task list using the intelligent ink editing feature of the software.

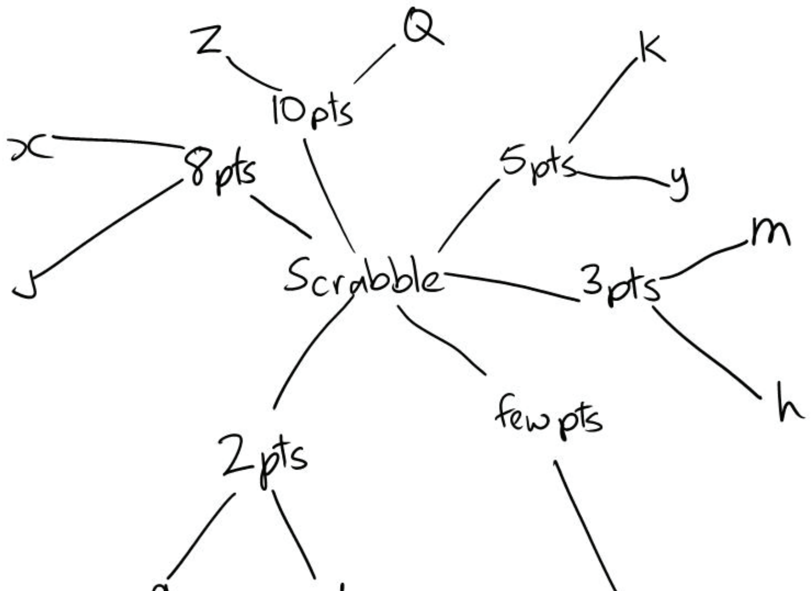


Figure 41 Testing scenario premade mind-map

Test Scenario 2: Task list		
	Description	Script
Task 5	Loading a Mind-map.	You have another mind-map that you have made and want to make adjustments to. Open the mind-map named “scrabble”.
Task 6	Relocating Branches	You are given a pre-made mind-map regarding the scoring of letters in a game of scrabble. However, some branches of the mind-map are not where they are supposed to be and you want to move them under the new branch. Relocate the branches with the heading “2pts” and “3pts” under the node “few pts”.
Task 7	Moving Nodes	The nodes are too packed together and you would like to space them out. Move the node “8pts” so the mind-map looks spread out.
Task 8	Changing Modes + adding more nodes.	You realise that you have missed out the letters that are worth 4 points such as “V” and “F”. Create a new node with the name “4pts” and add these two letters to the map.
Task 9	Moving Branch	You review your map and you find out that the letter y is only worth 4 points. Move “Y” under the new node (“4pts”) you just drew.
Task 10	Saving a Mind-map.	Finally satisfied with the adjustments, you want to save the mind-map. Save the new mind-map as “scrabble”.

Table 6 Test scenario 2 task list

After completing both tasks, the users are then asked to complete a post task questionnaire where they rate the ease of use of the system features and comment on the things that caused frustration or improvements that can be made. After the questionnaire, there is an informal interview to gather more qualitative information such as the ease of use for the software and the comprehensibility of tasks.

Pilot Testing

Before the actual testing can be commenced, the training and test scenarios need to be trialled to ensure the training and test cases are of equal difficulty and that they are easy to comprehend. To test our test, we performed a pilot test for our scenarios. A pilot test is a dummy run of the usability test. It was used to test trial the usability questionnaire to determine if they are suitable as they may be too difficult/easy/long/short and to make changes before the actual test if they are not satisfactory.

Also, the pilot test showed that participants might not have any previous experience with Tablet PCs and sketch lightly when writing on top of the touch screen for the fear of damaging the equipment. This resulted in strokes that were thin and often disjointed as the pen leaves the surface of the screen. This was not ideal as the recognition engine examines single strokes and a single stroke that is split into multiple strokes would cause complications. Therefore, users in the actual test were shown how hard to write on the tablet and were comforted with the knowledge that the force with which they wrote on the screen would not damage the Tablet PC.

Also, another observation noted was that the keywords were written at a slight angle. As the canvas contains no lines, there was no passive restriction on the user to write on horizontally. Although our application allowed a small leeway for words that are not written on a single line, the recognition rate would decrease as the keyword becomes more angled. For the actual usability exercise, we have asked each participant to write horizontally as much as possible.

For our pilot test, it was found that the preliminary training was lacking. For the training scenarios, the participants were originally walked through the training task list with the demonstrator writing using the tablet. However, there wasn’t enough user interaction with the tablet PC. The lack of “hands on” experience with the application was noticeable as the participant was observed hesitating at the very start of the test. Another problem was the tasks were originally too brief and had to be expanded, especially for the second scenario.

