2000

MapCAST : Real-Time Collaboration With Concept Maps

Brett Greay
Edith Cowan University

Edith Cowan University

Copyright Warning

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

mapCAST : Real-Time Collaboration with Concept Maps.

Brett Greay
Bachelor of Arts (Interactive Multimedia), Honours
Faculty of Science Technology and Engineering
School of Communications and Multimedia
Edith Cowan University
January 2000

Abstract

This thesis describes the development of the application mapCAST, a computer-based concept-mapping tool that allows synchronous collaboration via TCP/IP networks, such as the Internet. The useability and feasibility of mapCAST as a computer-based tool was examined and analysed in a real-world situation. Results indicate that mapCAST is successful as a collaborative tool in a situations involving knowledge organisation, but lacks certain functionality that many Macintosh users are accustomed to.

Key words: concept maps, collaboration, Internet

Declaration

I certify that this thesis does not, to the best of my knowledge and belief:

- incorporate without acknowledgment any material previously submitted for a degree or diploma in any institution of higher education;
- contain any material previously published or written by another person except where due reference is made in the text; or
- contain any defamatory material.

Signed 

Brett Greay, January 2000

Acknowledgments

I would like to thank my supervisor, Dr Ron Oliver, for offering helpful suggestions while this thesis was being developed, and for purchasing the books and software, without which mapCAST would have not being developed.

Thanks also to Dr Arshad Omari, for answering my technical questions, offering suggestions and solving my math problems. Never underestimate the power of a high school geometry book.

I must also thank Joe Luca for the loan of his PowerBook G3 during network testing.

Finally, I must acknowledge the REALbasic NUG (Network User Group) mailing list for answering all technical questions, and being a general knowledge pool for everything related to REALbasic. Mailing list details are available from the REALbasic web page, http://www.realsoftware.com.

Table of Contents

1.0 Introduction

The exponential growth of the Internet has allowed new ways of communicating and collaborating in many areas of society. In particular, educational domains, especially those involving distance education, have benefited from the different ways in which material; instruction, collaboration and feedback can now be delivered.

With new and existing Internet technologies facilitating new interactions and flexible delivery, real-time collaboration with tools such as computer-based concept maps can now be realised. Concept maps, whether electronic or the traditional 'pen and paper' type, are visual expressions of one's knowledge. Concepts, or 'ideas', are connected via relationships to form propositions on the subject matter. The advantages of electronic concept maps can now be applied in different domains, such as open learning.

mapCAST, a real-time collaborative concept mapping program, was developed to facilitate this realisation as an appropriate concept-mapping tool, and to allow the advantages of electronic concept mapping to be investigated.

Research Aims

The purpose of this study was to develop a functional piece of software, mapCAST, and to evaluate it as a collaborative tool. This study arose from opportunities afforded by new and existing technologies to explore network-able, collaborative concept mapping tools.

Concept maps, and the learning and organisational advantageous they posses, are not a new idea, nor is using them in collaborative situations. This research was done to realise the potential advantages that these two ideas can produce, and to explore the functional requirements needed, through the development of an easy to use network application for the MacOS. At this stage, a Windows version is not planned.

What was Achieved

The major achievement was developing mapCAST to the stage where it became a functional Macintosh application, suitable for collaborative concept mapping. Its success as a functional computer-based tool was determined by testing it in a real-world situation.

1.1 Concept Mapping

A review of concept mapping can be divided into two distinct categories. The first describes concept-mapping theory; followed by a description of some of the existing computer-based concept map tools and efforts to promote collaboration.

Concept Mapping Theory

Concept maps can be described as a visual expression of one's knowledge, or as Kremer and Gaines define, "an intuitive visual knowledge representation technique" (1996, p. 1). The art of concept mapping "is a technique for externalising concepts and propositions" (Novak & Gowin, 1984, p. 17). Figure 1 shows an example of a concept map. Maps consist of nodes, the concepts, and are linked by lines, the relationships, to form propositions and a semantic network. The visual construction of concept maps can also imply relationships through directional arrows or distance between concepts (Novak & Gowin, 1984, p. 35).

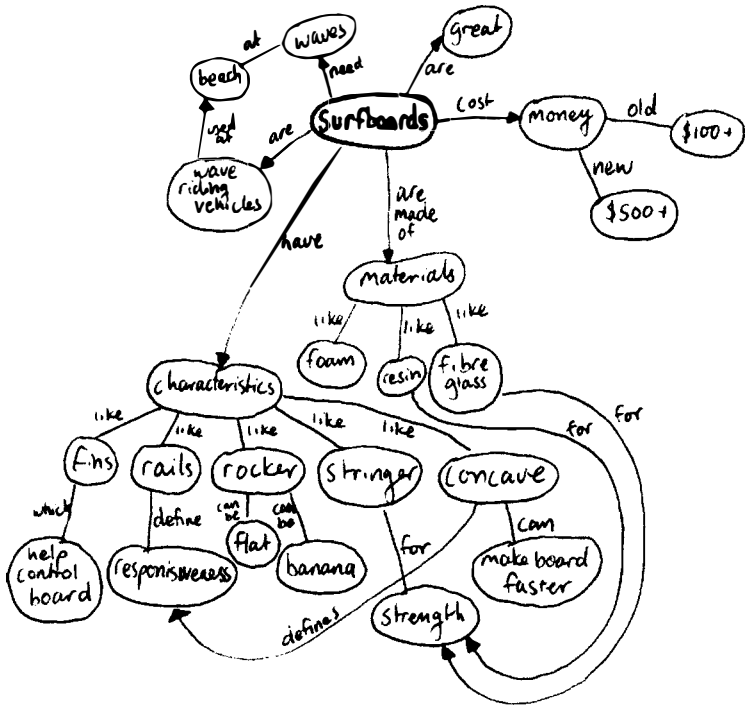

Figure 1: *A concept map on surfing, using pen and paper*

"Learning how to Learn" (1984) is perhaps the seminal text in concept mapping theory in education. It's authors, Novak and Gowin, are considered prominent developers of concept maps, relating them to Ausubelian educational theory. This theory is based on the premise that educating people is influenced by their existing

cognitive structure, or what they already know. Novak and Gowin state that concept map tools can ascertain this as they have "been developed specially to tap into a learners cognitive structure and to externalise what the learner already knows" (Novak & Gowin, 1984, p. 40).

Novak and Gowin clarify what a concept map *is* and what it *represents* to the creator. They explain that concept maps are "powerful tools for observing the nuances of meaning a student holds for the concepts embedded in his or her map" (p. 35) and that "concept mapping may be a creative activity", when done in groups it can serve as a useful social function that leads to discussion (p. 17 – 20).

At the time of publication of "Learning how to Learn", the personal computer industry was at its infancy and it is therefore safe to assume that electronic concept-mapping programs did not exist. Novak and Gowin justify revisions of maps as an important task, telling educators students must revise and redraw. This is one of the many advantages of a program like mapCAST, or any other electronic concept map program; a mundane task for today's application, but one certainly worth doing. Novak and Gowin:

> *Concept maps need to be redrawn. The first concept map a person makes is almost certain to have flaws...We find that a second map usually shows key relationships more explicitly...A secondary important reason for redrawing maps is to clean them up – to make them neater, correct spelling errors, and reduce clutter or crowding...[which increases] the meaningfulness of the composition (1984, pp.35-36).*

It is this ability of (relatively) easy revision that makes computer-based concept mapping less frustrating; "the electronic medium is flexible and forgiving" (Anderson-Inman & Horney, 1997, p. 1).

The advantages of concept mapping have been discussed in research literature from the late seventies, through to current studies. Novak and Gowin (1984) and students understanding of learning materials, as well as tools for evaluation; Anderson-Inman and Horney (1997) using maps for brainstorming and synthesising information as pre-writing strategies; Kremer and Gaines (1996), Gaines and Shaw (1995) and the University of Calgary's Knowledge Science Institute with their research into concept maps as hypermedia components and collaborative interfaces; Ferry, Hedberg and Harper (1997) with teachers organizing curriculum; Kennedy and McNaught (1997) and their use of concept maps for the design of multimedia learning tools.

Existing Concept Mapping Applications

Many computer-based concept mapping programs and languages exist. Each has unique feature sets and different process models for developing concept maps with a computer. The following reviews these features, and identifies common and desirable characteristics amongst them that a computer-based concept-mapping tool should employ to ensure success.

Rautama and Tarhio (1998) discuss "alternate ways to deliver concept maps" (p. 273) on the Internet via a web browser. They outline a proposed extension to HTML (HyperText Markup Language) named CMML or Concept Mapping Markup Language (p. 274). They suggest that CMML could be assimilated into future specifications of the HTML language to facilitate the sharing of concept maps without extra plug-ins or programs. CMML would be interpreted and rendered by the web browser, and being a markup language, could be authored as a text based language by anyone who understood the syntax (akin to HTML). Although not entirely interactive, it is certainly an astute suggestion to propose a defining standard for online concept maps that could easily be integrated into such a popular application as a web browser. It is this popularity of the 'network' that Rautama and Tarhio (1998) wish to take advantage of to share concept maps:

> *Traditionally, it has been difficult to share concept maps with people in distant locations…Computers and networks provide us with new methods for collaboration…Computer supported concept mapping makes the editing process of a map more flexible than the old pen and paper method. (p. 273)*

Sung, Chen, Lin & Chang (1998), concur that electronic concept mapping is advantageous over traditional methods, and based on this, discuss a computer based concept mapping learning system that was developed. They outline four disadvantages of conventional methods (p. 692-693):

1. Communication such as feedback and interactions between learners and instructors is inconvenient. Teachers cannot observe the construction process of the student and provide suitable feedback.
2. Construction of concept maps, especially those involving many propositions, can be difficult for students, especially novices.
3. 'Paper and pencil' concept maps are difficult to revise. Students can spend a majority of their time making the map clear and legible, neglecting the "learning characteristics [embedded] in the construction of [the] concept map." (Sung, Chen, Lin & Chang, 1998, p.692).
4. 'Paper and pencil' methods are not efficient for evaluation.

Twidale, Rodden and Sommerville (1994) offer some advantages of computer based concept maps, in their discussion of their program Designers Notepad:

- As a structure builds up, the maps can be arranged, clarifying the emerging structure.
- Alternative structures can be experimented with quickly and easily.
- Colour and shape may be used as for pen and paper, but can subsequently easily be changed (p. 109).

Twidale, Rodden and Sommerville in their discussion of Designers Notepad, outline research that is closer to mapCAST's intentions, namely collaboration, than similar texts. Again although real-time collaboration between users does not exist, the application running under a UNIX based system is quite advanced and evidently (as the paper suggests) easy to use. It implements such features as the ability to attach notes to each node, with an option to export all notes, and the ability to have sub-designs, ie a node that links to another embedded concept map.

