A comparison of two approaches to Web access for blind users

MEng Third Year Project (PR3)

Simon Thompson
(sjt104@york.ac.uk)

Supervised by: Dr. A. D. N. Edwards
(Alistair.Edwards@cs.york.ac.uk)

Department of Computer Science
The University of York
Heslington
York
YO10 5DD

Submission Date: 16th March 2000

A comparison of two approaches to web access for blind users

Abstract

The content and size of the World Wide Web has increased over the past few years. With this development and increased number of users, certain accessibility problems have surfaced. One particular group of users who may not be able to access Web information is blind users.

Various tools have been developed to help blind users access the web. Two of these include Betsie, a script that parses the content of web pages before presenting information to the user, and BrookesTalk, a specialist web-browser. This project aims to compare the two tools that are based around different technologies and to highlight good and problematic areas of each.

Both Betsie and BrookesTalk were found to present the user with an adequate representation of pages, however certain specific problematic areas were found. Users of Betsie were often noted to become disorientated within a page and it was noted that this was because information extraction was difficult. This did not occur for users of BrookesTalk who were able to extract key document information by using the conceptualisation tools that form a part of it.

An implementation of a similar system was added to Betsie and whilst found to be useful, sometimes gave excessive information.

Various navigational issues were raised with BrookesTalk, this included a focus problem and form filling problems. It was also noted how a search within page may be a useful feature to add to BrookesTalk.

Acknowledgements

I would like to thank the five testers who volunteered to take part in the evaluation process of this project and provided interesting comments and views on the tools tested. My thanks also go to Alistair Edwards for supervising this project and to Ben Challis for his assistance in using the Human-Computer Interaction Lab. I would also like to thank Stephanie Booth for agreeing to proofread this report.

Table of Contents

1. Introduction 5
   1.1. Initial Project Objectives 5
   1.2. The Internet as a resource 6
   1.3. The development of the Web 7
   1.4. Accessibility Issues & the Web 8
      1.4.1. Disability Law 9
   1.5. Research Into web accessibility 10
   1.6. Web page design guidelines 11
      1.6.1. Images & Animations 11
      1.6.2. Image maps 11
      1.6.3. Multimedia 11
      1.6.4. Forms 12
      1.6.5. Frames 12
      1.6.6. Scripts, Applets 12
      1.6.7. Document layout 13
      1.6.8. Verify Hypertext 13
      1.6.9. Verify Accessibility 13
   1.7. Project Undertakings 13
   1.8. Report Outline 14
2. An Overview of Betsie and BrookesTalk 15
   2.1. Betsie 15
   2.2. BrookesTalk 19
      2.2.1. Features of BrookesTalk 20
   2.3. Summary 22
3. Initial Evaluation 23
   3.1. Personal Observations 23
      3.1.1. Preparing the tools for use 23
      3.1.2. Applying the tools 27
   3.2. User Evaluation 28
      3.2.1. Techniques to test the tools 29
      3.2.2. Results and analysis 32
   3.3. Summary 34
4. Development of Betsie 37
   4.1. Generating a document summary 37
      4.1.1. Providing access to the summary 37
      4.1.2. Generating a page summary 38
   4.2. Towards an automated document abstract 44
      4.2.1. Abstract generation algorithm 45
      4.2.2. Implementation of the abstract generation algorithm 45
   4.3. Summary 46
5. Further Evaluation 47
   5.1. Evaluation Design 47
      5.1.1. Selecting target websites 48
      5.1.2. Websites used for testing 48
      5.1.3. User Response 49
   5.2. Results and analysis 51
      5.2.1. Questionnaire results 51
      5.2.2. Interview results 53

5.3. Summary 55
6. Conclusion and recommended work 56
6.1. Comparison of the tools 56
6.1.1. General navigation 56
6.1.2. Images 56
6.1.3. Conceptualisation tools 57
6.1.4. Search 57
6.1.5. Forms 57
6.1.6. Tables 58
6.1.7. Installation Issues 58
6.1.8. Financial Implications 58
6.2. Development ideas for each tool 58
6.2.1. Ideas for developing Betsie 59
6.2.2. Ideas for developing BrookesTalk 59
6.3. Areas for further work 61
6.4. Self-criticism 62
7. References 63
Appendix I 65
Code for Betsie 65

Table of Figures

Figure 1 Diagram shows how a blind user may browse the web, the arrows indicate requests and the dotted lines show two-way data flow 15
Figure 2 A screenshot of a web page that uses frames, the scroll bar can be noted to be a visual cue splitting two windows into distinct text areas 16
Figure 3 Tasks the two volunteers were asked to undertake 30
Figure 4 Questionnaire for initial evaluation of the tools 31
Figure 5 Screen shot showing how the modified Betsie script gave information to the user about the summary and provided a link to it 38
Figure 6 Code extract showing the additional variables needed to be configures resulting from the addition of the page summary 39
Figure 7 Example Perl substitution using pattern-matching statements 40
Figure 8 Words removed from a document before keywords are extracted 42
Figure 9 Endings of words removed to leave stems of words 43
Figure 10 Closed questions each user was asked to complete after use of each tool 49
Figure 11 Structure of the interview used to obtain further user comments 50

Index of Tables

Table 1 Table showing results of the evaluation questionnaire 32
Table 2 An example hash table giving key and value pairs 43
Table 3 An example rank table using frequencies to match to stemmed words with that frequency 44
Table 4 Answers given by each user to the selection answer questions in the second evaluation 51

1. Introduction

This chapter covers the initial ideas for the project and some indication of the stimulus behind undertaking this project. It should be noted that where the term blind is used, this refers to people with no sight, partially sighted refers to people with some sight and visually impaired refers to both blind and partially sighted people.

1.1. Initial Project Objectives

In recent years, the Internet has expanded massively, with both company and home users having the power to access it. It is becoming a vital if not invaluable source of information for all purposes – from research papers, to on-line shopping, to personal holiday photos. Whatever you may wish to use it for and whoever you are, you should be able to access the wealth of information distributed over it. For many people, using office or home PCs and browsing with their favourite software, this is not a problem. Sighted users can see pictures on pages and can read the text provided saving or printing pages considered useful and rapidly discarding obviously irrelevant pages. Unfortunately, not all are able to access the Internet in the same way. Take for example, blind users, they cannot read the text on screen or see the pictures that sighted users can, instead they have to rely on other users or means to access the information provided. This project aims to consider various ways of accessing information on the Internet for blind people.

Two browsing tools have been developed aimed specifically for blind users, these are BrookesTalk, a specialist web-browser, and Betsie, a gateway script, the details of these tools will be described fully later. The project will consider the two browsing methods, evaluate their effectiveness, and identify any shortcomings. Betsie is a piece of open source software written by Wayne Myers at BBC Digital Media [Betsie 1999] and hence provides the opportunity for modification. BrookesTalk is in active development by The Speech Project [Speech Project] at Oxford Brookes University who have been contacted concerning this project and are prepared to read any conclusions that are found. It is important to note that BrookesTalk is a multipurpose browser in that it is also designed for partially sighted users by providing a visual representation of pages as well as an auditory representation. This is an important observation to make as many partially sighted users wish to use the sight available to them and have other aids rather than to simply rely on an auditory mechanism. Edwards and Stevens [Edwards & Stevens 1997] note that “if a person receives conflicting information on different senses it will usually be the visual signal that is heeded”, and this may be an explanation as to why partially sighted people prefer to use their available sight when possible. As both of the software tools have the ability to be modified or developed as deemed necessary from conclusions drawn because of this project, they will be the only products considered. Whilst other products do exist, the implementation of changes to a commercial project are likely to be harder to achieve and hence of less practical use.

The aims of this project are:

1. To investigate the functionality and effectiveness of Betsie and BrookesTalk;
2. To implement changes that will result in advantageous use of Betsie;
3. To re-evaluate Betsie and BrookesTalk;

1.2. The Internet as a resource

During recent years, the use of the Internet has exploded with the development of the World Wide Web. The British government aim to "connect" every school in the country to it within the next few years. It is used by millions of people every day from across the globe and provides a vital means of communication between people, companies, and countries. A recent statement by Tony Blair PM [Blair 1999] claimed that businesses that did not develop their web presence would be in serious danger in the future. Whilst this may be extreme, it does show the potential of the Internet both now and in the future.

The term "the Internet" has many different meanings to different people, it encompasses a huge variety of systems and communications protocols, including the World Wide Web or web, email, 'ftp' (for transferring files – file transfer protocol) and gopher. Some of these systems are familiar to all users, for example the web and email, but others may only be known about and used by specific groups of people. For example, gopher is mainly used within the scientific world. For the majority of users who say they use the Internet, what they mean is that they use the web and hence this project will specialise in this area.

With the expanding use of the Internet and the information available on it, it would seem unfair if everyone could not have access to the information provided. In fact, it could be that with careful design of web pages, information could be presented to all users in a simpler and more accessible fashion than is currently used.

"The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect."

-- *Tim Berners-Lee, W3C Director, and inventor of the World Wide Web*

An example of how accessible design can be advantageous is outlined below, a telecommunications company may do one of the following to send out a bill to a customer:

1. Send out a normal printed bill;
2. Send out a Braille version of the bill;
3. Send out a taped version of the bill;
4. Send an electronic version of the bill for use with a blind person's screen reader;

At this point, it is necessary to define the term 'screen reader'. This is a piece of software that has the ability to pass text displayed on screen to a suitable output device. The term screen reader does not mean solely speech based output as many screen readers have the ability to be connected to other output devices including Braille pads which present the user with a tactile representation of the text on screen. Many current screen readers including the one used for evaluation purposes in this project JAWS [JAWS], include a speech synthesiser that can be used with a standard Windows compatible soundcard.

Whilst the latter three of the examples given above are all suitable for a blind person, the second can only work if the blind user can read Braille and the third would be expensive in terms of tape and time to record the tape. In contrast to this, the fourth option allows the blind user to browse their bill at leisure and at little cost to the telecom company (as their records will be stored on computer in any case!).

This project aims to concentrate on the needs of blind users, but accepts that the blind are not the only group of people who may have difficulty accessing the web.

1.3. The development of the Web

The web is a collection of sites distributed over thousands of computers around the world, each site consists of a collection of pages, known as web pages. These pages are built using hypertext mark-up language (HTML). HTML is documented and specified as a document type definition (DTD) by the World Wide Web Consortium or w3c [W3C], which is made up of individuals and company representatives from around the world (including IBM and Microsoft). Suggestions are made when new specifications and modifications are being designed which are discussed and then published as a standard.

On 24th December 1999, HTML 4.0.1 [HTML4.01 1999] was published. In theory, all web pages should conform to this standard (a validation service [VALIDATER] is provided as a quick, simple and easy check for this). With the millions of users worldwide each wanting their own web page, this is however, unlikely to happen. It is worth noting that HTML 4.0.1 (and 4.0 [HTML4.0 1998] to which 4.0.1 corrects a number of errors) is designed with total accessibility in mind. Prior to HTML 4 have been HTML 3.2 [HTML3.2 1997] and HTML 2.0 [HTML2.0 1995]. HTML 2.0 was an early standard developed when the web was first appearing however it had little support for anything other than text or links and hence users developed new tags, many of which were implemented in the then, leading browser, Netscape Navigator. At the time, many of the tags (mark-up codes used to identify or label elements within the document, e.g. `<p>text</p>` would be used to indicate that "text" is classed as a paragraph) included in HTML 3.2 were not rendered by the many browsing tools available. Since that time, newer versions of the browsers have been released which do handle all the tags in their own way.

As the web has developed, people have demanded more graphically rich pages in the same way as companies have moved from simple black and white leaflets to coloured, glossy brochures. The HTML standard was however developed simply to structure documents, and not to provide layout thus allowing individual browsers to render pages in an appropriate manner providing scope for specialist uses. In the early days of the web, many people used the same browser, Netscape Navigator, and page designers turned to the way it rendered tags to provide layout. For example, the heading tags `<h1>`, `<h2>` … were used to display different sized text because this is how Netscape visually renders these. Whilst this technique has the desired effect for viewing in Netscape, the underlying structure of the document may be destroyed. For this reason, the w3c have developed a method for adding layout information to pages known as "cascading styles sheets" [CSS] or more simply, style sheets. Style sheets allow localised definitions of how the browser should visually render a page, thus leaving HTML for use in determining the document structure.

When page designers misuse the HTML language to provide document layout rather than structure, specialist browsers that render tags in different ways are affected. For example, the `<h1>` tag may be used by a designer to achieve larger text, however a specialist browser for blind people may represent the enclosed text as louder or with emphasis. When the tag is misused, the meaning of the page may not be correctly conveyed to the blind user. Further

aspects of web accessibility are discussed below but it should be clear that correct use of HTML helps to minimise accessibility problems.

1.4. Accessibility Issues & the Web

When the HTML 4 standard was developed, the issue of accessibility was raised and the resulting standard was aimed at providing fully accessible web sites. Clearly, this will only be possible if pages do adhere to the HTML standard. As such, page designers should use the free validation service [VALIDATOR] provided by the w3c. Other tools are also available for checking the syntax of web pages and include "Bobby" [BOBBY] developed by CAST. Bobby is a tool that analyses web pages and reports on potential accessibility problems for users. The following section discusses some of the common issues concerning accessibility for blind users.

One of the problems affecting many screen readers is the inability to represent information displayed that is non-textual. This causes a problem when textual information stored in an image or picture is incorporated in a page. For example, often designers wish to indicate that a particular section of a web site is new and represent this with the word "new" surrounded by some form of coloured star. This would cause a problem for a screen reader because it has no way of accessing the text stored in the image and hence the meaning could potentially be lost for a blind user. HTML 4 provides a mechanism for providing a short textual description of the image in the 'ALT' attribute added to the tag as well as a link to a file that can contain a detailed description of the image in the 'LONGDESC' attribute. The long description is optional because not all images require a full description - a photograph may do, however an image used to depict "new" only needs alternative text of "new" to fully describe it. This attribute can be used by blind users because many web browsers support displaying the text instead of the image that can then be accessed using a screen reader.

The requirement for 'ALT' text was only added in the HTML 4.0 standard, previously it was optional and as such, many web sites do not include it. It should also be noted that providing 'ALT' text of the filename is often of little use, for example even to a sighted user 'fig09a.jpg' has no meaning as to the content of the image. It has been suggested [Jenkins 1997] that future image standards should themselves provide the option to include a description of the image within the file itself.

As an aside, many current browsing tools do not provide support for the 'LONGDESC' attribute and as such the w3c have suggested using a specific link following an image referenced as just 'D' to a file containing a description.

It should be noted that not only images provide problems with accessibility for blind users, other new technologies also do. Some examples are Shockwave flash [SHOCK], a tool for generating multimedia presentations using images and vector manipulation for animation, and Java [JAVA], a programming language designed to run on a "virtual machine" which is emulated in software on the client computer. When these technologies are used, images often form an integral part of what is being described. Haywood [1997] commented on how the effects of the then new technologies such as ActiveX [ACTIVEX], (a technology developed by Microsoft used for programming which allows modules that perform specific operations to be included in applications and web pages), and Java would adversely affect browsing. If we

consider existing browsers, it can be seen how specialist technology use can cause problems for blind users. This is because little or no support for technologies is available – Lynx (a well used text based browser) for example, has no support for ActiveX that can often be used for key features of pages because of the ease of use by the page designer. Often, no consideration is made for users without the capability of ActiveX. This technology renders all users of Unix based machines disabled to a certain extent because, at the time of writing, ActiveX is only supported on the Microsoft Windows 9x and NT platforms.

Despite the efforts of the w3c to ensure all web pages are accessible by using their standards, in the real world pages are not going to be ideal and hence specialist applications and technology are required. This may be through pages being designed to previous HTML standards, or insufficient testing of the site as well as a designer totally failing to consider users unlike themselves (i.e. Blind or in some other way disabled) or simply through bad site design.

1.4.1. Disability Law

Both the UK and USA have laws protecting disabled users, these take the form of the 'Disability Discrimination Act' [DDA1995] and the 'Americans with Disabilities Act' [ADA 1990]. The American law effectively states that it is illegal not to provide accessible information, whereas UK law claims that it is only illegal to refuse accessible information when asked for it. In America, a case has tested this in terms of computing information regarding blind users' access to AOL [Mendels 1999], however no UK cases have yet been brought. Edwards [1997] discusses the feasibility and use of legislation and it is important to note that the imposition of legislation has implications on the freedom of speech of the author – a person may not be capable of providing a fully accessible page and the law should not stop them from publishing what they can write. Edwards brings forward the point that the Internet as a whole is not owned by anyone and hence implementing any such law would be difficult. He goes on to imply that some sort of prestige for accessible pages would be better in terms of encouraging others to do likewise. At present the w3c provide a banner for pages that conform to the standards, this is accepted as a form of prestige attributed to a well-designed page. Cast, with their tool "Bobby" have also implemented a similar system and increasing numbers of pages are being classed as Bobby accepted implying their accessibility.

At present, many blind users do use the web with limited accessibility, either through specially designed web browsers (which often suffer from lack of support for new technologies and tags) or through commonly used visual web browsers using a screen reader. Haywood [1997] performed tests on the use of these specialist and adapted browsers, however some of her conclusions as to the use of adapted browsers are no longer valid – for example, Microsoft Windows 98 provides greater accessibility options and their Internet Explorer (versions 4 and 5) provide accessibility handles – these provide information to external software regarding the page in use. Jaws for Windows (a popular screen reader) provides support for these accessibility handles which make browsing in Internet Explorer easier. One of the comments made by Haywood [1997] was that it was difficult to identify links in that the user had to listen out for the screen reader noting a change of colour. When using Jaws and Internet Explorer, links are easily identified because the screen reader is made

aware that the text is a link by using accessibility handles. It is for this reason that testing of Betsie shall be undertaken using Internet Explorer and Jaws – so that any problems with accessing pages can be attributed to Betsie and not through lack of understanding or functionality of the screen reader/browser combination.