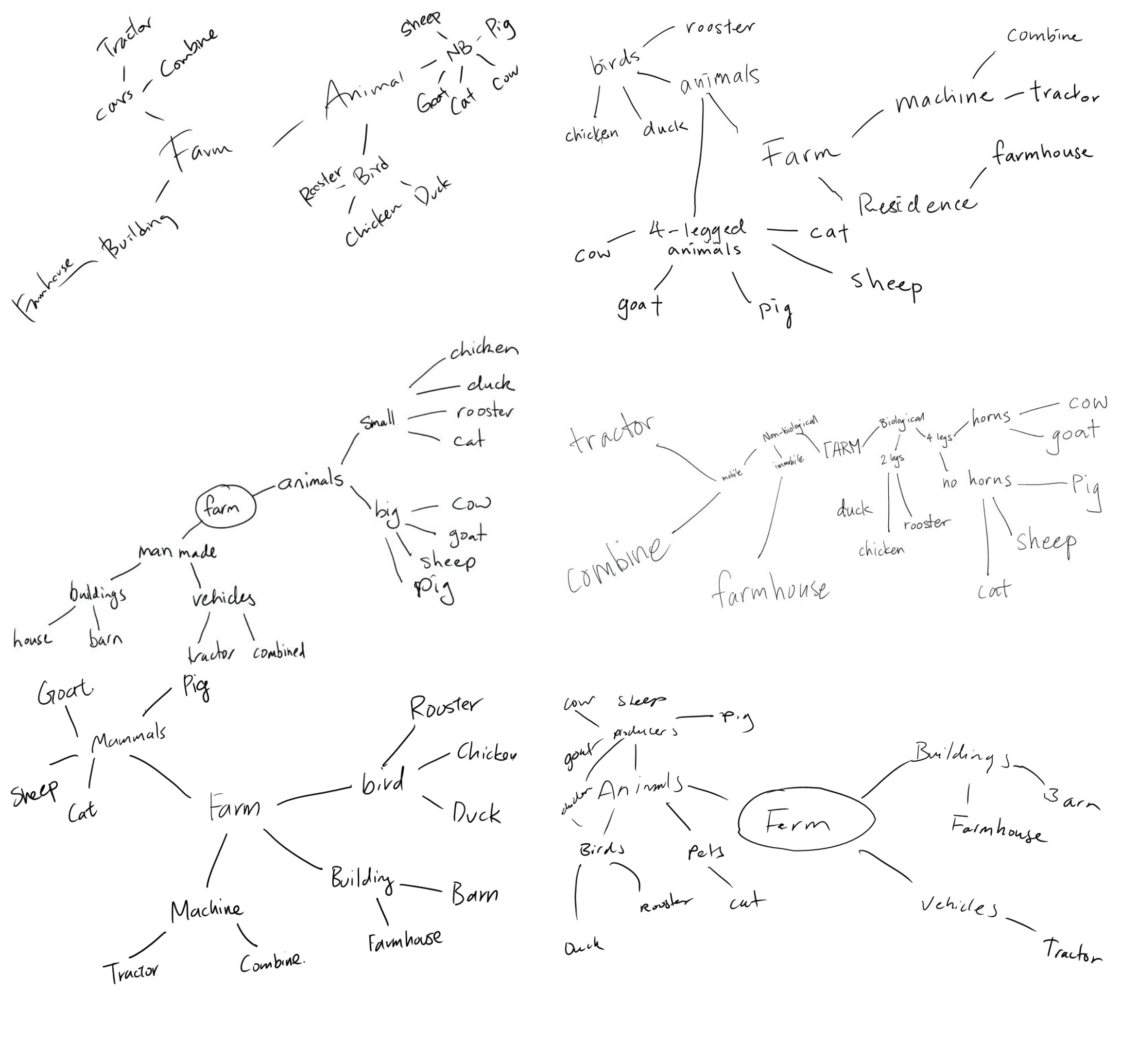
In the revised usability test, the ink reflow scenario was extended, with some tasks similar to the first scenario, forcing the user to switch between the available modes (Task 8).

Also, the training approach was modified and that during this scenario training, the participant was given the tablet and stylus and the instructor guided the user to use the appropriate features.

5.2 Evaluation Results

Evaluation Analysis

After pilot testing the mind-mapping tool, eight participants of various backgrounds ranging from an age bracket of 20-25 years were invited to partake in this exercise. The participants have had no prior experience with the tool. The following describes the usability issues discovered and other technical issues found, derived from observations of the user, the questionnaire and the video capture obtained from the test.



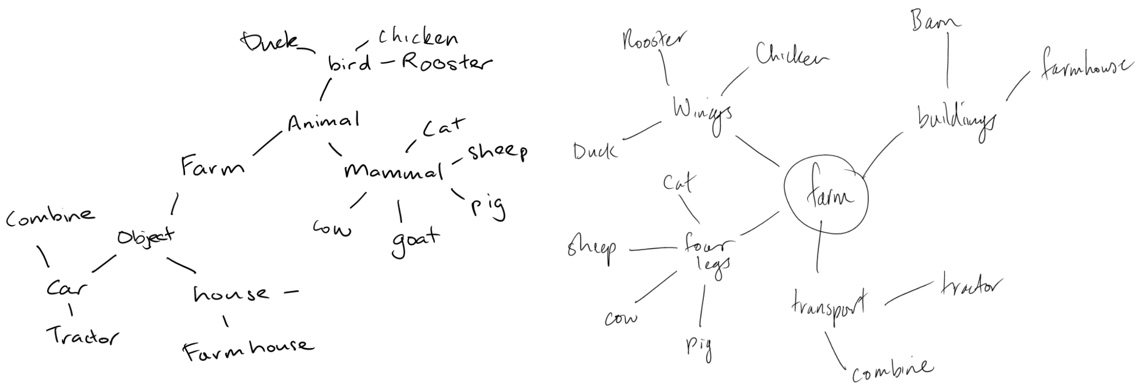


Figure 42 Mind-maps drawn by the participants.

Observations

The common problem that was observed throughout the course of the usability test was the level of feedback from the system. This problem, the "Visibility of system status" is one of the many usability issues from Nielsen and Mack's (1994) list of usability heuristics. Nielsen stated that users of the software should be kept informed of the system state via feedback from the application. Participants commented on the lack of feedback in two areas: editing mode change, colour change

For the editing mode change, there was not enough feedback for the user confirming the mode switch. The system does have a default colouring of buttons when they are selected. However, participant three pointed out the mode switch was difficult to determine as it was very faint and commented that it would not be noticeable for many users. As a result, participant three was confused at the lack of response and was observed clicking the mode switch buttons numerous times just to be sure. This lack of response was resolved by changing the font of the selected mode button. The currently active mode would have its button text set to bold in order to stand out from the other modes.

As mentioned previously, there was a lack of instant colour change for the colour of the ink stroke and this was first noticed by participant one during the first trial. The user had to unselect the selection of ink strokes before the colours were updated. This proved confusing and frustrating for participant one as there was no immediate feedback to show that the colour had been changed. The participant ended up selecting the colour many times and only by trying to unselect and reselect the ink stroke to try again did he realise that the colour was changed but was not updated.

