TBHelp
A Help Authoring Tool for ToolBus Applications

Zohra Bouguillara and Tesla K.S. Ganpat

Masters thesis
Computer Science Institute, University of Amsterdam
Department: Programming Environments
Supervisor: Prof. dr. P. Klint
Date: August 13, 2001

Abstract

In large software systems the user interface and system are often complex and difficult to navigate. It is necessary to provide users with online help to support users in effectively using applications. Such online help systems can include providing online documentation such as quick tours, online manuals, tutorials and other collections of information. Building online help systems can be a time-consuming and complex process, especially when large navigational structures are required. Help authoring tools can save substantial effort in building help systems for applications by automating the process of creating and organizing help content as much as possible. In this paper, we investigate how such help authoring support can be developed and provided in the context of a component interconnection architecture, the ToolBus. After a review on modern help authoring tools and online help systems, we developed a component-based help authoring tool, TBHelp. The software components of TBHelp are interconnected with the ToolBus software architecture and each component automates part of the help authoring process. TBHelp generates HTML-based help systems that can be viewed on any HTML browser and provides basic help features.

Acknowledgements

There are many people who have supported us during our research. Unfortunately, thanking everyone separately would take some time and a lot of paper (think trees). So we have decided to thank everyone by sending them happy thoughts. However, because the “happy thougths”-connection is sometimes down due to a hardware failure, we'll thank some people explicitly on this page.

First of all, we would like to thank our supervisor Prof. Dr. Paul Klint, who always seemed to find time in his busy schedule to help us through our research and promised to stand up for us as long as we didn't misbehave too much. We also want to thank Dr. Mark van den Brand for always lending a helping hand when needed and who put up with all our giggles and “angel-like” behaviour.

We don't want to forget our families and friends, who we tortured with our interesting mood-swings over the last few months. And if they saw us at all, they even managed to encourage us to keep going towards getting our masters degree.

Some of our friends like Stephen Ho, Gerald Stap, Fred Kreek, David Rasch and Stephan Vinkenborg showed us that even with our disability (lack of Y-chromosome) we still could succeed very well in the world of computer science. They certainly deserve some special thanks. We bonded with some “Y”-less people like Thnimphorn Panyasak, Linda van der Pal and Nuray Gokalp, who in many frustrating phone conversations and email-sessions, convinced us that emigrating instead of finishing our study, was not an option. They also deserve our special thanks.

Last, but definitely not least, we want to thank each other for ever being born.

Contents

1 Introduction **11**
- 1.1 Online Help Systems 11
- 1.2 The Need for Help Authoring Tools 11
- 1.3 The Development Process of TBHelp 12
- 1.4 Outline 13

2 Online Help Basics **15**
- 2.1 Components of Online Help 15
- 2.2 The Help System Design 17
- 2.3 Overview of help formats 18
  - 2.3.1 WinHelp 18
  - 2.3.2 Microsoft HTML Help 19
  - 2.3.3 WebHelp 21
  - 2.3.4 JavaHelp 21
- 2.4 Comparing Help Format Features 23
- 2.5 Other platform independent online help formats 24
- 2.6 Conclusions 29

3 Comparison of Help Authoring Tools **31**
- 3.1 Basics of help authoring tools 31
- 3.2 Doc-To-Help Professional 2000 32
- 3.3 ForeHelp Premier 2000 33
- 3.4 HDK/Virtual Media Suite 2000 34
- 3.5 RoboHELP 2000 35
- 3.6 Comparing Doc-To-Help, ForeHelp, HDK and RoboHELP 39
- 3.7 Conclusions 41

4 The TBHelp Generator **47**
- 4.1 Why The TBHelp Generator? 47
- 4.2 Features of the TBHelp Generator 48
- 4.3 TBHelp Generator Component Architecture 53
- 4.4 The Main Component Functionality 54
- 4.5 The Indexer Component Functionality 59
- 4.6 The Toc Component Functionality 59
- 4.7 The Search Component Functionality 61
- 4.8 The Preview Component Functionality 62
- 4.9 The StartBrowser Component Functionality 64
- 4.10 The Help Component Functionality 65

5 Implementation of the TBHelp Generator **67**
5.1 The ToolBus Architecture 67
5.2 TBHelp Generator ToolBus Architecture and Communication . . 69
5.3 The Main Component 76
5.4 The Indexer Component 78
5.5 The Table of Contents Component 79
5.6 The Search Component 79
5.7 The Preview Component 80
5.8 The StartBrowser Component 81
5.9 The Help Component 82
5.10 The TBHelp Help Format 82

6 The TBHelp Viewer **83**
6.1 TBHelp Viewer Features 83
6.2 TBHelp Viewer Implementation 85
6.2.1 Program GenViewer 85
6.2.2 The Helpscript 86
6.3 Applying the TBHelp Viewer 87
6.3.1 Generating a Viewer 87
6.3.2 Integrating Help System and TBHelp Generator 87
6.3.3 Help request handling 88

7 Discussion, Future Research and Conclusions **91**
7.1 Discussion 91
7.2 Future Research 96
7.3 Conclusions 97

A Installation Guide **103**

List of Figures

2.1 HTML Help example (Internet Explorer 5 Help system) 20
2.2 A JavaHelp system displayed in the Help Viewer window 22
2.3 A manual page in Xman 28

3.1 Creating a TOC with drag-and-drop 37
3.2 Application help 40

4.1 TBHelp Generator: An example help system. 49
4.2 TBHelp Generator: The Main GUI 49
4.3 TBHelp Generator: The *Application* menu 50
4.4 TBHelp Generator: The *Topic* menu 50
4.5 TBHelp Generator: The *Tools* menu 50
4.6 TBHelp Generator: The *Help* menu 51
4.7 TBHelp Generator: File-Dialog. 51
4.8 TBHelp Generator: Generate a table of contents. 52
4.9 Component overview of the TBHelp Generator 54
4.10 TBHelp Generator: No help system name was entered/selected. . 55
4.11 TBHelp Generator: Attempt to create an existing help system. . 55
4.12 TBHelp Generator: Attempt to open a non-existing help system. 55
4.13 TBHelp Generator: Select or enter a topic name. 56
4.14 TBHelp Generator: Add a new topic. 57
4.15 TBHelp Generator: Example contents for a topic. 58
4.16 TBHelp Generator: Creating index for a help system. 59
4.17 TBHelp Generator: Creating index for a help system is done. . . 59
4.18 Creating a table of contents. 60
4.19 TBHelp Generator: Removing old files. 61
4.20 TBHelp Generator: Generate exclude file. 62
4.21 TBHelp Generator: Generate include file. 62
4.22 TBHelp Generator: Configure archive. 62
4.23 TBHelp Generator: Index help topic files. 63
4.24 TBHelp Generator: Adding search done. 63
4.25 TBHelp Generator: Form for issuing search requests. 63
4.26 TBHelp Generator: No index was generated yet. 63
4.27 TBHelp Generator: No table of contents was generated yet. . . . 63
4.28 TBHelp Generator: No search capabilities were added yet. 64
4.29 TBHelp Generator: Select browser that should be used. 64
4.30 An example help system. 65

5.1 The ToolBus Software Architecture 68

5.2 TBHelp Generator ToolBus Architecture 70

6.1 Generation process for a help viewer. 84
6.2 TBHelp Viewer: Graphical User Interface. 84
6.3 TBHelp Viewer: Not all fields are filled. 84
6.4 TBHelp Viewer: Incorrect browser location. 85
6.5 TBHelp Viewer: Incorrect help system location. 85
6.6 TBHelp Viewer: A viewer already exists. 85
6.7 Help request handling. 88
6.8 Result of a specialized help request. 89

Chapter 1

Introduction

1.1 Online Help Systems

Putting together a help system for an application was once considered a low priority task. Developers would spend all their time building applications, but would virtually ignore the need to provide an effective online help system. A comprehensive online help system is necessary to familiarize users with modern complex user interfaces. Today, online help system development is being recognized as an essential part of the software development process. The development of an online help system should start at the beginning of software development and continue throughout the development process.

Online help is more than just putting a manual as a block of text online. It combines topics, links, a table of contents and index into a complete integrated system. These basic online help features are usually displayed in a help viewer or web browser.

In the past, help was mainly offered through long printed manuals. Presenting documentation in an online format has many advantages over printed documentation. For instance, online help is faster to use due to effective navigational structures such as an index, table of contents, search, etc. Users can get exactly the information they want or need with minimum effort. Online help is also cheaper to create than a printed manual. The single biggest cost when releasing a software product is usually the printed manual. By shifting the documentation to an online help format, software vendors can substantially reduce the cost of goods and the cost of shipping. In addition, online help is easier to maintain. Adding a new option or a new feature for a product can be documented by updating the help files and release it to production. Online help can be linked directly to a program it documents. This way, it is integrated with the application it provides help for. The integration will lead to lower support costs, as the users will have the answers they need when they want.

1.2 The Need for Help Authoring Tools

The process of creating a help system can be complicated. Especially when the documentation is large and highly structured. When constructing a help system, an author will need to implement navigation structures and display features in addition to writing the help system content. This means that a lot of manual coding is involved in, for instance linking various topics and constructing a table of contents. In practice, creating navigational structures for a help system can be a time-consuming task, but with the proper tool the process can be streamlined. A *help authoring tool* (HAT) aids help authors in the process of creating and organizing help content as much as possible. This allows the author to focus on the help system's content, design and usability instead of the mechanics of implementing specific features. A help authoring tool consists of two main parts, a generator and a viewer. The generator is used to create the help system and the viewer enables the user to view the help system. A help authoring tool consists of a set of tools to write, process and view online help. Major help authoring tools can generate help files in different help formats. A help format provides the means to display navigational structures, such as a table of contents or an index, and help topics. Each type of help format provides specific features which define the appearance and functionality of the help system. For instance a help format determines the platform(s) the help system will operate on and the structure of information presented in a help viewer. Commercial help authoring tools can generate help files for help systems that run on the major computing platforms, including Mac, Unix, and Windows.

1.3 The Development Process of TBHelp

Our research focusses on the following central question:

How can a help authoring tool be developed, that runs on top of the ToolBus software architecture?

We extracted two questions from the central question. The first one focusses on finding out whether we should reuse existing help technology and if so, what technology? The second question focusses on the basic requirements a help authoring tool that runs on the ToolBus, should meet. These two questions are listed next:

- *Is it possible to reuse existing help technology in the development of a help authoring that runs on top of the ToolBus?*
- *Which basic requirements should a help authoring tool, built on the ToolBus, meet?*

To answer the research questions, we performed a number of studies to determine the benefits and drawbacks of several off-the-shelf help authoring tools. Many commercial help authoring tools exist that run on a Windows platform. We discuss online help systems, capable of running on Windows and Unix machines, and their features. We give an overview of the online help formats used by existing online help systems. A comparison of several major help authoring tools will guide us towards a set of basic requirements for a help authoring tool that is built on the ToolBus software architecture.

In the absence of a flexible platform independent help authoring tool, which fulfills these basic requirements, we decided to implement TBHelp, which is a component-based help authoring tool for ToolBus applications. The components of TBHelp do not operate in isolation, but interact to form a complete help authoring environment. We use the ToolBus software architecture [2], a component interconnection architecture, to integrate the components. The component-based approach of TBHelp provides flexibility in dealing with change. The components are independent and can easily be replaced. TBHelp provides developers of ToolBus applications with a help authoring environment for presenting online information to ToolBus application users. Providing a help system standard relieves developers of the task of inventing their own help systems.

TBHelp is defined by two workflows: the TBHelp Generator, which provides a help authoring environment for ToolBus applications, and a viewer. The TBHelp Viewer enables the integration of a help system, built with the TBHelp Generator, with applications running on the Toolbus software architecture. TBHelp systems do not have to be linked to an application, but can also function as stand-alone online help. The TBHelp Generator provides a complete authoring environment to construct online help systems. The generator enables the automatic creation of a table of contents and an index for a help system. TBHelp also supports full-text search of a predefined set of help files. The help content is displayed through a web browser.

Currently, TBHelp only runs in a Unix environment, because of its dependency on the ToolBus. The ToolBus is implemented to run on Unix machines, but plans are made to port the ToolBus to Windows in the near future.

1.4 Outline

In Chapter 2 we discuss the basics of online help systems. In this chapter we describe the elements that constitute an online help system. In addition, we discuss several techniques for presenting online help on different platforms and compare these techniques. In Chapter 3 we introduce concepts of the help authoring process. Several major help authoring tools will be discussed in this chapter as well. Their comparative features are outlined and discussed. Chapters 4 to 6 describe TBHelp. In Chapter 4 we describe the design and features of the TBHelp Generator and list the reasons for developing it. In Chapter 5 we discuss the implementation details of the TBHelp Generator. Chapter 6 describes the TBHelp Viewer features, design and its implementation. Finally, in Chapter 7, we present results and draw some conclusions.

Chapter 2

Online Help Basics

Each help system has the same basic objective: to present information in the most productive manner. Variations in the use of platforms and browsers make each help system unique in its approach.

In Section 2.1 we discuss the basics of online help systems. Several design options that help system developers need to consider are discussed in Section 2.2. In Section 2.3 we give an overview of the four major help formats used today. We proceed in Section 2.4 with a comparison of the supported features by each help format. By comparing these help formats we are able to decide upon a help format used in the online help system for ToolBus applications. In Section 2.5 UNIX-based online help systems are discussed.

2.1 Components of Online Help

[8] describes online help or help system as

> *...an integrated system of short, succinct topics that can be displayed in the context of using a product to help users understand the context and to accomplish tasks...*

Online help also includes other forms of online documentation such as quick tours, online manuals, tutorials, and other collections of information. Online help is more than an online tutorial. It is a complete integrated system with the following elements:

- *Topics*
  Topics are the blocks of text that represent the written content in a help system. Each topic is displayed on a single screen. Help topics are the basic units of information that make up a help system. Topics communicate the information or provide the assistance users are looking for. A help author should organize the help information into a logical framework. Usually, this results in a hierarchy of topics.

- *Links*
  Links connect topics and allow the user to navigate from topic to topic. There are different types of links that can be used in online help. Hyperlinks are used to link topics within a help system or to a site on the

Internet. Pop-ups are little windows which pop-up on top of the current window. Generally, pop-ups are used to define words or display notes or tips. A browse sequence is a link that lets the user move forward and backward through a set of topics arranged in a specific order. Browse sequences let the users step through groups of topics, like paging through a book. *See Also* links or *Related Topic* links provide a way to show the relationship between related topics. See Also links can be used as another way to group topics that might be necessary or interesting to users.

- *Table of contents and index*
  The table of contents provides a hierarchical view of the topics in the online help. The index lists all keywords or phrases for finding topics in an online help system.

- *Full-text search*
  Full-text search features allow users to search for words or phrases that are not included in the table of contents or index. A full-text search enables a user to search all help files defined for a help system.

The table of contents, index and links are navigational structures which provide the means for a user to browse through the help system files.

Online help has many advantages over printed documentation. It is faster and easier to use. Rather than having to page through a printed manual, the users can search for a specific topic or keyword to get exactly the help they need. Online help enables users to choose their own path through the file with navigational aids such as keywords, browse sequences and tables of contents. More sophisticated help files can interact directly with the users through prompts and dialog boxes that can identify the types of information to present and the order in which to present it. Online help consists of short topics, which provide procedural information. In general, it provides information on how to accomplish a specific task through a set of short instructions. There are two general styles of online help, which are listed below.

- *Context-sensitive help* is linked directly to a program. It has the ability to display a specific topic from a location in the application, such as a button in a dialog box. When a user presses F1 or clicks the help button in a window or dialog box, a context-sensitive link between program and help file identifies the topic to be displayed and then displays it. Context-sensitive help usually enables users to get help on a particular window, dialog box, command, or toolbar button. Context-sensitive help promotes a high degree of integration between the application and its online help. From the users' perspective, help is part of the application.

- *Stand-alone help*, provides online access to task, reference, or tutorial information, independent of application software. Stand-alone help is not connected by context-sensitive links with an application.

There are three ways in which online help can be invoked. Help can be initiated by an application. With this invocation mechanism it is the application that determines when help is needed and what to present. When help is initiated by a user, for example by pressing the F1 function key, the system decides what

information to display, according to the user's current context. Specific information can also be requested from a Help menu, Help key and Help buttons, provided by most applications. Within most applications, the primary way for a user to request help is by pressing the help key.
Help topics often contain hyperlinks that refer to related help information. To display a help topic, a user selects a title in the topic tree or a hyperlink within the topic display area. Navigation commands enable a user to return to previous topics or to the main page of a help system.
Although help can be displayed in a number of ways, the most common way is through a help viewer window. Most help viewer windows consist of a toolbar, content pane, and navigation pane.

2.2 The Help System Design

Developing online help needs good planning. A help project without a solid plan will likely encounter many problems. These problems can add up and negatively affect the final online help systems. Project planning helps to ensure that a help project runs smoothly and that a usable online help system will be delivered. When developing an online help system certain design options need to be considered. For instance, how will help be delivered to end users? Through a web browser or pop-up window? On what platform will the help system operate? What navigational structures will be supported? A development process suggested by [31], defines five phases. These phases describe a sequence of actions users perform when they initiate a help request. It is the typical workflow for which users of an online help system request help. For each phase certain design choices need to be made by developers.

1. *Invoke the online help system*: to deliver online help to users, the online help system needs to have a certain invocation mechanism. Users need to know how to get help whenever they need it. Will online help be offered through a help menu entry or by pressing a help button?

2. *Specify the problem*: after successfully invoking help, a user needs to specify the problem he wants help for. With this characterization of the problem he can browse through the index, table of contents or initiate a search request. What kind of navigational structures should the online help system provide?

3. *Read the information*: subsequently, the user reads the information. How will the information be presented to the user? Some choices include a tutorial, examples, task instructions, shortcuts, troubleshooting, reference information or glossary of terms. What fonts will be supported? Can the help screen be resized?

4. *Navigate*: After reading the information, the user might need additional information and decide to navigate through the help system again. What kind of information will be offered to users? Which topics will be available?

5. *Apply the information*: the online help system delivered the requested information to the user, so now the user will be able to apply the information

to his problem. How can a user apply the information presented to him? Does he need to memorize the information, close the help screen and apply the information to resolve the problem? Or is the help content offered in a separate window? Can the online help system automatically apply the information to accomplish a specific task? An online help system that is able to apply the presented information automatically to accomplish a certain task, actually takes over part of the interface. An online help system that has fully integrated this feature is called a successful online help system according to [32].

2.3 Overview of help formats

Before writing online help it is important to consider the type of online help to be created. Different help formats are available to deliver online help. Having many help formats to choose from, however, makes it even more difficult for help authors to decide which help format to use in which situation. These help formats allow online help to be delivered to users on a variety of platforms such as Windows, Unix, MacOS, over Intranets and the Internet. The most common help formats are WinHelp and HTML help. The majority of work lies in the creation of help for Windows applications. WinHelp (or RTF-based help) was once most common, but HTML-based and Java-based help formats are becoming more popular. Many of the help authoring tools have a version that generates HTML-based and/or Java-based help. In RoboHELP, a major help authoring tool, it is possible to generate online help in several different formats without having to develop in more than one. Most help formats used in todays online help systems include HTML Help, WinHelp, WebHelp and JavaHelp. In the next sub sections we will discuss these help formats. To decide upon a help format used in an online help system for ToolBus applications, we will compare the features of these four help formats.

2.3.1 WinHelp

WinHelp is a system for providing online help for Windows applications and runs exclusively on Windows systems. WinHelp was first released to the public as part of the Windows 3.0 Software Development Kit. WinHelp for Windows 3.1 provided a hypertext table of contents, index, links between topics, pop-up windows and context sensitive help. With the release of Windows 95, Microsoft provided significant improvements to WinHelp. With Windows 98, Microsoft introduced HTML Help (see the next section), which is now the new Help standard for Windows 98 and Windows 2000. WinHelp, however, will continue to be supported on future Windows platforms. Today, it is used to display application help or stand-alone help on early Windows versions. WinHelp is the most common and most widely recognized format for online help.

WinHelp generates .HLP files, which can be viewed on any Windows system without installing an additional viewer, because the WinHelp viewer is part of the operating system. A Windows help file supports a table of contents, a keyword index, full-text search, user defined bookmarks and annotations. WinHelp offers a help environment capable of running sound, video, and multi-media applications and linking to other systems and programs. Help authors need to write their help content in a word processor which supports Rich Text Format (RTF), because the WinHelp compiler expects RTF files. A project file (an ASCII document) lists all the RTF files, graphics and macros which are required to make the complete online help system. WinHelp is the first compiled online help format. This means that one or more source files (files containing the content that will appear in the compiled WinHelp file) and one or more instruction files (files containing instructions about how to compile the WinHelp file) are run through the WinHelp compiler and turned into a single compiled file. The result is a help file which may be run from Windows as a stand-alone online help. Some of the WinHelp source files include, the RTF files containing the actual topics that will appear in the online help, graphics, sound, video, multimedia and the tables of contents files. Instruction files used by the WinHelp compiler are a HPJ (Help Project File) file and map files. The help project file contains the basic instructions for what to compile and how to compile it. It also contains information on the appearance of the online help system. The map files contain information about context-sensitive links for an application. With the WinHelp format one can easily distribute online help and provide an environment familiar to the reader.