There are effectively three ways in which a blind user can access the web, these are summarised as follows:

- Use a standard browser and screen reader;
- Use a standard browser with a gateway parser and screen reader;
- Use a specialised browser;

Betsie falls into the second category and BrookesTalk into the third. The first category of access is generally unsuitable for providing meaningful access to web page because the majority of screen readers available have problems when tables and frames are used in page layout. This is due to the nature of screen readers that simply read text across the screen as they are unaware of any layout and formatting provided by the web page to the sighted user.

1.5. Research Into web accessibility

Much research has been undertaken into making the web accessible including that by companies such as Microsoft and IBM who have also joined the w3c and are involved in the development of web accessibility guidelines which form part of the w3c web accessibility initiative [WAI]. These guidelines set out ways in which web designers can improve the accessibility of their pages for all. Unlike this project, it does not only consider blind users but also other users who may experience problems accessing the web, for example, those using a mobile telephone or people using text only access methods.

Jenkins [1997] describes some of the problems that IBM experienced in developing an accessible web site. Much research and testing was undertaken to ensure that the site was accessible to all users and the research highlights some key groups who have problems accessing the web. These include people with mobility problems who may not be able to hold down a key or press multiple keys. This highlighted the need not only for IBM to produce accessible web pages, but also to produce accessible software as a whole and highlights specific items such as using standard input and output techniques to help reduce conflicts with assistive technologies as well as allowing individual users to have a profile whereby they can customise areas of the software. Jenkins also notes that as technology continues to develop and expand, making pages accessible will become more difficult because people as a whole use them in different ways. Jenkins concludes by stating how difficult is has been and continues to be to encourage each group in the ibm.com domain to convert their pages to be accessible. Jenkins also comments on how making departmental representatives who are responsible for web pages aware of the needs and capabilities of accessible web design. The aim of this is to provide accessible pages in the first instance rather than getting changes made afterwards. As more people involved in development become aware of web accessibility issues, software as in general should become more accessible because similar principles apply to both the web and software.

1.6. Web page design guidelines

The following presents a set of guidelines to web designers that if closely followed should provide a useful and informative site to both blind and sighted users with little effort on behalf of the page designer. These guidelines are based around ideas presented by IBM [SNS 1999] and the w3c WAI [WAI]:

1.6.1. Images & Animations

Provide ALT="text" for all images which is meaningful and concise. That is the text described earlier and used to provide users with a description of images for use when non-visual or text-only browsing is undertaken. Do not use animations unless the meaning can be properly conveyed with other means, this is also important even for visual users as moving images are distracting to the eye and hence can distract sighted users as their eye is drawn towards the changing information.

1.6.2. Image maps

Use client-side rather than server-side image maps and provide text descriptions of the link – browsers not using image maps can access this and provide equal functionality. Where server-side image maps are used, provide equivalent textual links. Image maps provide sets of co-ordinates and corresponding links that can be applied to images such that when the user clicks the part of the image surrounded by a set of co-ordinates, the corresponding links is activated. For example, a web page may contain a map of Europe, links could be set up using co-ordinates so that clicking on the UK would follow a link taking the user to a page about the UK. This would be an image-map (although a photo of a band could be used rather than a map of the UK and clicking each member of the band takes the user to a different personal page). It is clear how non-visual users would have problems following image map links unless alternative text is provided for each set of co-ordinates.

A server-side image map requires special software to be installed on the web-server and a file containing co-ordinates and corresponding URLs is stored on the server. When the user clicks over a mapped image, the co-ordinates of the pointer are passed to the web-server that then redirects the user to the appropriate page. Server-side image maps tend not to be used because the syntax for writing the image map is server-software specific and often difficult to configure. Client side image maps on the other hand have co-ordinates coded into a web page and the following of links is co-ordinated by the client web-browser rather than the web-server. Client side image maps form part of the HTML [HTML4.0 1998] standard and hence can be transferred to different web servers without the need to re-write the co-ordinate information. This is because the handling of the redirection is undertaken by the client. Client side image maps allow specialist browsers to handle image maps in their own way, so for example a non-visual browser may provide a set of links extracted from the image as access to the source code is possible. This option is not available when a server-side image map is used and hence the need for additional textual descriptions.

1.6.3. Multimedia

Provide text transcripts of audio and video presentations. For example, a site containing a RealAudio [RAUDIO] interview should also provide a transcript. Not only is this essential for blind users who may encounter problems when using a screen reader due to technical

limitations of soundcard output, but also for users who wish to read the page without being connected to the Internet. RealAudio is a standard used for “streaming audio” where chunks of audio are sent to the user and re-assembled in the correct order. It is intended to provide a better form of listening to audio on the Internet than downloading whole files because listening can begin before the whole “stream” has been received. The nature of the streamed audio makes it difficult to listen to whilst not connected to the Internet and hence a transcript may be necessary.

1.6.4. Forms

Forms are electronic versions of paper fill out forms but can have greater uses. For example, they may be used to request a brochure from a company, to fill in an electronic visitors’ book, or to provide a search string to an Internet search engine. Elements available for use in forms include check boxes and text boxes. A blind user may find it difficult to conceptualise the layout of the form and cannot see to click into each box provided, hence it is important to associate labels and tab indexing with all elements to aid the navigation of the form for blind users.

1.6.5. Frames

Provide titles for frames so that the user can track location. Provision of access into the site for non-framed browsers is also necessary. Frames are a feature of HTML that allows multiple pages to be opened within the same context and communication between the pages is allowed. This technique is often used to provide a navigation bar and a content window because following links in the navigation bar can cause the page displayed in the content window to be dynamically changed. Framed pages can cause problems when a browser and screen reader are used and this is discussed further in the next chapter.

1.6.6. Scripts, Applets

Provide alternative content for scripts and applets and ensure functionality or meaning is not lost when disabled. Scripts and applets are mini-programs that run when an event happens to a web page. Events include the loading of a page, or clicking or moving the pointer over some area of the page. Often this is used to generate dynamic content, that is, something about the page changes when an event occurs. An example of an event is when the mouse pointer is moved over an image or block of text. The event handler for the specific “onmouseover” event triggers a textual description to appear on the screen. As the content changes, this causes problems for screen readers and for blind users who have no visual cue as to the changing content. It is therefore vital that dynamic content is not required for full functionality of the page.

Applets run in an object frame within a page and will often contain dynamic content. Many pages use Java applets to provide functionality because of their platform independence provided a Java machine is installed on the client computer. Scripting languages on the other hand vary dependant on browser and operating system. Java applets also have compiled code and hence the source code is hidden from the client. The NatWest [NatWest] use this technology for online banking because it provides a secure way of accessing their databases. Many text based browsers do not support access to applets and screen readers will often have

problems either caused by dynamic content or not being able to access text within the applet object-frame, similar to the problem with text that is represented by images.

1.6.7. Document layout

Ensure HTML is used only for marking-up and identifying elements of a document and that the visual layout of the document is controlled with Cascading Style Sheets [CSS] rather than misusing HTML tags to provide features. Many designers misuse the heading tags (e.g. `<h1>`, `<h2>`) to provide text of different sizes, which may not convey the true structure of the document i.e. that an `<h2>` element is a subsection of an `<h1>` element. Cascading Style Sheets provide a page designer with the ability to specify attributes of each element in a page. Attributes include the colour, size, and typeface of text and even extend to the actual language the paragraph is written in. An attribute is also provided to change the speed at which any future auditory browser should read the text. At present, no existing browsers support this feature.

1.6.8. Verify Hypertext

Verifying hypertext ensures the underlying document structure is properly formed and hence specialist browsers that are not as fault tolerant to poor HTML will have no problems presenting the page. Use the w3c's validation service [VALIDATER] to perform this task.

1.6.9. Verify Accessibility

Check ease of use and content of the site with images and sounds disabled, verify the site using a tool, e.g. Bobby [BOBBY].

1.7. Project Undertakings

This project aims to analyse and compare two tools available for web access, these tools are BrookesTalk and Betsie. One of the many problems with adaptive technology is the cost involved to the user – a screen reader can be very expensive, as can a specialist browser. BrookesTalk and Betsie were chosen to be evaluated because they present two different approaches to making existing web pages accessible to blind (and partially sighted in the case of BrookesTalk) users. Both of the tools selected are free of charge and are able to take advantage of any conclusions found from this project because of the nature of their development. Betsie is an open source script and can therefore be modified and BrookesTalk is under development by a team at Oxford Brookes University, this will have the effect of allowing blind users to benefit from any conclusions drawn.

Betsie requires a web browser and a screen reader (it is assumed that all blind users who currently use PCs will have a screen reader) but other than that is platform independent because it acts as a gateway for obtaining pages and runs on a web server. BrookesTalk on the other hand contains a built in speech synthesiser and therefore only requires a Microsoft Windows based system.

It can be seen from this that Betsie has the potential to be very cheap for a blind user who already has a screen reader – a standard free web browser (Internet Explorer or Netscape Navigator) can be used. At present, BrookesTalk is also free because it is under development and contains an in-built speech synthesiser.

Much research has been undertaken in the field of speech synthesisers and Braille devices as well as some research into the use of dedicated browsers or browser/screen reader combinations. The advent of HTML 4 with its dedicated section on accessibility has also been the subject of much research and as such, this project aims to look at tools for accessing pages that do not conform to the latest standards or those designed without accessibility in mind.

1.8. Report Outline

The report for this project is structured in the following way, chapter 2 presents the user with an overview of the two tools to be evaluated including details of their features and operation. In order for the reader to understand the descriptions of evaluations and results, it is important to have reasonable background knowledge of the tools, which is provided by chapter 2. An initial evaluation of the tools was undertaken and details of this, including conclusions drawn are detailed in chapter 3 of the report. Chapter 4 of the report presents the reader with a description of the modifications made to Betsie as a result of the findings from the initial evaluation and includes details of the algorithms used. The last two chapters describe a re-evaluation of the tools following the modifications made to Betsie and conclusions drawn as a result of both the evaluations undertaken, and the background research completed concerning the area of study.

2. An Overview of Betsie and BrookesTalk

In order to understand fully the testing and conclusions drawn over the two browsing tools, it is important to first understand the features and results using of each tool. This section describes the ways in which the two tools can aid browsing for blind users.

2.1. Betsie

Betsie is an "open-source" CGI script written in the language Perl. This means that the source code for the tool is available to all and hence can be modified. The script runs on a web-server and acts as a gateway for blind people to access web pages. Figure 1 shows how a blind user may browse the web using various tools:

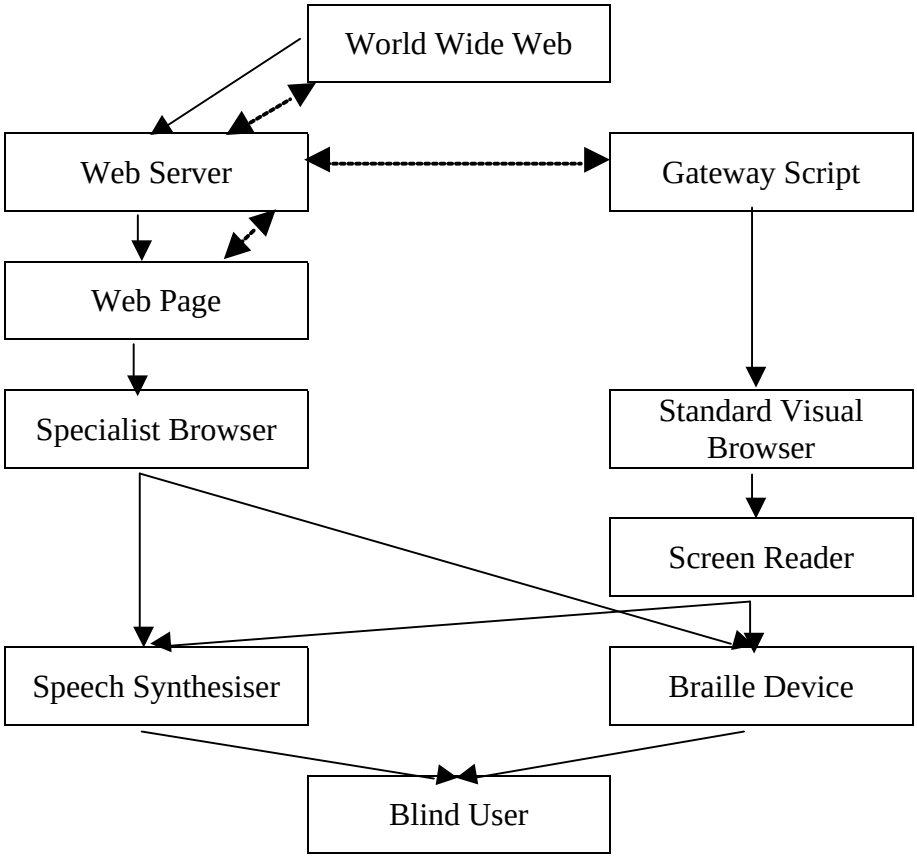

Figure 1 Diagram shows how a blind user may browse the web, the arrows indicate requests and the dotted lines show two-way data flow

In the context of web browsing, the term gateway means that the user makes web page requests to the script which causes the server to load, parse and return a more accessible version of the page to the user, this flow of information is shown in Figure 1 with dotted lines. Whilst other gateway scripts are available performing similar tasks, Betsie was chosen because it was specifically designed for presenting information to the blind. Other available scripts have been written with different applications in mind (one such example attempts to translate the language of text in a web page).

Betsie is the result of an attempt made by BBC Digital Media Services to allow blind users access to the BBC website [BBC]. This may have been as a result of a request made under

the Disability Discrimination Act [DDA 1995] or simply because the BBC perceived the need to provide blind users with access to their web site. Betsie aims to present the web-browser/screen-reader combination with a suitably modified page to provide the user a meaningful representation of the page being accessed. Past research, including that done by Haywood [1997] has shown that the combination of a screen reader and standard browser often has problems in reading out a page. This has been attributed to several factors, firstly that the screen reader can only read text presented on the screen and hence any visual distinction such as frames between boundaries is difficult for a screen reader to correctly recognise or that text presented as part of an image cannot be accessed. Screen readers also suffer from the same boundary problems when tables are used. Many screen readers do not recognise column boundaries when reading a page and an example of how a screen reader may be confused by a frame boundary is described below.

Figure 2 A screenshot of a web page that uses frames, the scroll bar can be noted to be a visual cue splitting two windows into distinct text areas

In Figure 2, the visual cue of a line and scroll bar between the two pages indicates that the left hand side is a navigation bar of some sort whilst the right hand side is text content and that the two sections are distinct and discrete areas of text. A screen reader will not recognise this visual cue and hence may instead read across the whole line in one go resulting in the following example output, “2000 Product > Bonne Marmite is closed…”. Betsie attempts to solve this problem by modifying framed pages so that each frame is layered horizontally across the screen thus avoiding the problem described above. It should be clear how the same problem could occur when tabled data is presented in a web page – the screen reader cannot use the visual cues of lines or justification to distinguish columns and as such reads across whole table rows in one go. Again, Betsie provides a solution for this problem by splitting each column of a table onto a separate line on the screen thereby separating each cell in the table. Not only is this important for accessing real data tables but also in providing the user with a representation of pages when tables have been (incorrectly) used for the layout of the page rather than simply to present data.

It has already been discussed how the use of images on the web has increased and how means of providing textual descriptions is available, Betsie is written to take advantage of this and instead of loading pictures it simply returns the text specified. Not only does this provide the user with an understanding of the image (where available), it also reduces download time because image data does not require loading.

It was stated earlier that Betsie was not specifically designed for partially sighted users, rather for blind users. Despite this, it still provides support for partially sighted users as a visual representation is presented to the user that read by the screen reader. To aid partially sighted users in using their available sight with screen reader assistance, Betsie converts all text to be displayed with larger characters and could easily be modified to represent the page using high text-to-background contrast. High contrast pages are generally found to be easier to read by partially sighted people because the difference between text and background is more easily noticed. In order for Betsie to provide larger text displays for partially sighted users, all references to the `<font>` tag are removed and a new `<font>` tag is placed at the start of the page resulting in the whole page being displayed in the same, large font.

Many web pages use forms to allow users to pass information to the web-server, this may be to provide data for a search query or for a guest-book where users can leave comments. The data passed from a form has to be handled in someway, this may be by triggering an email to be sent, starting a program on the client computer, or executing a script or program on the server. Whilst scripts and programs are different in the way they work, in terms of how Betsie accesses them, there is no difference. If a script is executed on the server then data from the form is passed to it using one of two methods called “get” and “post”. In effect, the result is the same but are different in practice, the “get” method encodes the passed data into the URL of the script being requested and is obtained from the script accordingly whilst the “post” method encodes data into a variable that is passed as part of the request for the script. In order to display the output of any scripts through Betsie, the script has to be called using Betsie. Passing data from the form to Betsie does this, and then the script is loaded from Betsie using the same mechanism as for any page. In order for this to function properly, data passed has to be forwarded to the script in the same way it would be if Betsie were not in use and thus Betsie appends data to the called URL if the “get” method is used or appropriately encodes the data into the document request if the “post” method is used.

The web is navigated by following links between documents, these can take the form of absolute links or relative links. An absolute link is that where the web-server and page or directory is specified whereas a relative link will only specify the directory or page and not the server. Both forms of link have their advantages, for example absolute links allow access to web sites that are not hosted on the local server whereas relative links allow sites to be moved between servers without the need to change every link each time the site is moved. Further to this is an optional tag that can be added to a page that specifies the base URL to which all links should be appended to form the whole link, this is the `<base>` tag. In order for blind users to navigate between pages and for all pages to be displayed through Betsie, the Betsie script must look for links in each page and modify them so that when a link is followed it is loaded through a request to Betsie. In cases where an absolute link is specified, this simply involves appending the URL to Betsie to the front of the link using the correct syntax to join them. This is more complicated when relative links are used and when the `<base>` tag is included. These cases are handled in the following manner. If the `<base>` tag is used, Betsie has to calculate what the current base URL is and then append the URL of Betsie to it appropriately. When relative URLs are used, Betsie calculates the web-sever URL and directory structure, when required, from the URL that was used to load the page

containing relative links. Once the Betsie enabled URLs are calculated, they are substituted in place of the original link.