In Nielsen's book, another usability heuristic "Recognition rather than recall", was relevant in our study. This heuristic stated that the actions, options for a system should be visible to the user so it can be easily recognised as opposed to remembering where to use the feature (Nielsen and Mack 1994). In the case of our IMM tool, participant two commented on the hassle to change colours by searching through the menu items in order to change the ink colour of the strokes. The participant also mentioned that the ink properties should be placed in an easily accessible location and in the post task questionnaire suggested that having a mini palette on the interface would improve the tool. This change was implemented before participant three did the study and was received well as there were no complaints by the other participants concerning this area after it was implemented.

A technical problem emerged when participant four selected a branch for moving and then switched to 'select node' moving mode. For task seven, participant four attempted to relocate the entire branch using the 'select branch' mode. There was ink overlap after the branch move, so the user decided to use the 'select node' mode to move the node to another position. However, the whole branch was selected at the time and when the user moved the whole selection, the system failed to reflow all the ink strokes. There was no problem with the branch move itself, but the problem lay in the mode change. The select node mode only caters for the movement of single nodes and not designed for moving multiple nodes at the same time. The system already has a node check in the 'select node' mode that only allows one node to be selected at any time. However, we did not account for ink strokes already selected by the user when the mode was switched. A simple solution would be to empty the ink strokes from the selection list when the user switches between nodes.

And lastly a technical problem was discovered concerning the reflow of the connector. Participant two caused a particular connector to behave abnormally when trying to move a node. This was caused by the connector end point being not close enough to the line between the two node centres as shown in Figure 43.

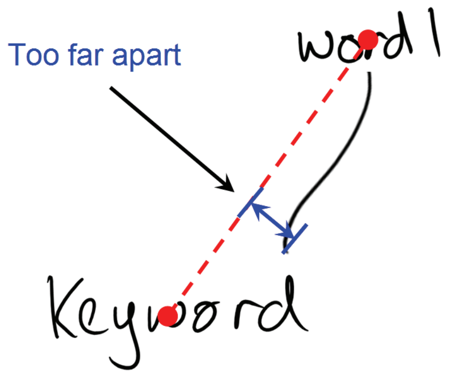


Figure 43 Connector problem from usability tests

The connector extension/compression relied on the connector lying on the horizontal plane. Since the connector needs to be rotated to horizontal plane for scaling and the angle is determined by the line between the centre points of the nodes, any deviation of the connector from this line will cause the connector to become scaled incorrectly.

Some positive signs observed were the way that the users intuitively used the other end of the stylus to erase ink. None of the participants used the delete widget when correcting mistakes on the canvas. Also, once users realised that selecting an ink stroke of a node automatically selects the whole node, they never went back to trying to encircle the complete node.

Questionnaire

The following are the results from the questionnaire that the participants filled out after completing the test.

	Participant Rating 1(never) - 5 (always)							
The participant uses mind-maps	3	1	1	3	3	3	1	3
The participant sketch ideas for making anything concrete	5	3	2	4	5	3	4	3
The participant thinks mind-maps are useful	4	3	4	5	4	3	2	4

	Scenario 1: General Inking (1 Strongly Disagree – 5 Strongly agree)							
The exercise was enjoyable	4	4	5	5	4	3	4	4
Easy to read and comprehend	4	5	3	5	5	4	4	5
Task length was acceptable	3	4	4	5	4	5	4	5
Creating nodes was easy	4	5	5	5	5	4	5	5
Creating connectors was easy	4	5	5	5	5	4	5	5
Adjusting the mind-map was easy	4	4	5	4	4	4	5	5

Reviewing the mind-map was easy	3	5	5	5	5	4	5	5
The mind-map application was enjoyable to use	3	4	5	5	4	4	4	5
The mind-map application was practical to use	3	3	5	5	4	4	3	4
Overall Rating of this experience (1-10)	6	6	9	10	7.5	8	9	9

	Scenario 2: Ink Reflow (1 Strongly Disagree – 5 Strongly agree)							
The exercise was enjoyable	4	4	5	5	5	4	4	4
Easy to read and comprehend	4	5	3	5	5	4	4	5
Task length was acceptable	3	4	4	5	4	5	4	5
Creating nodes was easy	4	5	2	5	5	4	5	5
Creating connectors was easy	4	5	5	5	5	4	5	4
Adjusting the mind-map was easy	4	4	3	4	3	5	5	4
Reviewing the mind-map was easy	3	5	3	4	5	4	5	5
The mind-map application was enjoyable to use	3	3	5	5	4	4	4	5
The mind-map application was practical to use	3	3	5	5	3	4	3	4
Overall Rating of this experience (1-10)	6	6	7	10	8	8	9	8

Table 7 Post-test questionnaire

From the user's questionnaire we can see that the participants tend not to use the mind-mapping technique but sketch ideas to plan out designs themselves. Also, they tend to agree that the mind-mapping technique can be useful.

For the general inking scenario, there was good feedback from the participants. Most found the tool satisfactory to create their mind-maps. The connectors and nodes were all rated well as the users did not have problems inking. Also, adjusting the map was easy and the user was quick to flip the stylus over and use the eraser. Reviewing mind-maps were also rated rather high as it was easy for the user to zoom out to view the complete map. With the complete map laid out before the participants, it was effortless for them to quickly find keywords on the map, enabling them to complete the task with relative ease.

The overall rating of the experience leans towards a positive response from the participants. The lower rating given by the early participants may be due to the usability issues that appeared during the tests that severely hurt the user experience. And the last participants generally had a better experience as the major problems were fixed right away after being identified. On the whole, all participants found the exercise enjoyable which is highly promising.

5.3 Summary

It is clear that the usability testing identified interaction problems that were overlooked during implementation. This was due to the fact that the implementation process focuses more on the actual working features of the mind-mapping tool. Several usability issues regarding the “visibility of the system state” and “recognition not recall” heuristics were uncovered. These problems were verified by the users becoming frustrated and commenting on these issues. We have amended the software to alleviate these two problems by moving the system functions to a visible place on the interface and displaying the various mode changes clearly for the user to see. Apart from these issues, there was a positive trend in the questionnaire that shows promise for our mind-mapping tool, with some participants thoroughly enjoying their interaction experience.