2.3.2 Microsoft HTML Help

Microsoft's HTML Help [17] is the new standard for Windows application Help and produces .chm files (compiled HTML files). HTML Help is written using standard HTML (HyperText Markup Language) to format the information. Microsoft HTML Help was first released in August of 1997, incorporated into Windows 98 and the release of HTML Help 1.2 in fall 1998. The HTML Help format is Microsoft's current application help format for Windows. Microsoft made a major transition for online Help by using its new HTML Help format instead of the WinHelp format. Due to the fact that many companies continue to produce WinHelp-based help systems, Microsoft will provide WinHelp support for many years in future operating systems for backward compatibility. HTML Help is an advanced help format that provides user assistance for applications running on Windows 98 or Windows 2000, and for users of Internet Explorer 4.0 and above, Microsoft Office 2000, or other new Microsoft applications.

A HTML Help file is a single file. It is a single file storage container that can store many HTML files, including images. The separate source files are compiled and compressed in the HTML Help file by a help compiler. The HTML Help compiler is a program that compiles all the source files into a compiled help (.chm) file. The compiled help file includes all the files that are generated for a particular help system. The source files are stored in a folder, which is referred to as the project file (.hhp). The compiled help file contains a compressed project (.hhp), index (.hhk), topic (.html), contents (.hhc) and other (.gif, .jpeg, etc.) files. The compiled help file can be distributed with a software product. A keyword index in an HTML Help file is made up of a list of keywords that are associated with topics and URLs, and is available in an HTML Help file on the Index tab. The HTML Help format lets users locate information through either a table of contents or through an index. Help authors can select different icons for headings and topic entries in the table of contents. The topic entries can link to a variety of destinations, including external HTML files, e-mail addresses, newsgroups, and files local to the user's system. Help authors can construct help systems tailored for specific user groups, based on any criteria, such as training level, type of job, or language. When the user selects the information type for the help system, HTML Help displays only the applicable contents entries. HTML Help must be displayed using Microsoft Internet Explorer version 4.0 or later running on a Windows system. The HTML Help windows are usually tri-pane (a window with three frames), meaning they contain three standard panes, which is the default. The top pane is for the ToolBar, the bottom-left pane is for the table of contents and the bottom-right displays the help content. An example of a HTML Help window is shown in Figure 2.1. This is in fact an example of a help viewer.

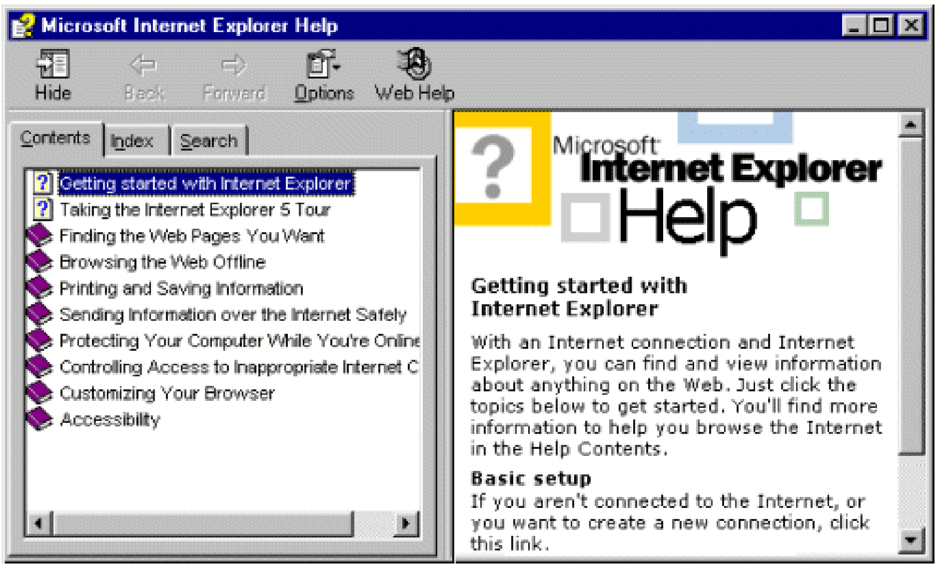

Figure 2.1: HTML Help example (Internet Explorer 5 Help system)

The left pane contains the navigation pane, which supports the table of contents, the index, text search, and other features. The right pane or topic pane displays HTML text and manages hyperlink navigation. The upper pane or toolbar pane contains and manages the toolbar buttons. Each help project has an index of important words and phrases that the user can search to find help topics on a subject. The index entries will be displayed when the user selects the index tab.

HTML Help currently runs only on Windows systems with Internet Explorer. Unlike plain HTML files, it is not cross-browser and cross-platform. HTML Help is also not designed for easy access via the Internet. To view a topic within an HTML Help file, the user must download the entire compiled file. The navigation pane and its tabs are controlled by an Active X control [18]. Active X controls, formerly known as OLE (Object Linking and Embedding) controls, are components or objects that can be inserted into a Web page or other application. These components add special features and actions, such as formatting features and animation. If a user does not have this control installed or if the help is being accessed through the Internet, the control will be downloaded from the Internet. If the user's security settings do not allow downloading of Active X controls, the table of contents, index and full-text search features will not appear. HTML Help stores each topic in a independent HTML file. When developing topics, HTML Help expects the use of separate HTML files for each topic. This often results in hundreds of HTML files. Therefore, HTML Help is not an option when developing help for large legacy documentation sets.

2.3.3 WebHelp

WebHelp, a help format developed by eHelp Corporation, is a variation on HTML Help. WebHelp can be run on Windows, Macintosh and Unix operating systems, and can be viewed with Internet Explorer and Netscape Navigator. WebHelp automatically detects the end user's platform and browser and will launch the appropriate files for optimal operation. This makes WebHelp display the help system consistently and reliably across multiple platforms and browsers. WebHelp supports standard HTML as well as other help capabilities, including dynamic table of contents, multi-level indexes, full-text search, context-sensitive help, related topics controls, expanding text, and more. WebHelp is an uncompiled format, which means that the HTML files that represent the topics of the help system, along with the other WebHelp generated files, are provided as separate files contained in the WebHelp output file. Because WebHelp is not a compiled format, all the WebHelp files can be stored locally on a user's computer. Therefore, a help compiler to compile the separate HTML files is not needed. WebHelp is a good option for providing help for a Web site or Web application. In this case, the WebHelp files can be stored on a server to enable web-based help. WebHelp is a single-source help format. Single-sourcing refers to the capability of producing more than one documentation format from one output file. Single sourcing enables help authors to write or import any HTML file, Word document or help project, store those help files in one project file and generate different formats from that one project file. WebHelp uses a bi-pane format to display the table of contents and index (on the left) and content (on the right).

2.3.4 JavaHelp

JavaHelp[1] is a new help format developed by Sun Microsystems that works with Java applications [6]. It is developed in the Java programming language. JavaHelp systems can be deployed on any platform that supports the Java Runtime Environment. One intent of JavaHelp is to provide an online help delivery system for HTML-based help that is independent of Windows and Internet Explorer. JavaHelp systems enable Java developers to provide online help for applications (both applet and stand-alone), JavaBeans components and applets in HTML pages. JavaHelp is a compressed file or a set of files used to run a help system for a particular Java application. All the help files needed for distribution with a software product have to be copied into a single folder, called JavaHelp or into a single file in the Java Archive (JAR) format, after compression of the help

[1] The JavaHelp distribution is free for download from Sun Microsystems JavaHelp web page at `http://www.java.sun.com/products/javahelp`.

files. Using compression can significantly reduce the size of a JavaHelp project. JavaHelp works equally well with help information that is not compressed into JAR files. This flexibility allows authors to view files during development without taking the time to compress them.
As shown in Figure 2.2, JavaHelp uses a tri-pane window. The tri-pane window,

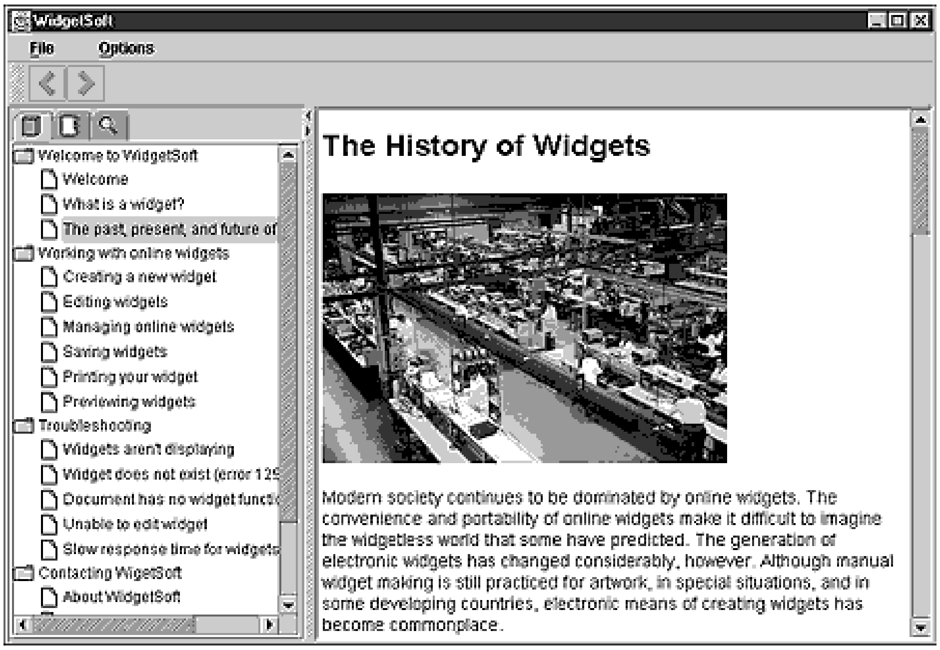

Figure 2.2: A JavaHelp system displayed in the Help Viewer window

called the HelpSet Viewer, is a HTML-based viewer and has a toolbar, a content pane and a navigation pane. The HelpSet Viewer enables users to select an item from the navigation pane and the corresponding help topic appears in the content pane (the right pane). If a help topic contains a link, the user can click it to display the corresponding HTML page in the content pane. The HelpSet Viewer functions like a Web browser, but is specifically designed to work with JavaHelp files. JavaHelp topic files are HTML files and these HTML files function the same way as in a Web browser. The HelpSet Viewer also supports pop-up windows and secondary windows.

The JavaHelp format is in its initial public release. JavaHelp is still under development and is not as feature-rich as other online help formats. For example, JavaHelp has no option for printing topics from the JavaHelp viewer, no support for scripting, cascading style sheets, or Dynamic HTML. In addition, JavaHelp shows a slow deployment speed. Particularly, when using graphics inside the help content. Some other features which are not supported by the JavaHelp format include links to external files and Active X controls.

Since JavaHelp can be deployed in an Unix environment, it is interesting to take a closer look at JavaHelp. We will discuss the possibilities of using JavaHelp as the online help format for an online help system for ToolBus applications. We are particularly interested in the process of creating and organizing a help system in this help format. How can one create a JavaHelp system?

Each online help system is implemented as a set of files, called the HelpSet. A HelpSet includes HTML-format topic files, navigation files such as indexes, and JavaHelp-specific configuration files. The following JavaHelp development process summarizes the procedures for creating an online help system:

- creating the HelpSet's directory structure;
- creating HelpSet data and navigation files;
- creating help topic files;
- testing the finished HelpSet.

The help files need to be written according to the specification of the JavaHelp format. For example, JavaHelp topics are written in HTML. A help author has to create these topics and apply the appropriate HTML tags. The JavaHelp map, table of contents, and HelpSet files are all written in the Extensible Markup Language (XML). The map file assigns a map ID (shorthand name) to each help topic. It maps ID strings to the uniform resource locators (URLs) of topic files. The help author has to create these files including all the required tags. All links between topics have to be declared one by one in XML. JavaHelp expects the files in the HelpSet to be organized in a predefined directory structure. Referring to a file by the wrong name or placing a file in a wrong directory, causes JavaHelp to fail when it tries to access the file. The directory and file structure are crucial for successfully running the help system. Creating and organizing a help system in this help format can be complicated and time consuming. Especially, when large complex navigational structures are involved.

JavaHelp is not recommended [6] when it is needed to incorporate an online help system into a non-Java application. Because of the different environments, other online help systems are better for different platforms. For example, a Visual Basic application would not be well suited for JavaHelp. Using the Java runtime environment only for the online help system is not worth the required resources. Instead, a more compatible help format such as WinHelp or HTML Help works better with the application and development environment.

2.4 Comparing Help Format Features

In the previous sections we discussed the four major help formats: WinHelp, MS HTML Help, WebHelp and JavaHelp. Table 2.1[15] summarizes the comparative features of these help formats. When help authors start designing their help system they should consider which help format to use. After setting up the requirements for an online help system, help authors can decide on a help format that meets their requirements.

MS HTML Help, WebHelp and JavaHelp are HTML-based help technologies. HTML-based help is any help system created using HTML files. HTML-based help systems are composed of HTML content files (as opposed to WinHelp's RTF files). Graphics are included using GIF and/or JPG files (WinHelp usually uses BMP files). HTML-based help systems can run from the Web, and they can be extended with Web-based technologies such as Active X, JavaScript and VB-Script. WinHelp lacks these features. When developing help for a new Windows product that does not have help, help authors should consider MS HTML Help.

MS HTML Help currently runs only on Windows systems with Internet Explorer. Unlike plain HTML files, it is neither cross-browser nor cross-platform. MS HTML Help is also not designed for easy access via the Internet. WebHelp can be used for HTML-based products that run locally on the user's computer and for Web-based applications. JavaHelp should be used when creating help for a 100% pure Java application. Accessing a JavaHelp system is not as fast as other HTML-based help formats and a feature rich set is not supported. It supports only basic text formatting and excludes DHTML and scripting. WinHelp should be considered when help is developed for Windows 95 or 3.1 applications or when maintaining large legacy documentation.

2.5 Other platform independent online help formats

A help system for ToolBus applications will run in a Unix environment. All the help files will be created and viewed on Unix machines. Therefore, we will explore more platform independent online help formats.

HyperHelp

HyperHelp by Bristol Technology [28] is an online help delivery system for Unix and Linux applications. HyperHelp supports all the basic features of an online help system, such as search for keywords, hyperlinks, full-text search, graphics, index and table of contents.

The HyperHelp development system consists of two separate applications: the HyperHelp Compiler and the HyperHelp Viewer. The HyperHelp Compiler is used to generate an online help system. The HyperHelp Viewer is used to display online help from an application. The HyperHelp Compiler takes all the files containing the help content as input and converts these into a help file (the .HLP file). HyperHelp can use the same RTF, project, and bitmap files as Microsoft Windows Help. This means that an existing Windows Help system can be converted into a Unix version by the HyperHelp Compiler. An author can write the help topic text in several tools. Once the help topics are authored, the appropriate HyperHelp Compiler that matches the chosen help file format must be used. HyperHelp Compilers support the following formats: Rich Text Format (RTF), Maker Interchange Format (MIF) created with either a Unix or Windows version of Frame Technology's Framemaker, Standard Generalized Markup Language (SGML), HyperHelp Text (HHT) created with an ASCII editor such as vi, emacs or Windows Notepad. In addition to writing the topics, special control codes must be inserted in each help topic to support navigation features. Some control codes, such as the code used to assign a keyword to a help topic, must be inserted into the help topic as a footnote. Other codes, such as the code used to create a link to another topic, are applied directly to the text. Once the help system has been coded, compiled, and tested, the help programmer needs to add code for invoking help to the parent application. The parent application invokes the HyperHelp Viewer and displays the appropriate help topic when an user selects an item on the help menu.

	WinHelp	**MS HTML Help**	**WebHelp**	**JavaHelp**
Browsers	No browser required	IE 4.0 or later	IE, Netscape or other	No browser required
Platforms	Windows	Windows 2000, 98, 95, NT 4	Platform independent: Windows, Unix, Mac	Platform independent: Windows, Unix, Mac
Runtime files	None	None	None	JRE 1.2.1 or later and JavaHelp 1.0 (or 1.1) components
Tri-pane window	Yes	Yes	Yes	Yes
Secondary windows	Yes	Yes	Yes	Yes
Files generated	.HLP and CNT files	.CHM file	Multiple HTML and WebHelp support files	.HS file and/or .JAR files
Compression	Yes	Yes	Yes (some files)	Yes
Deployment speed	Fast	Fast	Fast	Slow
Table of contents	Yes	Yes	Yes	Yes
Multi-level index	Yes	Yes	Yes	Yes
Full-text search	Yes	Yes	Yes	Yes
Hyperlinks	Yes	Yes	Yes	Yes (not external)
Pop-ups	Yes	Yes	Yes	Yes
Browse sequences	Yes	Yes	Yes	Yes
Glossary Tab	No	Yes	No	No
Related topics	Yes	Yes	Yes	Yes
Favourites tab	No	Yes	No	No
Websearch	Yes	Yes	Yes	Yes
DHTML effects	No	Yes	Yes (IE and NN 6.0)	No
Images	Yes	Yes	Yes	Yes
Multimedia	Yes	Yes	Yes	No
Scripts	No	Yes	Yes	No
Forms	No	Yes	Yes	Limited
Hotspots/Image maps	Yes	Yes	Yes	Yes
Context-sensitive help	Yes	Yes	Yes	For Java applications only
Embedded help	No	Limited	Yes	For Java applications only
Drop-down text	No	Yes	Yes(depending on the limitations of the browser)	No

Table 2.1: Help format comparison

	WinHelp	HTML Help	WebHelp	JavaHelp
Expanding text	No	Yes	Yes(depending on the limitations of the browser)	No
Style sheet	No	Yes	Yes (IE 4, NN 4 or later)	Yes
Pure Java	No	No	No	Yes
Active X controls	No	Yes	Yes (IE 4 or later)	No
Import Word documents	Yes	Yes	Yes	Yes
Import HTML documents	Yes	Yes	Yes	Yes
HTML content	No	Yes	Yes	Yes

Table 2.1: Help format comparison (continued)

NetHelp

NetHelp [27] is an online help format developed by Netscape Communications Corporation. It is an open standard, HTML-based help format and authoring system. NetHelp files can be viewed using a web browser and need not be re-compiled to move to other platforms. NetHelp2, the current version of NetHelp, was created to provide a help platform for Netscape Communicator. It is based on technology built into Communicator. This means that users of NetHelp2 systems need to have Communicator installed in order to access NetHelp.

NetHelp lacks some essential features. There is no support for hierarchical subheadings in a page. Every topic has the same heading format. Currently, there is no support for multi-file search. The only search feature is Find In Page. NetHelp is incompatible with standard HTML. Every URL is supposed to start with 'nethelp:', which makes the pages completely incompatible with regular browsers.

InterHelp

ForeFront's InterHelp technology [35] provides a cross-platform, HTML-based, browser independent online help format. The help content can be viewed on virtually any HTML browser on virtually any platform. The help files can reside locally on the user's machine or remotely on the Web. The only requirement is a Java-enabled browser, such as Microsoft Internet Explorer or Netscape Navigator.

The core InterHelp functionality is contained within four Java applets, which provide the basic navigational structures, such as a keyword index, table of contents, and full-text search. InterHelp enables authors to display small topics in a window that pops up over the current help window or in a secondary window. Unlike pop-ups, secondary windows remain up when the primary window has been closed. Secondary windows are primarily for displaying topics of hotspot links. InterHelp also provides an application programming interface (API) for context-sensitive help. An application can access the API to display any topic within the help system. The application can call the topic by its HTML file name. InterHelp technology implements these help features by using a combination of JavaScript and Java applet technologies.

Emacs Info

GNU Emacs [20] is an open source text editor. A component of Emacs is its online help system, called Emacs Info. Emacs Info is a hypertext browser for reading manuals online. These manuals or Info files are written in the GNU documentation format Texinfo. Texinfo [21] is a documentation system that uses a single source file to produce both online information and printed output. This means that instead of writing two different documents, one for the online help and the other for a typeset manual, only one document needs to be written. To make a document suitable for printing, a Texinfo source file needs to be processed by the TeX typesetting program (the Texinfo language is different from plain TeX). This program creates a DVI file that can be printed as a book or report. Info files are the result of processing Texinfo files with the program Makeinfo or with the Emacs command M-x texinfo-format-buffer. Makeinfo is a utility that converts a Texinfo file into an Info file. Info as a stand-alone program is used to view info files on an ASCII terminal. A Texinfo source file is a plain ASCII file containing text and @-commands (words preceded by an @) that tell the typesetting and formatting programs what to do. By convention, Texinfo source file names end with a '.texi' or '.texinfo' extension. A Texinfo file can be written with any text editor. Emacs has a special mode, called Texinfo mode, that provides various Texinfo-related features. Texinfo files can be converted into HTML files by Makeinfo using the –html option. Currently, the available output formats for Texinfo by Makeinfo are Info and HTML. The Emacs Info browser is required for viewing Info files. From the same Texinfo source file it is possible to specify a menu-driven, online Info file with nodes, menus, cross references, and indices. A HTML output file suitable for use with a web browser can be created from that same source file also.