The task of link conversion is slightly simplified in that not all links need to be modified. This is because Betsie checks if the linked page is within a list of “safe” domains. If the page URL is not listed as safe, then the original link is left intact but with the word “External” added to indicate the page loaded by following the link will not be Betsie parsed. This feature is implemented because Betsie was originally written for the BBC website [BBC] and indicates that Betsie is not tested as safe to use on pages not within the BBC domain. Some organisations may also require a notification by way of disclaimer concerning to the content of web pages that are not under the control of that organisation, this feature provides a convenient way to indicate this when using Betsie. When installed on a web-server, it is possible to modify the list of “safe” domains to allow a specific domain or indeed any domain to have linked pages parsed by Betsie.

It was discussed earlier how certain types of object that can be included in a web page, for example Shockwave Flash, may cause problems for screen readers. Betsie provides little in the way of technique to avoid this problem other than to remove images included in a page using the `<object>` tag (rather than the usual `<img>` tag) as well as modify links within the `<object>` tag to be loaded using Betsie.

In the previous chapter, it was discussed how Java applets whose content may be dynamic or inaccessible to screen readers and scripting languages such as JavaScript, often used for dynamic changes in a page, can cause problems for blind users. To counter this, Betsie takes several steps in an attempt to make the page more accessible. Firstly, Java applets are removed from the page, not only does this stop the problem of inaccessibility but also reduces the complexity of Betsie. This is because a Java applet has to be downloaded and run on the client computer. In order for Betsie to properly support Java applets, it would have to detect the presence of Java in a page and then make a second request to the web-server for this data. The Java applet data would then be loaded and returned to Betsie. Once received, Betsie would have to forward the data to the user knowing that it was an applet i.e. without parsing the data received as if it were a web page. Again, this not only helps blind users with page accessibility but also reduces the time for the page to be downloaded.

Whilst Betsie totally removes Java applets from pages, it does not remove all scripting language functions. There are two potential ways in which a scripting language can be utilised within a page, these are functions that are defined within the header information page (enclosed in the `<head>` tag) and script items that form part of the body (within the `<body>` tag) of the page. Script items that are located within the body of the document are generally only executed when the page is loaded. An example of this is a script item to show the date the page was last modified obtaining the information from the file date rather than a text area that the page maintainer has to remember to update each time a change is made. As script items within the body do not generally alter the content of a page once loaded, Betsie leaves them intact. On the other hand, script items outside the `<body>` tag are generally functions that are called dynamically as the page is being viewed in response to events occurring on the page. Some example events that may occur are, the mouse being moved over an element of a page, or an item being clicked as well as the page loading and page close events. Betsie

therefore removes all references to event handlers (the action to take on events is an attribute of a tag and may include a call to a function or a line of script) as well as scripting language code that is outside of the <body> tag. This approach to handling scripting languages may not be perfect, but in the majority of cases is effective in producing pages that are more accessible.

The general presentation of web pages is not the only aspect of web browsing that Betsie attempts to make more accessible for blind users. Anyone who has browsed the web for any length of time will know the frequency that pages are not found, caused by both incorrect and outdated links and pages being updated. To help blind users in the event this happens, Betsie replaces the commonly seen "error 404 document not found" error page with its own error page that attempts to provide the user with additional information including the URL of the page being accessed providing a blind user with information sighted users obtain by looking at the address entered. Betsie also attempts to keep the user informed about other errors that may have occurred including socket errors – a socket is opened by Betsie to a web-server to load pages – which may occur for a variety of reasons from a fault on the server running Betsie to other network errors. Betsie also provides an error message in the event of a timeout, this is where the web-server containing the requested page fails to respond within a certain amount of time, this may be of use to some users.

A further underlying feature of Betsie is its ability to follow redirects to different pages. A redirect may be triggered by two means either from the server or from within a page. A server-side redirect is generally not noticed by the user but Betsie must be able to act appropriately to server redirects in order that the correct page is obtained. A web page redirect is subtly different in that it may cause the current page to be reloaded after a period of time or a different page to be loaded. The code for this to occur is stored within a web page in a <meta> tag which also enables additional document information to be coded into the <head> of a web page. The HTML form of redirect is no longer recommended by HTML 4.01 [HTML4.01 1999] because some browsers do not support the feature. In order to stop any page Betsie loads from refreshing automatically, the tag is removed then any re-directs scheduled to occur in less than 99 seconds are followed, and the resulting page displayed to the user.

2.2. BrookesTalk

In comparison to Betsie, BrookesTalk is a different approach to web accessibility, rather than being a 'gateway' layer for a blind user to use an existing browser, it is a specialist browser that handles all document requests and the presentation of information to the user itself. This specialist browser also removes the need for a separate screen reader application because a text to speech engine is included with the package. The supplied speech engine is a part of the Microsoft Speech Development kit. At present BrookesTalk is in development and is available as a free piece of software which allows a blind user to access the web with no cost implications other than a computer fitted with a soundcard, unlike Betsie which requires a screen reader. It should be noted that this restricts the output of BrookesTalk to auditory only i.e. output to a Braille or other tactile device is not possible.

BrookesTalk presents the user with two windows when launched, the upper one presents the user with a large font version of the textual information being read, the colours for the

background and text can be adjusted to allow the user to modify them to suit their own needs, but defaults to black and yellow – a high contrast display. This window helps visually impaired users who do have limited sight to access the page without the need to rely solely on auditory output. The lower of the two windows presents a full visual representation of the page as would be seen in a normal web-browser. Again, this enables partially sighted users to access the visual part of the page without some of the potential problems described above when a traditional browser and screen reader are used – this is because the browser and speech engine being used are specifically tied together and hence information about the document structure is known, unlike when using most screen readers. This capability of the browser has enabled certain special features to be implemented.

2.2.1. Features of BrookesTalk

The standard browsing using BrookesTalk has basic functionality like Internet Explorer may have, but is limited in its capabilities, for example, files other than HTML and images cannot be retrieved and 'plug-ins' to support further technologies are not available. Java and JavaScript are also not supported. A problem with forms also exists but this will be discussed later. BrookesTalk does provide the means to move forwards and backward pages as is available in, for example, Internet Explorer or Netscape Navigator.

In addition to basic functionality, BrookesTalk provides some 'specialist' features that have been the subject of much research at The Speech Project [Speech Project], a description of these features follows. Once a page is loaded, BrookesTalk processes the contents of the page in order to extract various pieces of information to provide specialist features designed for blind and partially sighted people that are not found in conventional browsers. Many of these features have been implemented in order to ease the cognitive load of a blind user whilst browsing the web. They are designed to help the user obtain a conceptual [Zajicek & Powell 1997] model of the page. This is the ability to understand the structure and content of the page without the visual cues provided for sighted users. BrookesTalk itself is a development of an earlier research project investigating conceptual models that resulted in the software WebChat being developed.

Headings Menu

Pages that are well structured will include headings for sections of a document, for example, a chapter in a book may be called, "The Light Fantastic", but within the chapter may be subsections such as, "The Sending of the Eight". If a page is properly designed and uses HTML tags correctly, this would be structured as `<h1>`The Light Fantastic`</h1>` and `<h2>`The Sending of the Eight`</h2>`. This shows how the text enclosed by the `<h2>` tag is a subsection of that within the `<h1>` tags. The headings menu can be made to read all headings or used interactively, pressing return during interactive reading sets the current position to that heading where reading of the page can be continued.

Links Menu

A menu is provided which lists all available links in a page in order. The text provided which forms the link is read rather than a URL to guide the user as to the result of following the link. Links can all be read together or can be read out individually in an interactive manner where pressing return will result in the link being followed.

History Menu

The history menu acts as a collection of links to previously visited documents and reads out the URL of each document. The history list is only retained during the current session. As with the links menu, by reading the URLs one at a time, the document can be retrieved by pressing return.

Document Keywords

In order to help a blind user quickly grasp the contents of a page, a list of keywords is provided. This list is generated by parsing the contents of the page. Keywords are then calculated according to frequency of use within the document. The keywords that are listed are those that are the highest scoring for each page and no association with previously visited page is made even if the pages are within the same domain.

Page Abstract

BrookesTalk provides the user with an abstract of the page being accessed. Whilst keywords can help a user understand some of the content of a page, single keywords were perceived to be out of context and hence the usefulness of them alone was called into question. An abstract or abridged version of the text was therefore added to help with the understanding of the page content.

The algorithm used to generate the abstract is reasonably complicated and hence is only briefly discussed here. A more in depth description of the algorithm including an example relating to BrookesTalk is discussed by Zajicek, Powell and Reeves, [1998] in “Orientation of Blind Users on the World Wide Web”.

Effectively the algorithm looks for regularly occurring groups of words, in this case groups of three words, also called “tri-grams”, and then allocates a score to each tri-gram before selecting sentences with the highest scores to be included in the abstract.

In order to minimise the diversity of tri-grams found through grammatical differences, stems of words were used when looking for tri-grams rather than whole words. Stems are generated by removing bound morphemes from the ends of words. “Bound morpheme” is a linguistic term and refers to chunks of words that can only exist as part of another word and include for example, “ing”, “ed”, “es” and “’t”. These are all endings of words that can change because of grammatical requirements and therefore need to be ignored when generating tri-grams as the ultimate meaning of the word group is the same – runs and running are spelt differently because of grammatical requirements but are both forms of the verb “to run” and hence convey the same meaning. It should be noted that English is a poor language morphologically and that other languages, for example, Arabic have many more bound morphemes. In other languages, bound morphemes may not only be attached to the ends of words but can also be split within a word. This technique of abstract generation is therefore only reliable for the language it has been set-up for, in this case, English.

Once tri-grams have been collected, each is allocated a score relating to frequency, number of keywords in the tri-gram and the number of content words contained in the tri-gram. Content words are words that add to the meaning of the sentence rather than being required because of linguistic requirements, some non-content words are “and” and “is” which do not add to the

meaning of a sentence but merely join words together. Tri-grams are then ranked according to score and appropriate sentences selected.

Page Summary

As BrookesTalk has access to the source code of the web page being accessed, it is able to generate a summary of the page. This is available through one of the menus and provides the user with key information about the structure of the page. This information includes the number of words – a blind user cannot look at a page visually and see if it contains lots of text or is just a collection of images. A count of the number of headings and extracted keywords is also generated as well as the number of 'META' [HTML4.0 1998] keywords that are encoded in the document `<head>` section.

Help Information

At any time, a user may hold the CTRL key and press another key that will prompt BrookesTalk to give auditory, context sensitive help to the user.

A set of configuration tools is also provided to allow the user to adjust the speed of reading and the voice used in order to cater for individual needs.

2.3. Summary

After reading this section of the report, the reader should be familiar with the features and their uses of each of the two tools, Betsie and BrookesTalk. The next section of the report will look at an initial evaluation of the tools in order to find problematic areas and to look for potential improvements that could be implemented.

3. Initial Evaluation

In order to assess potential areas for development, an initial evaluation was undertaken to look at the usability and usefulness of Betsie and BrookesTalk. The evaluation consisted of two parts, personal observations of the software, and a set of tasks undertaken by several users. The external users help to discover any particularly difficult problems or particularly good features. This chapter presents the initial evaluation and the results of it.

It should be noted that this evaluation was intended to identify problems and advantages of each of the tools and hence only a small number of people were involved. An effect of using a small quantity of people is that objective, potentially inaccurate results may be obtained. In this case, this can be discounted because the object of the evaluation was to find potential problems with the tools and to get ideas on development potential for the two tools. Development ideas for Betsie were of special importance because of the ability to immediately implement and test them.

3.1. Personal Observations

In order to evaluate the software with other users effectively, I thought it necessary to familiarise myself and become comfortable using the tools so that I may provide help and understand problems users may encounter. Doing this also provided me with an early insight into potential problems that I could then include in my test for other users to see if they also had the same problems, as well as to identify immediate and potential future areas of development. Items noticed as needing immediate development applied only to Betsie as the source was available for changes to be made and were only those considered necessary to proceed with further evaluation.

3.1.1. Preparing the tools for use

The tools are designed for blind users and this project is aimed at evaluating their effectiveness when used by blind users. In order for a blind user to use either tool, installation of software is required and as such, a description of the installation process follows. Ease of installation is an important observation to make when assessing the usefulness of the tools for blind people because many installation programs do not give anything other than visual representations during installation.

Installation of the Betsie

In order for a blind user to use Betsie, some degree of software installation is required. For the majority of users this will purely be the installation of web browser and screen reader, and, as these will vary from user to user and ease of installation from application to application, will not be discussed within this report. Once a browser and screen reader are available to a blind person then nothing further needs to be done other than knowing the URL of a server hosting the Betsie script.

Betsie is a server-side script and hence requires a web server and sufficient permissions on that server to install a Perl script onto it. It is fortunate that all Computer Science students at the University of York have access to a file/web server and have the ability to install server scripts. This meant I was able to install a local copy that could easily be modified and that did

not rely on external hardware or other factors over which I may have no knowledge or control.

The installation of Betsie was found to be simple on the web-server because little is required in the way of configuration and customisation. Elements of the script that do require customisation are clearly marked. In fact, the most difficult part of installing Betsie on the student fileserver was finding the location of Perl 5 – the version of Perl installed in the default Perl location was version 4 and Betsie contains methods that are only supported in Perl 5.

Below is shown a list of variables that require customisation for the local web server (other than the path to the Perl installation):

```
$pathtoparser = “http://atlas.cs.york.ac.uk/~sjt104/cgi-bin/$name”;
$parsecontact = “sjt104\@york.ac.uk”;
$localhost = “atlas.cs.york.ac.uk”;
@safe = qw (york.ac.uk virgin.net ac.uk);
```

The first of these variables `$pathtoparser` gives the URL of the script. This is used when modifying links within a document to allow followed links to be loaded using Betsie. The `$parsecontact` variable stores an email address to be included in error messages generated by Betsie in order to allow users to contact the person responsible for Betsie on the particular web-server. Note that the @ symbol in the email address is escaped (prefixed by a “\”), this is necessary because of the syntax of Perl.

The nature of decoding URLs passed to Betsie requires that the script knows the name of the server it is installed on and this is specified in the `$localhost` variable.

Finally, an array of domains over which Betsie is safe to use is specified in the `@safe` variable. The domain list may contain partial domains as well as fully qualified server identifiers. This list of domains is used when links are parsed in order to identify links to mark as external. This feature was discussed in the previous chapter and can be used to mark pages that have not been tested as safe with Betsie and also to identify that following the link will take the user away from the current web site to a different one. Any link that does not contain any of the safe-listed domains will also not be loaded using Betsie. The example given above indicates that anything within the `york.ac.uk`, `virgin.net` and `ac.uk` domains are safe to be parsed. Some accepted web-server names would include `kipper.york.ac.uk`, www.cs.york.ac.uk as well as www.bham.ac.uk. It should be noted that repeated domains cause no problems as can be seen from using `york.ac.uk` and `ac.uk` – the York domain would be included in the more general `ac.uk` domain but was specifically included to allow easier removal of non-local computers during testing.

It is possible to allow any domain without having to include all domains in the list of safe domains by modifying the sub-routine called `safe`. This should be changed to be “`return 1;`” which has the effect of saying all URLs passed to it are safe to be parsed using Betsie.

A further variable `$maxpost` is also provided to be modified, this can usually be left unchanged but is included to stop hackers attempting to overload a web-server by sending excessive data in a post request. This works by limiting the amount of data that can be accepted. Many web-servers have a default limit to the amount of data permitted when a post

request is received and hence the `$maxpost` only needs to be modified if the server-default is a lower value than that specified within Betsie.

Once Betsie has been configured for use on a server, all that requires to be done is that it be copied into a directory with permission to execute scripts and that the script be set to executable. The technique to do this depends on the web-server and hence it is suggested that a competent and knowledgeable user of the server perform this.

It can be seen that whilst Betsie is easily configured for use, it is unsuitable for installation by a non-computer literate person. Once installed on a web-server, no further installation or configuration is required in order that any client may use Betsie for web browsing. The only installation required on behalf of a blind user is as discussed above – a screen reader and web browser, which would be required for web browsing in any case.

Installation of BrookesTalk

BrookesTalk is different from Betsie because it runs solely on the user's computer and requires no resources on external computers. In fact, BrookesTalk only requires that the Microsoft Windows operating system be installed on the user's computer.

The installation process for BrookesTalk is documented in the manual that is distributed as part of the package. The manual is aimed at partially sighted users as it is printed in reasonably large sized text. Unfortunately, this is of little use to a blind user and hence a sighted user will be required to help with installation of the software. It should be noted that an electronic version of the manual is included on the CD and providing a blind user is aware of this could load the file and access the information with a suitable screen reader. In order to help with installation for blind and partially sighted users, it may be advantageous to make use of the "auto-run" feature of Windows. Auto-run is part of the operating system that automatically runs a specific file in a CD's root directory whenever inserted into the CD drive. This could for example be used to either automatically start the installation process or to load the text file containing the instructions for installation.

The installation process directs the user to "click start, then run" and to then run three separate install programs. If the directions are followed as described in the instructions, the user will encounter problems with running the first two install programs. This is because the actual programs are located within a sub-directory of the CD and hence the person installing will have to find the correct path to the install programs in order to complete the first two steps of the installation process.

BrookesTalk is built around Microsoft speech tools and the first installer that is run installs the speech development kit. This involves three steps, confirming that the user accepts the licence agreement, adjusting the install location and finally responding to the server confirming installation is complete. After installation of the speech engine, installation of a text to speech engine is required. Once executed, the install program simply asks the user to confirm agreement of the licence. The program then installs and completes with no further user interaction.