Chapter 6

Discussion

Mind-mapping is a unique method of brainstorming and idea generation. It was originally invented by Tony Buzan and is based on ‘radiant thinking’, an notion that describes the way the human brain functions, associating ideas with one another via ‘relationship hooks’ (Buzan and Buzan 2000). In this thesis, we have identified potential issues present with computer based mind-mapping tools. We hypothesise that the user becomes distracted by tool operation similar to user interface design (Goel 1995). However, Bailey and Konstan (2003) and Plimmer and Apperley (2003) discovered that sketch based computer tools do not suffer as much comparatively. We predict that widget-based computer tools are detrimental to the idea generation process as the users become immersed in fine tuning their mind-map. The motivation for our project is to create a powerful mind-mapping tool that can replace widget-based computer tools with a more user friendly interaction thus maintaining the traditional pen and paper method of mind-mapping. The approach we adopted was to first explore the literature on current mind-mapping tools; sketch tools that have already been implemented by others; and to research on ink recognition and editing challenges and that may be useful. From this literature we defined the requirements for a sketch based mind-mapping tool and went on to design, implement and usability test a prototype.

6.1 Tool Discussion

The literature on mind-mapping suggests that mind-mapping is indeed a useful mechanism a wide variety of applications. Proven applications of mind-mapping include note taking and studying, critical thinking planning and teaching. Studying and note taking and results from the study by Farrand, Hussain et al. (2002) have showed that memory retention is improved with a 10% better recall than non mind-map techniques. Mind-mapping has received praise from nursing field (Mueller, Johnston et al. 2002) for facilitating critical thinking in patient care plans. Planning is another instance of a mind-map application. Students have used mind-mapping in a team management scenarios for business case studies (Mento, Martinelli et al. 1999). These studies demonstrate the powerful capability of mind-mapping that applies both sides of the brain to cultivate critical thinking, studying, problem solving and memory recall.

Traditionally, mind-mapping has been carried out using pen and paper but with the digital age, there are computer based tools that support this technique. Four popular mind-mapping applications have been examined to discover their methods of mind-map creation and to determine the pros and cons. From this we have verified that there is a general trend among these tools. These applications are very restrictive in many ways; such as the positioning of the nodes are preset and sub nodes are minimised and attached next to its parent. A study conducted by Ware, Purchase et al. (2002) has shown that a grid like layout is not easily understood. Another problem with these tools is the constant use of widgets to perform tasks and editing node properties. This distraction interrupts the thinking process of the user as their presence encourages the user to arrange their mind-map components. Although there are drawbacks to using a computer based mind-mapping tool, there are a few redeeming features and benefits that arise such as editing support and the ability to store digital copies have encouraged the use of these tools.

We have also examined a selection of sketch tools and the benefits they provide in web page and multimedia design over widget based computer tools. DEMAIS (Bailey and Konstan 2003) is an example of such a sketch tool that has been rated more favourably than its computer based counterpart. Their study has shown that their sketch tool fostered creativity more than the computer based version. In addition, Yeung (2007) has found that the increase in formality for the design process decreases the number of alterations users make. Coyette, Vanderdonckt et al. (2007) have also found this to be true in web design in that users that use a low fidelity tool tend to reiterate over their plans more. These studies show that sketch interface minimises the loss of the pen and paper benefits while allowing computer support.

To implement a sketch based tool, we must first look at the problems and challenges that are present in ink oriented tools. The challenges identified in a sketch tool were ink division, ink grouping, ink structure and ink reflow.

When a user sketches on the canvas, the system must be able to recognise and differentiate the type of strokes. This is where ink division is required and there have been numerous ideas and measures suggested from related works. Sezgin, Stahovich et al. (2001) suggested using measures such as Ink speed, ink curvature used to detect corners to differentiate shapes. Lank, Thorley et al. (2000) have used the size of the bounding box as a measure in separating UML components and text as the components are generally much larger. Gross (1994) proposed an innovative method of recognising ink strokes by the order it is drawn on a 3 by 3 grid. Patel (2007) has created a tree based partitioning system that divides single ink strokes into text and drawing.

Stroke grouping is the next step to understanding the user's strokes and is an essential part of ink recognition as a single stroke may not have any meaning whilst a cluster of strokes may make up a keyword. Spatial and temporal grouping are the two measures to group ink strokes. Chiu and Wilcox (1998) used agglomerate clustering technique where the closest ink strokes, spatially and temporally, are grouped together one by one. Landay and Myers (1995) used ink stroke enclosure to group ink where strokes that lie inside another are grouped together. From these examples, the general consensus is that the spatial relationship between ink strokes is a very reliable method of grouping. Temporal location is also used but cannot be used standalone as there are instances where the temporal data may be misleading.

After the ink has been grouped into mind-mapping components can the mind-map hierarchy be found. In ink structuring, Freeman and Plimmer (2007) have proposed a uinque way to determine the structure of diagrams by overlaps between a connector and a node. Also, they suggested using implementing a list to store the relationships between nodes via the connection links.

Ink reflow is the last challenge of sketch tools. All the above processes must occur before any ink reflow can be accomplished. A structure is required in order to effectively reflow ink in a coherent manner that is consistent to the mind-map. Golovchinsky and Denoue (2002) describe a simple technique for resizing annotations on digital documents. Arvo and Novins (2005) discuss the possibility of using linear interpolation for morphing ink connectors to preserve the look of the mind-map. Reid, Hallett-Hook et al. (2007) have based their connector reflow algorithm and addressed several problems that were present in Arvo and Novins' (2005) work. They have created an improved method to maintain the appearance of a reflowed connector. With a better understanding about the challenges of sketch based tools and digital ink, we then proceed to the design and implementation of our mind-mapping tool.