Designers' Notepad was "initially developed to support software design...due to features of the early stages of the software design process that are similar to the learning activities undertaken in many domains. These features include brainstorming, planning, organising, the refinement of ideas and the consideration and selection between alternative options." (1994, p. 107). As such, Twidale, Rodden and Sommerville explain that the interface of Designers Notepad is an iterative process, constantly revised to allow the program to be used effectively:

> *A key difference between the DNP [Designers Notepad] and other design tools (and indeed other concept mapping tools) is that most other systems have an implicit process model embedded in them...the process model may not only vary between users but an individual user may use different models and variations depending on the problem faced. In cases where there is a mismatch of models, the tool will not be effective, it will feel awkward to use and is likely to be abandoned in favour of traditional, more flexible methods. (p. 111)*

This embedded implicit process model is exceedingly apparent in SemNet (1993), another computer based concept-mapping tool. The user is forced into a set process of creation-naming-relating-organising via dialog boxes that severely limits the quick creation of concepts as they come to hand, ultimately limiting the intended advantages of electronic maps (see figure 2). Fisher (as quoted in Twidale, Rodden and Sommerville, 1994, p. 110), one of the many designers of SemNet states of her own program that "...assigning names to relations, as one is prompted to do in SemNet, is not a 'natural, cognitive act'-that is, not a natural part of spontaneous thought processes".

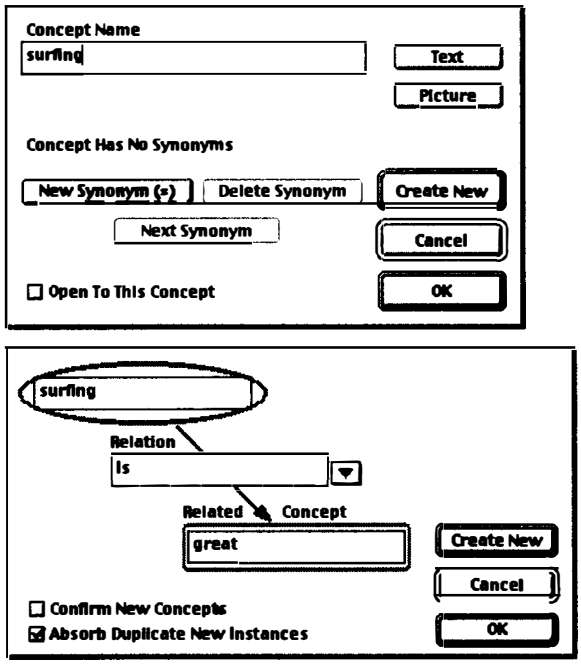

Figure 2: *Creating a proposition in SemNet*

As such, flexibility of the interface for Designers Notepad was a key issue for Twidale, Rodden and Sommerville, designing it so no specific model was adhered to for creation and revision of the map:

> *The aim was to enable the user to focus quickly on the domain issues, to be able to quickly enter many ideas as these occur and not to have to be concerned initially with the form or type that the idea should have. This can only occur is she trusts that it will be easy to edit structures with any refinements that occur to her later. (1994, p.108).*

Sung, Chen, Lin and Chang (1998), Reader and Hammond (1994), and Twidale, Rodden and Sommerville (1994) all agree that computer based concept maps offer the advantage of relatively easy revision, suggesting that the tools available for revision must be available and intuitive to the user. Revision leads to (better) organisation. Organisation leads to structure. Structure leads to clarification and a greater understanding of the material to be learnt.

Although SemNet does have particular characteristics that some users may find limiting or frustrating, it has an excellent feature that "provides a continuously updated quantitative analysis of the knowledge structure being created."(SemNet, 1993). This quantitative data includes, show all concepts alphabetically, by creation order, by number of instances (relationships), and many more.

Reader & Hammond (1994) have also implemented an electronic concept map used as a learning tool. Using a custom concept-mapping application developed in HyperCard, Reader & Hammond examined the effectiveness of concept maps in learning information from a Hypertext system. Their results suggest that the actual process of creating a concept map based on recently learnt material from a Hypertext system helps retain knowledge better than ordinary note taking:

> *Advocates of concept mapping argue that by encouraging learners to represent their knowledge using a node-link formalism, learners are forced into activities that aid the organisation and integration and integration of knowledge, and that the map itself can serve to communicate the learner's knowledge more effectively than text. (p. 99).*

The actual HyperCard program is rather rudimentary (Reader & Hammond, 1994, p. 101), but includes the ability to attach notes to each node, aiding the learning process.

Perhaps the most popular computer based concept-mapping program is Inspiration (1998), available for the Windows and Macintosh platforms (see figure 3). Inspiration, currently at version 5, is quite advanced yet relatively easy to use. Its popularity comes from many of its more notable, user-friendly features which include:

- RapidFire™ - when people experience "bouts of creativity" (Twidale, Rodden and Sommerville, 1994, p.110), RapidFire™ allows the entry of many ideas quickly, without leaving the keyboard. Ideas are separated by the return key, and Inspiration automatically creates the graphically nodes.
- Child Windows – like Designers Notepad (Twidale, Rodden and Sommerville, 1994), Inspirations has the ability to create concept maps inside concept maps, that is, a child concept map attached to a parent node in the main concept map.
- Notes – again, like Designers Notepad, the ability to attach notes to nodes exists.
- Arrange – Inspiration allows the concept map to be arranged to improve readability. Arranging can include, tree or cluster layouts.
- Node Design – nodes can take on numerous graphical forms, dictated by colour, font, shape, size, picture and symbols.
- Outline – an excellent feature analogous to the propositions put forth by Rautama and Tarhio (1998). A concept map can be viewed as a text based outline, allowing the creation and revision of concepts and their relations. One click on the diagram button, and the text is converted to the familiar graphical form.

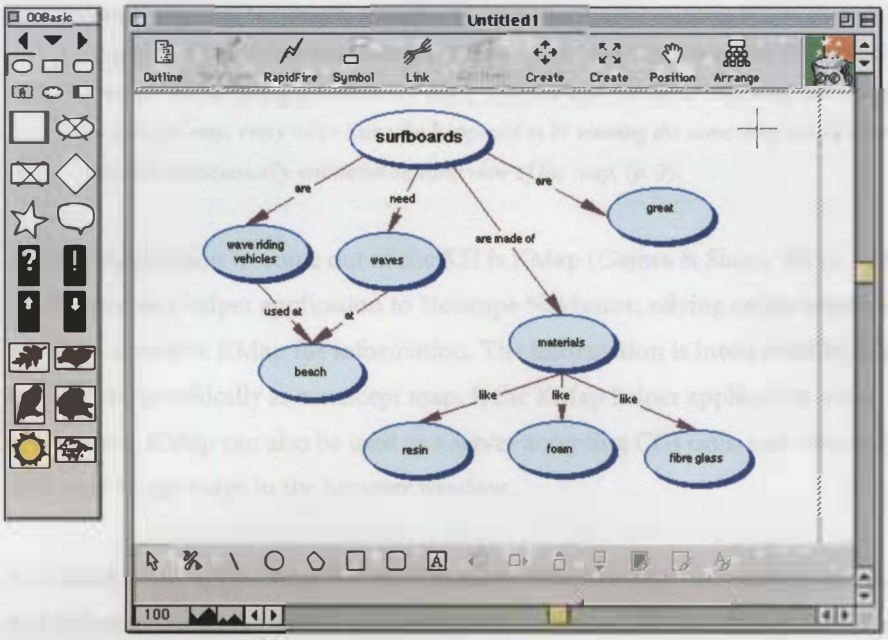

Figure 3: *Inspiration toolbar and document window*

The Knowledge Science Institute (KSI) at the University of Calgary (Alberta, Canada) have published several papers discussing collaboration with electronic concept maps, the web and hypermedia. Kremer and Gaines (1996) (also comparable to Rautama and Tarhio, 1998), discuss ways to embed concept maps into HTML documents using a program called Smart Ideas. "Smart Ideas allows one to draw concept maps using an editor, but every node and link drawn in the map is also represented in an underlying hyperspace which is potentially shared with other users and other concept maps." (p. 2). The paper suggests that Kremer had Smart Ideas networked to allow groups of students on different workstations to collaborate in real-time on the same "concept map workspace" (p. 2), as is the intention of mapCAST. Unfortunately, I could not find a downloadable copy of Smart Ideas, as this seems to be the only existing software congruent to mapCAST.

Smart Ideas also comes in a plug-in flavour, which was developed to run in Netscape Navigator on the Windows 95 operating system only. Kremer and Gaines (1996, p. 4) explain that the plug-in would allow the viewing, rearranging and navigating of Smart Idea concept map documents, giving the user the opportunity to click on concepts and navigate through the Smart Ideas hypermedia database; a URL or another Smart Ideas document.

It is interesting to quote Kremer and Gaines (1996), in their discussion of future work as they (unknowingly) describe mapCAST:

It will provide access to all the mapping functions, as well as providing quick access to common network functions, such as controlling the map. Toolbars are extremely common in applications nowadays, as they provide quick access through visual recognition (usually icons) to program functions without traversing through a hierarchical menu. The toolbar will also implement 'tooltips', which are a visual aid explaining what the tool is, followed by the keyboard shortcut to the tool (in brackets). The 'tooltips' usually show up when the mouse has lay to rest over a tool icon for a period of three seconds.

Figure 4: *The Toolbar, showing a tooltip for the inspect tool*

The concept map window is the area where the user will create and manipulate map documents. This window has no extra functionality, and only one concept map can be open at a time. Document objects (nodes and relationships) can be moved with the arrow keys and deleted with the delete key.

The menu bar (figure 5) is designed to be familiar to any level of Macintosh user, with common commands grouped under each menu heading. These commands are consistent with the Macintosh Human Interface Guidelines:

- Short description of application and author under the APPLE menu.
- File, printer and application-wide commands under the FILE menu.
- Copy, paste, undo and preferences under the EDIT menu.
- List of windows that can be hidden under the WINDOW menu.
- At this stage, mapCAST does not implement help.

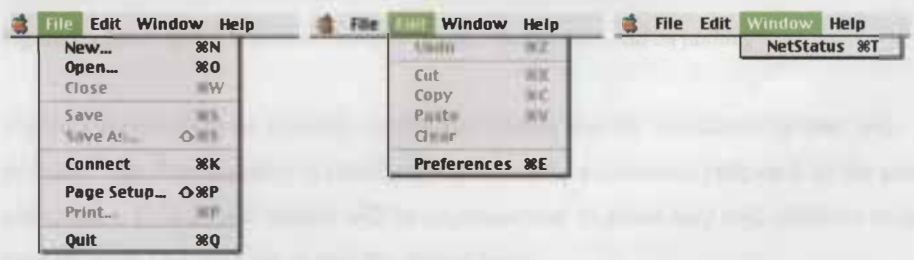

Figure 5: *(left to right) the File menu, the Edit menu and the Window menu*

Mapping Elements

To create a new map, the user selects File > New from the menu bar, which displays the New Map dialog (figure 6).

Figure 6: *New Map dialog*

Here the user can enter a title for the map, select the dimensions from a pull-down list of common paper sizes, and select the orientation. Although this version of mapCAST will not support scrolling of the window, the canvas created is the size that the user selects, and any attempt to re-size the window past the selected dimensions will result in the window snapping back to its largest size. After selecting the 'OK' button, a new map window is presented.

The user can create and manipulate concepts and relationships using the tools and clicking / dragging on the map window. The map window will also support contextual menus, a context-sensitive menu interface that was introduced in MacOS 8 and accessed by pressing the control key, followed by a mouse button press (in applications that support it).