Info files are divided into pieces called nodes. Each node contains the discussion of one topic. A node contains both text for the user to read and pointers to other nodes, which are identified by their names. The Info program displays one node at a time, and provides commands with which the user can move to other related nodes. Each node of an Info file may have any number of child nodes that describe subtopics of the node's topic. The names of child nodes are listed in a menu within the parent node. With Info commands it is possible to move to one of the child nodes. Generally, an Info file is organized like a book. If a node is at the logical level of a chapter, its child nodes are at the level of sections. Likewise, the child nodes of sections are at the level of subsections. All the child nodes are linked together in a bidirectional chain of 'Next' and 'Previous' pointers. The 'Next' pointer provides a link to the next section, and the 'Previous' pointer provides a link to the previous section. In addition to menus and to pointers, Info provides pointers of another kind, called references. These references enable cross-referencing through the documentation system. An Info file is entered through a node that by convention is named 'Top'. This node contains a summary of the file's purpose and a menu. From this top node it is possible to search the index menus or traverse the file by going from node to node. The stand-alone Info program provides command line driven options to alter menu specific items.

Info files can be read by entering Emacs Info mode or by invoking the Info program. To start the Emacs Info mode from within Emacs by typing Control-h or by selecting an entry from the help-menu. The stand-alone program can be invoked from the command line by typing 'info <filename>' or 'xinfo <filename>' (from an X Windows login).

An Info file can be read on virtually any computer terminal. This makes Texinfo a general purpose system. On the other hand, this brings with it a constraint, which is that a Texinfo file may contain only the customary 'typewriter' characters (letters, numbers, spaces, and punctuation marks) but no special graphics such as bitmaps. This makes Texinfo mainly suitable for text-based software documentation.

Unix manual pages: Xman

For many Unix programs there are manual pages. These are usually reference documents, but occasionally provide more substantive information. These documents are organized with several sections to categorize information. For example, the manual pages for most of the common user commands are in section 1. These documents can be read by invoking the Xman program [23], a manual page display program for the X Window System, or by typing 'man <programname>' from command line. The man command uses a pager, by default more, to display a man page one screen at a time. Man pages can also be read from within Emacs.

The initial window from Xman provides three options: Help which will pop up a window with online help, Quit, and Manual Page which will pop up a window with a manual page browser in it. Figure 2.3 shows an index for user commands from the Xman program.

Figure 2.3: A manual page in Xman

More information on Unix help systems can be found in [22]. They describe techniques applied to the problem of providing help systems mainly for the Unix operating system.

2.6 Conclusions

We use a minimal set of requirements for the development of a help system for ToolBus applications. The requirements are based on our discussion of several online help formats in this chapter. The requirements for the online help format used in TBHelp are listed next.

- The online help format delivers at least the following elements of an online help system: a table of contents, index, links, browse sequences, and full-text search. These navigation structures should be delivered through a tri-pane window in a help viewer. The help topics are usually shown in the right pane.
- The online help format used in TBHelp should run on Unix machines.
- The online help format should be browser independent.
- The help system should be linked to ToolBus applications for providing context-sensitive help.

HTML-based help formats run on any browser and any platform. This solution provides the most flexibility for help authors and application developers. Cross-browser and cross-platform systems consist of a set of individual HTML files (one per topic), which facilitates access to topics from a web site. On the other hand, there are more files to manage: in addition to the topic files, there are graphic files, files for the content, index and full-text search. Depending on the users' browser, a help system might appear differently or with inconsistent formatting.

WinHelp and MS HTML Help are a compiled help format, designed to operate on Windows operating systems. Therefore, these help formats are not suitable for an online help system for ToolBus applications. We will consider using a HTML-based help format, because the online help system we will design will run in an Unix environment. JavaHelp and WebHelp formats enable HTML-based online help delivery in an Unix environment. Additionally, HTML-based help delivered through a web browser provides platform and browser independence. It has none of the configuration requirements that complicate the distribution of Microsoft's HTML Help.

We have explored existing Unix-based online help systems, such as Xman, Emacs Info, JavaHelp and HyperHelp. These are all online help formats and do not provide help authoring assistance. Writing help files for these systems requires a lot of low level knowledge from help authors.

Chapter 3

Comparison of Help Authoring Tools

In the previous chapter we discussed the key features that constitute an online help system and contemporary online help formats. We will describe tools that facilitate the help authoring process in the next sections. These tools attempt to automate the process of creating and organizing help information as much as possible. Help authoring tools have the ability to generate several online help formats. In this chapter we give an overview of the various help authoring tools that are available and an evaluation of the comparative features of the four major help authoring packages: Doc-To-Help Professional 2000, ForeHelp Premier 2000, HDK/Virtual Media Suite 2000 and RoboHELP 2000. Additionally, we explore the possibilities of reusing existing help technology in the development of a help authoring tool that runs on top of the ToolBus. Finally, we conclude with a set of requirements for a help authoring tool, which delivers online help for ToolBus applications.

3.1 Basics of help authoring tools

The process of creating a help system can be complicated. When constructing a help system, an author will need to implement navigation structures and display features in addition to writing the help system content. Organizing the content and creating the navigation is the most involving aspect of help system design and construction. In practice, creating navigational structures for a help system can be a time-consuming task, but with the proper tool the process can be streamlined. A help authoring tool can automatically generate a table of contents based on the hierarchical structure of topics. Major help authoring tools have a table of contents editor that includes drag and drop functionality for rearranging the topic order. An index is an essential part of a help system. It provides quick access to topics by listing keywords. Creating a comprehensive index can also be a time-consuming process, regardless of the help format on which it will be deployed. A help authoring tool with an index generator can save a lot of time in a help development schedule while producing comprehensive multi-level indexes. An automated indexing process can also be used to ensure that the index remains up-to-date as new topics and index entries are added for a help project.

The goal of a help authoring tool is to automate the process of creating and organizing content as much as possible, allowing the author to focus on the help system's content, design and usability instead of the mechanics of implementing specific features on different help formats. Many companies have invested in creating solutions toward this goal. Ehelp Corporation [24], for instance, has invested over fifty man-years of research and development in creating help authoring software. Help authoring tools provide a complete set of tools to develop online help for application software. Help authoring tools contain tools to write, process and view online help. These tools enable authors to write online help that includes graphics, text formatting, and hyperlinks. Some help authoring tools provide a set of tools for integrating online help into an application.

Creating a useful help system requires thorough project management. High-level analysis, like examining the help system's overall structure, or testing the system's functional integrity and lower-level tasks, such as looking for broken links or unreferenced content need to be performed during the course of a help project. Help authoring tools directly support these efforts by providing project management features and reporting capabilities that specifically reflect the help authoring process.

The various help formats available today and the range of features they support, raise issues such as re-use and portability. A single-source help authoring tool enables the use of help content and navigation already contained in existing help systems for deployment on other help formats. With a help authoring tool that supports single-sourcing, help authors can create a help system in one help format and convert to another. Help authors are not limited to a particular help format once they have created the help system. Since help systems are typically a combination of new and existing content, existing help files can be reused in the new help system.

3.2 Doc-To-Help Professional 2000

Doc-To-Help [25] by WexTech is a commercially available tool for generating Windows-Help from Microsoft Word Documents. Doc-To-Help produces printed documentation, online help (for WinHelp, HTML Help, and JavaHelp formats), and HTML documents from a single source document. Doc-To-Help includes Helpsite, a Help-to-Web conversion tool; Quicture, a graphics handling utility for Word for Windows; Setup Wizard, a utility for creating professional installation programs for stand-alone Help systems; Smooth Scaling, a stand-alone Windows program for reducing the size of graphic images; and the AnswerWorks component, a natural language help query tool. AnswerWorks allows users to get answers to plain language questions. Users simply type in questions using their own words, and AnswerWorks analyzes each question and lists the most relevant Help topics. The returned list is typically smaller than a full text search. Doc-To-Help adds templates to a Word installation, using macros to add help authoring functionality. A manual has to be written first, using Doc-To-Help tools to add links, glossary, index information, and other WinHelp information.

Doc-To-Help is in fact a set of Microsoft Word for Windows (WinWord) templates and macros. The idea behind Doc-To-Help is to use the templates provided to produce rich documents on screen, then run the supplied macros for

automatic link creation. The result can be used as both a paper document and a hyperbook (an electronic book with hypertext features). Large structures such as the tables of contents, glossaries, and indices are all automatically translated to their hypertext equivalents. Doc-To-Help supports the marking of text as appearing in either the paper form or the hyperbook. The Help Compiler can be run directly from WinWord.
The following list summarizes the tools that are included in the Doc-To-Help package.

- *Doc-To-Help 4*, an authoring system for all help files and documents.
- *AnswerWorks*, a natural language interface which adds to the traditional search facilities, allowing users to ask questions in their own words.
- HelpXtender, includes comprehensive facilities for developing context-sensitive help.
- *Smooth Scaling*, a screen capture and anti-aliasing program.
- *Lotus Screencam*, a tool which captures on-screen actions that can be included in the help system.

Doc-To-Help supports a library of cascading style sheets which allow different presentational features to be defined for the HTML environment. It also supports the JavaHelp format, which allows for cross platform compatibility.

3.3 ForeHelp Premier 2000

ForeHelp Premier 2000 by ForeFront is a suite of help authoring applications which includes ForeHelp [19]. ForeHelp has applications for producing WinHelp, HTML Help, help for Visual Basic applications, JavaHelp, help for Windows CE, and InterHelp (a cross-platform, HTML-based, browser independent online help format). ForeHelp does not require a word processor in order to function. It provides a WYSIWYG (acronym for "What You See Is What You Get") editor. A WYSIWYG editor is one that allows the help content developer to create a page of text so that the developer can see what the end result will look like while the document is being created. A WYSIWYG editor can be contrasted with more traditional editors that require the developer to enter descriptive codes (or markup) and do not permit an immediate way to see the results of the markup.

ForeHelp, runs on Windows, but the output can run on any operating system that has a browser with Java and JavaScript. The output can be delivered via HTML Help, JavaHelp, Windows CE Help or InterHelp. Windows CE Help runs on compact computing devices, such as hand-held computers or palmtops. These devices have limited storage and memory capacity and, therefore, support a subset of the familiar help features: HTML 3.2, table of contents, index, bitmap graphics and jumps.

uForeHTML Pro is a tool that enables users to develop Windows CE Help from scratch or by converting any other help project. The testing or previewing of a Windows CE project can only be done on a CE device or a CE simulator on Windows NT After building a Windows CE Help project, ForeHTML

Pro generates one .HTP file containing all the topics with appropriate links. All popups are converted to jumps. If customization of the list of Help files is needed in Windows CE Help, ForeHTML Pro can generate the link (.LNK) file. ForeHTML Pro generates a table of contents from the entries in the Contents Editor and places the table in the main contents topic. ForeHTML Pro builds the index from assigned keywords. The tools provided by ForeHelp Premier 2000 include:

- *ForeHelp 4* is the main help authoring tool for creating WinHelp 4 files.
- *ForeHTML Pro 4* builds HTML Help, JavaHelp, Windows CE Help and InterHelp (a cross-browser/cross-platform help system from ForeFront).
- *Index Expert* creates a complete index using keywords and can add keywords to groups of topics at a time.
- *QuickContext* creates context-sensitive links for C++ applications.
- *ForeVB* creates context-sensitive links for Visual Basic applications.
- *QuickFix* offers help on context-sensitive help.
- *Help Buttons* enables help authors to customize their graphic and text buttons
- *ActiveX/OLE for Help*, a tool for linking ActiveX controls into a help file.

3.4 HDK/Virtual Media Suite 2000

Virtual Media [26] has been developing hypertext since 1993. Virtual Media currently produces three packages; Hypertext Development Kit (HDK), Virtual Media 2000 Suite and WebLite. These products are aimed at making the creation and management of on-line documentation, Web sites and help files easy and effective. These packages run on Windows V3.1, 95, 98, NT3.51, NT4 and 2000. Virtual Media Suite 2000 comes with HDK, HyperLite, MediaZoom, HelpMate and WebLite.

- *HDK 2000* the main help authoring tool.
- *HyperLite*, a tool for creating a graphical representation of multiple help systems or web sites. This tool manages and facilitates the navigation through large document collections.
- *MediaZoom*, this tool can zoom and hotspot large flowcharts, diagrams and maps in WinHelp and HTML. MediaZoom can display large graphics within a relatively small frame in WinHelp or Internet Explorer. Hotspot editing enables the linking of zones in a graphic to other pages and documents.
- *WebLite*, automatically converts Word, Framemaker and Word documents into HTML.

- *HelpMate*, this help authoring tool will scan an application and extract images of all the windows and dialogs in that application. After extracting a graphical database from an application's windows and dialogs, HelpMate builds context-sensitive help interactively.

HDK enables the creation of help systems and HTML. HDK 2000 includes all the tools needed to create online documents with graphics and formatting features. This package can automatically convert .RTF documents into HTML. WebLite's wizard makes it possible to create web information from Word, Framemaker and WordPerfect documentation. The help files and web pages are required to be written in Word, since HDK requires Word. WebLite requires any .RTF compatible word processor, (incl. WordPerfect, Framemaker and Word). HDK supports the following help formats: WinHelp 3.1 and 4 (require Windows), HTML Help and NetHelp. NetHelp is an open standard, HTML-based help format and authoring system. NetHelp files (by Netscape) can be viewed using a web browser and so do not need to be recompiled to move to Macintosh or Unix systems.

3.5 RoboHELP 2000

RoboHELP 2000 [10] is the latest release of eHelp Corporation's help development suite. RoboHELP can generate various online help formats, including WinHelp and HTML Help for Windows 3.x, Windows 95, 98 and 2000. Other cross-platform online help formats include WebHelp and JavaHelp. WinHelp and HTML Help can be created using Word for Windows and the standard Microsoft WinHelp and HTML Help compilers, with no interference of RoboHELP. A more convenient way is to use RoboHELP, which automates the help creation process.

The overall process for creating a help project with RoboHELP will be described next.

1. First, a new help project in RoboHELP must be created. An online help project is a suite of files containing topic text, graphics, formatting instructions, tables of contents, indexes, and so on that, when compiled, create an online help system. The help project file contains instructions that describe how to compile the online help project and what source files to use. RoboHELP uses the help project file to manage the files in a project.

2. In Word for Windows the topics, text, graphics, multimedia, and so on must be entered. This Word file(s) will communicate the information to the users.

3. To enable navigation between topics, jumps, popups, and other hyperlinks must be inserted.

4. The table of contents must be created, which provides a fast way for users to find information.

5. While writing topics, a help author can add index words and phrases (keywords) to those topics. After writing the topics and keywords, the Word document(s) must be indexed, to create an index.

6. The source files must be compiled into the designated help format now. The compiler may report some errors and warnings.

7. The compiled help file must be tested by evaluating the online help system. This can be done by running the help file(s) in a help viewer. For example, a WinHelp file can be evaluated by running the file in the WinHelp viewer. The help files must be recompiled after each change and then tested again.

This is the typical flow for creating a help project in RoboHELP. Next, we will describe how RoboHELP creates a table of contents.

The table of contents

All the topics written by a help author are summarized by RoboHELP in a topic list. This list can be viewed using RoboHELP Explorer, a three paned help viewer. RoboHELP provides three methods to create a table of contents.

- *Drag-and-drop*: the Table Of Contents (TOC) Composer shows in the right pane a topic list (see figure 3.1). In the left pane the created table of contents will be displayed. By simply dragging topic entries from the topic list (right pane) and dropping them into the left pane, a table of contents is created. The topics can be structured into a hierarchy of books (a group of related topics sorted hierarchically). After creating the initial table of contents, the location of topics in the table of contents can be changed by clicking and dragging the topic to another location. When users double-click a topic entry in the TOC Composer in the RoboHELP Explorer, RoboHELP displays the associated topic in the Word file for viewing or editing.

- *Auto Create TOC*: RoboHELP creates a quick table of contents by creating a single top-level book for each of the documents in the project. The topics are listed in the order they appear in the help documents. The entries in the table of contents can be reorganized with the previously discussed drag-and-drop method. The Auto Create TOC tool does not structure the table of contents automatically, but simply repeats the order of the topics in the help files. Manually organizing the topics in the table of contents can be time consuming.

- *Using the Contents (CNT) file*: the information for the table of contents is stored in a text file with an extension of CNT. This method requires the knowledge of several commands which will appear in the CNT file. For each topic the associated table of contents, indentation level and the associated help file can be specified by the commands.

RoboHELP uses a suite of macros and templates that are loaded into Microsoft Word. It also comes with external programs such as a program for managing help projects and help project information, and a number of utilities that assist the help author by converting help projects from one format to another, importing information, and debugging and enhancing help projects. These macros, templates and programs eliminate almost all of the mechanical

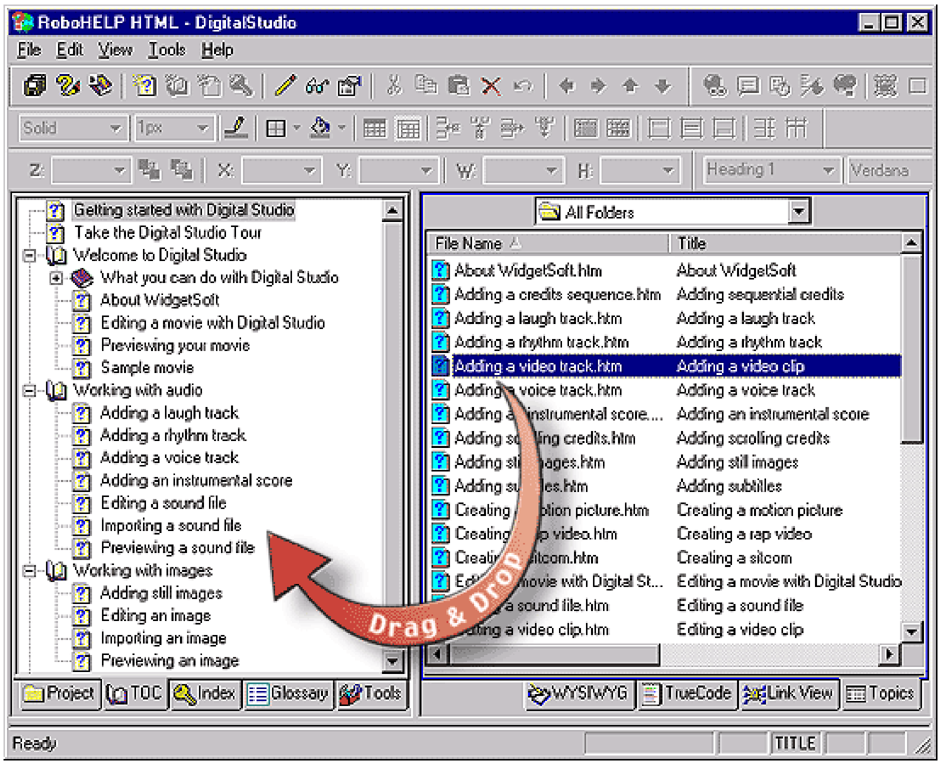

Figure 3.1: Creating a TOC with drag-and-drop

parts of the help creation process, making it possible for a help author to concentrate on the design and writing of help content. RoboHELP comes with two major components: RoboHELP Classic, for creating standard WinHelp files, and RoboHELP HTML, for generating files in the HTML Help standard used by Windows 98. Both the Classic and HTML editions of RoboHELP use a similar interface design. This makes it easy to move between the two tools, without having to learn new techniques.

RoboHELP HTML

RoboHELP HTML is a large program within the RoboHELP 2000 package. With this program, help authors can create HTML-based online help systems. RoboHELP HTML has its own editor. Therefore, the help files need not be written using Microsoft Word. RoboHELP HTML runs on Windows 98 and 2000. The HTML Help system itself is platform independent. This means that the help files must be generated on Windows and can be viewed on any computer that has an HTML browser. The What's This? Help Composer creates context-sensitive links for C, C++ and Visual Basic applications.

RoboHELP Classic

With RoboHELP Classic [29] one can create a new WinHelp project or open an existing one. This package requires Microsoft Word 95/97/2000 in order to function. The help files containing the topics, text, graphics, and so on are all created using Word. RoboHELP Classic includes technologies such as the RoboHELP Explorer, drag and drop Index Designer, Table of Contents Composer, management reports and the What's This? Help Composer. It also includes the Smart Index Wizard, a tool in RoboHELP Explorer used for automating the creation and maintenance of indexes and keywords. It scans an entire WinHelp project based on criteria that are entered by the help author to assign keywords. In addition, the Smart Index Wizard can suggest new and existing keywords based on the content of topics.

RoboHELP has an extensive set of features, but there are drawbacks to the use of RoboHELP. It is dependent on the use of Microsoft Word, which means that RoboHELP must be updated when there are new features, but also potentially whenever Word has changes. RoboHELP has a difficult learning curve, due to the complex user interface and large documentation set. In addition, it can be expensive to own because of frequent major upgrades.

Creating a JavaHelp system for ToolBus applications

There are basically three steps involved with creating a JavaHelp system with RoboHELP.