6.2 Implementation Discussion

The prototype we have constructed allows the user to sketch mind-maps with relative ease. Our mind-mapping software was developed in C# using Visual Studio because the .NET languages already contain digital ink support libraries. Also C# has a well established knowledge base and support such as MSDN. By implementing our tool in C#, the user only requires the .Net framework and the digital inking libraries on their computer for compatibility with our mind-mapping tool. We have chosen to construct our mind-map software with its usage on a Tablet PC. Our application is designed with using the stylus in mind and with the tablet in the clipboard mode. By doing this, we remove the inconvenience of switching between mouse and keyboard and vice versa while creating a mind-map. Our goal is to mimic the pen and paper environment and the tablet PC neatly adopts these characteristics as the user can perform all the system functions via the stylus. Although we have designed this software on a Tablet PC and the intended use is on such a machine, we have catered for the use on other systems without the benefit of a stylus. This allows users to sketch and operate the features of the tool by using only their mouse. Commands only applicable on a stylus such as the erase feature is also incorporated into the system interface for this very reason.

Based on the literature reviewed, we concluded that the components required for our application are open nodes which contain purely text, enclosed nodes which are text and a circle, and connectors. We then constructed personas and scenarios that demonstrate how users would utilise a mind-mapping tool. From these scenarios, we determined the criteria of the system using a use case diagram and creating a list of system requirements. The tool we have created has interactive space is similar to that of the traditional mind-mapping method as users only have to sketch on the large canvas. Furthermore, it provides the digital support that is not present in a pen and paper environment. The system provides the intelligent ink support not found in the widget based mind-mapping tools. The system is designed such that when the user draws a stroke, it gets passed into the divider to sort the ink strokes into shape and text strokes. Once divided, the strokes are then grouped together to form the components of the mind-map. After ink grouping, the structure of the mind-map is established. The instantiation of the mind-map hierarchy will allow intelligent ink support. This process is all completed in the background as the user writes, providing an unobtrusive environment for constructing a mind-map.

To create a mind-map, the user just writes on the large canvas, beginning with the main idea. Then a connector is drawn and a new keyword is written. This process of idea generation continues until the user has completed the mind-map. Our tool encourages a distraction free environment for the user as they do not have to worry about inserting mind-map components onto the canvas. As the user sketches on the canvas, the system is automatically categorising and grouping each ink stroke. The downside is that there is less data to work with compared to analysing the final completed mind-map. With less data, the system is more likely to misclassify ink strokes. However, the beauty of eager recognition is that it allows a smooth transition from writing ink to intelligent editing. Upon writing an ink stroke, the stroke data is passed to the ink divider engine based upon Patel's (2007) ink divider engine. The divider was a generic divider for diagrams and only differentiated between text and shapes. To cater for the domain of mind-mapping, a measure involving the bounding box width in the partitioning tree was adjusted. Also, the divider was extended to differentiate shapes into two categories, circles which represented enclosed nodes and lines that represented connectors. Ink strokes are then grouped according to their context with existing nodes on the canvas. Our technique ensures that strokes that are not drawn in succession are correctly grouped. We believe that this method enables text to be efficiently grouped and can be relevant to any text recognition application, being able to group paragraphs of text with ease. The method we implemented above grouped ink strokes that are on the same horizontal plane. Because of our approach, keywords that are written below one another become classified as separate nodes. Hence we have implemented node grouping to compensate for this issue. The next step for grouping would be the grouping of ink that is written at an angle. This is a challenge as the current system uses the bounding box of strokes which is insufficient to group ink strokes at an angle effectively.

Only when the ink strokes have been divided and grouped can the mind-map hierarchy be ascertained. The overall hierarchy of a mind-map is determined by the links that the connector makes. The benefit of such a hierarchy is that it paves the way for ink manipulation. However, a limitation of such a method is that it is very dependant on proximity measures and the relocation of a connector can easily disrupt the structure of a branch. Since the system stores the relationships between nodes, it makes intelligent ink move and file export possible. The exportability and ability to intelligently manipulate ink of our tool provides is truly an advantage over paper tools,

With a proper mind-map hierarchy, intelligent ink editing can be performed. We have identified 3 types of editing that the user will do; simple editing, node move and branch move. Simple editing is where a number of ink strokes are selected and changed such as move, size, colour and thickness properties. There is no intelligent editing present in this mode to allow the user to freely edit their mind-map. Node move is where the user only desires to relocate a node to another location while keeping the other nodes intact. This requires intelligent ink reflow as the connectors that are linked to the node needs to be reflowed also in order to keep the mind-map structure intact. The last mode we used was branch move. This is where a whole branch of the mind-map is moved under a new location on the canvas. Ink reflow is essentially a difficult process. The biggest challenge is to reflow the ink strokes while retaining the hand drawn appearance, maintaining the mind-map structure and at the same time preserving the overall appearance of the mind-map. Our tool effectively moves the components of the mind-map while retaining the structure of the mind-map.

As with all prototypes, we have identified limitations of the mind-mapping tool we have constructed. The accuracy of the ink recognition is a significant issue. Although the application performs well, the accuracy of the recognition is not perfect and ink strokes are not always categorised correctly. Due to the unrestricted nature of a pen and paper environment we are trying to emulate, the number of different strokes that the user can draw is limitless. For example, there are an infinite number of ways to draw a connector to link one node to another and each stroke has different properties and to sort them correctly is no small task. And because each user sketches and writes in different sizes and styles on the canvas, misclassifications will ultimately occur as our heuristics can only cater for the majority of ink strokes drawn. We have minimised this recognition problem in two ways. The first is the analysing ink strokes twice, passing the last ink stroke through the divider again when a new stroke has been drawn. The second technique we have used to reclassify strokes based on its context on the canvas to correct misclassifications.

Another limitation is the single stroke recognition of the divider. If the user accidentally lifts up the stylus when writing, the separation of a supposedly single stroke would essentially result in a misclassification as each stroke is treated separately. Multiple stroke recognition could potentially overcome this problem but would open up more problems and challenges hence we allow the user correction features and to manual ink classification as a quick remedy. Also, as the number of strokes drawn on the canvas increases, grouping each stroke takes a little bit longer to complete as there are more nodes to compare.

Lastly, there is limited intelligent reflow of ink when ink strokes overlap. The only reflow implemented due to ink overlap is the overlapping of nodes. When a node overlaps another, the other node is pushed away a certain distance. The system does not take into account the overlap of connectors and there exists layout algorithms that successfully eliminates such intersections (Reid, Hallett-Hook et al. 2007). However, these layout techniques completely rearrange the mind-map appearance. We considered this negative for the user as such a significant change in the mind-map look and feel would be disorientating for the user. Furthermore, the intelligent reflow would allow the user no control over the layout and forces the user to adapt to the map arrangement and not the other way around.