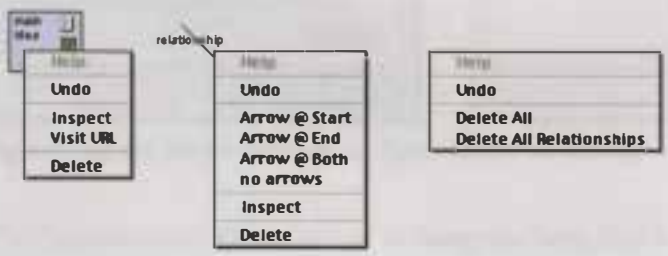

Figure 7: *(left to right) Contextual menus for nodes, relationships, and on nothing*

Figure 7 shows all three possible contextual menus and the functionality they will provide. This functionality is usually quick access to a common task such as the undo menu item. Contextual menus will be implemented to allow easy and intuitive map construction and revision as per the design brief.

To change a node or relationships appearance, the user 'inspects' the object with the Inspector tool, which shows the Inspector window (figure 8).

Figure 8: *Inspector window showing the Appearance tab*

The Inspector is a context sensitive, modal window, adjusting to each node or relationship. Replacing many menu items to change an object, such as used in other concept mapping applications, the Inspector was designed to provide a singular window where a user can easily change all of an objects properties via categories implemented as tabs.

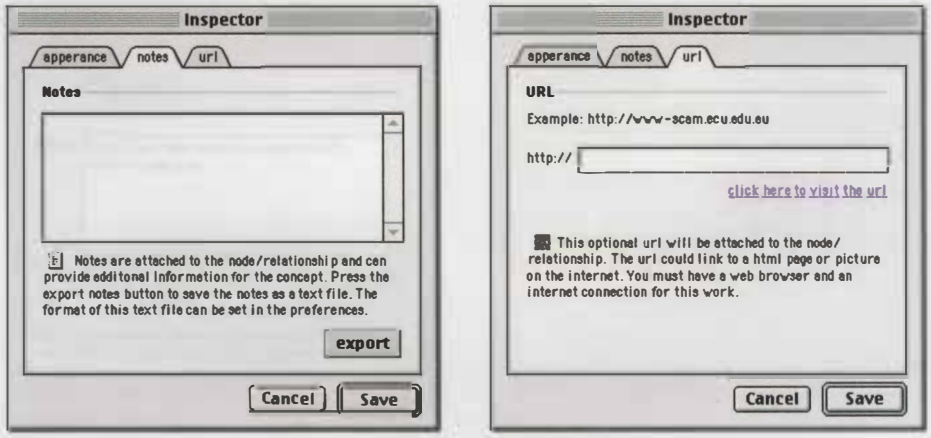

Figure 9: *Inspector window with notes tab (left), and URL tab showing*

The Appearance tab allows the user to change the label, font colour, fill colour, and if a relationship, the arrow mode. The Notes tab (figure 9) allows the user to add notes to the object. This feature could serve many purposes, but it's intention is to be used for further describing the concept, and making the concept more meaningful not only to the user when revising, but other collaborators who encounter it for the first time. These notes can be exported to a text file by pressing the 'export' button. The URL tab (figure 9) allows the attachment of a URL to the object. Again, this feature makes the construction and revision process more intuitive and meaningful, by

providing the option of a internet URL link that could relate or further explain the concept or relationship.

If either notes or a URL are attached to a concept, mapCAST indicates this to the user by providing a visual cue in the form of icons in the top right corner; a page icon for notes, and the ampersand symbol for a URL (figure 10).

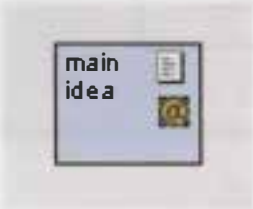

Figure 10: *A node indicating attached notes and URL*

During a collaborative session, users that do not have control cannot alter the map but may wish to inspect the concepts and relationships to examine their respective notes or URL. The Passive Inspector (figure 11) was designed to facilitate this, giving an overview of the objects label, its URL with an option to visit the hyperlink, and its notes with the opportunity to export them.

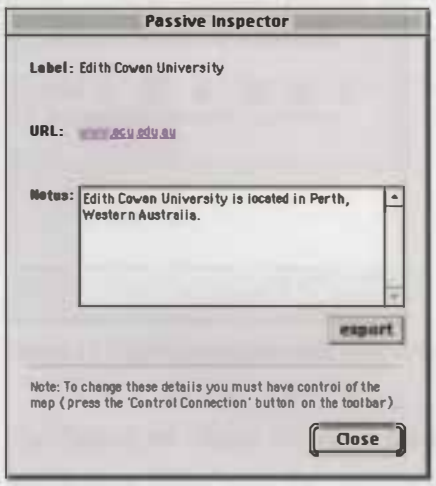

Figure 11: *Passive Inspector window*

To take control of the map, the user presses the 'Control Connection' button on the toolbar, which brings up a window showing network-related functions (figure 12). To control the map, the user presses the 'control' button. To disconnect from the host, or to stop hosting, the user presses the 'disconnect' button.

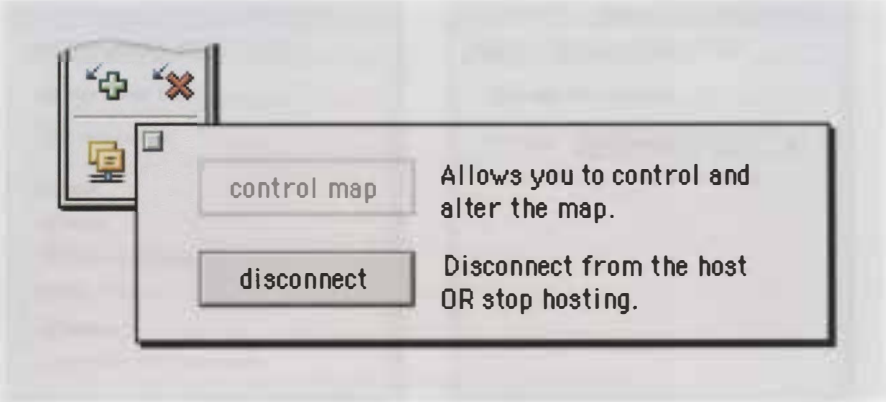

Figure 12: *'Control Connection' button window, accessible from Toolbar*

Preference Elements

To increase overall accessibility, the preferences window will allow the user to customise mapCAST, especially that related to the network ability of the application. Each preference item has a short description.

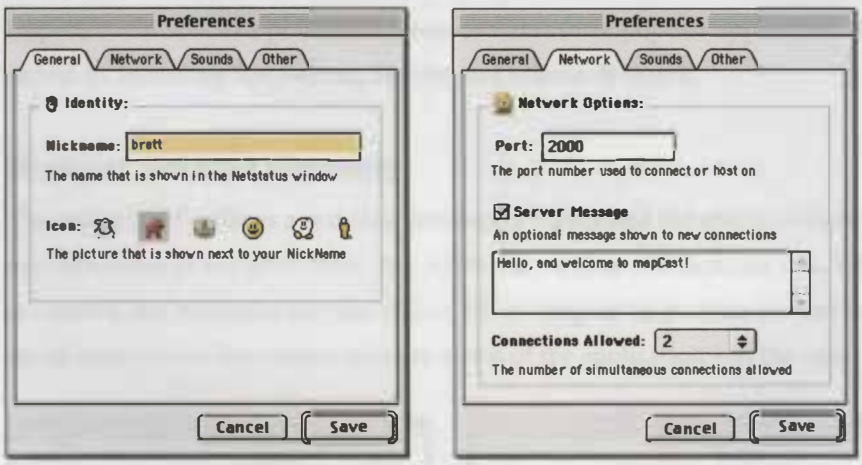

Figure 13: *Preferences window with General tab (left) and Network tab showing*

The General tab (figure 13) allows the setting of the users identity and icon. This information shows up in the users list in the NetStatus window, when connected. The Network tab (figure 13) allows the setting of the port used when in client or host mode. It allows an optional message to be shown when new users connect, and allows the user to set the amount of simultaneous connections when in host mode.

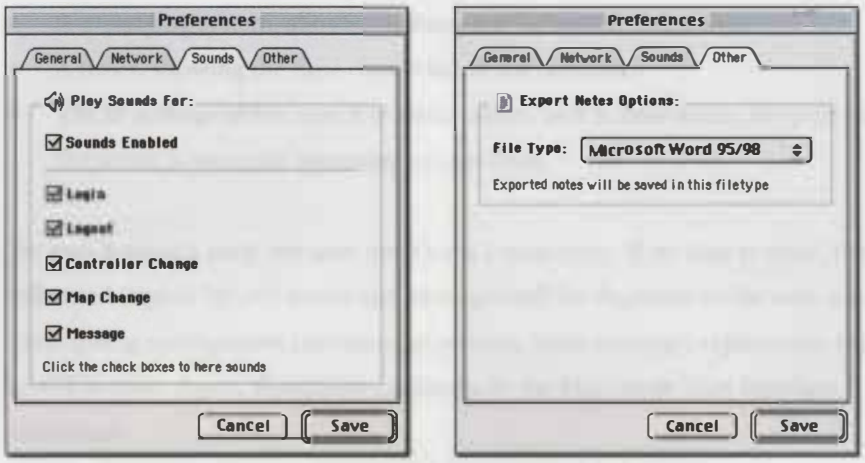

Figure 14: *Preferences window with Sounds tab (left) and Other tab showing*

The Sounds tab (figure 14) allows the user to hear audible cues when certain events occur while connected. These events include a user logging in, a user logging out, controller change, a map change and new message arrival. The Other tab (figure 14) allows the user to select the type of file saved when notes are exported via the Inspector window. The pull-down menu contains three popular application types to export to, including SimpleText, BBEdit, and Microsoft Word.

Network Related Elements

The design brief outlines a real-time messaging system and the ability to show the network status at any given time. The NetStatus window will facilitate this. When connected, the NetStatus window (figure 15) is designed to provide the user with visual feedback on the current network status of the application and the map.

Figure 15: *The NetStatus window (left), and the window expanded*

This concentration of all network related information in one window will simplify the task for the user to get network status and execute network functions such as instant messaging. The NetStatus window will provide:

- A user list displaying whom is currently collaborating on the map.

- A messaging system, allowing messages to be sent and received.
- A timer, showing the time connected to the network.
- The IP address of the host if in client mode, or if in host mode, the port on which the server is using for incoming connections.

To start hosting a map, the user must have a map open. If no map is open, the user is informed (figure 16). All errors and messages will be displayed to the user using the same dialog arrangement consisting of an icon, main message, explanation text, and an OK button. Again, this design conforms to the Macintosh User Interface Guidelines

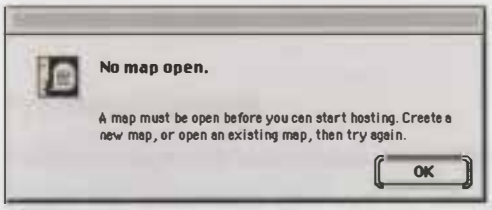

Figure 16: *'No map open' dialog*

If a map is currently opened, the Connect window is shown (figure 17).