- Creating the component files (the HelpSet and Java classes that implement the help system) of a JavaHelp system with RoboHELP.
- Compressing the files into a JAR file. This step is optional. In order to compress the components of a JavaHelp system, the Java Development Kit (available from Sun's web page) is needed.
- Viewing the JavaHelp system by running the HelpSet Viewer from the JavaHelp program or directly from RoboHELP.

After creating the HelpSet, adding the help system to applications can be done as summarized by the following steps.

- Import the JavaHelp system classes:

```
import javax.help.*;
```

- Find the HelpSet file and create the HelpSet object:

```
try{
   URL hsURL = HelpSet.findHelpSet\
                      (null, ''../help/myHelpSet.hs'');
   hs = new HelpSet(null, hsURL);
   } catch (Exception ex)  {
       System.out.println\
```

```
                  (''HelpSet  ''+helpsetName+'' not found'')
          return;
      }
```

- Create a HelpBroker object. A helpBroker is an agent that manages the display of help content for the application.

```
hb = hs.createHelpBroker();
```

- Create a Help menu item to trigger the help viewer:

```
JMenu help = new JMenu(''Help'');
menuBar.add(help);
menu_help = new JMenuItem(helpsetLabel);
menu help.addActionListener\
                    (new CSH.DisplayHelpFromSource(hb));
```

The first thing the application does is read the HelpSet file specified by the application. Then the viewer is loaded with the main page defined in the HelpSet. Figure 3.2 shows how a Java application runs locally and accesses help data installed on the same machine. The application, requests the creation of a JavaHelp instance, loads the help data on it and presents the requested help topic. It is also possible to initiate help by connecting an application to the ToolBus. The application requests help by sending a message to the ToolBus. The ToolBus handles the request by initiating a process for loading the HelpSet Viewer with the specified help data.

Since RoboHELP only runs on Windows, the above solution is not suitable for delivering online help for ToolBus applications. We want to be able to author the help files in a Unix environment. Additionally, the support for JavaHelp at the moment is not recommended. This help platform is in an immature state and has a slow deployment speed.

3.6 Comparing Doc-To-Help, ForeHelp, HDK and RoboHELP

In table 3.1 to 3.6 [16] the comparative features of RoboHELP, HDK, ForeHelp and Doc-To-Help are shown. In terms of feature-richness, there isn't too much differentiation between these help authoring tools. The supporting tools and wizards are effective aids in the process of building a help system and release help authors from the more technical tasks like building an index or the construction of links between topics.

Most help authoring tools can produce printed documentation, but the process is often cumbersome. Doc-To-Help provides the best solution when a true single source tool is needed. This tool is best suited for a linear help project such as a tutorial or a software user's guide organized around menu options. Because the help is driven from the same document that produces the print manual and because the compilation processes are lengthy, making fast revisions and testing parts of a help system are cumbersome. Doc-To-Help's components, AnswerWorks and HelpXtender, have no comparison with other help authoring tools.

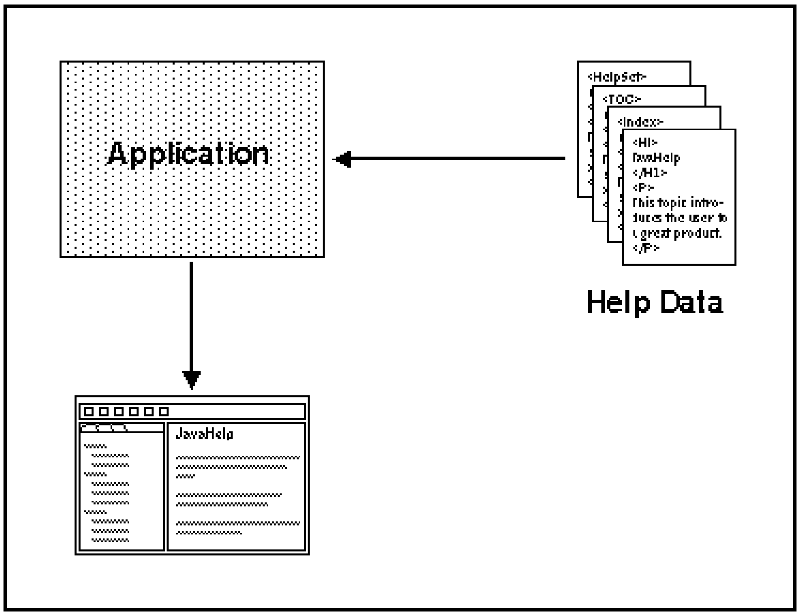

Figure 3.2: Application help

These components offer features that represent trends in cutting-edge technology.

ForeHelp can be seen as the opposite pole from single sourcing. One can compile the help files rapidly, test individual topics, and revise the entire project rapidly. This tool is best suited for non-linear relationships such as course catalogs. ForeHelp implements user feedback by providing a WYSIWYG editor. ForeHelp also has the ability to import help files created with other tools. ForeHTML Pro can import web pages and, optionally, all linked pages that may be local to the system. This makes it possible to insert entire web sites. ForeHelp keeps the existing HTML intact, including any scripts within the pages. Finally, in our opinion ForeHelp has the most feature-rich set. ForeHelp is able to build complex help systems which demand advanced functionalities, in addition to the basic online help system features.

RoboHELP is a general purpose help authoring tool. It is the proper tool for diverse range of projects. In terms of single sourcing, RoboHELP is the weakest of the four major help authoring tools. RoboHELP will only convert simple help projects into a clean Word document, and even then it's likely that the document will show some irregularities. This means that printing online documentation is difficult to enforce. RoboHELP inserts RTF codes directly in help files, which are compiled by the WinHelp compiler. Help developers can manually edit these RTF codes. RoboHELP is best suited for developers who want to tune the results manually. RoboHELP is not suited for complex help projects. If one wants to create Microsoft standard help files for a simple online help system, without specific requirements, RoboHELP is the proper tool to use.

HDK's basic approach is to create a help file with a stand-alone program, which is stored in a database. The topics are kept in Word documents, so help authors need to switch between Word and HDK as they create a help system. HDK excels at converting existing document sites into a help format. HDK handles the conversion of an existing database into a WinHelp system well. It can take old documentation residing in various databases and convert it to help files. This makes HDK well suited for handling large databases. Like ForeHelp, RoboHELP and Doc-To-Help, HDK can create help files for Windows 3.1, but it provides those help files with more functionality than the other authoring tools. HDK is therefore the proper tool for developing online help systems in standard WinHelp files. HDK is the most specialized help authoring tool, with few broad-based features. Another drawback to HDK, is its relatively inaccessible support. It can be difficult to find help on a specific task, because HDK lacks well structured and organized online support for the help authoring process.

3.7 Conclusions

To deliver online help for ToolBus applications we need to have a help format that enables online help delivery on Unix machines. Additionally, we want to automate the process of help authoring by using a designated help authoring tool, which runs in a Unix environment. The help authoring process can be time consuming and complex without a help authoring tool, due to the required manual coding techniques.

Doc-To-Help, ForeHelp, HDK and RoboHELP all run on Windows systems. This means that all the writing, organizing and compilation of help files needs to be done on a Windows platform, whereas the online help delivery can occur on other platforms as well, depending on the chosen output format. To deliver online help for ToolBus applications, HTML-based help files need to be generated with one of the previously mentioned help authoring tools running on Windows. The output format can either be JavaHelp, NetHelp, WebHelp or InterHelp. Once the help files are created on a Windows system, it will be possible to view the help content on Unix machines. JavaHelp comes with its own viewer, the HelpSet Viewer, whereas InterHelp and WebHelp can be viewed with any HTML-based browser. NetHelp requires Netscape Communicator. The only drawback to this approach is the fact that help files need to be generated on a Windows system. We want to be able to create the help files in a Unix environment, therefore we will not continue with this approach.

The four major help authoring tools studied in this chapter, are all commercial products. We explored the features offered by these tools and compared them. From this comparison we can set up requirements for our design of the help authoring tool, which will automate the process of creating and delivering an online help system for ToolBus applications. We define a set of requirements for a help authoring tool, which delivers online help for ToolBus applications next.

- The help authoring tool is required to run in a Unix environment.

Features	**Doc-To-Help**	**ForeHelp**	**HDK**	**RoboHELP**
Requires Microsoft Word or uses WYSIWYG editing?	Word	WYSIWY	Word	Both
Can use styles/templates?	Yes	Yes	Yes	Yes
Comes with a template library?	Yes	InterHelp only	Yes	Yes
Wizard-driven help file creation (or equivalent)?	Yes	Yes	Yes	Yes

Table 3.1: Help Authoring Tools: comparison of the application

Features	**Doc-To-Help**	**ForeHelp**	**HDK**	**RoboHELP**
Can tool produce Web sites in pure HTML?	Yes	No	Yes	No
What help formats are supported?	WinHelp 3 and 4, HTML Help, JavaHelp	WinHelp 3 and 4, HTML Help, JavaHelp, Windows CE Help, InterHelp	WinHelp 3 and 4, HTML Help, NetHelp	WinHelp 3 and 4, HTML Help, JavaHelp, WebHelp, Oracle Help
If tool produces a proprietary help format, what are the differences/advantages between this and standard formats?		InterHelp: cross-platform, browser-based	Extended HTML gives similar features to InterHelp and WebHelp	WebHelp: cross-platform, browser-based
Wizard-driven help file creation (or equivalent)?	Yes	Yes	Yes	Yes

Table 3.2: Help Authoring Tools: comparison of the output formats

Features	**Doc-To-Help**	**ForeHelp**	**HDK**	**RoboHELP**
Is application single sourcing: can it produce paper as well as online manuals from the same project?	Yes	Yes	Yes	Yes
Does the appliaction support conditional compiling: i.e. the ability to mark text as 'online only' or 'print only'?	Yes, Conditional Text feature	Yes, using up to 16 conditions	Yes, marking of text for processing by 'fuzzy logic' conditional compiler	Yes, using build tags
How are 'print only' features like page numbers handled when project is compiled as online help?	With numbering feature, a project is automatically numbered, regardless of output	When importing a print file, cross-references become hyperlinks	Page references become hyperlinks, footnotes, become popups, etc.	Ignored

Table 3.3: Comparison of Help Authoring Tools: single sourcing

Features	**Doc-To-Help**	**ForeHelp**	**HDK**	**RoboHELP**
Does the package include an indexing utility?	Yes	Yes: Index Expert	Yes: auto-indexing	Smart Index Wizard
Does the package include a browse sequence feature?	Yes	Yes	Yes	Yes
Can the package generate browse sequences automatically?	Yes	Yes	Yes	Yes

Table 3.4: Comparison of Help Authoring Tools: index and browse sequences features

- A component which automates the process of adding a robust full-text search capability must be included. To support the more skilled user, a powerful search facility is necessary to allow quick access to discrete pieces of information.
- A component that facilitates the process of creating a table of contents.
- The process of creating an index or other browse sequences must be automated by the help authoring tool.
- The tool must be able to produce a HTML-based online help format.
- Help authors must be able to write their help content in an editor. The use of a standard editor, such as vi or emacs, must be supported.
- The help authoring tool must be able to produce paper as well as online manuals.
- The tool must be easy to use and provide online help for the supported features.

Features	**Doc-To-Help**	**ForeHelp**	**HDK**	**RoboHELP**
Does the application have a graphical overview of topics and links?	Yes	Yes: Navigator or Grapher tool	Yes: Outliner	Yes: Link View
Is there a mechanism for reporting broken links?	Yes: Diagnostics tool	No	Yes	Yes: Link View or Broken Links report
What reporting and statistics are offered (for a help project)?	None	Reporter tool	Reporting tool	A number of customisable reports
What testing/debugging/diagnostics tools does the package include (over and above reporting of compiler output)?	Navigator tool	Instant testing of topics, topic preview, QuickFix and reproting tools	project analysis and auto-fixes or suggested remedies. Application emulation	BugHunter, Activetest mode, WinHelp Inspector

Table 3.5: Comparison of Help Authoring Tools: project management

Features	**Doc-To-Help**	**ForeHelp**	**HDK**	**RoboHELP**
What paper manuals are shipped with the product?	One manual per tool	ForeHelp 4 User's Guide, ForeHTML Pro 4 User's Guide, Add-on tools User's Guide	Quick-and-Easy Guide	User Guide, WebHelp Resource Guide
What bonus on-line documents ship with the product?	None	Samples and product tour	Tutorial, Document Design Guide, Troubleshooting Guide, Knowledge Base	Overview, samples, tutorial
Does the application include extended or assisted scripting for macros?	No	Yes, script editor for HTML-based help and macro editor for WinHelp 3/4	Yes, script editor for HTML-based help and macro editor for WinHelp 3/4	Yes, macro editor for WinHelp 3/4
What graphical tools does the package include?	Smooth Scaling, Lotus Screen-Cam	Graphics Manager	Animated bitmap generator, MediaZoom screen capture	Image workshop, Graphics Locator, ReSize, Software Video Camera, image map editor

Table 3.6: Comparison of Help Authoring Tools: miscellaneous

Chapter 4

The TBHelp Generator

This chapter describes the design and features of the *TBHelp Generator*, which is an important part of the development process of TBHelp. The TBHelp Generator is one of two parts of the TBHelp Help Authoring Tool (HAT). Section 4.1 lists the reasons for designing the TBHelp Generator, instead of reusing existing help technologies. In Section 4.2 we describe the main features of the TBHelp Generator. The design of the TBHelp Generator is component-based and complies with most of the basic requirements we defined in Section 3.7. Section 4.3 describes the component architecture of the TBHelp Generator. Each of the components defined in this section has its own functionality. Sections 4.4 upto 4.10 describe the functionality of each of the software components, defined in Section 4.3.

4.1 Why The TBHelp Generator?

We had several reasons to build the TBHelp Generator, instead of using one of the existing HAT's described in Chapter 3:

- The four major HAT's described in the previous chapter, only run in a Windows environment. To generate help systems in a help format that can be viewed in a UNIX environment, help authors must create their help systems in a windows environment. TBHelp was in the first place created for ToolBus applications. ToolBus applications currently only run in a UNIX environment. Users who create a ToolBus application, would rather write their help system in that same UNIX environment, instead of having to turn to a windows environment. TBHelp has been designed to run in a UNIX environment. The TBHelp Generator creates help systems in a HTML-based help format, which is a browser-independent and platform-independent help format. We call this the *TBHelp help format*. The *TBHelp* help format is described in Chapter 5.

- Efforts are being made to port the ToolBus software architecture to a windows-platform. TBHelp has been written mainly in Java and Tcl, which both run in a UNIX environment and windows environment. Once the ToolBus software architecture runs in a windows environment, minimal changes to TBHelp will allow help authors to create help systems in any environment he/she likes. The needed changes are described in Section 7.1.

- HAT's that run in a UNIX environment (e.g. JavaHelp, Emacs Info etc), only consist of the viewer part of a HAT. Creating help systems in a UNIX environment is therefore time-consuming because a lot of low level knowledge is needed by the help author. Although some of these HAT's or viewers like JavaHelp, provide templates for creating help systems. Managing for instance a table of contents or index of keywords, takes a lot of work. The TBHelp Generator automatically generates an index, and provides tools that relief the help authors task of producing a table of contents.

- Most well known and frequently used HAT's are commercial products. They are usually expensive and do not conform to the wishes of ToolBus architecture users. Users are not able to adjust the HAT's to their specific needs, due to copyrights. Adding necessary functionality for instance, can only be done by the owner of the HAT.

- Creating an own HAT allows us to add new functionality or update the HAT with functionality, when necessary. The source code is easily accessible and understandable.

- The TBHelp Viewer is written especially for ToolBus applications. The TBHelp Viewer provides the functionality for integrating a help system in *TBHelp format*, with a ToolBus application. This means that a help system that should be connected to a ToolBus application, needs to be either generated by the TBHelp Generator, or manually transformed to support the TBHelp help format. Because other HAT's are commercial software, their output format cannot be moulded into the correct format needed by the TBHelp Viewer.

4.2 Features of the TBHelp Generator

This section lists the features of the TBHelp Generator. A help system created with the TBHelp Generator, can be used as a stand-alone help system, by opening the *index.html* file of the help system in a browser. The help system can also be integrated with a ToolBus application, which is described in Chapter 6. An example of how a help system looks when created with the TBHelp Generator is shown in Figure 4.1.

A help system created with help of the TBHelp Generator contains an alphabetically ordered index, a table of contents and search capabilities. Once a help system has been created, it can be used stand-alone by opening the *index.html* file of a help system in a browser. The help system can also be connected to the ToolBus application it was created for. Chapter 6 lists the TBHelp Viewer features, which allows the user to connect a help system to a ToolBus application.

A help author, creating a help system with the TBHelp Generator, has access to the different features of the TBHelp Generator by a graphical user interface, which is depicted in Figure 4.2.

The GUI has four menu's, through which the help author can access the different features of the TBHelp Generator:

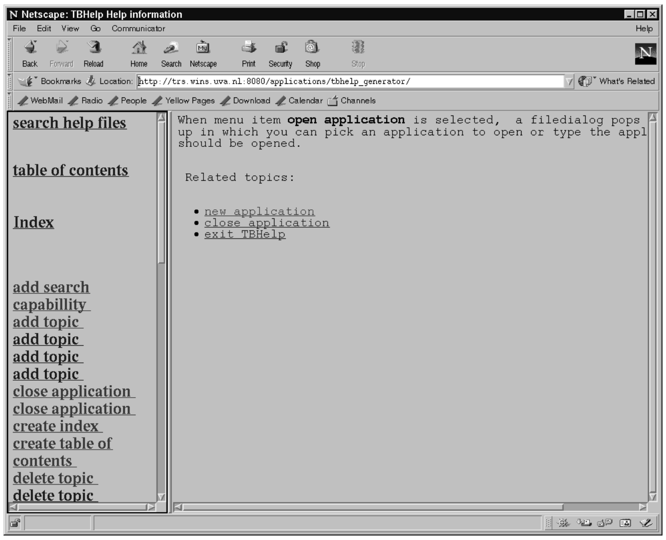

Figure 4.1: TBHelp Generator: An example help system.

Figure 4.2: TBHelp Generator: The Main GUI

- The *application menu*, provides the help author with capabilities for opening a help system, creating a new one, closing an opened help system and exiting the TBHelp Generator (Figure 4.3).

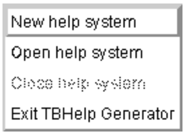

Figure 4.3: TBHelp Generator: The *Application* menu

- The *topic menu* lets the help author add, delete or edit help topics of a help system (Figure 4.4).

Figure 4.4: TBHelp Generator: The *Topic* menu

- The *tools menu* provides the help author with tools for generating an index, creating a table of contents, adding search capabilities and viewing the intermediate results while creating a help system (Figure 4.5).

Figure 4.5: TBHelp Generator: The *Tools* menu

- The *help menu* provides help and some additional information about the TBHelp Generator (Figure 4.6).

The Application Menu

When you select the *New help system* menu item from the Application menu, a file-dialog pops up in which the help author can enter the name and location for a new help system (Figure 4.7).

The location for the new help system is entered in the field labeled *Enter path or folder name* and the name of the new help system is entered in the field labeled *Enter file name*. When the help author selects the *Open help system* menu item, a similar file-dialog allows the help author to select an existing help system. Selecting the *Close help system* menu item, results in closing the currently opened help system and selecting menu item *Exit TBHelp Generator* closes the TBHelp Generator.

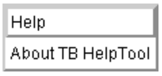

Figure 4.6: TBHelp Generator: The *Help* menu

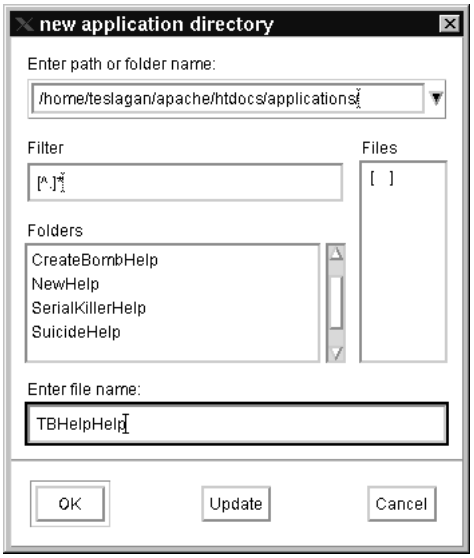

Figure 4.7: TBHelp Generator: File-Dialog.

The Topic Menu

Once an existing help system is selected or a new one is created, it is time to start adding help topics, and if necessary, editing or deleting help topics. Selecting any of the three menu items *Add Topic*, *Edit Topic* or *Delete Topic* results in a file-dialog, in which the help author can enter or select the name for a help topic. This file-dialog is similar to the ones used when creating a new help system or opening an existing one (see Figure 4.7).

The Tools Menu

If the help author selects the *Generate index* menu item from the *Tools* menu, an alphabetically ordered index of keywords is generated for the currently opened help system. This index makes it possible for end users of a help system to select help topics of interest by selecting keywords. Selecting menu item *Generate TOC* will assist the help author to create a table of contents for the currently opened help system. The window from Figure 4.8 will be displaying help topics, possibly previously added structure and newly added help topics.

Full-text search capabilities can be added to the currently opened help system, by selecting menu item *Add search*. The help author can preview his/her help system by selecting menu item *Preview*. A browser is opened, displaying the currently opened help system in it. An example was shown in Figure 4.1.