Finally, the user is required to install the BrookesTalk browser and the program to install this does run from the location specified in the manual. During install, the user is presented with a series of messages to which a response is required. Firstly, a welcome screen to which the

user merely has to acknowledge the licence agreement, this is followed by a prompt to enter user information. Default values are pre-filled into the form. These values are extracted from the Windows registry and relate to those specified during Windows installation. The next two prompts ask the user for an installation directory and a folder under which to place BrookesTalk in the Start-menu. Once again, default values are specified. A confirmation of all the details specified then follows before installation begins. During the install process, an error message is generated. This is however, documented in the manual provided. On completion of installation, some icons are created in the Windows Start-menu. The window in which the icons are generated remains the attention of focus rather than returning to the install program for the user to acknowledge completion of the install process.

Once the three steps of the installation are complete, the user no longer needs access to the CD and can run BrookesTalk by simply selecting it from its location in the Start-menu.

Summary of the installation process

Clearly, the two tools have vastly different mechanisms for installation, the first may require no interaction from a blind user – they may simply need to load their screen reader and browser and then enter the URL of a server that provides Betsie. On the other hand, a blind user may need to install or request that Betsie be installed on a web-server. For the majority of users the former is the most likely case and hence from the point of view of a blind user, they could immediately start using Betsie to browse the Internet without the need to do anything.

BrookesTalk, on the other hand, requires interaction from a user to install the application and from the description detailed above, it is clear that this process may be difficult for a blind user. This is because they may not be aware of the location of the electronic version of the installation guide stored on the CD and hence assistance from a sighted user will probably be necessary. It should however be noted that the version of BrookesTalk used is an evaluation version and hence any formal release would probably be equipped with a better install application. It is hoped that any formal release would not require the user to run three separate programs, some of which have to be located from sub-directories on the CD.

Suggestions for improving the install method of BrookesTalk are as follows:

- Provide a better install program so that only a single application needs to be executed to complete installation;
- Include an auditory guide to the installation process – when installing the Jaws [JAWS] screen reader, audio cues helped guide the installation process. If a blind user has no screen reader then they may be unable to access the information provided in each stage of the installation;
- Ensure blind users can access installation instructions. This could for example take the form of including Braille instructions or automatically loading a file containing instructions when the CD is installed for use with a screen reader. Automatically beginning installation when the CD is inserted in the drive is also another viable option.

By studying the design of an installation package specifically aimed at blind users, for example that for Jaws [JAWS], and implementing some of the mechanisms used, blind users should be able to use the tool rapidly and with greater ease. Some features of the Jaws install package worth noting for potential inclusion are the computer-auditory guide to installation and the cassette tape that accompanies the user manual and describes the installation process.

3.1.2. Applying the tools

Whilst being able to install the tools is important in order to use them, it is also a process that only requires doing once per user. Much more important is the task they were designed for – making the web more accessible to the blind and, in the case of BrookesTalk, partially sighted. The following details some of my observations of the tools from experience using them. This information is based on personal familiarisation of the two tools. It is important to note that as they are based solely on personal experience the view of the software is objective.

Comments on Betsie

On first beginning to use Betsie, it was noted that certain URLs caused Betsie to generate an uninformative error message. By analysing the code, it was determined that the cause was a result of checking input URLs for valid characters. The URL that was requested contained the character "~" as is used by many web-servers to indicate unofficial or user web pages. The regular expression used to check passed URLs was modified to include allowing "~". The error message generated was also updated to include an indication of what the error really was.

It was noted that whilst browsing documents where the content was already known or when pages were short, it was easy to quickly grasp the context of the pages. In cases where pages were being visited for the first time, the context of the page was found to be difficult to grasp. This was worsened when long pages were loaded as well as pages that originally contained a table to provide a navigation bar as this was often placed at the top of the page due to the structure of the table used. This resulted in a list of links to other pages that had to be read before any useful content of the page was found.

When framed pages were loaded, no problems were found with the screen reader reading between page boundaries and navigation was found to be simple. On one page, a client-side image map had been used and this was found to be unusable because the links within it were not provided in a textual format.

A problematic area of Betsie was discovered when password protected documents were accessed. This only occurred when server-side authentication was implemented rather than form based authentication. In server-side authentication, the server sends a specific HTTP code to the client browser, the client browser interprets this and prompts the user to specify a username and password. When Betsie is used to access password protected documents, it receives the HTTP codes from the web-server rather than the browser. Betsie considers all unknown codes (i.e. those not stating document returned OK) as an error and therefore does not prompt the user for a username and password. This means that Betsie is unable to access the requested page and hence cannot return it to the user.

Comments on BrookesTalk

On first use of BrookesTalk, I noticed that the default voice speed was excessive and after a small amount of time configuring the software, a more comfortable speed was found. The process of doing this was found to be simple using the inbuilt menu and voice assistance was provided whilst accessing the configuration screen. This was found to be helpful with non-visual access to the menu.

Several navigational issues were found to be difficult, firstly from the perspective of a visual (and presumable partially sighted user), I discovered that typing a URL into the address box in BrookesTalk and pressing return did not cause the page to be loaded. This was unexpected because personal experience of using other web browsers has always allowed this technique. Instead, the F1 key had to be pressed and the URL entered into the dialogue box presented.

It was noted that whilst loading a page, I was kept 'up to date' with information about the state of loading by being told the amount of data loaded and the total amount left to load. This was found to be disconcerting when told "loaded 29k of 28k" and loading was assumed complete when "loaded 30k of 30k" was said. Often further data was then loaded. Informing the user of the program's state is vitally important, especially if the user cannot see text as it arrives at the computer, however I would suggest that an alternative to the current method could be implemented.

The content menus (e.g. list of links, summary, etc…) that are provided in BrookesTalk were found to be useful in some cases. The different ways of accessing them – selecting to either read the contents or use the 'active' version (for example the links list can simply be read or read in an active manner where pressing return follows the link) was found difficult at first. When pages contained large amounts of information, I found the summary to be useful at times. Often I found that a search within the page text would have been useful, especially if I thought the page to be relevant and wanted to look for a specific topic within the page.

3.2. User Evaluation

In order to obtain a wider perspective of the two tools, a user evaluation was also conducted. The idea behind this was to draw on ideas and comments of other users and to identify further problems with the tools not personally experienced. Two people were asked to participate in the evaluation, one using each of the tools. Each was given a set of tasks to complete that were designed to evaluate the tools in terms of their use in non-visual browsing and potential problems drawn from my experience. This technique should ensure any problems I encountered were not exclusive as well as highlighting further problems. The technique also allows simple testing into how well the tools perform when some of the common problems with accessibility (as discussed earlier) are presented to them.

The evaluation asked the users to perform a series of tasks aimed at finding problems and useful features of the tools as well as assessing their use as accessibility tools for blind users. After completing the tasks, a questionnaire was given to the users to obtain their comments and views. Whilst the use of sighted users does not fully represent blind users, the results and conclusions drawn should still be valid as the aim of the evaluation is to identify problems. It could be assumed that blind users would also experience any web-accessibility problems experienced by sighted users. The use of sighted users may also bring out further ideas for

development based on existing experience of browsing – the idea of the tools is to provide blind users with improved web accessible. Sighted users will have existing and potentially different experiences of the web and hence ways in which sighted users perceive and conceptualise pages may be useful to include in a tool for blind people.

3.2.1. Techniques to test the tools

During early research, I was informed of a project several years ago that was called “Battle of the Browsers” which aimed to evaluate the effectiveness of both specialist and common visual browsers when used for non-visual web access. It was considered that similar techniques may be useful in testing Betsie and BrookesTalk and hence research into this began. Unfortunately, no information could be found despite extensive web research and postings to various mailing lists read by blind users. The mailing lists used were “blind-L”, a mainly American based list, and the British mailing list “BCAB”, that is aimed at blind computing professionals. Neither of these lists generated a suitable response and people interpreted “Battle of the Browsers” to refer to the competition between Netscape and Microsoft to develop a better, more feature rich browser which occurred during the late 1990’s. It was therefore decided to base the evaluation on some common ways that user interfaces are tested – by providing a set of tasks followed by a user debriefing which took the form of a questionnaire.

Evaluation Tasks

Initially when creating the list of tasks, it was thought that a sample set of web pages could be created, this was however rejected in favour of evaluating the tools against real web pages. This was decided because the tools are designed for use on real web pages that may not contain accurate HTML code and it was thought that writing web pages containing realistic, common errors would be a difficult task. It was therefore decided that the evaluation tasks should be centred on real web pages. The choice of page was important in order to ensure that the volunteers did not already know the information being sought in the tasks. It was decided after visiting various web sites that the Birmingham University web site be used. The volunteers used to undertake the task would be unlikely to have an in depth knowledge about the University and it’s website content but would have sufficient experience of a University based web site from the use of YorkWeb [YorkWeb]. It was assumed the volunteers would be from the University of York and have some knowledge of the Internet.

The design of the tasks and questionnaire took place in parallel in order to ensure the data retrieved from the questionnaire was relevant and that the tasks would obtain data and test certain features of each tool. A summary of the aspects of the tools to be tested follows. Each item noted has a corresponding task that the user was asked to complete (see Figure 3 for the full list of tasks).

Navigation

The user should be able to adequately move between pages using links. This is evaluated by several of the tasks where the user has to move to different sections of the web site.

Retrieval of textual information

The majority of web pages present information in a textual form, it is vital that users are able to retrieve data from blocks of text without becoming disorientated and frustrated when large

amount of text are presented to them. Tasks 7 and 9 require the user to extract information from paragraph text. Task 7 requires data extraction from a mixed media page (the page includes both paragraphs and tables), whereas task 9 is based on a page containing dense text.

Tables

Extraction of information from tabulated data is vital because many web pages now incorporate this feature for data presentation. Tasks 6 and 8 ask the user to find data from a table.

Images

Ability to understand the content of an image where ALT text is provided. This is vital considering it is the only means a blind user can access image-based information. This is tested by task 2 in which the user is asked to describe what photos are displayed on the page.

Forms

Forms are widely used on the Internet to allow data to be sent from the user to a web-server. They are used for applications such as guest-books and search engines, users should therefore be able to access this technology. Task 10 requires the user to perform a search using a form.

Non-HTML pages

Many sites incorporate information not stored in a traditional HTML page. Users should be able to cope reasonably well in such cases. Task 11 requires the user to navigate through a directory listing. In a visual browser, this would be navigated in the same way as an HTML page containing links. This task checks the availability of similar navigation when using non-visual access.

Tasks

Thank-you for you assistance, please complete the following tasks that will help evaluate the effectiveness of two web-access tool designed for blind people.
Remember that it is not you being assessed but the software and that all results will remain anonymous.
The evaluation is based around Birmingham University's website. Some tasks may be difficult and other may be easy, if you are having problems skip to the next task or ask for assistance.

1. Find the page within the Undergraduate Prospectus on the City of Birmingham.
2. What photos are on this page?
3. Return to the main University page.
4. Find the list of departments.
5. Find the Department of Computer Science and load their web page.
6. Find the internal phone number for the departmental library.
7. What is the dialling code for external access?
8. Find the post of Mr Bertram Dandy.
9. What bus number services the Vale Halls of residence from the Department of Computer Science?
10. Find the technical report archive and find the date and authors for the report "Interaction between object and space systems revealed through neuropsychology".
11. Return to the Department of Computer Science main page and find their information for students. Find the name of a key text for the course "Human Resource Management", module code "comm273".

Figure 3 Tasks the two volunteers were asked to undertake

Not only does the collection of tasks in Figure 3 require the user to undertake specific tasks. Also required is some general web-access skill. This includes the ability to navigate successfully between pages and also to extract and note required information in the same way that a visually orientated user might.

In order to obtain feedback from the users, as well as the tasks outlined in Figure 3, the users were also asked to complete a questionnaire.

Evaluation Questionnaire

Figure 4 shows the questionnaire that was given to the users. It should be noted that the questions were structured in a negative format to encourage the user to think about their answer – not all statements would necessarily be suitable to agree or disagree with. In order to reduce bias on the questions, two levels of agreement and disagreement were included as well as a neutral answer. This should cover different user opinions as well as stopping users from making up an opinion when they really have none. Open-ended questions are also included at the end in order to allow the user to specify and provide any comments and thoughts not covered by other questions.

Questions about the browser

Thank-you for your time. It would be appreciated if you could spend a few minutes completing this questionnaire. Please select the answer you feel is most appropriate.

1. I found it difficult to find the department of Computer Science's web page
   strongly agree agree no preference disagree strongly disagree
2. I found it difficult to find the phone number of the library
   strongly agree agree no preference disagree strongly disagree
3. I found it difficult to find the post of Mr Dandy
   strongly agree agree no preference disagree strongly disagree
4. I found searching for the report difficult
   strongly agree agree no preference disagree strongly disagree
5. I found the process of reading a page long and tedious
   strongly agree agree no preference disagree strongly disagree
6. I found it difficult to get the general idea what a page was showing
   strongly agree agree no preference disagree strongly disagree
7. I found it difficult to understand what a picture was trying to show
   strongly agree agree no preference disagree strongly disagree
8. I found links difficult to find
   strongly agree agree no preference disagree strongly disagree
9. I found navigation between pages difficult
   strongly agree agree no preference disagree strongly disagree
10. I would be happy to use this browsing method again
    strongly agree agree no preference disagree strongly disagree
11. I would need more practice using this method before being comfortable using it
    strongly agree agree no preference disagree strongly disagree
12. What did you like about this browsing method?
13. What did you dislike about this browsing method?
14. What did you find hardest?
15. How could using this browsing method be made easier?
16. Any other comments?

Figure 4 Questionnaire for initial evaluation of the tools

Question 1 was aimed at evaluating the ease at which a long list of links could be navigated. The page with a link to the Department of Computer Science contained such a list in that a link to each departmental web page was given. The second and third questions were aimed at evaluating the ease at which data could be retrieved from a table. The tables in question were of different sizes but located on the same page. The fourth question tested the ability of the user to use a fill out form and the ability to retrieve data from responses. Questions six, eight and nine were designed to get an idea of how well the user conceptualised and recognised the content of each page whereas question seven tested the ability to extract information about images from a page. Finally, questions ten to sixteen were included to obtain comments from the users and to find out how confident they felt using the tools provided. The results obtained from the questionnaire along with conclusions drawn from it are given below.

Performing the Evaluation

An explanation of the reasoning for undertaking the evaluation was given to both users along with a description of the structure of the evaluation. At this point, it was stressed to the users that the tasks were not necessarily all possible to be completed and that it was the tools, and not the user that was being tested.

The users were then given time to familiarise themselves with the browser before being supplied with a sheet containing the tasks to be performed in a non-visual manner. Each user was allowed to perform familiarisation in both visual and non-visual forms and was permitted to use web pages with which they were familiar. This was to allow the user to become used to the nature of auditory browsing within a familiar context. A help sheet was provided to each user that contained the keyboard commands required to perform various operations – this was important because each user was used to visual web-access using a mouse to perform actions. The users were able to ask for help at any time if they felt it was required.

3.2.2. Results and analysis

As only two users were involved in the evaluation, a world representative view cannot be found. It is however, possible to draw ideas for development and highlight problems with the tools from the results obtained from this evaluation. Table 1 shows the results from the questionnaire and an analysis follows below.

No	Question	Betsie user	BrookesTalk user
1	I found it difficult to find the department of Computer Science's web page	No preference	No preference
2	I found it difficult to find the phone number of the library	No preference	Agree
3	I found it difficult to find the post of Mr Dandy	Agree	Disagree
4	I found searching for the report difficult	Not possible	Not possible
5	I found the process of reading a page long and tedious	Agree	No preference
6	I found it difficult to get the general idea what a page was showing	Agree	Disagree
7	I found it difficult to understand what a picture was trying to show	No preference	Strongly agree
8	I found links difficult to find	Agree	Strongly disagree
9	I found navigation between pages difficult	Disagree	Agree
10	I would be happy to use this browsing method again	Disagree	No preference
11	I would need more practice using this method before being comfortable using it	Strongly agree	Agree
12	What did you like about this browsing method?	Ability to search	Didn't have to read
13	What did you dislike about this browsing method?	Difficult to know location in page, not know if loading	Loading, can't tell if something is on a page or a link
14	What did you find hardest?	Finding the bus number	-
15	How could using this browsing method be made easier?	Didn't read all links automatically	Provide a search in tool and previous page key
16	Any other comments?	-	Prefer English accent!

Table 1 Table showing results of the evaluation questionnaire

There seems little point in including the answers generated from the actual task evaluation because the tasks were purely to test certain features and sufficient information can be extracted from the results of the questionnaire and from observations of the users whilst performing the tasks.

The results to questions 1 and 8 indicate that general navigation of web pages using the two tools was not difficult, and whilst the Betsie user indicated that some links were difficult to find, when coupled with one of the comments made, this problem could be attributed to the screen reader not enumerating all links stored in a page. It should be noted how the links list in BrookesTalk clearly helped the user with navigation.

The results to questions 2 and 3 give conflicting views. Both questions concerned the extraction of data from tables. The Betsie user appeared to have no real problems in finding the telephone number of the library, this was contained in a small table. The user did indicate some concern over extracting data from the larger table containing the post of Mr Dandy. It is interesting that the user of BrookesTalk found problems with extracting information from the smaller table and found the extraction of data from the larger table, where more information was present, easier. Not only do the opposite views present an interesting confliction, when coupled with the knowledge that the auditory version of both tables was presented in the same way, further confliction is apparent. It is therefore probable that the confliction of views received can be attributed to the different users rather than any specific fault with either tool. If this is assumed, the fact that the BrookesTalk user may have become used to the nature of an auditory table when accessing the second table could be an explanation for some of the confliction of views. The user of Betsie may not have perceived the tables in the same way as the BrookesTalk user and the fact that both tools present tabular information in the same auditory way reinforces this. It is therefore accepted that both Betsie and BrookesTalk have adequate auditory representation of tabular data but that users would need to become familiar with the nature of the representation.