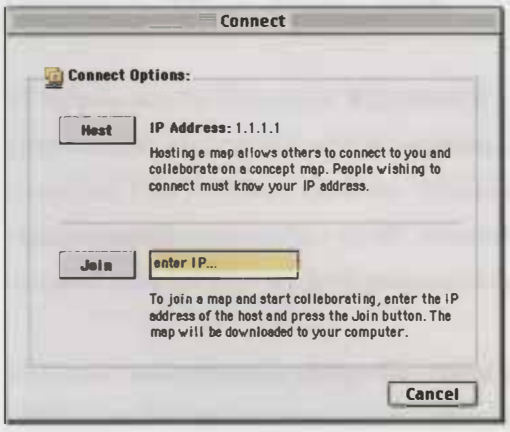

Figure 17: *Connect window*

From here, the user can start the hosting session by pressing the 'Host' button. The IP address field is populated with the computers network address. Clients wishing to connect to this host must enter this address into their 'Join' text-box and press the 'Join' button.

3.0 Development Cycle

The development cycle of mapCAST occurred in three major stages:

- The mapping module, where the code is created to draw, revise, store, and retrieve concept maps. The interface was also designed and the mapping tools were developed.
- The communication module, where the code is created to join and host a concept map, as well as form and send, receive and parse, all network data.
- Testing, which was an iterative process, occurring throughout the development cycle.

Pre-Development

Before development commenced, a suitable development environment had to be found. This period coincided with the release of version 7 of Macromedia Director. Director looked promising as I have an extensive background in using Director, and the release of version 7 brought new functionality, including the Director multi-user server. Unfortunately, feasibility tests proved programming this type of application in Director to be difficult, if not impossible.

REALbasic was the alternative. REALbasic is a MacOS application for designing and programming applications, with an emphasis on visual 'drag-and–drop' GUI (Graphical User Interface) elements. "REALbasic's programming language is an object-oriented version of the BASIC programming language...It was originally designed to be used for teaching programming" ("REALbasic", 1999).

REALbasic was eventually chosen for several reasons, which the REALbasic manual ("REALbasic", 1999) outlines as some of its key features:

- Includes an object-oriented implementation of the BASIC language with automatic garbage collection. Garbage collection is a programming term describing the 'cleaning up' of objects in memory once they are no longer used or referenced. Garbage collection is performed to save memory space.
  Programming manual garbage collection into an application is generally considered a laborious task. Java is another programming language where garbage collection is performed automatically at runtime.
- Has high level support for GUI elements, such as windows and buttons
- Development is native on the PowerPC; meaning that REALbasic was written to take advantage of the PowerPC processor speed and does not include legacy code written to support 68k processors. A 68k version is available.

- Ability to build stand-alone applications for either 68k, PowerPC or both (known as a Fat application) and Windows 95/98/NT.
- Is multi-threading, meaning that the application can be written to perform several tasks at once. An example of this is the ability to copy files in MacOS 8 (one thread) and empty the trash (another thread) simultaneously.

Despite being relatively new, REALbasic has an established developer base. This is evident through numerous web pages dedicated to developing applications in REALbasic, most notably the REALbasic developer ring (http://www.webring.org/cgi-bin/webring?ring=xbasic;list), and the Hotline site, café.realbasic.com. These resources contain reusable classes and code that can be incorporated into existing and developing applications.

Mapping Module

The initial coding of mapCAST centred on the need to nodes (or concepts) and relationships, followed by how to store the map both internally, and externally as a file.

Development began with a basic interface, which consisted of a toolbar with tools, and a map window, followed by experiments with adding, deleting, and manipulation of objects in the map window. The map window contains a REALbasic control called a Canvas. A Canvas is a rectangle area that allows the display of images, has methods that allow drawing down to pixel level, and receives all mouse events.

The basic design behind the mapCAST mapping module revolves around this Canvas, and several other classes (some programming knowledge may be needed to understand the following). The super-class is called *genericDragObject*, and contains all the code needed to draw objects (such as nodes and relationships) on the Canvas. Two sub-classes also exist, *node* and *relationship,* whose super (or parent) is the *genericDragObject* class. The area where all the map data is stored is an array, which is of type *genericDragObject*.

When a user adds a node, for example, by pressing the mouse on the Canvas while the 'add node' tool is selected, the following occurs:

- The Canvas receives the *mouseDown* event, at an X and Y location. The X, and Y location (see figure 18) is a graphics coordinate system based on pixel's;
- The *mouseDown* event passes the X and Y location onto a local method called *AddNode*;

- *AddNode* appends a new node object to the map window's *genericDragObject* array. The node object has many properties such as label, colour, width and height. Upon instantiation, the new node object requires the X and Y location (which the *AddNode* method provides), so it knows where to draw itself;
- Once the new node object has been appended to the *genericDragObject* array, the canvas has to refresh itself so it shows the new node on screen. This is simply a matter of looping through the array and telling each *genericDragObject* (whether a node or relationship) to draw itself on the Canvas.

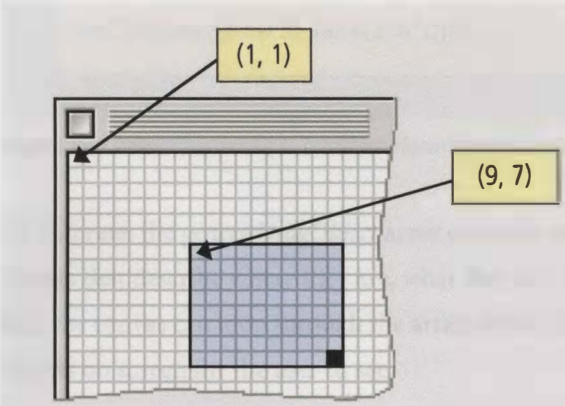

Figure 18: *The window X, Y coordinate system*

The node identifies itself as a node by is *kind* property. Based on this property, the *genericDragObject* will draw either a rectangle (for a node), or a line with arrows (for a relationship). At this stage, the *genericDragObject* class only contains code to draw rectangular nodes. Theoretically, given the X and Y coordinates, and the width and height properties of a node, the code could be modified to draw any number of shapes (including polygons). The difficult part is to program the geometry required to draw these objects, and to get these objects to report where they are relative to the mouse. As an example, figure 19 shows how a relationship reports to the user that it was selected.

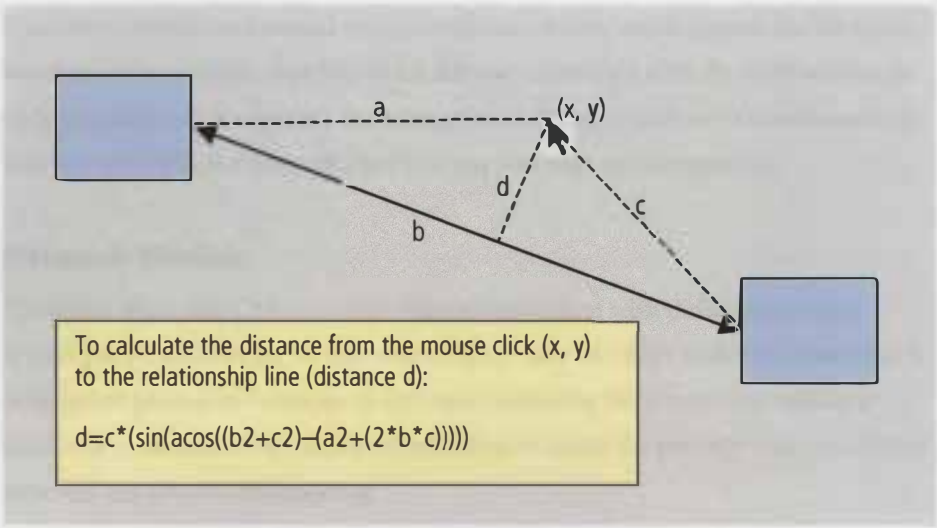

Figure 19: *Geometry required to calculate object locations based on mouse click coordinates*

To reiterate, the *genericDragObject* array contains information (proprieties) from objects that describe where they are, what they are, and what they look like. Based on this, the canvas can loop through the array, extract these properties, and draw the objects on screen for the user to see.

The next step was developing a method for saving this *genericDragObject* data externally as a file on disk.

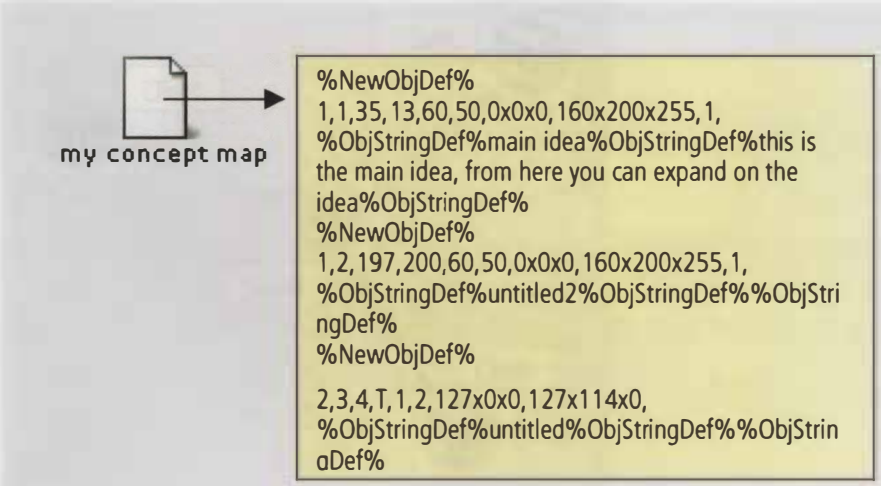

Figure 20: *Map file showing how data is stored on disk*

When a user saves the file to disk, mapCAST goes through the *genericDragObject* array, extracting all the objects properties. With this data, it creates a string and writes it all to a text file (figure 20). The text file is delimited by special tokens. When mapCAST reads in the text file, the objects and their associated property values are delineated by the tokens.

Once the internal and external structure of map objects was designed, the interface was finished to provide visual tools for the user to interact with the tools written in code to create and manipulate the concept map. Being conscious of the networking side of mapCAST, the network interface was also built simultaneously.

Network Module

The application mapCAST is a stand-alone application with the ability to have several people collaborate on the same concept map in which each participant can be in different geological locations. A network (including the Internet) is used as a backbone to facilitate the transfer of information about the concept map, its current state and the people collaborating.

Typically, an application that involves the transmission of data over a network is achieved via a classic client/server set-up (figure 21). In this model, a client application connects to a server - an application dedicated to nothing more than hosting all clients that connect to it - using an agreed protocol to redirect and provide information such as HTML or email. Although a server can, through extensions, interface with other sources such as a database, it usually provides no other functionality.