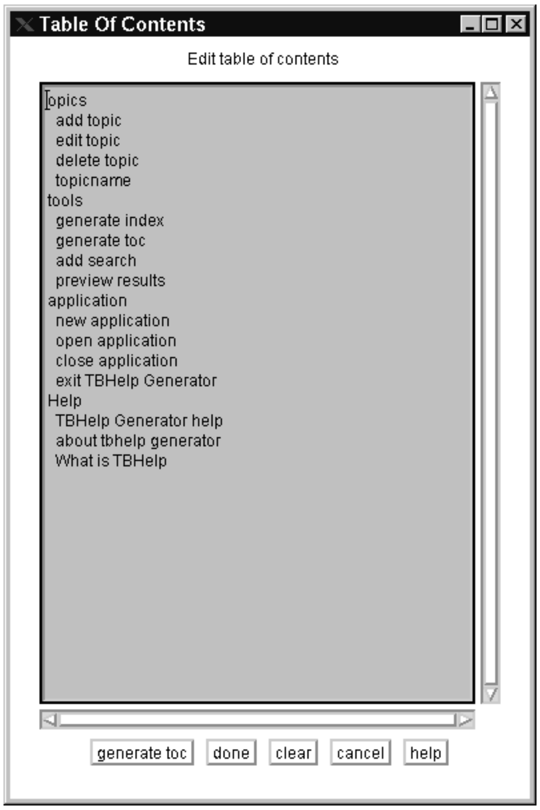

Figure 4.8: TBHelp Generator: Generate a table of contents.

The Help Menu

The *Help* menu item from the helpmenu, connects the TBHelp Generator to its help system. This help system has been created using the TBHelp Generator itself. The *About* menu item list some details about the TBHelp help authoring tool.

4.3 TBHelp Generator Component Architecture

The TBHelp Generator is a component based software application. It is made up of several components, each with its own functionality. The division of the TBHelp Generator into a set of separate components has several reasons [33].

- Components are independent, so implementation changes to one component can be made without affecting the others.
- Components add abstraction, hiding implementation details behind explicit interfaces, making it safe to change this implementation when needed.
- Components can be easily replaced by other components offering at least the same set of services.
- Components support reuse, so that new systems can be quickly built by assembling them from existing pieces.
- Component systems can be a mixture of self-made and bought components, making it possible to benefit from new developments in commercially available components.
- Components support customization, making it possible to produce a tailor-made component.

The component architecture of the TBHelp Generator is shown in Figure 4.9.

The seven components of the TBHelp Generator are listed below.

- *Main* provides, among other things, the graphical user interface for the TBHelp Generator. It provides the functionality for creating new help systems, opening existing help systems, adding help topics, editing help topics and deleting help topics (the menu items of the *Application* and *Topic* menus).
- *Indexer* generates an alphabetically ordered list of keywords (an index) for the currently opened help system.
- *Toc* assists the help author in creating a table of contents for the currently opened help system.
- *Search* adds *full-text* search capabilities to the currently opened help system.

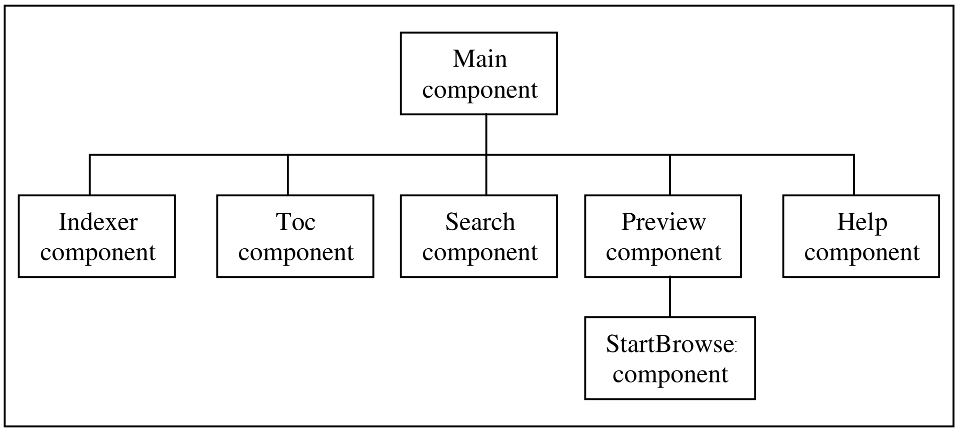

Figure 4.9: Component overview of the TBHelp Generator

- *Preview* checks whether the currently opened help system has an index, a table of contents and search capabilities. It then asks the user to enter the browser in which the help system should be displayed.
- *StartBrowser* displays the currently opened help system in the browser selected in the *Preview* component.
- *Help* connects the TBHelp Generator to a help system for the TBHelp Generator itself.

Inter component communication only takes place between the *Main* component and the other components. The *Preview* component also communicates with the *StartBrowser* component. Components *Indexer*, *Toc*, *Search*, *Preview* and *Help* do not communicate with each other. The following six sections describe the functionalities of the TBHelp Generator components.

4.4 The Main Component Functionality

The *Main* component provides the help author with the following functionalities:

- A graphical user interface, which serves as the help author's access to the features provided by the TBHelp Generator. This graphical user interface was previously shown in Figure 4.2. When a help author starts the TBHelp Generator, this graphical user interface is displayed.
- The functionalities for the menu items from the *Application* menu and *Topic* menu are also provided by the *Main* component (see Figures 4.3 and 4.4).

The Main GUI

When the help author selects the *New help system* menu item, a file-dialog pops up, in which the help author can enter a *location* and a *name* for the new help system (Figure 4.7). Pressing the *OK* button in the file-dialog results in the creation of a directory called *name* in directory *location*. If for some reason, no name for the new help system was entered, the message from Figure 4.10 is shown.

Figure 4.10: TBHelp Generator: No help system name was entered/selected.

Help information for the new help system will be stored in directory *location/name*. If an attempt is made to create an already existing help system, a message like the one from Figure 4.11 is displayed.

Figure 4.11: TBHelp Generator: Attempt to create an existing help system.

If the *Cancel* button is pressed, nothing happens and the message disappears. If the *Open* button is pressed, the existing help system is opened and ready to be updated.
Selecting the *Open help system* menu item also results in the display of a file-dialog that allows the help author to enter the name and location of an existing help system. Opening a help system means that name and location of the help system the help author wants to update, are saved. This way the TBHelp Generator knows where to update help information for a help system. If a non-existing help system is selected, the message from Figure 4.12 appears.

Figure 4.12: TBHelp Generator: Attempt to open a non-existing help system.

If the *Cancel* button is pressed, nothing happens and the message disappears. If the *Create* button is pressed, a directory with the given name at the given location is created.
Selecting the *Close help system* menu item notifies the TBHelp Generator that

the location and name of the currently opened help system no longer needs to be stored. Now the help author can open another help system, create a new one or simply exit the TBHelp Generator.
The *Tools* menu, *Topic* menu and *Close help system* menu item are disabled, while no help system is opened. When a help system is opened, menu items *New help system* and *Open help system* are disabled.
Exiting the TBHelp Generator is done by selecting the *close TBHelp* menu item. This results in closing the graphical user interface and any other message window that belongs to the TBHelp Generator. A possibly opened help system is closed in the process.

Adding, Editing and Deleting Help Topics.

The *Main* component also provides the functionality for adding new help topics, editing existing help topics and deleting help topics. When selecting one of the menu items *Add Topic*, *Edit Topic* or *Delete Topic*, a file-dialog appears in which the help author can enter the name for a help topic (Figure 4.13).

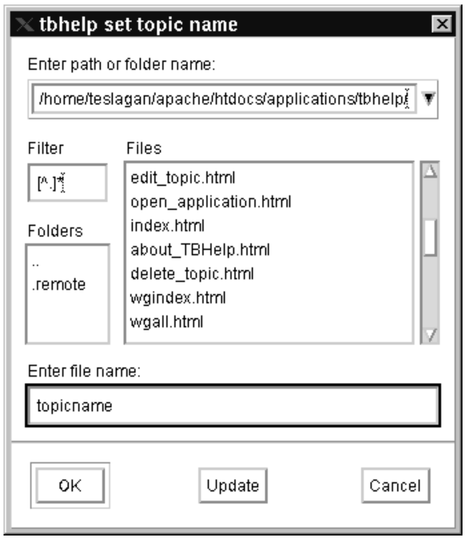

Figure 4.13: TBHelp Generator: Select or enter a topic name.

When no topic name is selected or entered, and the *OK* button is pressed, the file-dialog simply disappears.

To add a new topic to a help system, the help author selects the *Add Topic* menu item. Then the file-dialog from Figure 4.13 appears and a new topic name can be entered. After pressing the *OK* button in the file-dialog, the window in Figure 4.14 appears.

This window has a text field, containing the help topic name and two text areas. In the upper text area, the contents/description of the new help topic should be entered. The help author may choose to only enter regular text, but adding HTML tags is also permitted. Only HTML tags that are allowed between the <*body*> and </*body*> tags, should be used. The lower text area is meant for

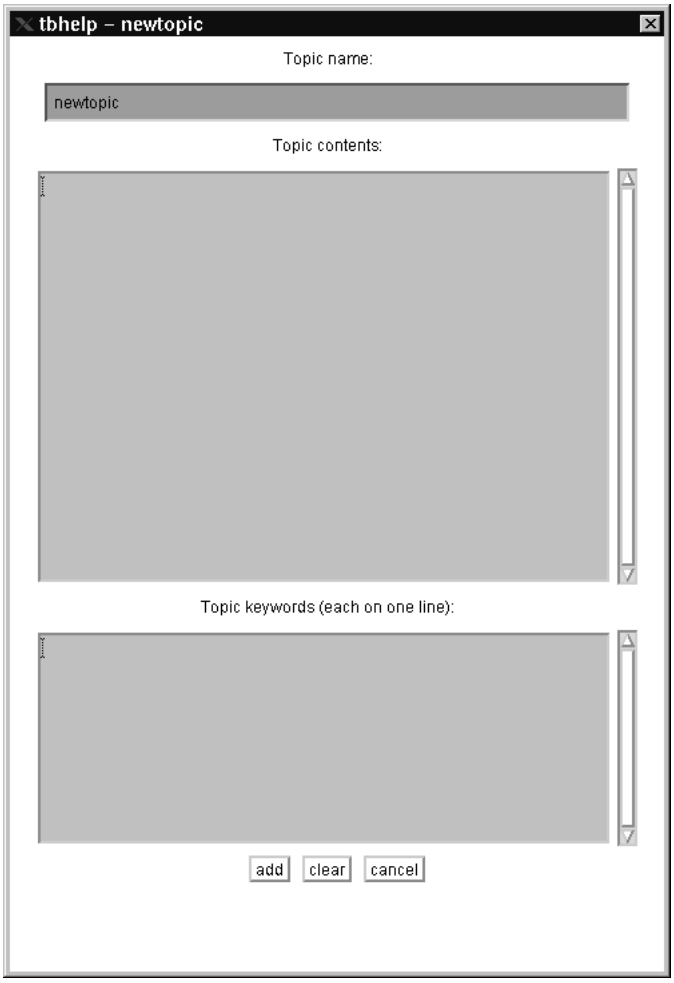

Figure 4.14: TBHelp Generator: Add a new topic.

keeping *keywords* about the topic. A keyword may exist of several words, but each keyword must be written on a separate line. Figure 4.15 shows an example.

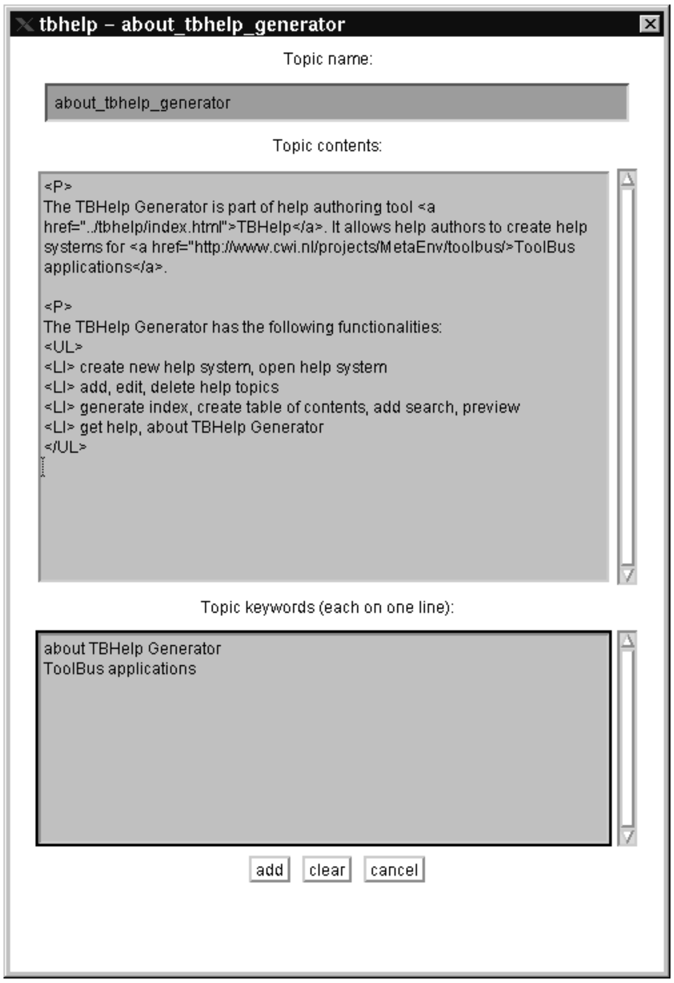

Figure 4.15: TBHelp Generator: Example contents for a topic.

Pressing the *cancel* button makes the window disappear, without creating the new help topic. Pressing the *clear* button, will clear the text field containing the topic name as well as the two text areas containing the topic contents and the topic keywords. When the *add* button is pressed, a HTML file *topicname.html* is created in the help system directory. The topic contents and keywords are stored in this html file. The obligatory HTML code is automatically generated, so help authors without any knowledge of HTML, do not have to be bothered with learning how to use HTML.

Editing a help topic is very similar to adding a new help topic. Once menu item *Edit Topic* is selected and a topic name has been selected or entered in the file-dialog, a window similar to the one in Figure 4.15 is displayed. Only the *add* button is now an *edit* button and the text areas for contents and keywords are filled with information filtered from file */location/name/topicname.html*.

Pressing the *cancel* button has no other consequences than making the window disappear. If the *clear* button is pressed, the text areas are cleared. Pressing the *edit* button regenerates file */location/name/topicname.html*, so it contains the edited topic contents and/or keywords.

If a topic should be deleted, the help author selects menu item *Delete Topic*. When the file-dialog from Figure 4.13 appears, the help author can enter or select the file *topicname.html* of the topic that should be deleted. Entering *topicname* without the *.html* extension, has the same effect. When the *OK* button in the file-dialog is pressed, the topic is deleted. Pressing the *Cancel* button in the file-dialog will make the file-dialog disappear, in which case the help topic will not be deleted.

4.5 The Indexer Component Functionality

When adding, editing or deleting topics, the help author assigns keywords to each of the help topics. The *Indexer* component extracts the keywords from the help topics. It then orders the keywords alphabetically after which it generates a HTML file called *keywords.html*, containing the sorted list of keywords. Each keyword is represented by a link to the help topic HTML file from which the keyword was extracted. While the *keywords.html* is being created, the message from Figure 4.16 is displayed. When the indexer is done, this message is replaced by the message from Figure 4.17. Pressing the *close* button makes the last message disappear.

Figure 4.16: TBHelp Generator: Creating index for a help system.

Figure 4.17: TBHelp Generator: Creating index for a help system is done.

If a new index needs to be created, because for example, help topics have been added or deleted, or their keywords have been edited, selecting the *Generate index* menu item will replace the existing index file with the new one.

4.6 The Toc Component Functionality

The *Toc* component is a flexible tool that aids the help author in creating a table of contents for a help system. The *Toc* component displays the graphical user interface from Figure 4.18, which we call the *toc window*. If no table of contents exists for the currently opened help system, the *Toc* component extracts the names of all help topics, after which they are displayed in the *toc window* (Figure 4.18).

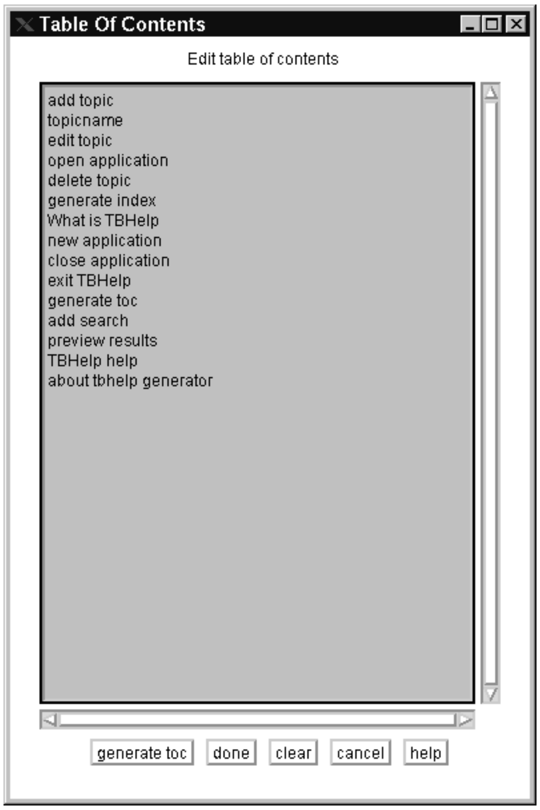

Figure 4.18: Creating a table of contents.

The help author can now add structure to the help topics and, if necessary, edit the list of help topics. Structure is added to the help topics by adding white spaces, changing topic names, adding topic names, removing topic names, changing the order of the topic names and adding collective names to group topics. If a table of contents has previously been generated for the currently opened help system, a HTML file called *toc.html* will exist, containing the table of contents. The *Toc* component will extract the topic names and the structure of the table of contents in *toc.html* and display them in the *toc window*. The *Toc* component will also check whether new help topics were added to the help system, after the table of contents was created. If this is the case, the names of the new help topics are added to the *toc window*. The help author can now edit the table of contents in any way he/she wishes. Pressing the *cancel* button will leave the table of contents unaltered and the *toc window* will disappear. If the help author presses the *clear* button, the text area of the *toc window* will be cleared and the help author can manually write the structure and contents for the table of contents. Pressing the *generate toc* button results in deleting file *toc.html*, after which a table of contents is regenerated containing the new structure and contents. The help author can edit and generate the table of contents as many times as needed, the *toc window* will not disappear until the *done* button or *cancel* button is pressed. When the *generate toc* button is pressed, each line in the *toc window* that matches the name of an existing help topic, will be transformed into a link to the matching help topic HTML file.

4.7 The Search Component Functionality

Webspider *Webglimpse* [34] is a manager for HTML files that should be searched. It adds search capabilities to a WWW site automatically, by adding small search boxes to the bottom of all HTML pages, or by creating a separate HTML page containing a search box. Webglimpse then allows search requests to cover the neighbourhood of a HTML page or the whole site. It is even possible for the search to efficiently cover pages of other sites, that are linked from the main site.

Webglimpse in combination with *glimpse* [34], can work to index and search a "corner" of the web, defined by the links you place on a starting page. Glimpse is the core search engine package and basis for webglimpse. Glimpse is a very powerful indexing and query system that allows the user to search through all his/her files very quickly. It can be used by individuals for their personal file systems as well as by organizations for large data collections. In case of the TBHelp Generator, glimpse allows users to search through the help files created by the TBHelp Generator.

The *Search* component uses glimpse and webglimpse to add full-text searchs capabilities to the currently opened help system. When the help author selects the *Add search* menu item, the *Search* component executes a series of steps that results in the search capabilities being added to the opened help system. The different steps needed to add search capabilities to a help system, are listed below.

- During the process of adding search capabilities to a help system, glimpse en Webglimpse each generate a set of configuration files. When the help system has been changed (e.g. by adding new help topics), the process for adding search capabilities must be repeated. In the first step of the *Search* component we remove these configuration files. While the old configuration files are being removed, the message from Figure 4.19 is displayed.

Figure 4.19: TBHelp Generator: Removing old files.

- In the next step, the *Search* component generates a *glimpse exclude file*. This file contains information about files that shouldn't be indexed by the search engine. While this step is being executed, the message from Figure 4.20 is displayed.

- The *Search* component also generates a *glimpse include file*, which contains information on what files should be indexed by search engine glimpse. During this step, the message from Figure 4.21 is shown.

Figure 4.20: TBHelp Generator: Generate exclude file.

Figure 4.21: TBHelp Generator: Generate include file.

- Now a configuration file is created for webglimpse. The message shown during this step is displayed in Figure 4.22.

Figure 4.22: TBHelp Generator: Configure archive.

- Finally, the *Search* component indexes the files denoted in the *glimpse include file*, leaving out the files denoted in the *glimpse exclude file*. Furthermore, a HTML file is generated, containing the search form with which users can issue search requests. The Figure 4.23 shows the message displayed during this step.