The results from question 7 brought out surprising results. The user of Betsie was clearly able to find and access pictorial information from within the page whereas the user of BrookesTalk was unsatisfied with the nature of pictorial representation. It should be noted that BrookesTalk did in fact, in no way present the user with any information about pictures within the page being accessed, despite the fact that ALT text was available. This indicates that developments to BrookesTalk remain and that presenting textual representations of images where available is one important development to implement.

It should be noted that task 10 was found to be impossible when using either tool. The cause of this was investigated, in the case of Betsie, it attributed to an incorrect link in the web page pointing to the script that performed the search. This resulted in Betsie modifying the link to the script to point somewhere on the server running Betsie. In the case of BrookesTalk, the problem with the task was that the user was unable to navigate and complete the provided form when non-visual access was used. It is therefore clear that a problem exists with BrookesTalk and forms. Once it was apparent that the problem with using Betsie and the script was attributed to a faulty web page, the script was tested using a different website and navigation and successful completion of a web form was found to be possible. This problem

means that the results from question 4 will be discounted, however a note of the problem with BrookesTalk should be kept.

It is clear from the results to questions 5 and 6 that the user of Betsie found conceptualising pages more difficult than the BrookesTalk user. This was apparent during the evaluation as the Betsie user became impatient and frustrated because they were unable to quickly identify the content of pages being accessed. The user noted as a response to question 13 that knowing where you are in the page was sometimes difficult. The user of BrookesTalk, on the other hand, was noted to make active use of the menu system when accessing pages and it could be assumed that their differing responses to questions 5 and 6 may be attributed to this fact – that they were able to obtain content detail of pages quickly. Despite the Betsie user complaining about understanding the content of pages, they clearly did not have problems with general navigation as they were able to find and move between pages quickly and easily and disagreed with question 9 when asked if they found navigation difficult. It is interesting that the user of BrookesTalk, whilst finding the content of pages easy to understand, complained about navigation. It was noted that the user used the links list for much navigation. When the user considered a page irrelevant, they found it difficult to return to the last visited page and even commented about this when asked what was found hardest. It was noted from the actions of the user that they tried a common "back" keyboard shortcut (CTRL + left arrow), but that this was found not to work. The user therefore resulted to traversing to previous pages by using the history menu. It was noted that the user found this difficult, and said that they attributed this to the fact that the history menu listed URLs rather than page titles. It is important to note that BrookesTalk does provide a key to move back a page, however it is not the same as used in many popular visual browsers and hence the user's difficulty in performing the action.

Both users indicate that they would need additional time using the browsers before becoming comfortable with them and the Betsie user notes that they would not be willing to use the method again. This would generally be expected when people who are used to visual web browsing are asked to perform non-visual, auditory browsing. This is because a certain degree of adjustment is required to transfer from visual to non-visual web access. It was noted from conversation with the users after the evaluation had been completed that one of their reasons for answering the questions in the way they did was because they were used to visual browsing and that the transition to auditory browsing would be difficult. It was interesting to note from the conversation with the two volunteers that they thought blind users would have less difficulty using the tools and would probably be confident using non-visual browsing more quickly if not used to being able to build visual conceptualisations of pages.

3.3. Summary

From this evaluation, it is clear that several points about the two tools' representation of web pages are worth noting. These key points are discussed in the following section.

Images were sufficiently represented when Betsie was used to access web pages. It was however noticed that when browsing using BrookesTalk, images were not represented in an auditory format. This is clearly not acceptable for blind users – partially sighted users would be able to use the visual representation BrookesTalk provides to access image information. This does not remove the problem for blind users and hence implementing an auditory

version of the ALT text provided as an attribute to the `<img>` tag is an essential improvement for a future BrookesTalk release.

It was noted that when long or unfamiliar pages were presented to the Betsie user, they often became frustrated and commented on the inability to quickly grasp the content of pages. Whilst this may be attributed to the user's transition from visual to non-visual browsing, the degree of frustration was not experienced by the BrookesTalk user, who was noted to make use of the specialist menus in BrookesTalk to access internal page information. Some form of conceptualisation based on the ideas in BrookesTalk being integrated into Betsie is one area that provides potential and scope for development.

Whilst not perfect, the representation of tabular information was found to be adequate when both of the tools were used. No suggestions for development in either tools in the way each presents tables has become apparent and hence the current representation of data is accepted as sufficient to present the data in a more accessible format.

General navigation using both tools appeared to be simple, however a number of small points are worth noting. Whilst the user of Betsie complained that not all links were identified whilst pages were being read, this is attributed to the screen reader not enumerating all links rather than a problem with Betsie and hence this comment should be discounted. It was interesting that the implementation of links in the two tools was different in that BrookesTalk links are followed by selection from a list of links in the menu whereas Betsie leaves links in context in the document. When implemented with the Jaws screen reader, links were presented to the user in an inline manner and could be followed immediately. This was an interesting concept and addition of this feature into BrookesTalk could enhance its navigational power. This would allow the user to immediately follow links from the context they were in rather than having to navigate the list of links to see if a particular section of text of interest to them contained a link. It was noted that the user of BrookesTalk found navigation to visited pages difficult, this was attributed to the different keyboard shortcut used in common visual browsers and that implemented in BrookesTalk. Clearly, if the user had read the reference card provided with BrookesTalk, this may not have been such a problem, but including the keyboard shortcut of CTRL + left arrow may result in fewer problems for users moving from one browser to another. Whilst not apparent from the evaluation, it was noted from personal, visual experience of BrookesTalk, that if the user clicked into the window displaying the visual page representation (maybe to follow a link), a focusing problem was found. The problem was that clicking into the different window removed focus from the text-speech reading window and hence pressing arrow keys to change the reading mode or to read more text was impossible. By clicking back into the speech window, this problem was resolved. At first, this was confusing because links were being followed by clicking them in the visual representation. The result of this was that pages loaded by following links could not be read because focus remained on the visual representation window.

An interesting problem with BrookesTalk was identified, this was that whilst support for forms was present, it proved impossible to navigate and complete forms when non-visual browsing was used. It was found that when browsing with the aid of the visual representation, forms could be used and results obtained provided that navigation around the form was undertaken using the mouse. Clearly, this is not possible for blind users who will not be able

to see the location of the mouse pointer in order to complete this task. Again, this may be associated with the focus problem discussed above but needs resolving to help accessibility for blind people. The user of Betsie was able to complete forms by pressing the tab key to move into the box. It is not sure whether this would always be the case, of special note is when a page designer has not included the `tabindex` attribute for each form element.

Finally, it is worth noting that the user of Betsie used an interesting feature of the web browser (not part of Betsie) to help retrieve information, this was the in-built search within page. It was noted how this was used to help determine if specific information was located within a page. Whilst BrookesTalk does have a search mechanism, this is actually an interface to various Internet-wide search engines rather than a search within page. It is therefore suggested that some form of search within page be included in BrookesTalk in order to help users find specific information within a page.

As a result of this evaluation, it was decided that some of the useful features of BrookesTalk might be included in Betsie. The following section explains the developments made to Betsie in order to attempt to improve its ability to present blind users with more accessible web pages.

4. Development of Betsie

The initial evaluation indicated that whilst Betsie was reasonably good at presenting web pages in an accessible format, it lacked the assistant features of BrookesTalk. The features in BrookesTalk that were found to be useful with orientating the user on the page were the specialist menus designed to give a conceptual model of the page – such as document abstract and summary. It was suggested by the user of BrookesTalk that these helped when trying to assess if the document being accessed was relevant or not. It was therefore decided that an implementation of some of these features into Betsie could be advantageous.

At this point, it is important to define two terms that will be used which are summary and abstract. The term "summary" will be used to refer to structural information about a page, for example, number of words and number of links. The term "abstract" is used to refer to a set of extracted sentences from the document that should concisely describe the content of the page.

4.1. Generating a document summary

Based on the content of the document summary in BrookesTalk, it was decided that Betsie would provide the following information in a summary:

- Number of words;
- Number of sentences;
- Number of paragraphs;
- Number of links;
- Document Keywords;
- Document title;

4.1.1. Providing access to the summary

The first decision to be made was where to place the document summary and abstract. BrookesTalk has an advantage over Betsie because it can present this information through program menus. Clearly, it is not possible to do this when using a gateway script because the script has no way of generating or altering menus within a browser. This kind of implementation would not only be a security risk, but information would also have to be known about the many browsers available thus removing Betsie's ability to be browser independent. Three realistic options were considered:

1. Use JavaScript or another scripting language to provide a "pop-up" window containing the information when the user performs a certain action related to the document.
2. Place the summary information somewhere in the page. Either the start or end would seem sensible locations.
3. Store the summary in a separate web page and provide a link from the original document that was loaded.

The first of these three options is not really a viable solution because of the issues discussed earlier concerning page scripts – it involves the use of scripting languages to create dynamic actions on the web page. It was discussed earlier how dynamic elements and event handlers could cause problems with some screen readers, hence this option was rejected.

The solution that was chosen was in fact a combination of suggestions two and three. This was to provide a link at the beginning of the loaded document to the summary page – thus giving the user the opportunity to load the summary if required. To aid the user in deciding whether the summary should be loaded, the number of words in the document is enumerated before the link to the summary. A full implementation of example two was not used because it was thought that many pages may not need the summary to be used and hence this information may simply serve to annoy the user if placed at the beginning of every page loaded. Similarly, it was decided not to place the page summary at the end of the document, this was considered pointless – by the time a user reached the summary, they would already have read the page.

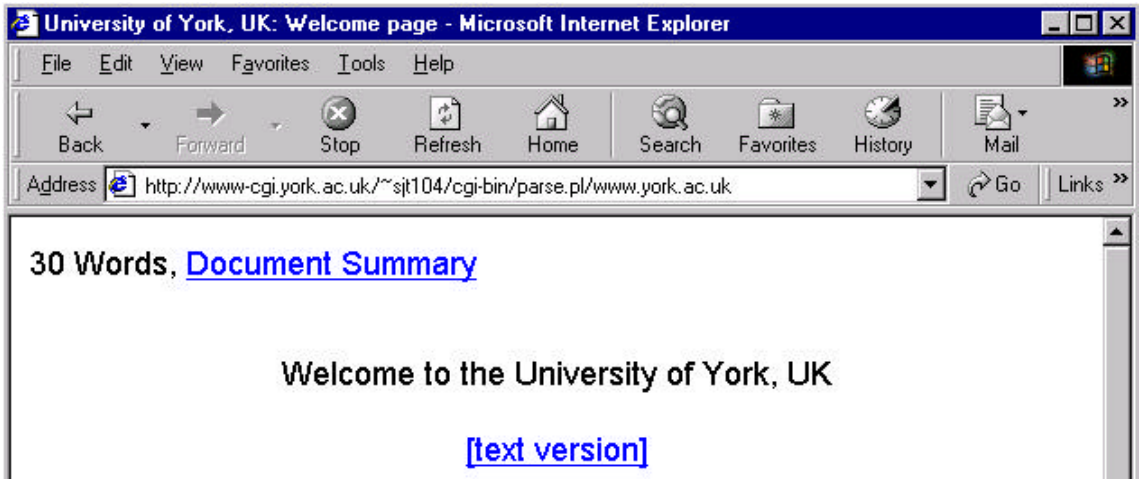

Figure 5 Screen shot showing how the modified Betsie script gave information to the user about the summary and provided a link to it

Figure 5 shows a screenshot of how the modified version of Betsie gives the user information about the content of the page in terms of the number of words and provides a link to the summary information that is stored in a separate page.

Initially, the link was to a constant, global file name. This was considered unsuitable for general use because multiple users may be using the gateway script and may accidentally load the summary for a page being viewed by a different user. The design was modified to use unique filenames for the summary page that related to each user. It was therefore decided to append the user's IP address to the name of the file stored on the web server running Betsie and hence a unique summary page would be stored for each user.

4.1.2. Generating a page summary

The language in which Betsie is written is Perl [Perl]. The Perl language has very powerful regular expression handling capabilities based on mathematical regular expression methods. Perl was originally designed as a text processing language that has been developed to include may other features. Fortunately, Perl's powerful text processing capabilities could be put to

good use when processing the HTML code that forms the underlying structure of a web page. It was therefore decided that the summary should be generated using Perl as an additional sub-routine called from the existing Betsie script. This was a simple feature to add because of the design of Betsie – a page is loaded from the server and a function called to process data with the whole page content passed as a variable. This function modifies the page data passed ready for further parsing. The first parser function effectively separates HTML tags to be one per line allowing the second parsing function to process each individual tag. In order to add a document summary, it was felt necessary that the whole document content should be available and therefore the first of the parser functions was modified to include a call to a further function that generates the page summary (and abstract). A wrapper was added around the call to the function to allow anyone installing Betsie onto a web-server the ability to disable the summary generation. This was included late on in development but was considered important in order that a single distribution of Betsie be available (thus easier to maintain) rather than to have two separate scripts depending on if the web-master did not wish the summary pages to be generated on their web-server. This option for local customisation for Betsie requires some additional variables to be configured by the user, these are shown in Figure 6.

```
$summarypath = “/usr/fsa/ug97/sjt104/web/summaryfiles/”;
$summaryurl = “http://www-users.york.ac.uk/\~sjt104/summaryfiles/”;
$generatesummary = “yes”;
```

Figure 6 Code extract showing the additional variables needed to be configures resulting from the addition of the page summary

Figure 6 shows three variables added because of adding the ability of a web-master being able to disable generation of a document summary. The first of these, `$summarypath`, gives the local path on the server running Betsie to where summary files should be saved. The second variable, `$summaryurl`, gives the base URL to where summary files are accessible from on the web. It should be noted that in many cases, this might not be the same URL as where Betsie is executed from – for example, CGI scripts are executed on a separate machine at the University of York compared to that from where web pages are served. This is designed to guard against attacks on the web-server for various reasons when using CGI scripts. The final variable that must be configured is the `$generatesummary` variable. This should be set to “yes” if the summary is required, or anything else if not. This variable is tested before the function to summarise a page is called and also before adding a link to the summary page from the Betsie processed page.

Once called, the function to generate a page summary takes the parameter passed to it – the complete content of the page – and copies it (this is necessary because of the way Perl parameter passing works). Three copies are in fact made which are used for the document summary as well as keyword weight calculation and sentence extraction. The latter two of these operations are used during document abstract generation and the process to perform each of these tasks is described later.

It was decided that the title and number of words and links would be generated as part of the summary, as well as a list of keywords. Further to this, it was decided that the title should be extracted from the page title specified by the page designer. This should be stored within the HTML tag `<title>` within the `<head>` section of a document. The title variable is

initialised to an empty string and then set to the document's real title by using the following code extract:

```
if ($pcopy =~ /<title>(.+)<\/title>/i) {
    $doc_title = $1;
}
```

The nested regular expression that forms part of the "if" statement will evaluate to true if the opening and closing `<title>` tags are found and within them one or more characters exist. If evaluated to true, the `$doc_title` variable is set to the last matching text string, in this case, the parenthesised "one or more character" (`.+`) part of the regular expression.

Once the document title has been extracted, the number of links is counted. The technique for performing this action relies on a specific feature of HTML. This feature is that links are contained within `<a>` and `</a>` tags. Using an in built text processing function of Perl called "split", the number of links was calculated. The split function takes an expression and a string and returns an array, each entry in the array contains the text that was located between each occurrence of the match expression. The match expression used to split the page text was the tag `</a>`. This was because it is assumed that page authors will close their link tags (otherwise all following text will also form part of the link) and also because the closing tags do not have additional parameters which would have to be included in the match expression. It should also be noted that the `<a>` tag might also be used to identify local anchors within a page, this is used to enable references and links within a document. Once an array has been created, another Perl feature is used to calculate the number of links. This is the ability to obtain the length (number of items) of an array by evaluating the array in a scalar context (this can be done by adding 0 or using the specific "scalar" function). If the length of the array is found to be greater than zero, then the number of links is set to the length minus one, otherwise, the link count is set to zero. It is important to take one off the value of the array length because of the technique used to count the number of links. This is because when a link is found, the text is split into two, that being the text on either side of the `</a>` tag, this has the effect of the array length being one greater that the actual number of links.

Having counted the number of links and extracted the document title, a word count needs to be undertaken. It was decided to generate the word count by using a similar technique to the link count, this time using a space as the split character. In order to assume that only words, and not tags are counted, as well as ensuring that one space only exist between words, a degree of processing of the page needs to first be undertaken.

The first step to be taken was to remove the `<head>` information. This was done using the Perl substitution using pattern matching and is the technique used for all of the text processing described below. Some example code, although not all code used, is shown in Figure 7

```
$pcopy =~ s/<head>.+<\/head>//is;
$pcopy =~ s/>/> /sg;
$pcopy =~ s/\&amp;//sg;
$pcopy =~ s/\&(\w|\d|\#)+;
$pcopy =~ s/\s{2,}/ /gs;
$pcopy =~ s/^ //gs;
$pcopy =~ s/ $//gs;
```

Figure 7 Example Perl substitution using pattern-matching statements

The first example shown in Figure 7 is used to remove page-heading information. This example matches when the <head> tag pair is found with any text within it. The matched string is substituted with the text enclosed by the second and third "/", in this case, nothing. It should also be noted that the "i" character at the end refers to non-case sensitive matching so that pages with capital letters in tag content, e.g. <TITLE> will be included in the substitution.

The second example is used to ensure that all HTML tags are followed by a space, this is required because some words may only be visually split because of images or other tags occurring in the source code. Example four would include example three that is included to remove special HTML characters such as "&" and "©" which are specified in the document with special code strings of the form "&". The final three examples are used to remove white space to ensure that only one space exists between each word and not at the beginning (example six) or end (example seven) of lines. Example five replaces multiple spaces with a single space. This processing of spaces is vitally important for an accurate word count. It was described earlier how a link would cause a split into two giving one more link than the actual count each time. A similar case to this will occur if multiple spaces exist because the split function will create empty records in the array because a split will occur for each space found even if no text exists between each occurrence of the match pattern. It is for this reason that multiple spaces are removed.