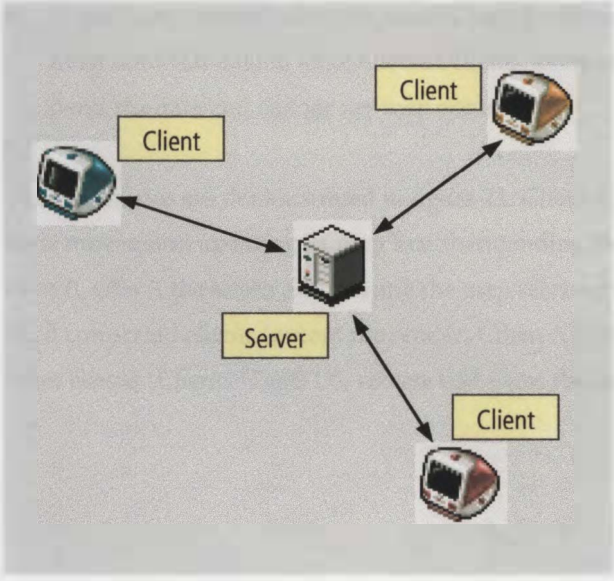

Figure 21: *Typical client/server setup*

Collaboration of a concept map via several mapCAST clients does not involve a dedicated server as such. Instead, a client can host a map, allowing others to connect, eliminating the need for a dedicated server. That is to say, the mapCAST application can function in both a client and server capacity.

The networking side of mapCAST was both easy and hard to program. Easy, because the data to send was mainly small string chunks describing a change in the map. It was also hard because mapCAST can act as both a client and a server, so many of the methods in code must accommodate the application in either mode.

To demonstrate the complexity of this, one must understand the notion of 'control' in mapCAST. Only one person, or collaborator, can make changes to the map at one time. This person is known as the Controller. The Controller can change add, delete, move concepts and change concept/relationship information. All other collaborators can only watch their maps be updated, and request control if they wish to change the map.

To code this type of set-up, the following must be done. For example, if a map change occurs...

- ...and I am a *client* and *in control*, I must update my map, then send the map change data to the server.
- ...and I am a *client* but *not in control*, then I will receive the map change data. I must parse this data, and change my map accordingly.
- ...and I am a *server* and *in control*, then I must update my map, then send the map change data to all connected clients.
- ...and I am a *server* but *not in control*, then I will receive the map change data. I must send this data to all connected clients, except the client who sent it, then parse the data and change my map accordingly.

These scenarios are demonstrated in figure 22. Client A, is in control and makes a node movement, updating its map first then sending the data to the server, Host B. Host B, who is the server and hosting the map received the information and sends it to all connected clients (except the sender, Client A), then updates its own map. The other clients (Clients C and D), receive and parse the data, then update their maps.

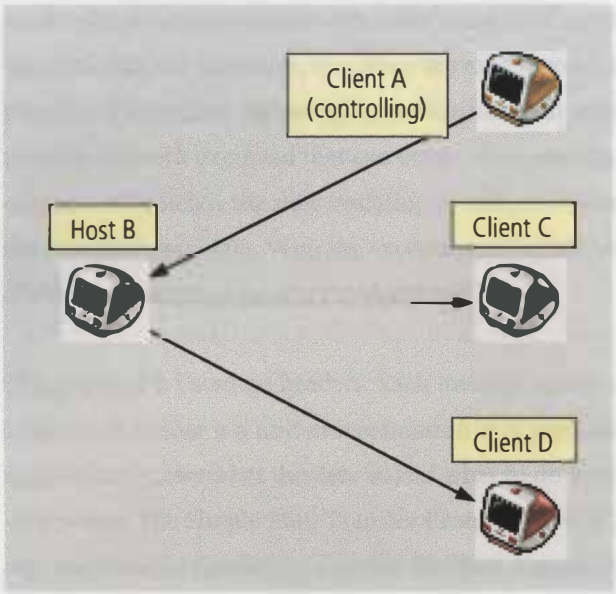

Figure 22: *Data Flow in mapCAST*

In server (host) mode, mapCAST initially contains an empty array of client sockets bound to a single port. When the user starts to host a map, a new socket is appended to the array. This new socket proceeds to listen on the specified port for a new client connection. When the socket receives a new connection, it again appends a new socket to the array to listen for the next client connection (see figure 23).

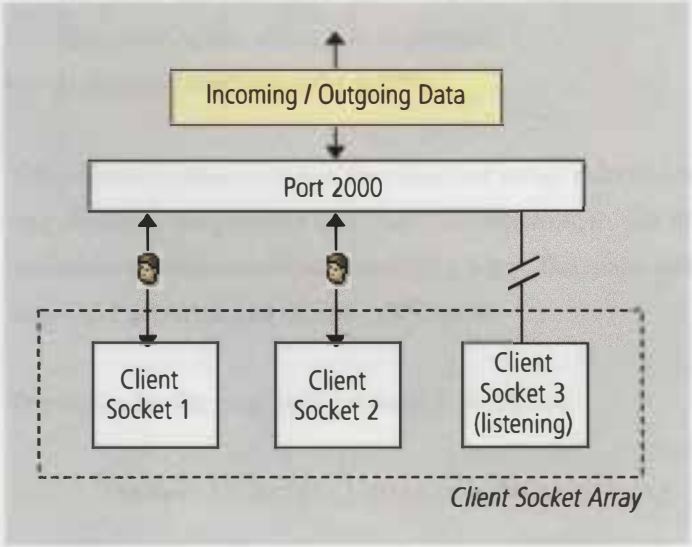

Figure 23: *mapCAST server client socket structure*

In client mode, a socket is created and attempts to connect to the server entered by the user, on the specified port. The default port for mapCAST is 2000.

To enable mapCAST to function as a network-able application, a custom protocol had to be developed. This protocol (see appendix D) allows the mapCAST

application to communicate with other mapCAST applications across a TCP/IP network (like the Internet), and allow the exchange of concept map data. The protocol is extremely lightweight, sending minimal amounts of data. This reduces the possible network overhead that can occur when sending large amounts of data, the outcome of which is the slow updating of each connected clients map as they wait for the modified map data. With the exception of initially sending the whole map to new clients, data sent and received is under 500 bytes.

The protocol is based on headers. Each message sent to and from mapCAST contains a header. A header is a unit of information that precedes the data sent, so an application knows what the data is, and what to do with it. Many protocols are based on headers. The Simple Mail Transfer Protocol (SMTP) uses a combination of letters and numbers for headers to describe the data. For example, when a SMTP client connects to a SMTP server to send email, the following occurs:

- Client connects to server
- The server replies with a `220` command;
- The client sends the `EHLO` command, proceeded by its IP address;
- The server replies with a series of `250` commands;
- The client sends the `RSET` command;
- The server replies with a `250` command;
- The client sends the `MAIL FROM:` command, proceeded by its email address;
- The server replies with a `250` command;
- And so on (Neuberg, 1999, p.579).

This communication between the client and server is an alternating dialogue, and very similar to the protocol that mapCAST employs, in that the client issues a command and the server responds with a reply. The major difference is that the mapCAST protocol uses all numeric headers.

The syntax for the mapCAST protocol is as follows:

```
Header || data1 | data2 | data3... |||
```

where:

```
||    = ASCII Character 12 - Header/data delimiter

|     = ASCII Character 14 – Internal data delimiter
```

```
|||  = ASCII Character 11 - End of data
```

An example, the structure of a client sending its login data:

```
000 || Nickname | iconID |||
```

The following outlines the sequence of communication between a mapCAST application in client mode connecting to a mapCAST application hosting a map:

- Client connects to server (the host map);
- Server checks allowed connections. If the server is full, it sends the `000` command, and closes the socket. Otherwise it allows the connection, and opens a new socket for the next connection;
- If the client does not receive the `000` command from the server, it sends its login data via the `000` command, proceed by its nickname, and icon number;

```
000 ||  brett  | 5 |||
```

- Server receives login data, and asks the client if its ready to receive the map by sending the `200` command proceeded by the length of the map in bytes;

```
200 || 165578 |||
```

- Client receives the 'ready for map' command, and if all is ok, returns 'ready to receive' by sending the `200` command;

```
200 || |||
```

- Server receives the go-ahead to send the map, and does so by returning the `201` command, proceeded by the map title, followed by the map data (a string);

```
201 ||  untitled map  | <map data> |||
```

- Client receives the map data, and returns the `201` command indicating that the map was received ok.

```
201 || |||
```

- The server receives the ok, and sends a series of commands. If enabled, the server message (`001`, proceeded by message);

```
001 ||  Welcome to my Server!  |||
```

- It then sends the connected user list (`002`, proceeded by nickname, followed by icon number, and followed by a unique user ID assigned by the server);

```
002 ||  Ron Oliver       | 5 | 1 |||
002 ||  Arshad Omari     | 4 | 2 |||
002 ||  Brett Greay      | 4 | 3 |||
```

- Finally it tells new client that it does not have control of the map (`202`, proceeded by 'false').

```
202 ||  F  |||
```

Once a client has connected to a server, the majority of data sent and received is that describing changes to the concept map. For example, the sequence of events when a user in control moves a concept node:

- User moves node;
- mapCAST will update its own map first;
- mapCAST checks if it is connected, and if it has control;
- If it has, it passes the moved node as a parameter to a local method;
- As each node (and indeed all objects) has a unique ID, this local method extracts the ID, and its new location defined by an X and Y coordinate relative to the concept map window;
- With this extracted information, the method forms the command string to send as defined by the mapCAST protocol. In this case, the `250` command:

```
250 || objectID | newX | newY |||
```

- The method then checks if it is a client or host mode. If it is a client it sends the command to the host. If is a host, it loops through the array of client sockets and sends the command to each.

Testing

Testing was an iterative process throughout development, coinciding with every compiled alpha version, which usually occurred fortnightly. Because of time

concerns, all testing was done by myself. This made it easier to debug and fix anomalies, as there was no time overhead waiting for beta testers to report bugs. This situation is not entirely desirable, as many minor bugs were discovered during the examination of mapCAST. The purpose of this in-house testing was to discover bugs in the map and network components and to make sure all tools and functions were executing correctly before placing the software in the hands of the evaluators.

Testing the map component was a relatively easy process. The most difficult part to test was the networking side of mapCAST. As there was no immediate access to a hub (or several computers to plug into it), a virtual network was created by connecting the primary development machine, a Power Macintosh 9600, to a PowerBook G3 laptop, via their ethernet cards, using a 'crossed' ethernet cable. By assigning each computer their own unique IP address, they will happily communicate.

When a new alpha compile needed testing, the newly compiled application was copied to the laptop via AppleTalk, opened, and a hosting session with a new blank map was started. The application on the development computer was then opened, upon which a connection to the host application on the laptop was established. Performing network related activities, such as joining a host and modifying a map, OTSessionWatcher was used to examine the data that mapCAST was sending and receiving, both as a client and server. OTSessionWatcher is an extremely helpful piece of software when debugging network applications. Created by Stairways Software (http://www.stairways.com), it displays the data transmitted through Open Transport TCP streams, and allows examination of the data and size of packets both sent and received.

Problems

It should be noted that mapCAST was compiled using a beta version of REALbasic (version 2.1 a17). It was necessary to use this pre-release version because of the extra functionality provided than the last official release (which is 2.02). This, of course, had a negative impact on the development cycle, as with any use of beta software. Sometimes, it is particularly hard to distinguish whether an error or crash was because of my programming or bugs in REALbasic.