When the *Search* component is done with executing the above steps, it displays the message from Figure 4.24. Pressing the *close* button makes this message disappear and the help author can now issue search requests on the help information from the opened help system.

The search form needed to issue these search requests, is stored in file *wgindex.html* and it looks like the one from Figure 4.25.

If help topics have been added, deleted or edited, the search capabilities can be updated be selecting menu item *Add search* again.

4.8 The Preview Component Functionality

Component *Preview* checks the status of a help system. It will report a missing index, table of contents or search capabilities. The *Preview* component is represented by menu item *Preview*.

The *Preview* component first checks whether the help author has already generated an index of keywords for the opened help system, by checking whether file *keywords.html* exists. If not, the message from Figure 4.26 is displayed.

Then the *Preview* component checks whether the help system contains a table of contents, by checking whether file *toc.html* exists. If not, the message from Figure 4.27 is displayed.

Figure 4.23: TBHelp Generator: Index help topic files.

Figure 4.24: TBHelp Generator: Adding search done.

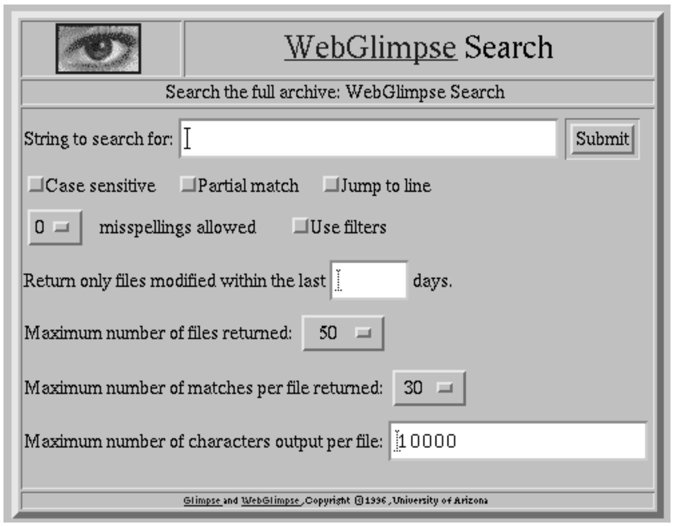

Figure 4.25: TBHelp Generator: Form for issuing search requests.

Figure 4.26: TBHelp Generator: No index was generated yet.

Figure 4.27: TBHelp Generator: No table of contents was generated yet.

The *Preview* component also checks whether the help author has added search capabilities to the opened help system, by checking whether file *wgindex.html* exists. If not, the message from Figure 4.28 is displayed.

Figure 4.28: TBHelp Generator: No search capabilities were added yet.

When an index and a table of contents are created, and search capabilities have been added, the *Preview* component generates file *index.html*. This file contains the frame layout for displaying a help system created by the TBHelp Generator. A titlepage name *mainpage.html* is also created for the help system. Once these files have been generated, the help system is ready to be previewed and the *StartBrowser* component comes into action.

4.9 The StartBrowser Component Functionality

The *StartBrowser* component opens a help system in a browser, after component *Preview* has checked the help system. The help author is first asked to enter a browser location and name (Figure 4.29).

Figure 4.29: TBHelp Generator: Select browser that should be used.

The *StartBrowser* component then opens the specified browser with the help system. The first time the *StartBrowser* component receives a request to open a browser, it starts a new browser. Future requests to display a help system will be serviced by an opened browser.

An example of what a help system looks like once opened in a browsers, is shown in Figures 4.1 and 4.30. Figure 4.1 displays the index of the help system in the left pane and a help topic in the right pane. Figure 4.30 displays the table of contents in the left pane and the form in which search requests can be entered in the right pane. The left frame displays the table of contents. At the top, links to the index and the search form are added. The right frame displays the title of the help system. When clicking on the index link in the left frame, the index will be displayed in the right frame. Clicking the search link, will open the search form from Figure 4.25 in the right frame.

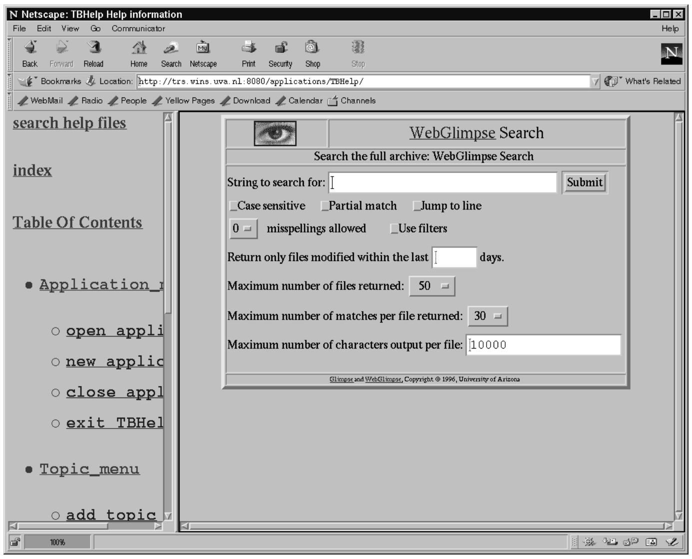

Figure 4.30: An example help system.

4.10 The Help Component Functionality

The *Help* component opens the help system for the TBHelp Generator in a browser. This help system was created with the TBHelp Generator itself, so it resembles the one in Figure 4.1. The help system for the TBHelp Generator contains an index of keywords, a table of contents and search capabilities, so the help author can efficiently find relevant help information about working with the TBHelp Generator.

The first time a help author selects the *Preview* menu item or the *Help* menu item, a new browser is started. After that, selecting any of the two menu items will use an opened browser.

The *Help* component services two kinds of help requests. The first one is a *global* request, which opens the title page of a help system in a browser. We call the second kind *parameterized* requests, because then the help system is searched for help information about a set of specified help topics. A browser will be opened with the results of the search.

Chapter 5

Implementation of the TBHelp Generator

This chapter discusses the implementation details of the TBHelp Generator on top of the ToolBus software architecture. In Section 5.1 we briefly describe the ToolBus software architecture. More information on the ToolBus architecture can be found in [2]. In Section 5.2 we describe the ToolBus architecture of the TBHelp Generator. This section defines the seven software components from which the TBHelp Generator exists. The implementation of the software components are each discussed in Sections 5.3 upto 5.9. Finally, Section 5.10 describes the TBHelp Help Format, which differs from other HTML-based help formats described in 2. The TBHelp Help Format complies with the basic requirements from Section 2.6.

5.1 The ToolBus Architecture

The ToolBus utilizes a scripting language based on process algebra to describe the communication between different software components. A ToolBus script describes a number of processes that can communicate with each other and with software components living outside the ToolBus. This section globally describes the ToolBus and its architecture.

Software applications grow larger because the complexity of the tasks we want to automate increases. The applications become heterogeneous because large systems may be constructed by re-using existing software components. It is more than likely that these components have been developed using different implementation languages and they may run on different hardware platforms. Systems become distributed because they have to operate in the context of local area networks. To successfully develop an application, communication between its components should be correct. The question of how to connect a number of independent, interactive components and integrate them into a well-defined, cooperating system is important for two reasons. New systems can be built at lower cost by connecting already existing components. The second reason is that decomposition of a single monolithic application into a number of components improves and simplifies the maintenance of the application. Adding functionality can be easily done by adding components and upgrading functionality can be

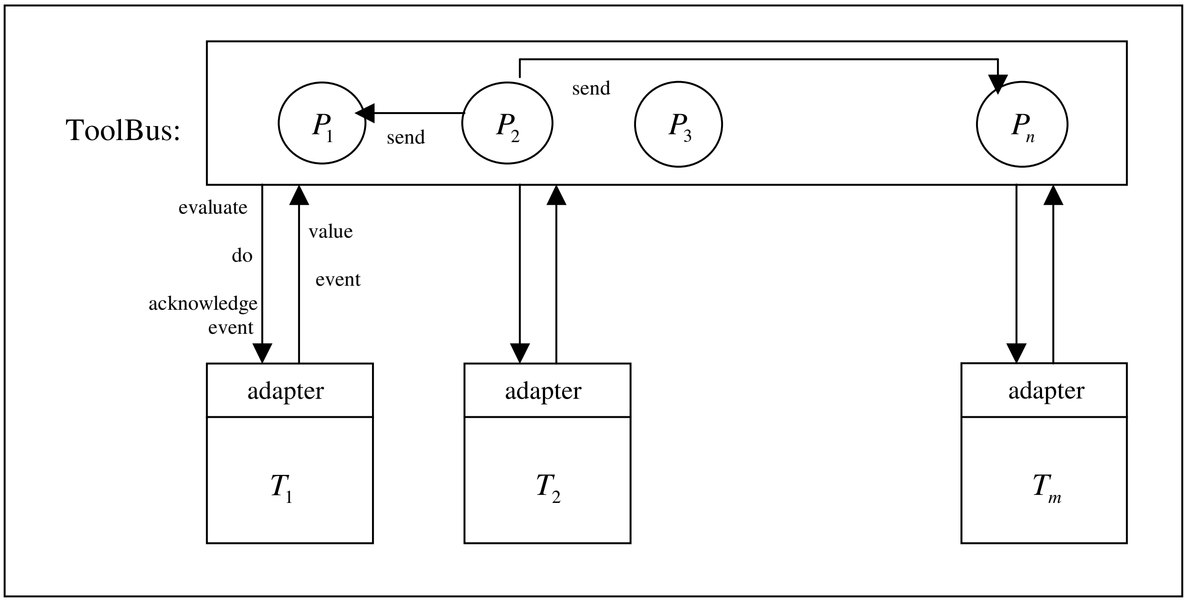

Figure 5.1: The ToolBus Software Architecture

done by replacing one component with another [2][3]. The ToolBus is designed to solve problems described in this paragraph.

The ToolBus is a component interconnection architecture that resembles a hardware communication bus. It controls the interactions between different components of a software application by forbidding direct inter-component communication. Instead, all interconnections are controlled by a process-oriented *script* that is based on process algebra. This script formalizes all the desired interactions among the components. Exchanging (complex) data values among software components that may be implemented in different programming languages is referred to as the *data integration problem.* A single, uniform, data representation called *Annotated Terms* is used [4]. A language-dependent adapter that translates between the internal ToolBus data format and the data format used by the individual software components of an application, makes it possible to write every software component in the language best suited for the task(s) it has to perform.

Annotated Terms form an open, efficient, concise and language independent solution to the data integration problem in distributed applications. It is considered open because of its independence of any specific hardware or software platform. It is simple because the procedural interface contains only about ten functions and efficient because the storage capacity needed for the data structures is small and data sharing is exploited. Data transmissions between components is fast because a compressed representation together with fast encoding and decoding is used. Data transmission preserves any sharing in the data structures. The Annotated Terms solution is also language independent because data structures can be created and manipulated in any suitable language. Annotations in the Annotated Term solution can be used to transparently extend the main data structures to represent non-structural information [2][5].

The global architecture of the ToolBus is shown in Figure 5.1. The ToolBus serves the purpose of defining the cooperation of a variable number of components T_i (i = 1,..,m) that are to be combined into a complete software application. The internal behaviour or implementation of each component is irrelevant: they may be implemented in different programming languages, be generated from specifications, etc. In general an adapter is needed to adapt a component to the common data representation and message protocols imposed by the ToolBus.

The ToolBus itself consists of a variable number of ToolBus processes P_i (i = 1,..,n). With ToolBus processes, activities inside the ToolBus are meant (not for instance operating level processes). The parallel composition of the processes P_i represents the intended behavior of the whole system. Components are external computation activities, most likely corresponding with operating system level processes. They come into existence either by an execution command issued by the ToolBus or their execution is initiated externally, in which case an explicit connection command has to be performed by the ToolBus. Although a one-to-one correspondence between components and processes seems simple and desirable, this is not enforced and components controlled by more than one process as well as clusters of components controlled by one process, are permitted.

ToolBus processes can communicate in two ways. A ToolBus process can send a message which should be synchronously received by another ToolBus process. These messages are intended to request a service from another process. A second communication mechanism allows a ToolBus process to send a note which is broadcasted to other ToolBus processes. ToolBus processes only receive notes they have subscribed to. The sending of notes amounts to asynchronous selective broadcasting.

Communication between ToolBus and components is based on the handshaking protocol. A ToolBus process may send an *evaluation message*, *do* message or *acknowledge event* message to a component. A component should react to an evaluation message by sending a value message to the ToolBus process and when a component sends an event to a ToolBus process, the component expects an acknowledge event message from the ToolBus process.

5.2 TBHelp Generator ToolBus Architecture and Communication

The TBHelp Generator is a *ToolBus application*, which means it is build on the ToolBus software architecture. In this section we describe the ToolBus architecture of the TBHelp Generator and the communication between the different components.

Figure 5.2 displays the ToolBus architecture of the TBHelp Generator components.

As explained in Section 5.1, ToolBus applications utilize a scripting language based on process algebra to describe the communication between different software components. Communication in the TBHelp Generator is described by a script that consists of seven processes. These processes communicate with each other and with the different software components that make up the TBHelp

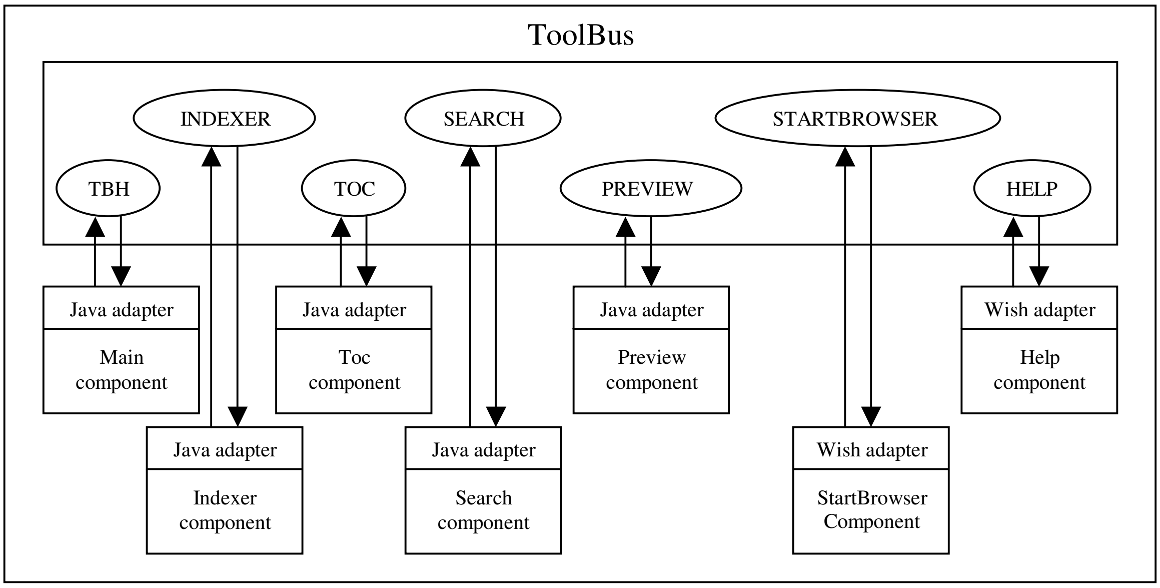

Figure 5.2: TBHelp Generator ToolBus Architecture

Generator. The seven processes are listed and explained next.

We start with process *TBH* in Listing 5.1, which is the ToolBus process that takes care of the communication between the *Main* component and other ToolBus processes.

Listing 5.1: TBH process

```
process TBH is
  let
    Tid : tbhelp,
    N : int,
    Path : str,
    Name : str,
    Loc : str,
    Keys : list
  in
    execute(tbhelp,Tid?) .
    (
      rec-event(Tid, genIndex(N?, Path?)) .
      snd-msg(index(Path)) .
      rec-msg(index(done)) .
      snd-ack-event(Tid, genIndex(N, Path))
      +
      rec-event(Tid, genToc(N?, Path?)) .
      snd-msg(toc(Path)) .
      rec-msg(toc(done)) .
      snd-ack-event(Tid, genToc(N, Path))
      +
      rec-event(Tid, addSearch(N?,Path?)) .
      snd-msg(addSearch(Path)) .
      rec-msg(addSearch(done)) .
      snd-ack-event(Tid, addSearch(N, Path))
      +
```

```
27         rec-event(Tid, preview(N?, Path?)) .
28         snd-msg(preview(Path)) .
29         rec-msg(preview(done)) .
30         snd-ack-event(Tid, preview(N, Path))
31         +
32         rec-event(Tid, help(N?, Keys?)) .
33         snd-msg(help(Keys)) .
34         rec-msg(help(done)) .
35         snd-ack-event(Tid, help(N, Keys))
36         +
37         rec-event(Tid, help) .
38         snd-msg(help) .
39         rec-msg(help(done)) .
40         snd-ack-event(Tid, help)
41         +
42         rec-event(Tid, terminateApp(N?)) .
43         snd-note(terminateApp) .
44         snd-terminate(Tid, "TBHelp Generator") .
45         shutdown("terminating toolbus").
46         snd-ack-event(Tid, terminateApp(N))
47       )* delta
48     endlet
49
50 tool tbhelp is {command="java-adapter -class TBHelpGenerator"}
```

Process TBH waits for any of the seven following events, generated by the *Main* component.

- Event *genIndex(Path)* which is a request from the *Main* component to generate an index for the help system located at *Path*. When process TBH receives this request, it sends message *index(Path)* to process *INDEXER*, to handle the request. Process TBH then waits until it receives a message *index(done)* and an acknowledge event is sent back to the *Main* component, to denote that the genIndex request is being handled (lines 12-15).

- Event *genToc(Path)* which is a request from the *Main* component to create a table of contents for the help system located at *Path*. When process TBH receives this request, it sends message *toc(Path)* to process *TOC*, to handle the request. Process TBH then waits until it receives a message *toc(done)* and an acknowledge event is sent back to the *Main* component, to denote that the genToc request is being handled (lines 17-20).

- Event *addSearch(Path)* which is a request from the *Main* component to add search capabilities to the help system located at *Path*. When process TBH receives this request, it sends message *addSearch(Path)* to process *ADDSEARCH*, to handle the request. Process TBH then waits until it receives a message *addSearch(done)* and an acknowledge event is sent back to the *Main* component, to denote that the addSearch request is being handled (lines 22-25).

- Event *preview(Path)* which is a request from the *Main* component to open the help system indicated by *Path* in a browser, to check whether the help system works correctly. When process TBH receives this request, it sends message *preview(Path)* to process *PREVIEW*, to handle the request. Process TBH then waits until it receives a message *preview(done)* and an acknowledge event is sent back to the *Main* component, to denote that the viewer request is being handled (lines 27-30).

- Event *help* which is a request from the *Main* component to display the help system for the TBHelp Generator, in a browser. When process TBH receives this request, it sends message *help* to process *HELP* to handle the request. Process TBH then waits until it receives message *help(done)* and an acknowledge event is sent back to the *Main* component, to denote that the help request is being handled (lines 32-35).
- Event *help(Keys)* which is a request from the *Main* components to display selected help information from the help system for the TBHelp Generator. This help information should concern the topics in list *Keys*. When process TBH receives this request, it sends a message to process *HELP* to handle the request. Process TBH then waits until it receives a message *help(done)* and an acknowledge event is sent back to the *Main* component to denote that the help request is being handled (lines 37-40).
- Event *terminateApp* which is a request from the *Main* component, to terminate the TBHelp Generator. When process TBH receives this request, it sends a note to all other processes to terminate themselves. The *Main* component then terminates itself and shuts down the ToolBus (lines 42-46).

The *Main* component, which communicates with process TBH, is described in more detail in Section 5.3. We continue with the description and explanation of process *INDEXER*, which is listed next.

Listing 5.2: INDEXER process

```
process INDEXER is
  let
    Ind : indexer,
    Path : str,
    S : str
  in
  subscribe(terminate(<str>)) .
  execute(indexer, Ind?) .
  (
    rec-msg(index(Path?)) .
    snd-do(Ind, indexer(Path)) .
    snd-msg(index(done))
    +
    rec-note(terminateApp) .
    snd-terminate(Ind, "Indexer")
  )* delta
  endlet

tool indexer is {command="java-adapter -class Indexer"}
```

ToolBus process *INDEXER* takes care of the communication between the Indexer component and process TBH. It waits for a message *index(Path)* from process TBH, where *Path* is the location of a help system. After receiving this message, process INDEXER sends a request to the Indexer component, to generate an index for the help system at location *Path*. Finally, process INDEXER sends message *indexer(done)* to process TBH, denoting that the indexer request is being handled (lines 10-12). When process INDEXER receives a termination note from process TBH, it sends a terminate request to the Indexer component, after which this component terminates itself (lines 14-15).

The Indexer component, which communicates with process INDEXER, is described in Section 5.4. The next process to be discussed is process *TOC*, which is shown in Listing 5.3.