As well as the examples shown above, other text processing also occurs, this includes the removal of HTML tags – these should not be included within a word count – as well as the replacement of certain special characters, ",", ".", "?" and "!" which are replaced by spaces. This replacement ensures text of the form "only,then" is counted as separate words rather than as a single word.

It is fortunate that in order to generate a collection of keywords the previously described substitutions and removals also need to have been undertaken. If these substitutions had not occurred and HTML tags, for example, were left in the document, then it is likely that <p> would occur regularly within a document and hence be classed as a keyword. Also, the characters "," and "." have to be removed otherwise words ending in "," or "." would not be found to be the same. For example, it would be desirable that "hence," and "hence" would match as the same word. By using the result of the existing modified text, the keyword generation is possible without any repetition of work.

At this point, it is important to note that keywords that are listed are those words with highest frequency within the document. This is however a simplification and rather than being words that are found and counted, stems of words are found and counted. This has been discussed in an earlier section, but to clarify this point, a stem of a word is effectively the part of it not affected by grammatical requirements. Rather than trying to explain this concept, it is easier to demonstrate with examples. The words "take", "taking" and "takes" all have the same stem, "tak", however have different additions to the word that depends on grammatical requirements. It should be clear that the frequencies of "take", "takes" and "taking" are all different, however the frequency of the stem "tak" should be the sum of the three separate frequencies. It is for this reason, that grammatical rules require different word endings, that the stem of words is used for the frequency count rather than whole words. It should be noted

that in some cases the technique of using stemmed words might generate incorrect frequencies for a particular stem because the start of several words may be the same. It was however, thought that this would occur infrequently and therefore this method was accepted for use.

Having defined what is meant by the term “keyword”, a definition of the algorithm used will be outlined. Full code for the algorithm is located in Appendix I.

In order to ensure meaningful keywords are generated, certain words are removed. A word list has been generated containing words considered not to provide additional meaning to the document content, these are words such as “and”, “the” and “this”. The full list of words removed is given in Figure 8.

but, for, you, and, did, what, all, has, one, two, not, was, were, been, had, used, when, where, are, our, with, that, put, some, other, which, also, big, the, she, they, its, have, this, from, there

Figure 8 Words removed from a document before keywords are extracted

Figure 8 shows the words selected to be removed from a document before keywords are generated. It should be obvious that the words included in the list would not add to the meaning of a page but are simply included to link sentences. Such words would occur frequently within a document’s text and hence could be ranked as keywords if not removed. Observing the keywords generated when browsing various pages resulted in this list of words. Any word found that was considered to provide no content value for the document was added to the list. Initially linkage words such as “and” and “the” were placed in the list with further words added as experimentation proceeded. Perhaps worth a note is the implementation of this technique. A list of words to remove is stored in a variable within Betsie and a loop of code is executed. For each iteration of the loop, one word is selected from the list of words to remove and, using a case insensitive global substitution, all occurrences are removed. Whilst in many programming languages, this may take time when long documents are parsed, using Perl allows a quick solution because of its original nature and power for text processing. As speed of the algorithm was not a problem, it was accepted as suitable for use.

It should be noted that no words less than three letters long are contained within the list of words to remove, this is because any word found with length less than three characters was removed before keyword generation. Such words were considered to not add to the meaning of a page and were hence removed. Words that fall into this category include “a”, “me”, and “I”. Clearly, such words cannot contribute to the overall meaning of a document and this is the reasoning behind their removal.

It is also worth noting that any hyphenated words were split to form two separate words, this was included to ensure that the two words forming a hyphenated word were counted individually. Performing this task would ensure that words such as “web-page” would become “web page”. This is required because of the implementation used to separate and count words.

Once removal of certain words and characters is complete, a frequency of each stemmed word can be counted. In the implementation of this, the keyword generator splits the text into individual words by using the Perl split function as described earlier using a space for the

match pattern. In order for this to function correctly, all extraneous white space is removed and replaced with a single space. An array of words would now exist over which a loop can be performed.

It was described earlier how stems of words would be used rather than keywords, it is therefore necessary to remove all endings of words and retain just the stem. Figure 9 shows the list of word endings removed from words to leave the word stem.

ings, ing, ies, es, ed, ‘s, s, e, ‘t, y

Figure 9 Endings of words removed to leave stems of words

In terms of implementation, the order in which the word endings listed in Figure 9 are coded is important. This is because only one ending should be removed from each word, if this restriction is not imposed, further breakdown of the word can occur resulting in useless stems being generated. For example, “ies” is placed before “es” in the list to ensure that the “i” is also removed as would be required in the example of “families”. If only “es” were removed, the result would not match with the stem of “family”. The list given does only contain a small number of endings required by grammatical differences (e.g. family and families have the common stem “famil” but are spelt differently because of plurality), it is however complete enough to count the stems with reasonable accuracy. A regular expression is used to pattern the endings and to remove them only if found at the end of a word. This ensures that removal of characters does not occur mid-word.

In order to implement the frequency count, the “lc” Perl function was used, this converts the string passed to it to be all lower-case. It was essential to do this because of the way the frequency table was implemented. Perl contains various internal data structures, many of which have already been seen in use – numerical values, strings, and arrays. Included in this collection of structures is the hash table. A hash takes a key (in this case the stemmed word is used) and pairs a value to it (the frequency) using an in-built function to relate the data. An example of this is given in Table 2. Further information on the Perl hash can be found in any Perl text [Perl]. The key for the hash table has to be the same for each lookup hence the need to convert the stem to lowercase. The existing frequency for the word is looked up and incremented. If the current word is the first occurrence of the word then a value of one is added to the hash table that corresponds to the stem.

Stem	Frequency
famil	10
book	2
reduc	39
run	6

Table 2 An example hash table giving key and value pairs

Once the frequency count is completed, all that remains to be done as far as the keyword algorithm is concerned is to extract the highest-ranking words. In terms of implementation, this is slightly more complex. Fortunately, Perl allows data within hash tables to be extracted in list format. This takes the form of key then value repeated until the whole hash has been enumerated. The implementation takes advantage of this feature and progressively works through the list taking a key and its frequency and storing the key as a value into a second hash table. This second hash table uses frequency of stem as the key. This has the effect of

grouping all stemmed words with the same frequency together. An example of this is shown in Table 3, this shows how the stems “run”, “find”, and “peopl” have the same frequency of 18. A count of the highest frequency so far is also kept in order to allow the highest-ranking words to be extracted quickly. This is implemented by using a loop which decrements a counter and extracts words with the specified frequency.

Frequency	Stems
20	book
18	run find peopl
7	jump

Table 3 An example rank table using frequencies to match to stemmed words with that frequency

It was noted that a count of paragraphs and sentences would also be included in the document summary, this was actually implemented whilst processing text to generate the document abstract but will be described here. The process for counting the number of sentences was similar to that for counting words. Again, the split function was used and the length of the resulting array evaluated in a scalar context to give the number of sentences. In the case of words, the split pattern was “ “, a space, when splitting text to find sentences, the following characters were used – “.”, “!” and “?”. These three characters all have a grammatical meaning related to the end of a sentence.

The process for counting paragraphs was more complicated in that it was aimed at tolerating poorly designed pages. In theory it would be suitable to use the split function using only the HTML paragraph tag, `<p>`, as the match pattern. This was the approach used, however a certain amount of modification of the page HTML was undertaken first to ensure all paragraphs began with a `<p>` tag. In some web pages, the designer of the page does not implement paragraphs using `<p>`. Instead, they use two line-break tags, `<br>,` to indicate a new paragraph. It was therefore decided that a substitution of two consecutive line breaks with a paragraph tag would be made in order that properly distinguished paragraphs would result. As well as this, headings and lists (enclosed in the `<li>` tag) would also be classed as paragraphs and hence their HTML tags were substituted with a `<p>` tag before the split function was called. Once split, the matter of counting the paragraphs was trivial and whilst the result was not necessarily accurate, it was considered accurate enough to provide an estimate of the number of paragraphs in a page.

Once the document summary and abstract (described below) have been generated, the results are saved into a file by redirecting the Perl output using file handlers ready for use by the user.

4.2. Towards an automated document abstract

The automated document abstract is aimed at providing the user of Betsie with an overview of the content of the page being accessed. It should be noted that whilst keywords can have their use, they can often have little meaning when taken out of context. It was therefore decided that the keywords generated for the document summary should be available within some context – this forms the basis of the automated document abstract.

The basic outline for the abstract generation algorithm will first be given, before a discussion on the implementation of it is presented.

4.2.1. Abstract generation algorithm

In essence, the abstract generation algorithm uses keywords to allocate each sentence an importance score and then extracts sentences that have attained a sufficient score. The score that is allocated to a sentence depends on the number of keywords in that sentence. In order for this technique to work, each extracted keyword (or stemmed word) is allocated a value depending on frequency and whether or not it has appeared in a heading (within a <h?> tag) or occurs in the document title. It is therefore necessary to assign a score, or weight, to each word in the title and any word in a heading. In the same way as when generating keywords, words that do not contribute to the meaning of a page should be excluded (see Figure 8 for a list) from being awarded a weight. Highly ranking keywords which do not occur in headings or title are also attributed a value depending on their position in the keyword rank order.

A score for each sentence is then calculated by summing the values of each scoring word within the sentence and if over a predefined threshold, is added to the abstract. It should be noted that extracted sentences remain in the same order that they were in originally, and are not presented with highest-ranking sentences first.

4.2.2. Implementation of the abstract generation algorithm

In order to store the values of each word, two hash tables were created. The first of these stored the value to give any word that exists in a header or keyword rank. This value was dependant on the heading level or location in the rank order. The second hash table stored words as the key and score as the value. Words that were located in the title were all allocated the same score and stored in the word/score hash table. This was implemented by using the document title variable as defined earlier and removing non-content words before splitting the text on spaces and storing the scores into the hash table. In order to proceed with allocating weights, the HTML tags are required to be available in the document and hence the need for the local copy of the page made when the summary function was first called as described earlier.

After allocating title word weights, the weights of heading words were then calculated. The weighting of words in titles depended on the “level” of heading used, i.e. level one heading (<h1>) words received a higher score than level six heading (<h6>) words. In order to ensure each heading was scanned correctly, each was placed on a new line (implemented by substituting the closing tag e.g. replacing </h2>, with the closing tag and a new line character). Once it was sure that headings were on their own lines, the text was split into an array, each entry in the array being a separate line. A loop of code was then performed on each line of text (each array entry). Code was only executed if the current line contained a heading. The level of the heading was obtained and then HTML code removed, this was necessary to ensure tags did not appear as keywords by accident. The list of non-content words was then removed along with all one and two letter words before the text was split into individual words. Once complete, endings of words were removed in the same way as before to generate stemmed words. A lookup was then performed to find the current weight of the stem, if none existed or if the weight from the current heading was found to be greater than the stored entry, the corresponding weight for the heading level was stored. In this way, stemmed words would always be allocated the highest score possible even if they appeared in different levels of heading.

Having completed allocating scores for heading and title words, it was necessary to allocate keywords a score. The scoring system for this was similar to that for heading levels – keywords higher in the ranking (i.e. those occurring most frequently) were given higher weights than those with a lower rank. A loop was executed over the rank list of keywords allocating weights to each stem. It should be noted that if a stem was found that already had a weight from being in the title or in a heading, the weight was set to be the existing weight plus the weight corresponding to the rank position. In this way, both the frequency of the word, and any occurrence in a heading or the title could be taken into account. This shows how it is assumed that more frequently occurring words are more important to the document content than less frequent ones.

Once all relevant words are allocated a weight, it is then necessary to scan the document for high scoring sentences and to return those to the user as part of the abstract. In order to do this, the third copy of the text was used because of the need to split into paragraphs and then sentences (this was also used to generate the sentence and paragraph counts as described above). Splitting into paragraphs enabled sentences to be placed in the abstract and to remain in paragraph context – i.e. to group sentences extracted from the same paragraph within the same paragraph in the abstract. Once sentences had been separated, each was checked for stems of words and any found were underlined (this was implemented for testing of the system but was left in because it provides a user information as to why the sentence was included in the abstract). The score for each sentence was calculated as the sum of the weights of each word in the sentence. Any sentence whose score was over a set value was then included in the abstract with all others being discarded.

4.3. Summary

This section has discussed the developments made to Betsie. This was possible because of the open source nature of the language and was able to be efficient because of the powerful text processing nature of Perl. The developments to Betsie were aimed at trying to increase the ability of the user to orientate themselves and to provide a general view about the content of a page. In order to assess the use of this, a further evaluation was undertaken. Not only was this aimed at assessing the use of the new features of Betsie, but was also to ensure functionality was not lost as a result of changes. This second evaluation is discussed in the following chapter.

5. Further Evaluation

After modifying the code for Betsie, it was thought that a further evaluation was necessary, this was to evaluate the use of Betsie after modifications had been made and to ensure that the changes made had not adversely affected the performance of Betsie.

As with the first evaluation, the second evaluation used sighted users working in a non-visual context. This was because sighted volunteers were simple to find and it was considered that the difference between sighted and blind users' perceptions is not significant to affect the results. For this second evaluation only a small number of users were involved. This was because of time constraints and hence any conclusions and opinions should be viewed as objective and may not be truly representative of the blind community as a whole.

In this evaluation, instead of each user accessing the Web using different tools, each was asked to perform a set of tasks using all the available tools. The tools utilised within this evaluation were BrookesTalk, the original Betsie script, and the modified Betsie script. The purpose of this evaluation was to test the use of the summary and abstract information added to Betsie and hence, a full test of general web accessibility problems was not included. It was decided that if the user could still access and extract information easily then the modifications to Betsie would be deemed not to have reduced functionality. It is safe to assume this because only the way a user might navigate using the tool would be affected. This was known because no changes were made to any other features of Betsie.

5.1. Evaluation Design

It was decided that all volunteers would use each tool and that the order of use would be the same. The task assigned to each tool would also be kept constant. The idea of randomising the tool and task order was considered but rejected. This was because the aim of the evaluation was to assess the use of the abstract and summary and not to assess the general use of the tools on the Internet. Thus, randomising information would have little effect and may serve to complicate analysis of the data. By using the tools in the same order, each user will have the same experience of using them and will all have been exposed to the electronic voice and other factors to a similar degree. It was decided that BrookesTalk would be used first, then Betsie, and finally, the modified Betsie script. The idea behind this order was to introduce users to the ideas behind conceptualisation models provided in BrookesTalk and then to remove them from the user. The final task would then be used in a context where conceptual tools were again available. It was hoped that the users would then be prompted to use the conceptual tools when available in Betsie if found useful from browsing using BrookesTalk.

As with the first evaluation, each user was given a set of tasks to complete, followed by a questionnaire. In this case, because it was only the functionality of the conceptualisation tools being evaluated, the tasks and questionnaire did not focus on aspects of web pages such as forms, images, or frames. Instead, the objective of the task was to find an item of information from within a web site. Each site contained many web pages, any of which might have been the correct one or on a different subject. The requests made to use user required them to access several pages and try to assay if the page was relevant. In this way, the users should be able to test the conceptualisation potential of each tool. It was checked that pages surrounding

the desired page contained sufficient information so that it was not immediately apparent that any particular page was definitely relevant by simply reading the first line of a page. This approach means that the user is required to search and extract the information in the same way as one might in the real world when browsing for information.

5.1.1. Selecting target websites

In selecting websites to be used as part of the evaluation, several factors were taken into account. The first of these was that each site should require the user to traverse approximately the same number of pages. This was to ensure that any perceptions from the user of one task being more difficult than another was caused by the tools' operation and not because the user was actually required to digest the content of many more pages in any specific task.

Secondly, the content of each website was selected carefully in order to avoid the users having pre-existing knowledge about each task. This was to ensure the users actually had to find information rather than already knowing about it. To be sure that the users were knowledgeable about the topic they were researching, some background information was given. This was to present a more realistic view of the browsing – users will usually know something about what they are researching. For the sites used in the evaluation, this may not have been the case and hence the need for some background information.

It was also verified that the information each of the tasks required was considered sufficiently important within its page that it should be included in any conceptual information provided to the user.

5.1.2. Websites used for testing

The first website that was used with BrookesTalk was the Guild of students at Birmingham University (www.guild.bham.ac.uk). The user was asked to find the mountaineering society web page and to then find the page that talked about a "rack" which is a piece of mountaineering equipment. The users were required to navigate through a list of links to societies and to then browse the mountaineering society web page in order to find the specified information. It was hoped that the conceptual model information would help provide the user with an indication as to whether or not each page was relevant. For example, the list containing the link to the mountaineering society also contained links to many other (over 200) societies, and hence conceptual model information may help the user to determine if the page contained many links or not. The actual page containing information about equipment for mountaineering contained many references to the "rack" and hence it was hoped that it would occur in the document keywords or abstract.

The website that was used with Betsie was the RNIB website (www.rnib.org.uk). The users were directed to a section of the site about accessible information. The task was to find out about contrast and its use. This was located on a page concerning clear print guidelines. In order for the user to find this information, they were again required to traverse a collection of similar pages in order to find the information. Whilst not available with the version of Betsie used, it was checked that the subject matter chosen was sufficiently important to the document that it could be considered a key item of the page and should, therefore, appear in a keywords list or in an abstract.