Overall, we have created a unique tool for mind-mapping. Drawing upon the literature and our design, we have created a tool that bridges the gap between a widget based tool and the traditional methods of mind-mapping. The traditional method of mind-mapping on paper is intuitive and easy to utilise. However, pen and paper methods lack the flexibility to edit the mind-map in a significant manner. We have mimicked the simplicity of pen and paper and provide similar affordances. However, our tool also supports many beneficial features, allowing the user to move their sketches around the canvas and change the ink properties which is not possible on paper. With our intelligent ink support for manipulating branches and nodes, it allows the user's ideas to be moved and changed which is not possible on paper. The widget based mind-mapping tool provides a rich set of features for the user to aid their mind-mapping process. Our tool is similar in this aspect, giving the user a set of editing features. Yet our tool differs as it creates a writing environment that encourages the rapid brainstorming process which the widget tools do not effectively promote. Users are not interrupted by inserting connectors and nodes. They just sketch. They need not worry themselves with adding various mind-map components onto the canvas. The underlying system will automatically classify the ink in the background as they write. This allows intelligent editing the moment editing mode is enabled. We realise that ink recognition and manipulation is an intrinsically difficult task and errors/faults will ultimately occur. We have tried methods to minimise the amount with our novel methods and enabling manual correction by the user as the last resort. It is hopeful that the methodologies discussed today will be applicable to the other fields of ink diagramming where effective grouping and where ink manipulation is required such as organisation charts and flow diagrams.

6.3 Evaluation Discussion

We have conducted a usability evaluation on our prototype to reveal the usability issues that have been missed. As the mind-map implementation focuses on the technical side of the software, often the usability issues are overlooked. The test objectives we plan to achieve are split into two categories, general inking and ink reflow. The first category tests if the user can use the application to draw a mind-map, alter the sketches they have drawn and erase any strokes they have drawn. The ink reflow category attempts to assess if the user can selecting a group of strokes and edit their properties, select a node and relocate it and lastly choose a branch a attach it to another part of the mind-map.

To test each of the categories, the test is split into two scenarios for the participant to complete. The first scenario involves creating a mind-map categorising a collection of pictures. The second scenario involves the user editing a pre-existing mind-map and utilising the mind-mapping tool’s editing features. The participants must be trained to ensure they are accustomed to the controls of out software. Hence for the tasks in each of the two categories, the user is guided through a similar scenario with similar tasks to allow the participant to familiarise themselves with the layout and features of the tool. Also, as the two test scenarios are completed separately, the participant are only trained enough to complete one scenario at a time. This is to prevent the overload of information for the user and to achieve a better accurate usability result.

Before commencing the actual test, a pilot test was run to trial the exercise to verify the tasks were of equal difficulty and other improvements to the testing process to allow more accurate results. The main hitch in the pilot test was user training. As we were familiar with the tool itself, our views on the amount of training was not sufficient. Also, our original plan of demonstrating the participant through did not provide enough hands on experience for the user to remember the features. Hence, in the actual test the user was given complete control over the Tablet PC and in the training sessions, the participants were walked through the scenarios. It is interesting to note that without constraints (such as lines on a page), the user's actions become less predictable. An example is the pilot tester writing at odd angles on the canvas. As our tool only deals with writing in a horizontal plane, this is not accounted for by the system and the ink recognition becomes ineffective. And so it was noted to relay a message to each of the participants beforehand to ensure they write in a relatively horizontal manner.

During the test, participants are observed via the tester's visual feedback and video recording software that records the tablet display. After the test, the participants complete a post-test questionnaire regarding the exercise they completed. From the analysis from the data collected, two usability issues, 'visibility of system state' and 'recognition not recall', described by Nielsen and Mack (1994) showed up on occasions. An example was the mode change feature in the ink reflow scenario. A participant was observed selecting a particular mode several times as the system did not give visual feedback that the mode has changed. A similar example was the lack of update for the change in the ink colour after being adjusted by the user.

There were, however, positive signs observed during the exercise as users appeared relaxed while creating the mind-maps. Also is the fact that users opted to use the reverse side of the stylus to erase instead of the using the widget is highly promising. There were also positive results of the questionnaire, with the participants rating the ease of use highly. They did not encounter any difficult problems with sketching on the canvas and with the ability to zoom out, there were no difficulty reviewing the mind-map, enabling them to quickly complete the tasks. It was pleasing to note that after amending some of the issues that were identified by the first few participants showed an increase in the rating of the tool by the remaining participants. Overall, the mind-mapping tool shows promise given the constructive feedback from the participants.

From these comments, we are pleased at the performance of our tool. What remains is a formal comparative evaluation of our mind-mapping tool with existing widget based tool. It would be interesting to observe how our IMM tool would fare against these tools and if our low fidelity tool allows a more efficient method of mind-mapping.

6.4 Future Work

The following describes the possible short and long term directions that the mind-mapping tool can head towards. Extending our IMM tool in these areas would provide an improvement to the user experience.

Expanding the recognition components

Currently there are three components that are recognised; text, line and circle shapes. The recognition engine could be extended to cater for more mind-map components such as drawings or icons. The addition of these components could possibly facilitate user creativity.

Mind-map attachments

The computer based mind-mapping tools that were examined allowed the attachment of other files into their mind-maps. Such attachments included pictures, documents, spreadsheets and URLs to websites. Our tool currently does not support external attachments to the mind-map. A feature like this will allow the user to enrich the mind-map by supplementing additional information to their ideas.

Ink recognition efficiency

This is one area that requires improvement. In the current prototype, we have discovered the ink grouping algorithm is not as efficient as the mind-map increases in size. When the system groups the ink strokes, it is compared to every single node on the canvas for a possible grouping. With increasing number of nodes, there would be an increase in the time taken and for large mind-maps, this becomes a significant issue. If this comparison time can be minimised by finding a solution to efficiently group the strokes, the user experience would be further improved.