A socket bug (which is well documented in the REALbasic Developers mailing list, as are pleas for its correction) was introduced in the early beta versions of version 2.1. This is a particularly annoying bug as it only exists in the IDE (Integrated Development Environment), but compiled applications seem to work. Sockets require OpenTransport, which is the MacOS' networking software. OpenTransport

reports errors to any socket that implements it. The socket bug made OpenTransport report error code '100', meaning "OpenTransport not installed', when in fact it was. This only happened while mapCAST was running in the IDE and the work around was to re-compile the application each time ANY change was made to the network module of mapCAST.

Once mapCAST was nearing completion, *NilObjectException* errors began to appear in my compiled application. Put simply, a *NilObjectException* occurs when an attempt is made to access an object (in this case a graphics object), that does not exist. If not handled correctly, a *NilObjectException*, or any exception for that matter, will quit your application or worse case crash the system. The reason for this error was because of a gross underestimation for the overall memory requirements of mapCAST:

- mapCAST creates and manipulates 16bit graphic objects that require more than the 'usual' amount of memory.
- mapCAST can be a client, or server. The mapCAST application in server mode requires more memory because it accepts and redirects network traffic for all connected clients and itself.

The original requirements for mapCAST was a minimum of 2040k of memory, with a preferred allocation of 5000k (5MB). This was too low, and *NilObjectException's* occurred because the application did not have enough memory to create the graphics object used to draw the concept map. Presently, mapCAST runs comfortably with a minimum of 6500k of memory, and a preferred allocation of 12500k (12.5MB).

Late in the development cycle, 'REALbasic, the Definitive Guide' by Matt Neuberg, was released. This book received rave reviews on the developers mailing list, some referring to it as 'the bible'. Because REALbasic is a relatively new programming environment, this is the only book available (besides the REALbasic user guides), that describes how to program in REALbasic, with all its good and bad facets. Unfortunately, this book was only available in early November. It quickly became a valuable resource. So valuable in fact, that a majority of the mapCAST code was re-written using new techniques and knowledge gained from the book. In retrospect this was good because mapCAST became much leaner and optimised. By the same token, it was also bad simply because it occurred so late in the development cycle, adding one to two weeks to the overall development time.

4.0 Testing Methodology

Original Scope

The original scope detailed a course of analysis that would gather data from two distinct groups, each group appropriate for the data gathered. For technical data, or useability, a small, local multimedia company called iHealth was to be used. For content data, or functionality, staff from the School of Communications and Multimedia at Edith Cowan University was planned. Both groups would be set a task appropriate to their group type, upon which observation would occur, concluding with an interview to discuss mapCAST success as a collaborative tool.

As the end of the year grew closer, it became increasing apparent that a reduction in scope due to time restraints would be necessary. These restraints arose from the fact that mapCAST was not a stage where it was developed and stable enough to put into a testing situation. This is elaborated more in the Design Brief (section 2).

Revised Scope

The final developmental stage of mapCAST (alpha 9) represented a program where every feature worked, and few bugs were present that would hamper normal operation (although many minor bugs would become apparent in testing - see Results, section 4).

This stage coincided with a contractual stage in the company iHealth, which presented itself as a perfect situation to test mapCAST as a collaborative tool. Whilst development of mapCAST concluded, iHealth had begun work on a recently acquired contract. As I was involved in the process of designing the overall product to be developed and what was required, I forwarded the suggestion to use mapCAST as a tool to aid this initial conceptual process. This real world-task replaced the original two-group approach, and provided both technical and content data.

Purpose

The purpose of testing was not to discover bugs, but rather to evaluate mapCAST as a collaborative computer based tool by examining the useability criteria which embodies three major elements; interface design, functionality and useability.

4.1 Useability Criteria

The criteria for investigating the useability of mapCAST as a functional computer based tool can be described in three areas:

1. Interface Design
2. Functionality
3. Useability

Interface Design

Interface design governs the way the user interacts with the application, and the associated comfort of use. Testing the success of the interface design involved the following elements:

- Intuitive Icons – are the icons intuitive in their design, and easily recognisable in the function they served?
- Network Management Tools – is connecting to the Internet to join or host a map, controlling, and disconnecting, a simple task?
- Collaborative Status and Feedback – does the interface provide acceptable levels of audible and visual feedback, as to the current status of the map and those collaborating?
- Communication Controls – is it easy to send a message?
- Menu Design – are the menus intuitively designed and adhered to the Macintosh Human Interface Guideline?
- Toolbar Palette Design – is the Tools palette intuitively designed, allowing access to the most used and critical functions of the program?
- NetStatus Palette Design – is the NetStatus palette intuitively designed, allowing easy communication?
- Inspector Palette Design – is the Inspector intuitively designed allowing easy revision of nodes and relationships?

Functionality

Functionality refers to the scope of what mapCAST is designed to do. Analysing the functionality of mapCAST involves examining the following items:

- Creating Documents – how well does mapCAST support document creation?
- Opening and Saving Documents – how good is the ability to open and save documents?
- Map Creation Tools – are there sufficient tools to create a concept map?
- Map Revision Tools – are there sufficient tools to edit a concept map?

- Network Settings and Connection Management – how well does mapCAST implement the ability to configure the server, connect, disconnect and control a concept map?
- Collaboration – how good is the ability and scope of collaboration?
- Communication between Clients – how good is the ability and scope of communication tools available?
- Printing – how well is printing supported in mapCAST?

Useability

Useability refers to the adequacy of mapCAST to be used for its design purpose, in this case, collaborative concept mapping. Analysing the useability of mapCAST involves examining the following items:

- Speed of mapCAST – does the program function quickly enough to create concept maps (ie. quick screen redraw)?
- Network Speed – does mapCAST implement an efficient protocol that allows 'feasible' collaboration between clients (ie not constantly waiting for map to update)?
- Ease of Concept Mapping – does mapCAST generally allow easy creation and editing of concept maps?
- Reliability – does the program suffer from bugs or crashes that prevent normal use of the program?

4.2 Evaluation Process

Situation

There were three participants involved in the testing of mapCAST, this being sufficient enough to test mapCAST in a network situation. The minimum requirements to test mapCAST would involve two or more computers, and at least one participant on each computer to collaborate on the same map.

All participants were employees of the multimedia company iHealth, and were present in an office area housing a small LAN (Local Area Network). The LAN currently accommodates three computers and a network printer. Two computers connected via the LAN were used:

1. Participant one used a Power Macintosh 7600 / 225mhz (released 1997), 172 MB RAM, with a 17 inch monitor
2. Participants two and three used a Power Macintosh 6100/66 (released 1995), 40 MB RAM, with a 14 inch monitor.

Participants one, and the group consisting of participants two and three, were not able to visually nor verbally communicate as the two computers were in separate rooms.

All participants had not seen or used mapCAST before, but had had experiences with traditional 'pen and paper' concept maps, and limited usage of electronic concept mapping software. According to participants, 'pen and paper' concept mapping was typically used during the design phase of a project, to flesh-out the major ideas. Each participant also had 8 or more year's experience with Macintosh computers and the MacOS.

The task involved a project at iHealth that was in its infancy, the design stage. It involves the development of a CD-ROM intending to teach medical student's about haematology, the study of red blood cells. The participants, and myself, discussed and agreed to map out the overall requirements and initial screen designs intended for the final product. As this was not the first time the product had been discussed, all participants had prior knowledge of the subject matter.

This situation is a real-world process, strengthening the validity of the data gathered. Testing mapCAST in a company setting returned reliable data that determined its success. As the company develops multimedia with new technologies, they deal with

ideas and concept creation usually within a group situation on a daily basis. Therefore, they were able to accurately represent and evaluate mapCAST in both technical and content areas.

Tools Used

Three tools and techniques were used during testing:

1. Evaluation forms.
2. Observation.
3. Discussions.

The evaluation forms (see appendix C) implement a Likert scale, used to gauge user attitudes towards mapCAST. Evaluation methods using the Likert scale provided summative data, where reviewers measure the concept of interest for each statement by indicating on a scale. The Likert method was used because it provided the best and most accommodating platform to gather data for groups of related statements about mapCAST. Likert evaluation forms are also effective and easy to use, and provided useable data despite the small size of the testing group.

The statements were designed to gather useability data, and were broken up into three categories: interface design, functionality and useability (see section 3.1 for more details). Each category contained numerous statements relating to the section heading, where participants would record their sentiments by marking the scale, rated 'poor', 'average', and 'excellent'. In relation to a specific statement or overall category, a 'poor' result indicated that mapCAST did not achieve its goal, an 'average' result indicated a satisfactory level, with needed improvement, and an 'excellent' result indicated that mapCAST achieved its goal, and was designed or performed to the evaluators liking. Space was also provided for participants to document any other opinions or comments in each category.

Observations took place during the course of the testing session. Spending equal times with participant one, and the group consisting of participants two and three, observation was unobtrusive and transparent, as not to disturb the user. Observation provided notes as to the user interacted with mapCAST and its user interface.

Before the commencement of testing, and at the conclusion, informal and semi-structured group discussions took place. Providing a more relaxed atmosphere in which to gather data, these discussions provided the participants' general feeling about mapCAST as a computer-based tool, and any other salient points. The discussions were also an opportunity to discuss the task at hand.

Procedure

As with any new software, a learning curve ensures. Before testing began, the group was exposed to the underlying concepts behind mapCAST to increase the curve somewhat. The group was not 'walked through' each feature of the program as this could influence and possibly undermine the study. What was briefly discussed was the concept of how a mapCAST client can host a map, and how others can join and collaborate.

Once discussions had concluded and the task was mutually agreed upon, each of the participants went to their respective computers on the network and testing began. A period of playing with the software started off the session, with users feeling out the software and familiarising themselves with the user interface and available tools. This lasted for around ten minutes.

Participant one then began to host a new blank map, and participants two and three connected. Collaboration then began on the map, with control alternating between each participant in building, organising and revising the concepts and relationships on the agreed upon task. This alternating control was subsidised with instant messages usually seeking comments or approval from the collaborators. All communication between the participants during testing was through this messaging system provided by mapCAST.

The testing session lasted for approximately forty minutes, during which each participant contributed to the map while simultaneously completing the useability form and documenting any comments they felt were important. Equal times were spent at each of the two computers involved, observing and taking notes, especially noting comments spoken by participants.

Once the map was a stage where all collaborators were satisfied, the map was saved to a network drive, printed, and the participants regrouped for final discussions.

5.0 Results

The results are best separated and described under content and technical headings. These areas were chosen to actively describe the areas of mapCAST that were being tested. Content deals with the tools and the software's ability to function for its purpose. Technical data refers to how well the software was programmed, ie. does it do what its supposed to do.

Content Results

The testing session began with the participants offering their definition of a concept map. All three participants acknowledged that they had always referred to them as mind maps, but understood that concept maps and mind maps are essentially similar in both design, and the advantages they each offer.