The final website chosen was located on a local computer within the Department of Computer Science. It was deemed that this would not cause a problem with users' having pre-existing knowledge of the site in that the volunteers were all from different University departments. This task required the users to find the instruction manual for a specific piece of software and to find a list of modules within that software. Once found, the users were required to locate information about a module concerning mapping parts of the file-system. This was considered a realistic enough example in that a real user may know of a feature, however be unsure about the specific syntax for it and hence may be required to browse the module-listing page to find the appropriate information. As with the first website chosen, the user would encounter a list of links that should be indicated in conceptual information if used, i.e. that a high number of links to paragraphs would be present. The links surrounding the required page were also noted to contain sufficient information such that non-visual access to the page may have been difficult without the conceptual information available. Finally, the target page was verified to ensure that the concept being searched for was sufficiently important to the page that it might be included in conceptual information to provide the context of the page.

5.1.3. User Response

As with the first evaluation, comments and ideas from the users were sought. A questionnaire and structured interview was used to obtain each of the users' views. The questionnaire and interview was conducted after the user had completed the task with each browser, in this way, conflicting views from using the different tools would be minimised. The questionnaire part of the evaluation is shown in Figure 10.

Browser Evaluation Questionnaire

1. I found it difficult to identify links
   strongly agree agree no preference disagree strongly disagree
2. I found it difficult to understand what information the pages contained
   strongly agree agree no preference disagree strongly disagree
3. I found the process of discovering the contents of a page long and tedious
   strongly agree agree no preference disagree strongly disagree
4. I found it difficult to quickly identify whether or not the page was relevant
   strongly agree agree no preference disagree strongly disagree
5. I would need more time before being comfortable using this method again
   strongly agree agree no preference disagree strongly disagree

Figure 10 Closed questions each user was asked to complete after use of each tool

As with the first evaluation, all questions were phrased negatively as can be seen from the questions given in Figure 10. This was to ensure each user thought about the answer to each question rather than being able to just agree with statements.

Question 1 was a check to ensure that each user was able to navigate between pages and was included to ensure functionality of the modified Betsie script had not been lost because of the changes made. It was described earlier how functionality would be considered not to have been lost if each user were able to navigate and proceed with web access as if the summary information was not available.

The second, third and fourth questions were included to evaluate the ease at which users could extract information from web pages. Specifically, they were aimed at assessing how

well each tool supported the user in building up a conceptual model of the page being accessed.

The purpose of question 5 was to provide an insight into how well the user thought they coped with each tool. It was considered that if users thought they required a lot more time when using the tools that provided conceptual models, then they may not have fully understood, and correctly used the conceptualisation tools. This may therefore have affected views on their use.

After the user had completed the questionnaire, a structured interview followed. This is where a list of items to be discussed was pre-defined although not available to the user and each item is used to direct the interview. This form of evaluation is often used when user interfaces are being tested. This is because the technique allows the interviewer freedom to obtain extended information from any comments made by a user. It is often of use if the user makes a specific comment that was unexpected or interesting because it allows the interviewer to expand on users' views.

1. What did you like most about this browsing method?
2. What did you dislike most about this browsing method?
3. How could browsing be made easier and what improvements would you suggest?
4. What did you find most difficult?
5. What do you think others would find most difficult?
6. I felt the page summary did not help understand what information the page contained
   strongly agree agree no preference disagree strongly disagree
7. I felt the page abstract did not give an accurate representation of the page
   strongly agree agree no preference disagree strongly disagree
8. Did you find the page abstract useful?
9. Other comments.

Figure 11 Structure of the interview used to obtain further user comments

The outline for the structured interview is given in Figure 11, it should be noted that questions 6, 7 and 8 were only included when inquiring about BrookesTalk and the modified version of Betsie. Questions 1 and 2 aimed to obtain an overview of the tools from each user and question 3 was designed to obtain any ideas from the user for potential development of the tools based upon their knowledge and experience of visual browsing. Whilst question 4 aims to find out what the user found difficult when using the tools, question 5 differs slightly in that users are asked for difficulties they think other people may encounter. This is a commonly used technique to extract further information from users. Often users will perceive actions they took to be more difficult for other users to perform for various reasons. This type of question also allows users who may be worried about their opinions to offer further information in the context that the problems may also be experienced by others.

Questions 6, 7, and 8 were aimed at obtaining feedback from the user about the conceptual tools provided and the results of them should present an overview of the use of the document summary and abstract generated by BrookesTalk and Betsie. Finally, an "other comments" question is included to obtain any additional comments from the user that may not be covered by previous questions. This is the section under which the interviewer may be required to obtain further clarification and detail over any comments each user might make.

5.2. Results and analysis

To fully represent the views of each user, analysis of results will be split into two parts. The first of these will cover the results from the closed questions and the second on the information extracted from the structured interview. Using a structured interview means that comparing results from users is difficult and hence this part will be presented as a collection of ideas and notes extracted from the information given by the users.

5.2.1. Questionnaire results

Question	BrookesTalk	Betsie	Changed Betsie
1. I found it difficult to identify links	Strongly disagree Disagree Disagree	Disagree Strongly disagree Disagree	Disagree Strongly disagree Disagree
2. I found it difficult to understand what information the pages contained	Strongly agree Agree Agree	No preference Agree No preference	Disagree Agree No Preference
3. I found the process of discovering the contents of a page long and tedious	Strongly agree Disagree Agree	Agree Strongly agree Agree	Disagree Strongly agree Agree
4. I found it difficult to quickly identify whether or not the page was relevant	Strongly agree No preference Strongly agree	Strongly agree Strongly agree Agree	No preference Strongly agree No preference
5. I would need more time before being comfortable using this method again	Strongly agree Strongly agree Agree	Agree Disagree Agree	No preference Agree No preference
6-. I felt the page summary did not help understand what information the page contained	Disagree No Preference No preference	N/A	Disagree Disagree Disagree
7-. I felt the page abstract did not give an accurate representation of the page	Disagree No preference Disagree	N/A	Disagree No preference No preference
8-. Did you find the page abstract useful?	Yes Didn't use it No	N/A	Yes Didn't use it No

Table 4 Answers given by each user to the selection answer questions in the second evaluation

Table 4 gives the results to the questionnaire given to each user. Question numbers marked with a "-" are extracted from the interview part of the questionnaire and are included here because of their similar nature. From such a small number of users, it is very difficult to perform any kind of statistical analysis and hence the presentation of the results will be as before, an informal overview of the results.

The results to question 1 indicate that each user did not encounter problems when accessing links and hence it can be assumed that no problems with general navigation were found. It can thus be concluded that any developments to Betsie did not adversely affect its functionality.

It should be noted that a problem with one of the pages related to the BrookesTalk evaluation caused some problems for the users. This was that some of the links, when followed, resulted in an almost identical page being loaded. The difference between the pages was that following the seemingly identical link would result in a different (and correct) page being loaded. This might be an explanation as to why the three users all indicated problems extracting information from the pages which can be seen from the results showing general agreement with statements 2, 3 and 4. When talking to the users about these responses, it was

indeed noted that this confusion caused the users to become lost in the page structure. This was because the user thought that they had followed a link to a different page, which they had, however the page appeared to be identical in content. This problem may have been made worse by the fact that sighted users, who would have noted a specific change in URL and would acknowledge the following of the link by clicking, were used in the experimentation. By examination of the results to statements 2, 3, and 4, it can be seen that a general change from agreement of the statements to disagreement occurred when using the two versions of Betsie. This indicates that in the evaluation using the modified version of Betsie, it was easier to understand and extract useful information from the pages accessed. This could be attributed to one of two factors, firstly that the tasks for the modified Betsie script were easier than for the original Betsie script, or secondly, that the conceptual tools provided in the modified version of Betsie were helpful to each user. The results from statement 6 indicate that some of the conceptual tools were helpful to the users in that they all disagreed with the statement that the summary was not helpful. This means they thought that the summary was in fact useful in helping understand the page. It is however indicated that the performance of the document abstract in helping the user orientate and conceptualise the page was not as good for Betsie when compared to BrookesTalk. It was also observed from the talking to the users about the modified Betsie script that they considered the summary information to be useful but that the abstract was often too long to use and be meaningful. This is also apparent from the results to question eight which shows that two out of the three users did not find the abstract useful when obtaining information about a page. The fact that the users did not find the abstract useful and that the users did find conceptualising and extracting information from pages when using the modified Betsie script easier (indicated by the answers to questions 2-4), strengthens the case that the summary information was accurate and useful in helping navigation of the pages.

It is interesting to note from the responses to statement 5 regarding needing more time that the users thought they would need much more time to be comfortable using BrookesTalk than for either version of Betsie. It is also interesting to note that two of the users considered themselves competent using the modified version of Betsie compared to the original version. When asked about this difference, both claimed that this was because in using both versions of Betsie, a sufficient number of pages had been accessed in order to understand well the concepts and functions of the browsing method. The second user was observed to consider they would need more time after using the modified Betsie script. When asked why this was so, they claimed that they would need further time to learn when it would be appropriate to use the conceptual information provided. The response to statement 5 concerning BrookesTalk is interesting in that all three users thought they would need time to become comfortable using it. This can partially be attributed to the change from visual to non-visual browsing experienced by the users. It was also noted that the users claimed that the different ways of accessing the conceptual tool menus was complicated and thought that simply employing the active menu would be sufficient. The active menu was where items are read out and can be activated by pressing return. Also available is a way of reading the whole content of the menu in a non-active format that was found to be of no use.

5.2.2. Interview results

In order to represent the comments made by the users, it has been decided to present an overview of ideas based upon the users comments rather than enumerating all comments and then attempting to try to perform any kind analysis. The format for the interview results will be to split into those relating to BrookesTalk and those relating to Betsie.

BrookesTalk

Various aspects of browsing were brought up from talking to the users about BrookesTalk. The first point worth noting was that the implementation of a search within page would be a useful feature. The use of search was also noted during the first evaluation. Implementing a search feature would allow users to look for specific keywords associated to the information being researched. It was noted that this was frequently used during the Betsie evaluations as users searched for words relevant to the task as hand.

It was noted that when long lists of links were presented to the user, they found it difficult to search for a relevant link. This was shown when the users were required to find the Mountaineering society web page from a list of over 200 societies. Again, this is reasoning for implementing a search mechanism into the tool.

All of the users commented that it was often difficult to remember how much of the page had been accessed. The users suggested knowing the amount of page left to read may have helped indicate whether the page could contain relevant information further down in the page and hence if it was worth continuing to read the page. It was suggested that some way of checking how much of the page had been accessed was made available. The exact technique for doing this would have to be researched, it may be suitable to present a percentage of page viewed as a measure or to develop some other technique.

It was interesting to note that whilst users perceived the abstract information to be of use, they also commented on a mistrust of technology. Two of the three users made this comment and when asked to explain, said that they did not believe that a computer could generate an accurate representation of the content of a page. When asked if this view had any bearing on whether the user thought more time was necessary before being comfortable, an interesting response was received. This was that whilst a mistrust of technology did exist, users thought more time was necessary to learn when the use of summary and abstract information was useful and to learn the features of BrookesTalk in detail.

The topic of having to learn the features of BrookesTalk was discussed with the users. Comments were received about needing to learn when to use each of the conceptual tools, but most importantly, users commented on the different ways to access conceptual information. The users thought that having an active and non-active version of the menu was pointless and this was one aspect of the program that took time for the volunteers to learn to use correctly. It was also noted that the different document reading modes were strange and that paragraph and sentence were the only modes necessary. This was because the users found it difficult to understand the information being presented to them when not in small chunks and reflects a major problem of auditory browsing. This aspect is that the cognitive load of auditory browsing is very high, especially when the user is normally able to use visual senses to obtain data and information.

It was noted that several of the users complained about the loading mechanism BrookesTalk implemented. This was also an observation made by myself during personal evaluation of the page. The complaint was that the user did not actually care how much of the document was loaded – only that a link had been followed and another document was being loaded. It was noted that claiming "21k of 20k" was thought to be pointless and confusing. When asked how an alternative indication of loading could be implemented, general comments were received that being told "page loading" when a link is followed and "loading complete, ready" were useful but that anything else was pointless. It was suggested to the users that many people might become worried that the browser had stopped working if no information was given during loading. The users then agreed that a message such as "loading" would be appropriate occasionally during the loading process because this was of more use, and less confusing than being told the amount of data loaded.

One user commented that sometimes they found the voice difficult to understand. The user commented on how they considered this to make using the tool more difficult because they were already using a strange technique to access the web, that of non-visual access.

Finally, it was interesting to note that two of the users, whilst commenting that they found browsing using non-visual methods difficult, thought that both BrookesTalk and Betsie would be useful to blind people who may otherwise be unable to access information from the Web.

Betsie

One of the first comments made by all users was that the method of browsing using Betsie was simpler than that of BrookesTalk. This was because the users did not have to remember a collection of F-key combinations in order to extract information from the pages. One user said that the whole system of Betsie was simpler to use, but lacked some of the refinements of BrookesTalk. It was noted that it was much easier to read one sentence at a time (provided by the screen-reader and not Betsie) than when using BrookesTalk in one of its many reading modes. This was because of the use of auditory output to provide the user with a page representation that is found difficult to understand by sighted users.

It was mentioned by the users, after using the original version of Betsie, that the availability of the page summary and abstract in BrookesTalk was useful and a similar implementation into Betsie could be helpful. After using the modified version of Betsie, it was noted that the summary information was of use, however, the abstract generated was too long to be useful. One user suggested separating the abstract and summary into separately accessible pieces of information, whilst another suggested that the abstract should come before the summary information because, in general, it was of more use. Clearly, it would only be suitable to order information in this way if the abstract generated was sufficiently short and concise. It was clear from the users' comments that the abstract implemented into Betsie was not sufficiently short and that a different, or modified algorithm should be used to generate abstract information.

One user mentioned that they found the list of links provided by BrookesTalk to have been useful and commented that finding someway to integrate this into Betsie might be advantageous.

A comment from one user was received relating to the original Betsie script, this was that it was not as good for general browsing purposes as BrookesTalk. When asked to explain this, the user said that it was annoying to have to load a separate page containing the document summary and abstract, and that because of the implementation used, the user had to remember to refresh the page each time. An associated comment was made by another user who complained about having to refresh the summary page once loaded before it could be used. This is because of the technique used to store the summary page in that it will be loaded into a local cache by the browser and hence not reloaded from the server when the summary is requested for a different page.

Whilst not a feature of Betsie, rather one of the web browser used, all users noted that the availability of a search within page was useful, and again commented on its integration into BrookesTalk.

It was observed from all users that a degree of patience is required when using non-visual browsing. This is probably caused by users being accustomed to visual access to information. Users noted that it was easier to extract information from pages using the modified version of Betsie compared to the original one, but that extraction was simpler using the more powerful tools available in BrookesTalk

Finally, two of the users commented about problems of internation with the screen-reader. Whilst not directly a problem with Betsie, it is worth noting that users were sometimes disorientated by parenthesis being read aloud and found being informed of commas and full-stops (or “periods”) patronising and thought that this should be represented using internation.

5.3. Summary

This chapter outlines the evaluation that attempted to test the use of the developments made to Betsie. Two points were chosen over which information should be obtained, the first of these was to ensure functionality of Betsie had not been lost as a result of development. It was decided that this would be evaluated by ensuring that users could still navigate successfully between pages. The developments of Betsie were found not to have adversely affected its use, this can be concluded because each user was still able to navigate between pages and extract information contained within them when using the modified Betsie script.

As well as ensuring Betsie was still functional, it was thought necessary to evaluate the use of the additional features added. It is clear that whilst thought useful to BrookesTalk, the abstract that was generated by Betsie was considered excessively long and that further work to improve its accuracy is necessary. Users of the tools said that they thought the summary information provided could potentially be useful, but was not for any pages involved in this evaluation. Comments were also made by the users over their lack of trust for technology to generate meaningful data from documents but said they could probably learn to do so over time.

The next chapter brings together the results of the project to form a conclusion and outlines some of the areas of both tools recommended for development in the future.

6. Conclusion and recommended work

This chapter of the document brings together the results of the evaluations to present a comparison of the two accessibility tools studied. This will include an outline of problematic and useful features of each tool. Following the comparison, an outline of developments required to the two tools will be given. Suggestions on areas for further work as well as a project criticism are also given.

6.1. Comparison of the tools

The initial aim of this project was to compare the two web accessibility tools, Betsie and BrookesTalk. In order to perform this comparison, aspects of each tool will be compared with corresponding aspects in the other tool. It is important to note that some problematic areas are not covered here, these relate to tool specific aspects that do not adversely affect the application use. An example is the loading information messages that BrookesTalk provides to the user. Whilst being an issue that requires discussion, it is not an issue that has a corresponding aspect within Betsie and hence discussion is left to the section on development ideas. This is because of the differences in approach used by the two tools – one is a gateway script, the other is a full browser.

6.1.1. General navigation

The two tools being evaluated by this project are designed to aid web navigation by blind users. In order to do this, several core features are required to be available. This first of these is link capability. BrookesTalk presents the user with a menu containing a list of links available in the page currently being viewed. It is possible to follow a link by pressing return whilst being read out from the links menu. It is also possible to follow a link by clicking on it in the visual representation of the page being accessed. Using this method results in a window focus problem and is unavailable for use by blind people. In contrast to this, Betsie simply provides inline links within the current loaded page, the activation of which is dependant on the screen reader being used. In many cases, this will be the ability to follow a link whilst it is being read. Whilst the BrookesTalk method of link implementation means that links within a page are easily found, they can be out of context when in the menu and hence the method used by Betsie can be advantageous. On the other hand, one user involved in an evaluation task using BrookesTalk was noted to locate information and to build up a mental picture of the structure of the web site through the use of the links menu. It is difficult to say which method is better, however, it is suggested that greater understanding of the result of following a link could be obtained from the inline method of link access.