Listing 5.3: TOC process

```
1  process TOC is
2    let
3      Tid : toc,
4      Path : str,
5      N : int,
6      Keys : list
7    in
8    subscribe(terminate(<str>)) .
9    execute(toc, Tid?) .
10   (
11     rec-msg(toc(Path?)) .
12     snd-do(Tid, toc(Path)) .
13     snd-msg(toc(done))
14     +
15     rec-event(Tid, help(N?, Keys?)) .
16     snd-msg(help(Keys)) .
17     rec-msg(help(done)) .
18     snd-ack-event(Tid, help(N, Keys))
19     +
20     rec-note(terminateApp) .
21     snd-terminate(Tid, "Toc")
22   )* delta
23   endlet
24
25 tool toc is {command="java-adapter -class Toc"}
```

ToolBus process *TOC* takes care of the communication between component Toc and process TBH. Process TOC waits for message *toc(Path)* from process TBH, where *Path* denotes the location of a help system. After receiving the message, process TOC sends a request to the Toc component, to create a table of contents for the help system at location *Path*. Process TOC then sends a message *toc(done)* back to the *Main* component, denoting that the toc request is being handled (lines 9-11). When process TOC receives a termination note from process TBH, it sends a termination request to the Toc component, after which the component terminates itself (lines 13-14).

The Toc component, which communicates with process TOC, is described in more detail in Section 5.5. Process *ADDSEARCH* is shown in Listing 5.4 and discussed next.

Listing 5.4: SEARCH process

```
process ADDSEARCH is
  let
    Aid : addsearch,
    Path : str
```

```
5   in
6     subscribe(terminate(<str>)) .
7     execute(addsearch, Aid?) .
8     (
9       rec-msg(addSearch(Path?)) .
10      snd-do(Aid, addSearch(Path)) .
11      snd-msg(addSearch(done))
12      +
13      rec-note(terminateApp) .
14      snd-terminate(Aid,"Add search")
15    )*delta
16  endlet
17
18 tool addsearch is {command="java-adapter -class AddSearch"}
```

ToolBus process *SEARCH* takes care of the communication between component Search and process TBH. It waits for message *addSearch(Path)* from process TBH, where *Path* denotes the location of a help system. When process SEARCH receives this message, it sends a request to the Search component, to add search capabilities to the help system located at *Path*. Process SEARCH then sends a message *addSearch(done)* back to process TBH, denoting that the add search request is being handled (lines 9-11). When process SEARCH receives a termination note from process TBH, it sends a termination request to the Search component, after which the component terminates itself (lines 13-14).

The Search component, which communicates with process SEARCH, is described in more detail in Section 5.6. Process PREVIEW is shown in the next listing.

Listing 5.5: PREVIEW process

```
process PREVIEW is
  let
    Vid : preview,
    Path : str
  in
  subscribe(terminate(<str>)) .
  execute(preview, Vid?) .
  (
    rec-msg(preview(Path?)) .
    snd-eval(Vid, preview(Path)) .
    (
      (
        rec-value(Vid,"noindex")
        +
        rec-value(Vid,"notoc")
        +
        rec-value(Vid,"nosearch")
      )
      +
      (
        rec-value(Vid,"ok") .
        snd-msg(startBrowser(Path))
      )
    ) .
    snd-msg(preview(done))
    +
    rec-note(teminateApp) .
    snd-terminate(Vid, "Preview")
```

```
29    )* delta
30    endlet
31
32 tool preview is {command="java-adapter -class Preview"}
```

Process PREVIEW takes care of the communication between process TBH and the Preview component. Process PREVIEW waits on a message *preview(Path)* from process TBH. When process PREVIEW receives this message, it sends an evaluation request to the Preview component, to check whether an index was generated, a table of contents was created and search capabilities were added to the help system at location *Path* (lines 10-11). If the Preview component sends one of the values *noindex*, *notoc* or *nosearch* back to process PREVIEW (lines 14-18), then message *preview(done)* is sent to process TBH, denoting that the preview request has been handled (line 20). If the Preview component sends a message *ok* back to process PREVIEW, message *startBrowser(Path)* is sent to process STARTBROWSER, to display the help system at location *Path* in a browser (lines 22-23). Process PREVIEW then sends message *preview(done)* to process TBH, to denote that the preview request is being handled (line 20). When process PREVIEW receives a termination note from process TBH, it sends a termination request to the Preview component, after which the component terminates itself (lines 28-29).

The Preview component, which communicates with process PREVIEW, is described in more detail in Section 5.7. Listing 5.6 shows the STARTBROWSER process.

Listing 5.6: STARTBROWSER process

```
1  process STARTBROWSER is
2    let
3      Sid : startbrowser,
4      Loc : str
5    in
6      subscribe(terminate(<str>)) .
7      execute(startbrowser, Sid?) .
8      (
9        rec-msg(startBrowser(Loc?)) .
10       snd-do(Sid, startBrowser(Loc))
11       +
12       rec-note(terminateApp) .
13       snd-terminate(Sid, "Browser")
14     )* delta
15     endlet
16
17 tool startbrowser is {command = "wish-adapter -wish wish8.3 \
18                                 -script startbrowser.tcl"}
```

Process STARTBROWSER takes care of the communication between the StartBrowser component and process PREVIEW. Process STARTBROWSER waits for message *startBrowser(Path)* from process PREVIEW. Process START-BROWSER then send a request to the StartBrowser component to let the user enter the location and name for a browser (lines 9-10), after which the help system at location *Path* is displayed in this browser. When process START-BROWSER receives a termination note from process TBH, it sends a termination request to the *StartBrowser* component, after which the component terminates itself (lines 12-13). If the *StartBrowser* component started a browser, this browser is also closed.

The StartBrowser component, which communicates with the STARTBROWSER process, is described in more detail in Section 5.8. Process HELP is shown in Listing 5.7.

Listing 5.7: HELP process

```
1  process HELP is
2    let Hid : help,
3        AppID: str,
4        Keywords: list
5    in
6      subscribe(terminate(<str>)) .
7      execute(help, Hid?).
8      (
9        (
10         rec-msg(help) .
11         snd-do(Hid, showHelpApplication)
12         +
13         rec-msg(help(Keywords?)) .
14         snd-do(Hid, showHelpKeywords(Keywords))
15       ) . snd-msg(help(done))
16       +
17       (
18         rec-note(terminateApp)
19         +
20         rec-msg(terminateApp)
21       ) . snd-terminate(Hid, "Help")
22
23     ) * delta
24   endlet
25
26 tool help is {command = "wish-adapter -wish wish8.3 \
27                          -script helpviewer.tcl"}
```

Process HELP takes care of the communication between process TBH and the Help component. When process HELP receives message *help*, it sends a request to the Help component, to start a browser with a help system that is specified in the Help component itself. This is possible because a Help component is generated for each help system separately (See Chapter 6). Process HELP can also receive a message *help(Keys)*, containing a list of keywords. When this message is received, process HELP sends a request to the Help component to select information about the keywords from the specified help system, and open the selected help information in a browser. The Help component, which communicates with the HELP process, is described in more detail in Section 5.9. When process HELP receives a termination note from process TBH, it sends a termination request to the *Help* component, after which the component terminates itself (lines 12-13).

5.3 The Main Component

The *Main* component is a Java program, saved in file *TBHelp.java*. This section describes the methods incorporated in class TBHelpTool, together with their functionalities.

- Constructor method *public TBHelpTool (String[] args)*. The constructor method initializes the GUI for the *Main* component (Figure 4.2). It also initializes a dialog, which will be used to display messages for users.

- Method *static public void main (String[] args)* connects the *Main* component to the ToolBus.

- Method *void recTerminate (ATerm term)* terminates the *Main* component when a terminate message is received from ToolBus process TBH.

- Method *void recAckEvent (ATerm term)* handles acknowledge events sent by process TBH.

- Method *public void actionPerformed (ActionEvent event)* handles the events generated by the *Main* component GUI. It also handles events generated by users interacting with dialog windows. If the help author selects one of the menu items *Indexer*, *Toc*, *Add search*, *Preview*, *Help* or *Exit TB-Help Generator*, a message is sent to process TBH to handle respectively, an indexer request, toc request, add search request preview request, or termination request.

- Method *public void AddTopic ()* is used to create a new help topic for the currently opened help system.

- Method *public void showTopicInterface ()* displays a GUI which allows the help author to add contents for a new help topic or edit the contents of an existing help topic. This GUI was shown in Figure 4.14.

- Method *public void CreateTopic ()* creates a HTML file named *<topicname>-.html*, containing the help topic contents and keywords, and the correct HTML code to open the help topic in a browser.

- Method *public void EditTopic ()* allows the help author to edit the contents and keywords of a previously created topic.

- Method *public void GetKeyword (File file)* extracts a list of keywords from a help topic HTML file, that was generated by method *public void CreateTopic ()*.

- Method *public void GetContents ()* extracts the contents of a help topic, from a HTML file that was generated by method *public void CreateTopic ()*.

- Method *public void DeleteTopic ()* allows the help author to delete an existing help topic file by deleting the HTML file for the help topic.

- Method *public String GetTopicName ()* returns the topic name of the currently opened help topic.

- Method *public void SetTopicName ()* opens the currently relevant help topic. This is done by displaying a file-dialog in which th help author can enter the help topic name. An example file-dialog was shown in Figure 4.7.

- Method *public void SetAppNewName ()* displays a file-dialog in which the help author can type the name for a new help system. An example file-dialog was shown in Figure 4.7.

- Method *public void SetOpenAppName ()* displays a file-dialog in which the help author can open a help system by selecting or typing a help system name. An example file-dialog was shown in Figure 4.7.

- Method *public String GetAppName ()* returns the name of the currently opened help system.

- Method *public void DialogMessage (String msg, int width, int height)* displays a dialog for the help author, with *size(width,height)*, a message *message* and a button labeled *close*, to close the dialog. An example was shown in Figure 4.10.

5.4 The Indexer Component

The Indexer component is a Java program, saved in file *Indexer.java*. This section describes the methods incorporated in class Indexer, together with their functionalities.

- Constructor method *public Indexer (String[] args)*. The constructor method initializes a window that will be used by the Indexer component to display messages to the help author.

- Method *static public void main (String[] args)* connects the Indexer component to the ToolBus.

- Method *void recTerminate (ATerm term)* terminates the Indexer component when a terminate message is received from ToolBus process INDEXER.

- Method *void indexer (String applicationPath)* is used to generate an alphabetically ordered list of keywords for the help system denoted by *applicationPath*.

- Method *public void MakeLinks (String keywords[], String applicationPath)* is creates a file named *keywords.html* for the help system denoted by *applicationPath*. This HTML file contains the alphabetically ordered list of keywords, generated by method *public void MakeLinks (String keywords[], String applicationPath)*. Each keyword is transformed into a link to the help topic file from which it was extracted.

- Method *public void DialogMessage (String msg, int width, int height)* displays a frame for the help author, with *size(width,height)*, a message *message* and a button labeled *close*, to close the frame. An example was shown in Figure 4.10.

5.5 The Table of Contents Component

The Toc component is a Java program, saved in file *Toc.java*. This section describes the methods incorporated in class Toc, together with their functionalities.

- Constructor method *public Toc (String[] args)* initializes the GUI which allows the help author to create a table of contents for a help system. It also initializes a dialog, which will be used to display messages for the help author.

- Method *static public void main (String[] args)* connects the Toc component to the ToolBus.

- Method *public void actionPerformed (ActionEvent event)* handles the events generated by the Toc GUI that was initialized in the constructor method. It also handles events generated by help authors interacting with dialog windows.

- Method *void recTerminate (ATerm term)* terminates the Toc component when a terminate message is received from ToolBus process TOC.

- Method *void toc (String applicationPath)* displays a frame in which the help author can edit the table of contents for the application denoted by *applicationPath*.

- Method *public void SetTocText ()* checks whether a table of contents already exist for the currently opened help system. If it does exist, this method extracts the old table of contents from the HTML file *toc.html*, and loads it into the frame that was displayed by method *void toc (String applicationPath)*. If file *toc.html* doesn't exist yet, this method extracts all topic names for the help system and loads this into the frame that was displayed by method *void toc (String applicationPath)*.

- Method *public void GenerateTocFile ()* generates file *toc.html* which contains the table of contents for the currently opened help system. Every entry of the table of contents is transformed into a link to the help topic file it represents.

- Method *String AddSpaces (String topic, int i)* returns string topic after adding *i* white spaces to the beginning of the string.

- Method *public void DialogMessage (String msg, int width, int height)* displays a frame for the help author, with *size(width,height)*, a message *message* and a button labeled *close*, to close the frame. An example was shown in Figure 4.10.

5.6 The Search Component

The Search component is a Java program, saved in file *AddSearch.java*. This section describes the methods incorporated in class AddSearch, together with their functionalities.

- Constructor method *public AddSearch (String[] args)*. It initializes a dialog, which will be used to display messages to the help author.
- Method *static public void main (String[] args)* connects the AddSearch component to the ToolBus.
- Method *public void actionPerformed (ActionEvent event)* handles the events generated by help authors interacting with dialog windows.
- Method *void recTerminate (ATerm term)* terminates the AddSearch component when a terminate message is received from ToolBus process ADD-SEARCH.
- Method *void addSearch (String application_path)* removes files that might have been generated during a previous attempt to add search capabilities to a help system. It then generates a glimpse include and a glimpse exclude file, after which it archives and indexes the help topics in the help system.
- Method *public void RemoveOldFiles ()* deletes any files that might have been generated during a previous attempt to add search capabilities.
- Method *public void Confarc ()* creates an archive, which will be used by webglimpse, to service search requests.
- Method *public void WGReindex ()* takes care of indexing the help topic files of a help system and creates search boxes in which the user can enter search requests.
- Method *public void GenIncludeFile ()* generates a glimpse include file, which is used by glimpseindex to select the help topics that should be indexed.
- Method *public void GenExcludeFile ()* generates a glimpse exclude file, which is used by glimpseindex to select the help topics that should be indexed.
- Method *public void DialogMessage (String msg, int width, int height)* displays a frame for the help author, with *size(width,height)*, a message *message* and a button labeled *close*, to close the frame. An example was shown in Figure 4.10.

5.7 The Preview Component

The Preview component is a Java program, saved in file *Preview.java*. This section describes the methods incorporated in class Preview, together with their functionalities.

- Constructor method *public Preview (String[] args)* initializes a dialog, which will be used to display messages to the help author.
- Method *static public void main (String[] args)* connects the Preview component to the ToolBus.

- Method *public void actionPerformed (ActionEvent event)* handles the events generated by help authors interacting with dialog windows.

- Method *void recTerminate (ATerm term)* terminates the Preview component when a terminate message is received from ToolBus process PREVIEW.

- Method *ATerm preview (String application_path)* check whether an index exists for the help system located at *application_path* by checking whether the help system contains a file *keywords.html*. If not, it displays a message, telling the help author that an index should be generated first. Otherwise the Preview component continues with checking for a table of contents in the help system, by checking whether file *toc.html* exists. If the file does not exist, a proper message is displayed, telling the user that a table of contents should be generated first. Otherwise the Preview component checks whether search capabilities were added to the help system at location *application_path*, by checking whether the help system contains a search box. This is done by checking whether field *wgindex.html* exists. If the help system doesn't contain search capabilities, a warning message is displayed, telling the help author that no search requests can be serviced. In the mean while, a message is sent to ToolBus process PREVIEW, with the request to open the help system located at *application_path*, in a browser.

- Method *public void GenerateFrames ()* generates file *index.html* for a help system, containing the frame structure for displaying the help system in a browser.

- Method *public void GenerateMainPage ()* generates file *mainpage.html*, which serves as a title page for the help system.

- Method *public void DialogMessage (String msg, int width, int height)* displays a frame for the help author, with *size(width,height)*, a message *message* and a button labeled *close*, to close the frame. An example was shown in Figure 4.10.

5.8 The StartBrowser Component

The StartBrowser component is a tcl script, saved in file *startbrowser.tcl*. This section describes the methods incorporated in the StartBrowser script, together with their functionalities.

- Method *proc rec-terminate* {*n*} terminates the StartBrowser component when a terminate message is received from ToolBus process STARTBROWSER.

- Method *proc startBrowser* {*location*} opens a browser with the help system located at *location*.

5.9 The Help Component

The Help component is a tcl script, saved in file *helptool.tcl*. This section describes the methods incorporated in the Help component, together with their functionalities.

- Method *proc rec-terminate* {*n*}, which terminates the Help component when it receives a termination message from process HELP.
- Method *proc showHelpKeywords* {*keywords*}. When this method is invoked, a file *index.html* is generated, which contains frame information for displaying the help system in a browser. Then a search request is issued on the list of *keywords*. Finally, a browser is opened with file *index.html*, containing the search results.
- Method *proc showHelpApplication* {} opens the *index.html* file of the TB-Help Generator in a browser.

5.10 The TBHelp Help Format

The TBHelp help format is a HTML-based help format, which resembles the WebHelp format described in Section 2.3.3. A help system in *TBHelp help format*, consists of a set of HTML topic files and configuration files, which reside in a directory named after the help system. The *TBHelp help format* structure consists of the following files.

- A help topic file named *<topicname>.html* for each help topic of the help system.
- A file named *keywords.html*, which contains an alphabetically list of keywords.
- A file named *toc.html*, which contains the table of contents for a help system.
- A set of configuration files for search engine glimpse.
- A set of configuration files for webspider webglimpse. *A* file named *archive-.cfg*, which is a configuration file for webglimpse.
- A file named *wgindex.html*, which contains the a search form in which a user can enter search requests.
- A file named *mainpage.html*, which contains the title page for a help system.
- A file named *index.html*, which contains a frames layout that allow a help system in TBHelp help format to be conveniently arranged when displayed in a browser.

Chapter 6

The TBHelp Viewer

The TBHelp Viewer is the viewer part of the TBHelp HAT. This chapter describes the development process of the TBHelp Viewer, from design to implementation. The TBHelp Viewer allows users to connect help systems created with the TBHelp Generator, to ToolBus applications. It consists of two main parts. The first part is a program called *GenViewer*, which generates a *Help* component (also called *viewer* or *help viewer*) for a specific help system that was created with the TBHelp Generator. The *Help* component allows a user to view the help system with a browser. The second part of the TBHelp Viewer, is a ToolBus script with one process. This process connects the *Help* component to a ToolBus application, after which the *Help* component is able to service help requests from the ToolBus application. We list the features of the TBHelp Viewer in Section 6.1. Section 6.2 describes the implementation details of the TBHelp Viewer. Finally, Section 6.3 describes some examples about how to generate a viewer, help request handling and integrating a help system with its ToolBus application.

6.1 TBHelp Viewer Features

Once a help system had been created with the TBHelp Generator, the help system can be connected to the ToolBus application for which it was created, instead of using it as a stand-alone help system. The TBHelp Viewer provides the user with the capabilities to generate a viewer for the help system, and connect this viewer to the ToolBus application. Once the viewer is connected, it can service help requests from the ToolBus application.

The process of creating a viewer for a help system and connecting the viewer with the corresponding ToolBus application, is depicted in Figure 6.1.

When the GenViewer program is started, a graphical user interface is displayed, in which the user can enter the necessary information about a help system (Figure 6.2).

The GUI contains four text fields in which the user should enter the following data.

- The help system URL.

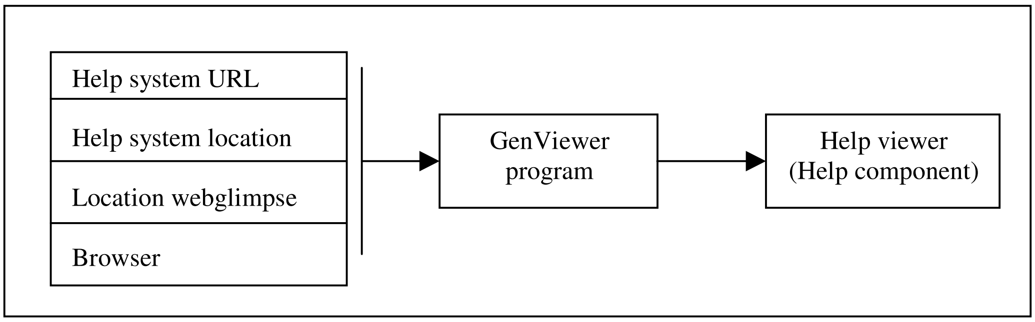

Figure 6.1: Generation process for a help viewer.

Figure 6.2: TBHelp Viewer: Graphical User Interface.

- The help system location.
- The webglimpse location.
- The location of the browser you want to use.

After entering this information in the text fields, the user can press either a *cancel* button or an *ok* button. When the *cancel* button is pressed, nothing happens and the graphical user interface disappears. When the *ok* button is pressed, GenViewer first checks whether all four fields were filled. If not, an error message like the one from Figure 6.3 is displayed. GenViewer also checks whether the browser location and help system location are correct. If the browser location is not correct, the error message from Figure 6.4 is displayed. If the help system location is incorrect the error message from Figure 6.5. The GenViewer program then checks whether a viewer already exists in the current directory. If this is the case, a message asking the user whether the existing viewer should be overwritten, is displayed (see Figure 6.6).

Figure 6.3: TBHelp Viewer: Not all fields are filled.

Figure 6.4: TBHelp Viewer: Incorrect browser location.

Figure 6.5: TBHelp Viewer: Incorrect help system location.