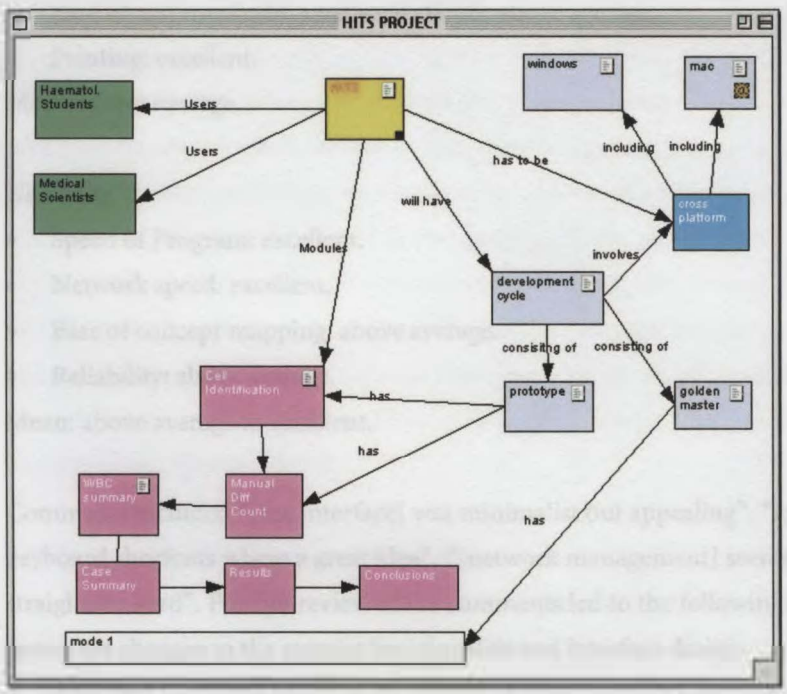

Figure 24: *State of concept map after testing concluded*

Figure 24 shows the state of the map when testing concluded. The following are average results for each category, rated either poor, below average, average, above average or excellent:

Interface Design

- Intuitive icons: above average.
- Network management controls: above average.

- Collaborative status and feedback: above average.
- Communication controls: above average.
- Menu Design: above average.
- Toolbar design: above average.
- NetStatus Design: average.
- Inspector Design: excellent.

Mean: above average.

Functionality

- Creating documents: excellent.
- Opening and saving documents: excellent.
- Map creation tools: average.
- Map revision and editing tools: average.
- Network settings and connection management: excellent.
- Collaboration: above average.
- Ease of communication between clients: above average.
- Printing: excellent.

Mean: above average.

Useability

- Speed of Program: excellent.
- Network speed: excellent.
- Ease of concept mapping: above average.
- Reliability: above average.

Mean: above average to excellent.

Comments included "[the interface] was minimalist but appealing", "tooltips and keyboard shortcuts where a great idea", "[network management] seemed very straight forward". Further review of the comments led to the following list of requested changes to the current functionality and interface design:

- A revised NetStatus window. Requested changes include a messaging system where the type-in area was much larger, and messages received were better labelled as to who sent them (either by the use of colour or bold type). The user list should also indicate the controller by having their name in bold type.
- Revised Inspector window, becoming a non-modal, global floating palette (much like the toolbar) that automatically updates and allows changes whenever an object is clicked.
- Modifiable text, including size, font and style.
- Spell checker.

- Alignment buttons
- The ability to select and drag more than one object at a time.
- A MacDraw like interface for manipulating objects, where objects have 'handles' on each side and corner for resizing.
- Double clicking objects to change text.

Observation notes showed the reasons for many of these requested changes. The participants expected 'standard' interface elements and functionality, present in many Macintosh applications, to be in mapCAST. What can be considered standard? A majority of applications that manipulate objects, like Adobe Illustrator, have minimum interface functionality. These include:

- Initiating object modification by double clicking.
- Copying and pasting objects.
- Selecting multiple objects.
- Changing the layer order of objects.

Most notably, all three participants struggled with constantly changing from the Move tool to the Inspector tool, to move and modify objects. They expected doubling clicking the object would initiate some change in the object (like showing the Inspector window). Similarly, observation uncovered users attempting other 'standard' functions such as copying and pasting objects, and selecting multiple objects at one time, both of which were not present in this version of mapCAST, and caused noticeable frustration in the participants. Contextual menus were not obvious at first, but were used extensively once discovered. However, participants still struggled with changing from tool to tool to perform simple tasks such as adding a concept node, then moving it.

Another notable observation occurred when moving from the first computer to the second computer used in the test. The group of participants two and three had extended the map window to full screen on their computer, while participant one's map window was much smaller, yet still showed the entire map. Confusion could occur if one collaborator placed a node at a location that was not visible on the other collaborators screen.

Technical Results

Technical results were obtained from all three categories, but focused mainly on the useability results. Overall, these results were above average to excellent, indicating that, technically, mapCAST was programmed at a level where the bugs or errors that did occur, did not hinder the expected useability and functionality of the

participants. This given that it is an alpha version of the application, using a beta version of REALbasic to compile it.

All participants reported that the application had excellent overall useability and stability. The speed of the program was classed as excellent. Processor intensive tasks such as screen redraw while resizing objects performed acceptably on the older machine that was used, the Power Macintosh 6100 (which was somewhat surprising). The network speed and reliability was also excellent, with one participant saying that the network implementation in mapCAST was "flawless". A major influence on this network speed and reliability was the fact that the test occurred on a small LAN. No other network activity occurred during testing, hence almost instantaneous map updates between the two computers. However, the favourable network speeds of mapCAST are attestation to the existing TCP/IP protocol accommodating the mapCAST protocol and its small overhead in transmitting data so well. Further testing of the network speed of mapCAST could involve more clients connected to the LAN, to testing it on a wider network with more network traffic, such as the Internet (ie. different collaborators in different continents).

All participants found the general reliability of mapCAST to be above average, representing the alpha stage of the program where full functionality is hindered by minor flaws. The application did not crash during testing, but bugs were discovered. According to participant comments, the bugs did not hamper the normal use of the program, but were "minor annoyances".

The following list describes bugs, both code and cosmetic related, that were discovered during testing:

- The delete key was not always functional when trying to delete objects.
- Undo information was not being set when performing functions from contextual menus. *
- The message box was not that entirely user friendly as the user had to scroll down to read each new message that arrived. *
- The NetStatus window, when closed via its close box and then reopened, did not refresh the user list or messages. *
- The 'tooltip' for the Connect toolbar icon was displaying 'Apple – K' instead of 'Apple + K'. *
- Toolbar and NetStatus windows would periodically disappear behind the map window when the window was full screen. This is not a bug that was programmed, rather a bug in the REALbasic development environment. The toolbar and NetStatus windows are defined as 'palettes', otherwise known as

global floating windows. The Macintosh Human Interface Guidelines specify that this category of window, when visible, should stay on top of all other windows.

- If the host map was blank (with no objects), mapCAST would inform connecting clients the host map was invalid, and subsequently disconnect. *
- One participant noted unusually high memory requirements for mapCAST, and requested these be reduced.

The asterix (*) denotes that this bug has since been fixed.

5.1 Analysis

The first salient point to arise from the test is that mapCAST as an application is complete, yet as a concept-mapping tool it is somewhat immature, lacking many 'standard' features, as well as those described in the participants' requests (see Content Results, section 4). Clearly, mapCAST needs added functionality relating to electronic concept mapping, but in its current state, it is still a functional tool.

This functionality can be validated by examining the map after testing had concluded (see figure 24, section 4). The map shows:

- Clearly formed structure of concepts and relationships.
- Organisation and classification of concepts by use of colour and space.
- Extensive use of notes to aid explanation of concepts.

The extensive use of notes was somewhat surprising, with participants using this feature to further clarify concepts, or elaborate concept labels. It would be wise to further enhance this feature in the next release of mapCAST.

It was clear during testing that the group of two felt more comfortable using mapCAST and revising the map than the lone participant. This was evident during observations as the group of two created and modified the majority of the map.

During testing, all participants used paper as an adjunct to mapCAST in the overall concept-mapping process. The paper was used for writing notes and initial concept labels and relationship design. I initially thought this indicated failure for mapCAST as the participants obviously did not feel comfortable with just the tools available. Discussions after testing however, exposed a common process. People, especially groups of people, involved in the concept mapping process find it easy to create and write initial ideas, concepts and relationships down on paper, mainly because it is quick and instant; people know how to use pen and paper, it being almost instinctive. Theses ideas are then transferred to an electronic concept-mapping program for further revision and organisation (two of the main advantages of electronic documents). Novak:

> *We usually work with Post-its™ of various sizes, using larger Post-its™ for the large general concepts of the global maps and smaller Post-its™ for specific concepts.... the team may work for a period of weeks, exchanging ideas on how...the global map might be improved. Electronic communication of maps, revised maps and suggestions can greatly facilitate this process. (1998, p.108-109).*

The use of paper could also indicate the participant's low level of exposure to mapCAST. Learning a new tool while creating a concept map, are two cognitive-intensive tasks that the user must complete simultaneously. This is apparent in one participant's comments, stating that three things had occurred during testing:

1. The learning of the application mapCAST.
2. Clarifying existing concepts and relationships.
3. Turning these propositions into map content, with the tools provided.

Between processes two and three above, all participants expressed that the notion of controlling the map in mapCAST, the lack of 'standard' features, and the functionality model currently implemented (the process of adding a node for example), can slow the overall process down for all collaborators. This is especially true when using mapCAST as a learning tool or for knowledge capture. Twidale, Rodden and Sommerville (1994, p. 111) warn that implicit process models that are incompatible with those of the user, can make the tool feel awkward, and be replaced with those that are more flexible, such as pen and paper.

The participants' final comments expressed that as a collaborative tool, mapCAST seemed to be successful, and had aided in organising and clarifying the task at hand. But they felt that all collaborators should know the subject matter, especially when organising existing knowledge.

Influences and Limitations

As with most testing situations, certain aspects (whether direct or indirect) may have influenced the evaluators impressions of mapCAST and subsequently influenced the final outcomes. These aspects include:

- Fast and non-congested network – the only network traffic at the time of testing was that generated by the two networked mapCAST clients, therefore updates were instantaneous, and may not have been generally representative.
- Physical proximity of evaluators – although in separate rooms, all three participants, and myself, were in the same building.
- Social relationships with evaluators – all involved in the testing know each other on a social basis. Evaluators may have felt obliged to give positive feedback on the software.

Limitations that limited the capacity to build mapCAST to a desired level, which may have affected the final outcomes include:

- Time – development of any new and custom network protocol (and indeed software) may take several years to fully mature and function correctly.

- REALbasic beta version – it was necessary to use a pre-release version of REALbasic to develop mapCAST because of the extra functionality it provided. As the version suggests, beta software contains bugs which hampered the development cycles of mapCAST and ultimately the final product.

6.0 Conclusions

The purpose of his study was to develop mapCAST as a functional and network-able concept-mapping program, and to evaluate it as a collaborative tool in a real-world/collaborative situation using existing network protocols. Clearly, the test results, and final outcome of the map (see figure 23, section 4), indicate that mapCAST was successful in providing a networked environment for collaborators to create and organise a concept map in the situation presented. Of course it would take further testing sessions to see have far the map could be constructed.

The useability and feasibility of mapCAST was analysed using a testing situation that produced largely qualitative data gathered from a small sample pool. Outcomes from analysis of this data strongly suggests that mapCAST needs further functionality as an application (including 'standard' features) and further enhancement of the collaborative components. This would ultimately make mapCAST more 'user-friendly' and more compatible with a wider range of users and their concept-mapping process models.