Improved grouping

Currently, the system effectively groups the appropriate ink strokes together on a horizontal plane. An effort has been made to account for keywords spread over multiple lines by grouping nodes together. An improvement to the existing setup would be to research and extend the current grouping technique to allow the system to group text which lies at different angles. This will allow users more flexibility when writing on the canvas and not interrupt the idea generation process as the user does not have to actively remember to write words on a level plane.

Ink reflow

As mentioned in the discussion, the current system only manages node/node intersections and does not cater for node/connector or connector/connector intersections. We have opted to not use to 'untangle' these intersections in fear of destroying the users overall layout of the mind-map. A possible research area would investigate existing layout algorithms and to see if they can be applied to specific areas of the mind-map or to relocated nodes only. Applying a layout algorithm on certain parts of the mind-map would preserve the user defined look of the rest of the canvas.

Merging of modes

A nice extension to the IMM tool would be to merge the editing modes together. By creating a seamless switch between modes would allow users to as opposed to interacting with the widgets. Utilising the other mouse buttons to differentiate between the three editing modes could possibly be an option for improvement to the system. For example, the use of right click selection could be designated as intelligent ink support and selecting a single node using this method could specify a node selection and when more than one node is selected could indicate a branch selection.

Formal evaluation

From the results of the usability testing, our prototype showed promising results. The users found the creation of the nodes and connectors easy to create and the tool enjoyable to use. It would be interesting to see how the mind-mapping tool fares with existing computer based tools. A formal comparative evaluation of our mind-mapping tool would prove useful at finding out how our mind-map stands with the commercial products while at the same time, would provide valuable feedback on the ineffective side of the software.

6.5 Summary

In this chapter we have overviewed the content of the thesis, briefly summarising the goal of this project and our motivations. We first found out how mind-mapping works and the advantages in critical thinking and idea generation they facilitate. Existing computer based mind-mapping tools were examined and their advantages and disadvantages revealed. We then discussed the literature concerning sketch tool and how the low formality of these tools allow better performance in design over their computer based counterparts. However, sketch tools do have various challenges in the form of ink division, ink grouping, ink structure and ink reflow. The surrounding literature is then thoroughly scrutinised for problems to pay attention to and possible techniques that can be incorporated into our mind-mapping software. We then discussed the implementation of our prototype, describing the methodologies which we adopted and the difficulties that surfaced during the course of the software development. We then discussed the usability issues of the prototype with two of Nielsen's usability heuristics being identified during our usability testing. Lastly we have discussed the future directions that this software can take in the technical aspects and in formal evaluation.

Chapter 7

Conclusions

Mind-mapping is a very valuable technique for generating ideas. It has a wide variety of applications and the success of this technique is shown especially in education, where students use have used mind-mapping for critical thinking, idea generation, planning and studying in different fields of study. Mind-mapping has traditionally been carried out using pen and paper. With the advent of the digital era there is a need to convert a hand drawn hard copy of a mind-map into an electronic form. We have created a unique mind-mapping tool that draws on the benefits of computer based applications and the informal nature of sketching tools. We set about this difficult task by firstly examining the numerous computer based mind-mapping tools in existence and comparing their pros and cons, keeping in mind the things to watch out for. We have also researched the various literatures surrounding sketch tools, again analysing the benefits of using such a medium. We then progressed into the ink support that would become the backbone of our intelligent system. With a firm knowledge base of sketch tool and its challenges, personas and use case scenarios are used in conjunction to determine the necessary features for our software. Our system is implemented following these design decisions and our prototype was usability tested. From our IMM tool, we have discovered novel techniques for our ink manipulation. The following describes the contribution from our research:

Digital support for mind-maps

The tool we have created attempts to bridge the gap between an informal hardcopy document and a computer system. We have taken the benefits from both mediums to allow the benefits of low fidelity tools whilst features present in a computer application. Users using this on the Tablet PC can use the software in a clipboard mode and a stylus, giving a similar feel to the traditional method of mind-mapping while being able to utilise editing features comparable to its computer counterpart.

Ink grouping

Ink grouping is another area where our contributions lie. It is a difficult process as the system must know what ink strokes to group and what to ignore. In many instances, the wrong ink grouping will cause major problems within the system. We have created our own grouping algorithm that effectively groups ink strokes that lie on a relatively horizontal plane with each other and text that lie in a paragraph. By providing a good basis for ink grouping, a more coherent structure can be obtained. In addition, we have implemented our own methods of context grouping, a way to correct misclassification of strokes based on their context on the canvas. These techniques can be applicable to other diagramming domains like UML diagrams and organisation charts and flow diagrams. By having well defined components, the difficulty of ink grouping and ink division can be alleviated.

Ink reflow

We have also made advancements in ink reflow. Our connector reflow is based upon Reid et al.'s strategies while using additional measures to ensure that the connector does not overlap the node it is linked to. Also, we have implemented node reflow where the node being shifted does not overlap the other keywords on the mind-map. However, for mind-map branch reflow, we have produced ourselves a novel technique of reflowing a complete branch while retaining the user defined appearance. Our research here may be applicable for other domains such as flow diagrams etc that require node and connector reflow.

With our intelligent mind-mapping software, users receive the advantages of both types of tools. With our tool, the user only sketches on the canvas without the need to specifically insert mind-map components. The system automatically recognises the strokes the user sketches and classifies them according to the corresponding mind-map components. This is not possible with the widget based tools which require the user to add new components which interrupts the idea generation process. Albeit similar to the pen and paper interactive style, the ink editing features of our system contains allow the user flexibility that is unavailable on pen and paper. Our mind-mapping software shows great promise and the feedback from the usability study demonstrates that this tool has potential given further improvements. It is hoped that the problems that have been identified here in this thesis would be of value to others. We are confident that other researchers attempting ink manipulation may find the task less daunting by reviewing the literature we have examined and the techniques that were attempted here while watching out for the issues that we have encountered. We hope that by merging the informal nature of sketching and the digital support of widget based tools, we allow users the freedom of traditional mind-mapping techniques with the affordances of computer editing and support.