6.1.2. Images

Images are used everywhere on the Web and as such, it is important for blind users to be able to access information that may be contained within them. The current HTML standard supports a feature to allow page designers to provide information about the content of an image in a non-visual manner. This is by using the ALT text attribute of the `<img>` tag. To support this, Betsie provides the user with the text specified by the ALT attribute when it is available. This has the effect that blind users are able to access information about images if provided by the web page designer. BrookesTalk does not however take advantage of this

feature of the HTML standard. This is a definite problematic area of BrookesTalk because it does not allow blind users access to hidden image information. Instead, BrookesTalk displays the picture in the visual page representation. Whilst this is excellent for partially sighted users, it does not provide access for blind people. This problem may be made worse on certain websites that use images to provide navigation around the site because a blind user will be unaware of the presence of images.

6.1.3. Conceptualisation tools

The initial version of the Betsie script contained no methods or tools to help users build up a conceptual model of a web page. That is to say, the user was left to obtain all information about the page with no electronic assistance. BrookesTalk, on the other hand has many features built into it to help users obtain information about the page being accessed. This includes information such as headings contained in the page, number of words and links as well as document keywords and abstract. It was found through evaluation tasks that these conceptual model tools were useful in helping a user to understand the content of a page quickly and easily. It is for this reason that some of these features were implemented in Betsie. The modified version of Betsie was found to produce reasonably good results in terms of document summary, however the abstract that was generated was frequently found to be too long and hence of only limited use. In this respect, BrookesTalk performs better than Betsie in that it helps users to quickly understand the content of a page. This is important because sighted users will often perform this task by simply glancing at the visual information.

6.1.4. Search

It is interesting that BrookesTalk provides a way to perform an Internet-wide search using a common search engines, however no search within page is available. Whilst not explicitly a feature of Betsie, the majority of web browsers available provide a search within page feature as standard and because Betsie is used through a web browser, is available for use whilst on pages returned by Betsie. It was noted that some users involved in evaluations used the search within page regularly to help decide if the page being accessed contained certain information that they were looking for. It is therefore suggested that by adding a search feature to BrookesTalk in association with the existing tools will result in greater functionality along with provision of easier access to information within a page.

6.1.5. Forms

It was noted during evaluation that Betsie was capable of allowing blind users to access web forms. This is an essential aspect to be included in an accessibility tool because many search engines and web pages use forms to obtain information from users. It was found that whilst BrookesTalk did support forms for partially sighted people, there was no support for blind users. This was because it was not possible to access each element of the form without resorting to clicking in the visual representation provided. Betsie only uses one browser window and, if the browser in use supports keyboard access to forms, (it is expected that all browsers do) then blind users can access and complete forms well.

6.1.6. Tables

Tables are becoming widespread on the Web and hence access to the data contained within a table is often important. It was found that both Betsie and BrookesTalk presented a similar representation of tabular information, this was to read each cell one at a time. Whilst this is sufficient for small tables, it is likely (although this has not been tested) that when large amount of data, or when similar data exists in several columns, that the user may quickly become disorientated about their location within a table. No alternative solutions to this problem have become apparent as a result of the evaluations undertaken and hence this is left as a future research topic.

6.1.7. Installation Issues

It was noted early on in this project that various issues regarding installation exist with both Betsie and BrookesTalk. In the case of Betsie, it is possible that no installation is required – this is the case if a user knows of an installation of Betsie and has a screen reader/web browser combination. The installation of Betsie, and/or a screen reader/web browser combination may be required in other cases. If Betsie is required to be installed, then either the user wishing to use Betsie needs to be computer literate and have access to a web server, or needs to ask another user to install the Betsie script. Depending on the installation level required, the difficulty level to use Betsie varies greatly. In contrast to this, all prospective BrookesTalk users will have to perform some installation. It was noted earlier how the installation process is complex for BrookesTalk because of the nature of the installation programs. It is difficult to draw any conclusions over which tool is easier to install because of the varying nature of installation requirements for Betsie.

6.1.8. Financial Implications

At present, BrookesTalk is a free product. Other than a computer with Microsoft Windows and a sound card (a standard for almost all computers sold within the last five years), a blind user needs to purchase no specialist software to access the Web using BrookesTalk. This is because BrookesTalk is distributed with part of the Microsoft Speech Development kit and hence a blind user may be able to access the web with no financial cost. In contrast to this, Betsie is a platform independent tool – provided a web browser and screen reader are available for the operating system the user wishes to use, Betsie will be able to help provide more accessible web pages. Using Betsie to access the Web does however require that the user is aware of an installation of Betsie on a web-server that can be used with the pages required and that a screen reader and web browser are available. Whilst web browsers are available free of charge for most operating systems, screen readers can be very expensive. In this respect, Betsie is potentially very costly to use. It could be considered that any blind user wishing to use a computer for any means other than web browsing would require a screen reader to be installed and it can be accepted that any blind user with a computer will have a screen reader available hence the cost of installation is difficult to gauge and compare.

6.2. Development ideas for each tool

This section outlines areas in which each tool requires development in order to provide greater use and functionality to blind users for web access. These developmental ideas have been extracted from the results of the two evaluations undertaken.

6.2.1. Ideas for developing Betsie

It should be clear that as a result of the second evaluation performed, the document abstract implemented into Betsie was considered useful, but too long. It is therefore recommended that the abstract algorithm be modified or replaced and that further testing be performed concerning the use of such a tool into web navigation and page conceptualisation for blind users. Research into other methods of automated document abstract generation is recommended in order that different methods and techniques may be evaluated to find an optimal system.

Different ways into presenting the summary and abstract information may also be worth considering. Implementing a different method of presentation may remove the requirement for the summary to be refreshed each time a different page is loaded through Betsie. During the second evaluation, some comments were received from a user that indicated that the BrookesTalk links mechanism was an excellent concept and therefore an inclusion of a separate set of links as well as inline links may be advantageous to browsing. Again, this would require research into presentation methods and user evaluations to test the performance of any implemented ideas.

It was noted early on in the project, but only mentioned as a passing comment, that Betsie struggled when password protected pages were accessed. It is therefore recommended that research into methods used for authentication is undertaken and some form of implementation to allow password-protected pages to be accessed is included.

Overall, it can be seen that the majority of development to Betsie concerns conceptualisation tools. This is not the case for BrookesTalk, and it more recommendations for development are given for BrookesTalk, this is not a reflection of poor design to BrookesTalk and it should be understood that BrookesTalk is a much more complex tool than Betsie. It is a whole web-browser in its own right and many of the development recommendations relate to general browser development and not to conceptualisation as in the case of Betsie.

6.2.2. Ideas for developing BrookesTalk

Three major issues concerning the development of BrookesTalk have been discovered as a result of this project. The first of these is the problem of image information representation. Images are widely used in web pages and therefore, it is essential that blind users should be able to access the information stored behind them. This is of special important because of the efforts of the w3c to implement the ALT textual description into the HTML standard. It is therefore vital that development of BrookesTalk is undertaken to ensure that, where available, ALT text is presented to the user of the browser. If this is not implemented then a blind user will not be able to access the information provided.

Secondly, the focus problem of the browser windows needs to be resolved. This problem could easily disorientate a blind user if they accidentally switched focus to the visual representation window because they may be unaware of the action and hence may not understand why the application has ceased to respond to keyboard actions concerning reading the page content. It is therefore important that a solution to this problem be found.

Whilst not apparent in the non-visual browsing exercises used for the evaluations, personal experience indicated a problem existed when links within pages referred to objects that were

not web pages. In many visual-based web-browsers, if an object is linked that is not a web page, the user is given the opportunity to save the file locally. This occurs regularly in web browsing because people wish to download files off the Internet. Whether the file may be a virus checker update, or a music file, blind users should still be able to access the information and hence providing a save to disk feature is important in any web browsing tool and therefore requires implementation within BrookesTalk.

It was noted during the first evaluation that problems with the installation procedures could cause difficulty to blind users and as such, modifications to the install procedure are recommended. It is accepted that the current version of BrookesTalk is only for evaluation purposes, but it is important to note that a formal release should provide a full and accessible installation procedure. This topic is also relevant to documentation provided with BrookesTalk. At present, the main way to access documentation is thorough visual means and whilst an electronic version of the documentation is provided on the installation CD, it is important that a method for indicating this to a blind user be implemented. Again, it is expected that a formal release of BrookesTalk would include such items.

By using Betsie and a screen reader/web-browser combination, it was apparent that access to inline links was useful. This means that whilst blocks of text are being read, if a link is encountered the user is notified and has the option to follow the link by pressing the return key. Implementing such a feature into BrookesTalk may reduce the cognitive load on a user if they wish to follow a link – it may be difficult to find a link relevant to a block of text within a long list of links.

The use of forms on the Internet is growing, and it was noted that unless some form of visual access was used, it was not possible to complete web-based forms in BrookesTalk. In fact, it was only possible to complete a form by clicking into the visual representation window. Clearly, this is not an option for a blind user who would not be able to see each element of a form. It is therefore necessary to research and implement non-visual ways of accessing form. If this is not undertaken, blind users may be excluded from accessing certain information and pages.

It was noted many times throughout the evaluations that users either used (in the case of Betsie evaluations), or requested the use of a search within page mechanism. The use of this has been reinforced several times throughout the evaluations and hence an implementation is recommended. The search mechanism would allow users to jump quickly to relevant information within a document without the need to listen to irrelevant items.

Finally, it was noted from the user evaluations performed that the information concerning page loading needs modification to simplify the use of the tool. The biggest problem with the existing system was the disorientation potential when messages such as “21k of 20k” or “22k of 22k” were received. The first of these examples could clearly confuse a user of the tool and also it gives no indication concerning the amount of data that remains to be loaded. The second example often confused users because they assumed data loading was complete when in fact further data remained to be loaded. It is accepted that some mechanism to reassure the user that information was still being retrieved is required so that users do not become concerned that processing has ceased. It is however recommended that either the existing

system being modified to be more accurate or a different message be implemented, for example simply, “Loading”.

6.3. Areas for further work

As well as developing the tools in the manner described above, suggestions on additional work should be considered. These suggestions provide scope for further research on the topic of web-accessibility for blind people.

Images

Whilst the ALT attribute of images allows page designers to provide a textual description of pictures, this is not always used. It was suggested how future image types could have an embedded textual description and hence research into the development of image standards to include such a description could have vast implications on web-accessibility. This is of special importance if image-editing applications were to query the user for a description whenever a file is being saved. This would prompt users to provide the information rather than having to remember each time that an image was placed in a web page. It is clear how the creation of a new standard or development of existing standards to include such text has great potential and hence research into this area is encouraged.

Abstract generation

It was noted how the method used to generate an abstract in Betsie was often over ambitious in that it provided too many sentences. Research and implementation of a different algorithm is therefore a potential area for future development.

Tables

It was noted that both tools used a similar method to present tabular information. This method was to represent one cell of a table at a time. Whilst this provides the user with all the information available in the table, once a sizeable amount of data is stored in the table, orientation of columns and rows would become difficult. This means that the user may quickly forget the column headings as well as forget which column they are currently accessing. In cases where numerical or similar data is stored in each column, the user may be unable to distinguish the context of the current cell. Investigation into ways sighted users conceptualise and work with tabular data may present new ideas for non-visual access. Not only has this the potential to help with web access, but could also be applied to spreadsheet applications. Studies into the ways sighted users manipulate data (e.g. comparison of columns 1 and 6) may lead to ways in which blind users can also manipulate fully data provided to them.

XHTML

Recently, the w3c decided that the HTML standard should be developed to be more powerful. It was considered that the XML [XML] standard was a good direction to use. XHTML [XHTML1 2000], issued on 26th January 2000, was the result of this. Further evaluation of the existing tools with this standard provides scope for further research. Testing the accessibility of XHTML in general could also provide an interesting research project.

Style-sheets

Style-sheets [CSS] have the potential to provide web accessibility with greater ease because of the vast amount of information that users can encode into them. This includes the ability to provide information concerning non-visual access. At present, no known browsers support the voice-speed and pitch styles included within the definition. Research with developmental browsers could provide indications as to the use of these specialist features and the effects in general on web accessibility when style sheets are used.

6.4. Self-criticism

Having completed the project, it is recognised that if a similar project were to be undertaken, several changes and improvements could be made:

- Due to time constraints, only a small number of users were involved in the second evaluation procedure. In order to obtain a better, more accurate set of results, a larger number of people should be involved in the evaluation process.
- Whilst difficult to find, it is suggested that evaluation procedures are undertaken using both blind and sighted users in order to obtain a more general view. This should help to reduce any differences in opinion through the use of sighted users.
- It is also accepted that not all aspects of web-access for blind people has been covered and in any further work, it would be necessary to investigate the effects of various technologies such as ActiveX, and their implications over the tools evaluated.
- The second evaluation performed involved several volunteers all using the same web-browser and page combinations. In order to increase the accuracy of results, it is suggested that further sites are evaluated using each browser and that the web sites used for each browser are randomised. This is necessary to minimise the impact on results if any specific task is found to be easier than others are.
- Part of the implementation of this project involved the development of Betsie. The document abstract algorithm used was found to be inefficient and generated an abstract that was too long to be of much use. It is therefore suggested that further abstract generation algorithms are investigated and implemented.

Despite these criticisms, it should be noted that much information has been learnt about web-accessibility that was not previously considered. Any future web sites that I may be involved in designing will be checked for their accessibility. It has been interesting to note that I have started looking at the accessibility of web pages I have visited and it is surprising the number of pages found which do not provide support for non-visual access.

Overall, this project has been interesting and enjoyable and has resulted in greater personal knowledge of web accessibility as well as learning a new programming language – Perl. This will have many uses in the future because of its general nature in use on the Web.

7. References

Speech Project	"The Speech Project - BrookesTalk - A Web Browser for the Blind and Visually Impaired" http://www.brookes.ac.uk/schools/cms/research/speech/
Betsie 1999	"BBC Education Betsie Site" http://www.bbc.co.uk/education/betsie/inverse/index.html
BBC	"BBC Online Homepage" http://www.bbc.co.uk/
NatWest	"The NatWest bank online" http://www.natwest.com/
W3C	"W3C – The World Wide Web Consortium" http://www.w3.org/
WAI	"W3C Web Accessibility Initiative (WAI) Home Page" http://www.w3.org/WAI
HTML4.01 1999	"HTML 4.01 Specification" http://www.w3.org/TR/html4
HTML4.0 1998	"HTML 4.0 Specification" http://www.w3.org/TR/1998/REC-html40-19980424/
HTML3.2 1997	"HTML 3.2 Reference Specification" http://www.w3.org/TR/REC-html32.html
HTML2.0 1995	"HTML 2.0 Materials" http://www.w3.org/MarkUp/html-spec/
VALIDATER	"W3C HTML Validation Service" http://validator.w3.org/
BOBBY	"CAST - Bobby" http://www.cast.org/bobby
Haywood 1997	Haywood, R, "Accessibility Issues of Web Browsers for Blind People", Department of Computer Science, The University of York, 1997
Blair 1999	Taylor, R, "Embrace the internet or go bust, Blair tells British business", The Guardian, September 13th, 1999 http://www.guardianunlimited.co.uk/Archive/Article/0,4273,3901437,00.html Jones, G, "Use Net or fail, Blair tells business", The Telegraph, September 16th, 1999 http://www.telegraph.co.uk/et?ac=002468295856081&rtmo=LbxLl3bd&atmo=ttttttd&pg=/et/99/9/16/ecnblair16.html
ADA 1990	"Americans with Disabilities Act", US Department of Justice, 1990 http://www.usdoj.gov/crt/ada/adahom1.htm
DDA 1995	"Disability Discrimination Act", 1995 http://www.disability.gov.uk/dda/index.html
Mendels 1999	Mendels, P, "Lawsuit Says AOL Shuts Out the Blind", New York Times, 1999 http://www.ilusa.com/News/110599aol_suit4n.htm
JAWS	"Henter-Joyce, Inc. - Developers of JAWS and MAGic!" http://www.hj.com/
Edwards 1997	Edwards, A, "Legislation and access to the World-Wide Web" , CD Edition of proceedings at the Sixth International World Wide Web conference, Santa Clara, California USA, 1997
SNS 1999	"Web Accessibility",1999 , IBM Special Needs Systems http://www.austin.ibm.com/sns/accessweb.html
Jenkins 1997	Jenkins, P, "Experiences Implementing Web Accessibility Guidelines in IBM", 1997 http://www.austin.ibm.com/sns/phillj.htm

Edwards & Stevens 1997	Edwards, A, Stevens, R, “Visual Dominance and The World-Wide Web”, CD Edition of proceedings at the Sixth International World Wide Web conference, Santa Clara, California USA, 1997
Zajicek & Powell 1997	Zajicek, M, Powell, C, “Building a conceptual model of the World Wide Web for visually impaired users”, Proc. Ergonomics Society 1997 Annual Conference, Grantham `http://www.brookes.ac.uk/schools/cms/research/speech/publications//40ergsoc.htm`
Zajicek, Powell & Reeves 1998	Zajicek, M, Powell, C, Reeves, C, “Orientation of Blind users on the World Wide Web”, M. Hanson (ed) Contemporary Ergonomics 1998 (Taylor and Francis, London) `http://www.brookes.ac.uk/schools/cms/research/speech/publications/56_ergsc.htm`
SHOCK	“Macromedia Flash” `http://www.macromedia.com/software/flash/`
JAVA	“java.sun.com - The Source for Java™ Technology” `http://java.sun.com/`
ACTIVEX	“ActiveX Controls - Microsoft Papers, Presentations, Web Sites, and Books, for ActiveX Controls” `http://www.microsoft.com/com/tech/activex.asp?RLD=18`
RAUDIO	“real.com” `http://www.real.com/`
Perl	Wall, L, Christiansen, T, Schwartz, R, “Programming Perl”, O’Reilly & Associates, 1996
YorkWeb	“University of York, UK: Welcome page” `http://www.york.ac.uk/`
XML	“Extensible Markup Language (XML)” `http://www.w3.org/XML/`
HXTML1 2000	“XHTML 1.0: The Extensible HyperText Markup Language” `http://www.w3.org/TR/xhtml1/`

Appendix I

Code for Betsie

Note that the code added as part of this project is located as the end of the script in the function called “summary”