Before injecting mapCAST into further experimental situations, it would be desirable to fix and enhance the application based on comments and results from this study. This would include:

- The inclusion of more 'standard' features.
- Adding more shapes for nodes, such as ovals, triangles and diamonds.
- More control over appearance of nodes and relationships such as border and line width, font face and font size.
- Printing of notes. At present notes attached to nodes and relationships can be exported to Microsoft Word (for example) to be printed. However, notes are not printed within mapCAST.
- Support for asynchronous collaboration. This can already be achieved via a shared network drive. Furthering this, a server application could be built that allowed the uploading of concept maps via mapCAST. The server could offer such features as date created/modified information, author information, a description, and the ability to store each saved state of the concept map, which would provide a of history of the map.
- Enhanced communication. A possible inclusion would be to incorporate live, streaming QuickTime audio as the main communication medium between clients. How easy this would be to implement I am unsure, but certainly warrants investigation.

Future Applications

The real-time collaborative concept mapping application mapCAST has proven that Internet-based concept maps can be successful. But this is based on a situation where three people, with prior knowledge of the subject, and two computers, used the software for knowledge organisation. More research is needed to investigate:

- how many groups of people or individual collaborators are feasible before the mapping process breaks down;
- which type of communication mediums enhance the collaborative process more effectively, including text, voice or video;
- other type of mapping processes mapCAST supports, not just those involving knowledge organisation.

Further investigation could be facilitated by increasing the amount of collaborators on a map with more clients connected, with either a single person, or groups of people per computer. A larger sample group would make the gathering of quantitative data via questionnaires or alike, more feasible. I envisage a classroom situation, where three to four groups of people (of no more than four people) are assigned to a computer. A task could be set out for the groups to collaborate on and complete. Such a situation would really test the ability of mapCAST as not only a collaborative tool, but also one that potentially aids the learning process.

References

Anderson-Inman, L. & Horney, M. (1997). Computer-based concept mapping: Enhancing literacy with tools for visual thinking. Journal of Adolescent & Adult Literacy [on-line]. Available WWW: http://proquest.umi.com/pdqweb?TS=926756757&RQT=309&CC=2&Dtp=1&Did=000000010440170&Mtd=1&Fmt=4 [accessed May 1999].

Ferry, B., Hedberg, J., & Harper, B. (1997). How do Preservice Teachers use Concept Maps to Organise Their Curriculum Content Knowledge? [on-line]. Available WWW: http://www.curtin.edu.au/conference/ascilite97/papers/Ferry/Ferry.html [accessed April 1999].

Gaines, B. R., & Shaw, M.L.G. (1995). Collaboration through Concept Maps [on-line]. Available FTP: ftp://ksi.cpsc.ucalgary.ca/KSI/RTF/CSCL95CM.RTF.Z [accessed November 1999].

Gaines, B. R., & Shaw, M.L.G. (1995). Concept Maps as Hypermedia Components [on-line]. Available FTP: ftp://ksi.cpsc.ucalgary.ca/KSI/RTF/ConceptMaps.RTF.Z [accessed November 1999].

Hibbard, J. (1997). Straight line to relevant data [on-line]. Available WWW: http://proquest.umi.com [accessed May 1999].

Inspiration [Computer Software]. (1998). Available WWW: http://www.inspiration.com [accessed November 1999].

Isaac, S., & Michael, W.B. (1995). Handbook in Research and Evaluation, Third Edition. San Diego, California: Edits

Kennedy, D., & McNaught, C. (1997). Use of Concept Mapping in the Design of Learning Tools for Interactive Multimedia. Journal of Interactive Learning Research, Vol. 8, No. 3/4, 389-406.

Kommers, P. (1997). Special Issue Preface, Concept Mapping. Journal of Interactive Learning Research, Vol. 8, No. 3/4, 281-287.

Kremer, R., & Gaines, B. R. (1996). Embedded Interactive Concept Maps in Web Documents [on-line]. Available WWW: http://www.cpsc.ucalgary.ca/~kremer/webnet96/webnet_kremer.html [accessed November 1998].

Neuberg, M. (1999). REALbasic: The Definitive Guide. California: O'Reilly & Associates, Inc.

Novak, J. D. (1998). Learning, Creating and Using Knowledge: Concept Maps™ as Facilitative Tools in Schools and Corporations. New Jersey: Lawrence Erlbaum Associates, Inc.

Novak, J. D., & Gowin, D. B. (1984). Learning how to learn. Cambridge, New York: Cambridge University Press.

Reader, W., & Hammond, N. (1994). Computer-Based Tools to Support Learning from Hypertext: Concept Mapping Tools and Beyond. Computers & Education, Volume 22, Number 1/2, 99-106.

REALbasic [Computer Software]. (1999). Available WWW: http://www.realsoftware.com [accessed March 1999].

Rautama, E., & Tarhio, J. (1998). Sharing Concept Maps on the Web. Proceedings of ICCE'98, Vol. 1. Global Education ON the Net, 273-280.

SemNet [Computer Software]. (1993). Available WWW: http://trumpet.sdsu.edu/ [accessed March 1999].

Sung, T.C., Chen, S.F., Lin, S.C., & Chang, K.E. (1998). Concept Mapping System with Scaffolding Learning. Proceedings of ICCE'98, Vol. 1, Global Education ON the Net, 692-700.

Twidale, M.B., Rodden, T., & Sommerville, I. (1994) The Use of a Computational Tool to Support the Refinement of Ideas. Computers & Education, Volume 22, Number 1/2, 107-118.

Appendices

Appendix A: Definition of Terms

Beta – Describes a state of a software development cycle. A Beta release of software is one that is not ready for release to the general public due to bugs (usually from the inclusion of new features), and other anomalies.

GUI – An acronym for Graphical User Interface. The user interface is how a user interacts with a program and can consist of buttons, sliders, windows, pictures, sounds and alike.

Internet – The Internet is a global network of computers that communicate via a standardised protocol known as TCP/IP. The Internet hosts numerous services such as web pages and email.

IP Address – An IP, or Internet Protocol address is a 32-bit binary value that uniquely identifies each computer on a network. An IP address consists of 4 8bit numbers separated by periods. For example: 139.230.169.149

LAN – An acronym for Local Area Network, dedicated to sharing data among two to several hundred computers.

MacOS – An acronym for Macintosh Operating System. An operating system controls hardware as well as being a platform for software to execute. At time of writing the current MacOS version is 9.0. MacOS is the only operating system for Apple computers, while Windows 95/98/NT is the predominant operating system for IBM and compatibles.

Object Orientated – Object orientated technology is implemented in many programming languages, including C++, Java and REALbasic. Object orientated languages consist of software 'objects' that have their own state and behaviour as defined by their variables and methods. Object orientated languages are considered advantageous to programmers because of the ability to code in modules.

Port – Most computers have one connection to a network, in which all data is received through for many network dependent applications such as web browsers, telnet, and e-mail. A port is a 16-bit number ranging from 0 to 65535 that maps all

incoming data to the right application. The 'well known ports' include 80 for HTTP services and 21 for FTP, and are reserved for each protocol.

Protocol – a definition of a language to communicate data between objects. For example, TCP/IP is a protocol used to communicate web pages (HTTP data) between a server and a client.

REALbasic – A MacOS implementation of the BASIC language. BASIC is an acronym for Beginners All-Purpose Symbolic Code – a simple but powerful object-orientated language.

Socket – A socket is one end of a two-way communication link between two computers on a network. When a connection is established between a server and a client, the client binds a socket to its assigned port number. This socket is then used to communicate data for the entirety of the connection.

TCP/IP – An acronym for Transmission Control Protocol / Internet Protocol. TCP/IP is the standard protocol for data transmission between computers on the Internet. TCP/IP sends packets of data between computers, routing through one node to another until the destination is reached, at the same time providing verification that the data was sent and received correctly.

Appendix B: List of Figures

Appendix C: Evaluation Form

Interface Design	poor	average	excellent
intuitive icons	|-----|-----|-----|-----|-----|		
network management controls	|-----|-----|-----|-----|-----|		
collaborative status and feedback	|-----|-----|-----|-----|-----|		
communication controls (ie messaging)	|-----|-----|-----|-----|-----|		
menu design	|-----|-----|-----|-----|-----|		
tools palette design	|-----|-----|-----|-----|-----|		
netStatus palette design	|-----|-----|-----|-----|-----|		
inspector palette design	|-----|-----|-----|-----|-----|		

Functionality	
creating documents	|-----|-----|-----|-----|-----|
opening and saving documents	|-----|-----|-----|-----|-----|
map creation tools	|-----|-----|-----|-----|-----|
map revision and editing tools	|-----|-----|-----|-----|-----|
network settings and connection management	|-----|-----|-----|-----|-----|
collaboration	|-----|-----|-----|-----|-----|
communication between clients	|-----|-----|-----|-----|-----|
printing	|-----|-----|-----|-----|-----|

Useability	poor	average	excellent
speed of program	|-----|-----|-----|-----|-----|		
network speed	|-----|-----|-----|-----|-----|		
ease of concept mapping	|-----|-----|-----|-----|-----|		
reliability (no bugs or crashes)	|-----|-----|-----|-----|-----|		

General Comments

Appendix D: mapCAST protocol

November 1999

m a p C A S T 1.0 - [TCP/IP Protocol]

This protocol is made up of headers, which define the type of information that is sent/received.
The syntax of the headers is:

```
Header || data1 [| data2 [| data3 ]... ] |||

||   = ASCII char(12) - Header/info delimiter
|    = ASCII char(14) - Internal info delimiter
|||  = ASCII char(11) - End of data
```

-- Messages sent TO the server

---

* Notes

These are typically from the mapCAST application acting as a client

* Headers

000 Sending Connection Info

```
000 || Nickname | iconID |||
```

001 Sending server chat

```
001 || message |||
```

200 Waiting for Map

```
200 || |||
```

201 Map was Received OK

```
201 || |||
```

```
202   Request Control Of Map
      202 || |||


------------------------------------------------------------
-- Messages sent FROM the server
------------------------------------------------------------

* Notes

These are typically from the mapCAST application
acting as a host, that is, hosting a map and all
connected clients

* Headers

000   Error
      000 || errorID |||

      errorID's
      -----------------------------
      000   Too many users
      005   Error sending Map Data


001   Broadcast Server Message (if enabled)
      001 || server message |||


002   New User Connected
      002 || nickname | iconID | userID |||


003   Sending chat from server
      003 || message |||


004   User Disconnected
      004 || userID |||
```

```
200   Client Ready for Map?
      200 || maplength(bytes) |||


201   Sending map data
      201 || mapname | mapdata |||


202   Send Map Controller (true(T) or false(F))
      202 || boolean |||


-------------------------------------------------------------
-- Common Messages
-------------------------------------------------------------

* Notes

Because the mapCAST application can act as a client and
as a host these messages can be sent from either.

* Headers

250   Move Object
      250 || objectID | newX | newY |||


251   Add Node
      251 || x | y |||


252   Add Relationship
      252 || startID | endID |||


253   Delete Object
      253 || objectKind | objectID |||
```

254 Delete All Objects

254 || |||

255 Delete All Relationships

255 || |||

256 Resize Object

256 || objectID | newWidth | newHeight |||

280 Inspect object

280 || objectID | objectKind | ObjectData |||

281 Inspecting (true or false)

281 || T or F |||

290 Undo

290 || |||