Bibliography

Alvarado, C. and R. Davis (2004). SketchREAD: a multi-domain sketch recognition engine. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM.

Arvo, J. and K. Novins (2005). Appearance-preserving manipulation of hand-drawn graphs. Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. Dunedin, New Zealand, ACM Press.

Bailey, B. P. and J. A. Konstan (2003). Are informal tools better? Comparing DEMAIS, pencil and paper, and authorware for early multimedia design. Proceedings of the SIGCHI conference on Human factors in computing systems. Ft. Lauderdale, Florida, USA, ACM Press.

Brinkman, A. (2003). "Mind mapping as a tool in mathematics education." The Mathematics Teacher **96**(2): 6.

Buzan, T. and B. Buzan (2000). The Mind Map book. London, BBC Worldwide Limited.

Chiu, P. and L. Wilcox (1998). A dynamic grouping technique for ink and audio notes. Proceedings of the 11th annual ACM symposium on User interface software and technology. San Francisco, California, United States, ACM.

Chung, R., P. Mirica, et al. (2005). InkKit: a generic design tool for the tablet PC. Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: making CHI natural. Auckland, New Zealand, ACM Press.

ConceptDraw (2007). ConceptDraw MINDMAP: Mind map software. Retrieved on 25th July 2007 from: http://www.conceptdraw.com.

Coyette, A., J. Vanderdonckt, et al. (2007). SketchiXML: A Design Tool for Informal User Interface Rapid Prototyping, Springer Berlin / Heidelberg.

Farrand, P., F. Hussain, et al. (2002). "The efficacy of the 'mind map' study technique." Medical Education **36**(5): 426-431.

Freeman, I. J. and B. Plimmer (2007). Connector semantics for sketched diagram recognition. Proceedings of the eight Australasian conference on User interface - Volume 64. Ballarat, Victoria, Australia, Australian Computer Society, Inc.

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

Golovchinsky, G. and L. Denoue (2002). Moving markup: repositioning freeform annotations. Proceedings of the 15th annual ACM symposium on User interface software and technology. Paris, France, ACM Press.

Gross, M. D. (1994). Recognizing and interpreting diagrams in design. Proceedings of the workshop on Advanced visual interfaces. Bari, Italy, ACM.

iMindMap (2007). iMindMap™ Official Mind Map Software. Retrieved on 25th July 2007 from: http://www.imindmap.com.

Inspiration (2007). Inspiration Software Inc. Retrieved on 25th July 2007 from: http://www.inspiration.com.

Junfeng, L., Z. Xiwen, et al. (2005). Sketch recognition with continuous feedback based on incremental intention extraction. Proceedings of the 10th international conference on Intelligent user interfaces. San Diego, California, USA, ACM.

Kara, L. B. and T. F. Stahovich (2004). Hierarchical parsing and recognition of hand-sketched diagrams. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM.

Landay, J. A. and B. A. Myers (1995). Interactive sketching for the early stages of user interface design. Proceedings of the SIGCHI conference on Human factors in computing systems. Denver, Colorado, United States, ACM Press/Addison-Wesley Publishing Co.

Lank, E., J. S. Thorley, et al. (2000). An interactive system for recognizing hand drawn UML diagrams. Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research. Mississauga, Ontario, Canada, IBM Press.

Mento, A. J., P. Martinelli, et al. (1999). "Mind mapping in executive education: applications and outcomes." The Journal of Management Development **18**(4): 390-407.

Mindjet (2007). Mindjet: Software for Visualizing and Using Information. Retrieved on 25th July 2007 from: http://www.mindjet.com.

Mueller, A., M. Johnston, et al. (2002). "Joining mind mapping and care planning to enhance student critical thinking and achieve holistic nursing care." Nursing Diagnosis **13**(1): 24-28.

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

Nielsen, J. and R. L. Mack (1994). Usability Inspection Methods, John Wiley & Sons, Inc.

Patel, R. V. (2007). Exploring better techniques for diagram recognition. Department of Computer Science, University Of Auckland. **MSc**.

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

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

Priest, R. and B. Plimmer (2006). RCA: experiences with an IDE annotation tool. Proceedings of the 7th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: design centered HCI. Christchurch, New Zealand, ACM.

Purchase, H. C. (1997). Which Aesthetic has the Greatest Effect on Human Understanding? Proceedings of the 5th International Symposium on Graph Drawing, Springer-Verlag.

Reid, P., F. Hallett-Hook, et al. (2007). Applying layout algorithms to hand-drawn graphs. Proceedings of the 2007 conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artifacts and environments. Adelaide, Australia, ACM.

Rubine, D. (1991). "Specifying gestures by example." SIGGRAPH Comput. Graph. **25**(4): 329-337.

Sezgin, T. M. and R. Davis (2005). HMM-based efficient sketch recognition. Proceedings of the 10th international conference on Intelligent user interfaces. San Diego, California, USA, ACM.

Sezgin, T. M., T. Stahovich, et al. (2001). Sketch based interfaces: early processing for sketch understanding. Proceedings of the 2001 workshop on Perceptive user interfaces. Orlando, Florida, ACM.

Sivathasan, S. and L. H. Ho (2005). Using mind maps in university teaching. Herdsa Conference, Sydney, Australia.

Thorsten, P., M. Carsten, et al. (2002). Developing CSCW tools for idea finding -: empirical results and implications for design. Proceedings of the 2002 ACM conference on Computer supported cooperative work. New Orleans, Louisiana, USA, ACM.

Vidal, V. V. (2004). "The Vision Conference: Facilitating Creative Processes." Systemic Practice and Action Research **17**(5): 385-405.

Ware, C., H. Purchase, et al. (2002). "Cognitive measurements of graph aesthetics." Information Visualization **1**(2): 103-110.

Ye, M., H. Sutanto, et al. (2005). Grouping Text Lines in Freeform Handwritten Notes. Eighth International Conference on Document Analysis and Recognition.

Yeung, L. W. S. (2007). Exploring beautification and the effects of designs' level of formality on the design performance during the early stages of the design process. Department of Psychology, University of Auckland. **MSc**.