If all went well, a *Help* component has been generated for the specified help system, in file *helpviewer.tcl*. Next to the *GenViewer* program, the TBHelp Viewer also contains a ToolBus script. This script, stored in file *helpscript.tb*, consists of one ToolBus process which is called *HELP*. The user can connect the *Help* component to a ToolBus application by adding the contents of file *helpscript.tb* to the ToolBus script of the concerning ToolBus application, and moving file *helpviewer.tcl* to the ToolBus application directory. If done correctly, the *Help* component is now able to service help requests from the ToolBus application.

A *Help* component generated with the TBHelp Viewer and connected to a ToolBus application, can service two kinds of help requests. A *global* help request opens the main page of a help system in a browser. A *parameterized* help request is a request for help information about a specified set of topics. In this case, the help system is searched for help information about the given topics, after which a browser is opened with the search results.

6.2 TBHelp Viewer Implementation

This section describes the implementation details of program *GenViewer* and the ToolBus script in file *helpscript.tb*

6.2.1 Program GenViewer

The *Help* component, has to be generated for each help system separately. Program GenViewer generates this help viewer, after the user has entered some information about a specific help system. Program GenViewer is a Java appli-

Figure 6.6: TBHelp Viewer: A viewer already exists.

cation, which is saved in file *GenViewer.java*. The methods in GenViewer.java are listed and described next.

- Constructor method *public GenViewer*, initializes the components needed for the GUI that is used to enter a help system URL, a help system location, the browser location and webglimpse location. This method also initializes a dialog which will be used to display messages to the user.

- Method *static public void main (String[] args)* creates an instance of class GenViewer.

- Method *public void actionPerformed (ActionEvent event)*, handles events generated by the GenViewer GUI or a dialog.

- Method *public boolean BrowserExists ()* checks whether the browser with name and location specified by the user, exists. If not, an error message is displayed and false is returned. Otherwise this method returns true.

- Method *public boolean HSExists ()* checks whether the help system with name and location specified by the user, exists. If not, an error message is displayed and false is returned. Otherwise this method returns true.

- Method *public boolean FieldsFilled ()* checks whether the user filled provided all the information necessary to generate a viewer. If not, an error message is displayed and false is returned. Otherwise, this method returns true.

- Method *public void GenScript ()* checks whether a *Help* component already exists, by checking whether file *helpviewer.tcl* exists in the current directory. If this is the case, method *Genscript ()* displays a dialog, asking the user whether the existing *Help* component should be overwritten. If not, the dialog disappears and nothing changes. Otherwise, method *public void Write ()* is called, which actually creates a *Help* component in file *helpviewer.tcl*.

- Method *public void Write ()* creates file *viewer.tcl* in the current directory, containing a *Help* component for a specified help system.

- Method *public void DialogMessage ()* creates a dialog containing a message and a close button.

6.2.2 The Helpscript

Once the *Help* component for a help system has been generated, the help system can be integrated with the corresponding ToolBus application. We do this with a ToolBus script in file *helpscript.tb*. Integrating a ToolBus application and its help system is done by integrating this ToolBus script with the script of the ToolBus application. An example is described in Section 6.3.

The help script takes care of the communication between the ToolBus application and the *Help* component. The ToolBus architecture of a ToolBus application with a *Help* component is shown in Figure 6.7. The ToolBus script provided by the TBHelp Viewer, was shown in Listing 5.7 and explained in Section 5.2.

The help script consists of one process, called *HELP*. This process waits for help requests from ToolBus processes that belong to the ToolBus application. If a help request is received, process HELP sends a *do_request* to the viewer component, which will further handle the request.

Process HELP can handle two kinds of help requests from a ToolBus application. The first one is a *global* help request (lines 10-11, Listing 5.7), and the second is a *parameterized* help request (lines 13-14, Listing 5.7).

6.3 Applying the TBHelp Viewer

This section shows how a viewer was generated for the TBHelp Generator help system, called *TBHelpGeneratorHelp*. It then describes how the help system was integrated with the TBHelp Generator, using the viewer. Finally we list an example of request handling for a *global* help request and request handling for a *parameterized* help request.

6.3.1 Generating a Viewer

Before we can connect help system *TBHelpGeneratorHelp* to the TBHelp Generator ToolBus application, we have to generate a help viewer for the help system. This help viewer will be able to service help requests from the TBHelp Generator ToolBus application, and display the results of help requests in a browser. The help viewer for the *TBHelpGeneratorHelp* help system, was represented in Chapters 4 and 5 as the *Help* component of the TBHelp Generator.

We generate a help viewer by starting program *GenViewer*. If started correctly, the GUI from Figure 6.2 appears. We enter the requested information about help system *TBHelpGeneratorHelp*, webglimpse and the browser, after which the *OK* button is pressed. The help viewer will be generated and stored in file *helpviewer.tcl*. If necessary the viewer can be regenerated, by restarting program GenViewer and again supply the requested information.

6.3.2 Integrating Help System and TBHelp Generator

This section describes how to integrate the help viewer we created in section 6.3.1, with The TBHelp Generator ToolBus application.

First we add ToolBus process *HELP* (from file *helpscript.tb*) to the ToolBus script for the TBHelp Generator ToolBus application (file *tbh.tb*). After adding process *HELP* to *tbh.tb*, we add *HELP* to the ToolBus configuration. The help viewer was presented as the *Help* component of the TBHelp Generator ToolBus application (Figure 5.1). Section 5.9 described the contents of file *helpviewer.tcl* (The *Help* Component).

Once process HELP is added to the TBHelp Generator ToolBus application, the help viewer is connected and process HELP is ready to receive help requests from the TBHelp Generator. The TBHelp Generator application can send help requests to process HELP, which will handle these help requests. Two kinds of help request are possible, *global* help requests and *specialized* help requests. The next section describes an example of both kinds of help requests.

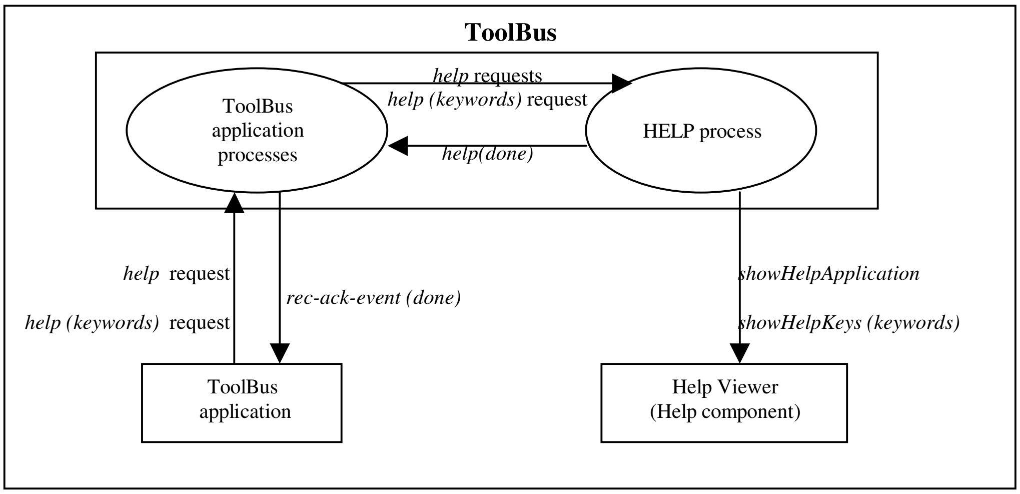

Figure 6.7: Help request handling.

6.3.3 Help request handling

In this section we describe an example of a *global* help request and a *parameterized* help request.

A global help request

When selecting menu item *Help* from menu *Help*, the TBHelp Generator sends a help request to its ToolBus script in file *tbh.tb*. We added lines 37-40 (Listing 5.1) to process TBH, to handle incoming help requests from the *help* menu item. When the global help request is received by process TBH, a message *help* is sent to process HELP. Process HELP then sends a help request to the *Help* component, which opens the help system for the TBHelp Generator ToolBus application, in a browser (Figure 4.1). In the mean while, a message *help(done)* is sent back to process TBH, to indicate that the help request is being handled. Handling a global help request from the TBHelp application is represented by Figure 6.7.

A parameterized help request

We use component TOC of the TBHelp Generator ToolBus application to illustrate a parameterized help request. When the button labeled *help* on the *toc* window from Figure 4.18 is pressed, a help message with a list of keywords is sent to the ToolBus process TOC (Listing 5.3). This list contains two keywords, *table* and *contents*.

Process *TOC* receives an incoming help request from component Toc (line 15), after which a help message with the list of keywords is sent to process HELP (line 16). Process HELP handles the help request by sending the help request with the list of keywords to the *Help* component. The *Help* component issues a search request on the TBHelp help system, on the list of keywords, and displays the results in a browser (Figure 6.8).

Handling a *parameterized* help request is represented by Figure 6.7.

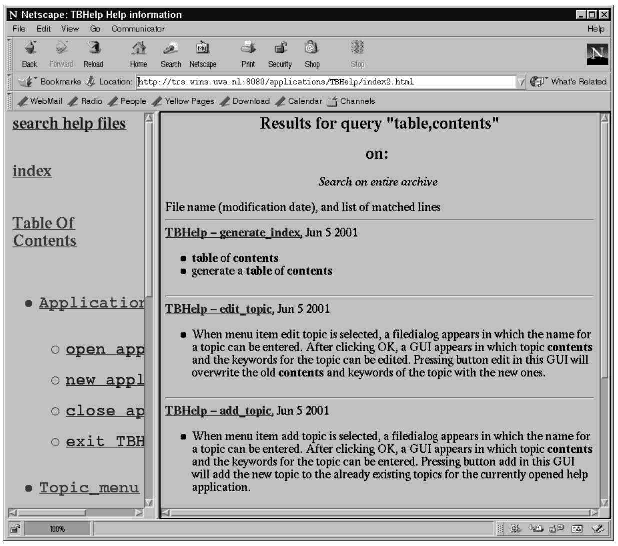

Figure 6.8: Result of a specialized help request.

In the meanwhile, process TOC receives a message *help(done)* (line 17) from the HELP process, after which it sends an acknowledge event back to component Toc, indicating that the help request is being handled.

Chapter 7

Discussion, Future Research and Conclusions

In Section 7.1 of this chapter, we discuss the results of our research. Section 7.2 defines some areas for further research. Finally, Section 7.3 briefly describes our conclusions.

7.1 Discussion

Limitations of TBHelp

TBHelp is not as feature-rich as the commercial help authoring tools discussed in Chapter 3. Our goal was to develop a help authoring tool that conforms to the set of requirements we defined in Section 2.6 and Section 3.7. TBHelp does not meet all those requirements. TBHelp has the following limitations:

- *Single source*: TBHelp lacks a true single source help format. TBHelp does not support the printing of complete online manuals. Instead, only separate help topics can be printed.

- *Support for an integrated editor*: TBHelp has no integrated support for an editor. Instead, a basic GUI for editing help topic content and keywords is provided.

- *Supported help formats*: TBHelp produces only one proprietary online help format. There is no support for other online help formats. Major help authoring tools, usually have the ability to produce help in more than one output format.

- *Support for project management*: Another limitation of TBHelp, is its lack of support for project management. TBHelp has no built-in support for reporting help project diagnostics. A help authoring tool that has the ability to report broken links or other statistics aids in controlling the help project status. Other lacking project management features include integrated facilities for testing and debugging the online help system.

TBHelp and XML

Most help authors are familiar with HTML- and/or RTF-based help formats. Both formats use markup tags to format a document. Most of these tags control document presentation and do not add information about the kind of data it contains. Specific markup languages have a tag set that is not extensible. It would be a lot easier for help authors if there was a simple HTML tag for adding a table of contents entry or an index entry.

Like HTML, XML [40] is a web standard, but it is more flexible. Internet Explorer 5 supports most of the international standards for XML 1.0. Netscape included standard support for XML in Netscape Navigator 6. Older versions of the browsers do not support XML. XML can be used to format any kind of textual data. In addition, XML can be used to *describe* the structure of textual data. This makes XML suitable for describing navigational structures. The navigation files in the TBHelp system, which are used to connect HTML-format help topic files, only make use of standard HTML tags. The application of XML tags or metadata is a better way of connecting the help topic files. These tags can be used to describe the meaning of data, for example <tocitem> and <indexitem>. The index-entry for an index file could be described by "<indexitem target="intro" text="introduction to...">. When using a standard web browser as help viewer, the help authoring tool should automatically collect all the data for these tags into an index file.

XML can be used for a variety of domain-specific markup languages. Current HTML-based help systems from Microsoft and Sun use proprietary tags to build navigation. Sun uses XML to build table of contents and indexes in JavaHelp. At this most basic level, XML is simply a database format for building a table of contents or a keyword index. [41] suggests that XML has more applications that make help systems easier to author and easier to use. In this article several potential uses for XML in online help systems, are considered, which we will discuss next:

- To make a distinction between help content for beginners and more experienced readers, help authors can mark individual sections of a help topic. According to the experience level of a reader, only marked sections will be displayed. These marked sections or *information types* as [42] denotes them, allow users to personalize their help systems by instructing the system to display the information that is appropriate for them.

- With XML one can define *one-to-many* relationships. For example, a keyword can be linked to one or more topics.

- Embedding XML tags in help topics could be used to instruct a compiler or a help authoring tool to perform certain actions. For example, metadata could be used to indicate that the current topic is a procedure and should always appear in a secondary window. This saves authors from repetitiously coding redundant information in each topic.

- By assigning a tag to a topic, authors can define a logical sequence for a group of pages so that a user could browse through them in any order. Creating and maintaining such browse tags is much simpler than working with hard-coded links.

- A *Help Markup Language* could easily define a standard format for table of contents entries and search keywords in HTML-based help. It could also define a standard <help> tag for providing context-sensitive help on web pages. A Help Markup Language might be expanded to define a tag set for standard help elements, such as section title, subtitle, steps, tips, and codes samples.

XML is not a replacement for HTML in the above examples. XML is used as a way to describe the data, while HTML will be used to format and display the same data. These technologies work together to create web documents.

XML improves the ability to create content from a single source and display it on any XML-enabled platform. Delivering help through the application of XML will provide users with more targeted information and simplifies the help authoring process.

The shift from HTML to XML is of no consequence to webglimpse. However, a minor change has to be made in the call to glimpseindex. Glimpseindex is the indexing program for search engine glimpse. When the TBHelp Generator adds search capabilities to a help system, a call to glimpseindex is made, which results in an index file being generated. This index includes the HTML and XML tags that are part of the help topic files. There are several ways to exclude the tags:

- When the call to glimpseindex is made, include the -z option. This option allows customizable filtering, using the file *.glimpse_filter*. Parts of files that should not be indexed, like the HTML and XML tags, will now be excluded. (The -z option should be included in the glimpseindex call in file wgreindex, which resides in the help system directory.). The -z option is not included by default, because the filtering may slow down the indexing process.
- Include the tag <input name=nonascii type=hidden value=on> in the webglimpse search form. HTML and XML tags will now be excluded from the search results. (The webglimpse search form is contained in file wgindex.html, which resides in the help system directory.)

Search Engines and Webspiders

Choosing a suitable search engine and/or webspider is not an easy task, because a lot of options are available. While some freeware still exists, a host of software companies have delivered commercial search engines to the market and most commercial web servers now have a built-in search facility [39]. A developer has to ask himself/herself the right questions when looking for a suitable search engine and/or webspider. For example:

- What platform will be used?
- How important is easy installation and low maintenance?
- What file types should be supported by the search engine?
- Where reside the files that should be indexed and searched?
- What kind of search engine should be used? A simple one or one that prompts for a number of search criteria? Another option is to use an initial simple search form, with a link to a more advanced search form.

- What search capabilities should be offered to users? Natural query language, websearch, boolean operators, proximity operators, phrase searching, thesaurus, concept searching, wild cards, exact match, fuzzy match, numeric operator, range operator, fielded searches, query by example, advisors etc.
- Should structured fields or metadata be added to documents?
- How should files that will be indexed, be collected? Should files for indexing be submitted interactively? Is built-in collection support needed? When files are submitted, should metadata be submitted with it?
- How should results be displayed? By title, author, description of summary, size, relevance ranking, number of files found, search terms used, date on which files were created or indexed, highlighting terms that were searched on, navigation between search terms (or hits) within the selected files etc. Should users be able to alter the results display?
- Should users be able to view the HTML form of the files?
- How much can be paid for the search engine?

Many more questions may be asked, depending on the particular situation for which the search capabilities are needed.

With the above list of questions in the back of our head, we looked at several commercial and non-commercial search products, like *FreeFind* [37], *ht://dig* which is combined with search engine *htsearch* [36], *Inktomi search software* [38] and *webglimpse* which works with search engine *glimpse* [34]. We tested FreeFind first, but because this webspider lacked the flexibility needed in the development process of TBHelp, we replaced it with webglimpse.

Webglimpse runs on a UNIX platform, which is the platform for which TBHelp was mainly developed. Webglimpse is easy to install and maintenance is low, which enabled us to focus on the development process of TBHelp instead of losing time in maintenance. Webglimpse supports the file format we wanted to index, namely HTML files. Webglimpse can index files in specified directories and at specified locations on the web. Webglimpse allows for simple and advanced search forms or a combination of both. Webglimpse has a lot of search capabilities, including the capability of adding and removing simple search forms to the indexed HTML files. Basic built-in collection support is available. Users of webglimpse can tailor the results display to their specific needs. Users can view the HTML form of the data files and webglimpse is free for educational institutions. Because of the reasons listed above, webglimpse in combination with glimpse turned out to be a suitable choice for us.

Convert Existing Help Systems

THBelp currently lacks the capabilities for automatically converting existing help systems to TBHelp help format. It is however possible to automatically convert LaTeXdocumentation [44] to a help format that partly resembles the TBHelp help format, so it can be integrated with a ToolBus application. Most users of the ToolBus software architecture have been writing their documentation in latex, which makes this a very attractive possibility. The following steps are required to convert LaTeXdocumentation to a HTML based help format and integrate it with a ToolBus application:

1. First convert the LaTeXdocumentation to documentation in a HTML based help format with program *latex2html* (e.g latex2html main.tex).

2. Then open the help system in the TBHelp Generator and add search capabilities by selecting menu item *Add search* from the *Tools* menu.

3. Now add a link to the *index.html* file of the help system, which jumps to the search form contained in file *wgindex.html.* (e.g. *Search*)The help system can now be used as a stand-alone help system with search capabilities, by opening the *index.html* file in a browser.

4. To integrate the help system with a ToolBus application, first generate a help viewer for the help system with program *GenViewer* as described in Chapter 6. The help viewer contains two procedures that each generate a file *index.html* when called. Delete the lines of code in file *helpviewer.tcl* that are used to create file *index.html.*

5. Then integrate the help system and ToolBus application as described in Section 6.3.2. Once integrated, the help viewer will be able to handle *global* and *parameterized* help requests from the ToolBus application as described in Section 6.3.3.

It is relatively easy to alter program GenViewer so a user can select whether a help system was either created with the TBHelp Generator or in LaTeX, in both cases making sure that a correct help viewer is generated. In this case, adjusting file *helpviewer.tcl* in step 4 will not be necessary.

TBHelp on Windows

The TBHelp generator is a ToolBus application. It only runs in a UNIX environment, because the ToolBus software architecture currently only runs in a UNIX environment. However, efforts to port the ToolBus software architecture to a Windows platform are being made, which will open the way for porting TBHelp to a Windows platform. If the interfaces between the ToolBus software architecture and software components remain the same, porting TBHelp to a Windows environment will not pose many problems. The seven components of the TBHelp Generator, defined in Section 5.2, are all platform independent because they are implemented in either Java or Tcl/Tk. The only problem arising involves webglimpse and glimpse. The glimpse search engine does not run on a Windows platform, which means that webglimpse cannot be used. Connecting another search engine to the TBHelp Generator, will therefore be necessary. Porting the TBHelp Viewer to Windows will not pose any problems, because the program is completely written in Java and generates Tcl.

Templates

The TBHelp package does not use templates, because we do not consider the use of templates very user-friendly. Templates are not a transparent solution and it requires a lot of low-level knowledge from the user. Besides, using templates, would degrade the TBHelp HAT to only being a viewer, as defined in Chapter 3.

JavaHelp instead of TBHelp

JavaHelp is a good solution for offering online help that is platform independent. JavaHelp is becoming more popular and will have a broad base of users in the future. There are a few disadvantages to the use of JavaHelp, which are discussed in Section 2.3.4. How to use JavaHelp and how to integrate help systems created with it, with ToolBus applications was discussed in Section 3.5. The development of a help authoring tool which produces the JavaHelp help format should solve this problem. This means that a complete JavaHelp HelpSet containing the required XML-format files, must be generated.

7.2 Future Research

This section lists some possible areas for further research:

- Extensive research about using XML instead of HTML, to structure table of contents, index and help topics, should be done.
- More research about webspiders and/or search engines should be done.
- Further research to whether and how TBHelp can be ported to a Windows environment should be done.
- Help systems in a lot of help formats exist. Research about how to convert existing help formats to TBHelp help format and/or how to improve the help format, should be done.
- TBHelp should provide capabilities for merging several existing help systems into a single help system. Research about how this can be done is needed.
- The algorithms used for creating a table of contents and generating an index, are very simplistic. Research in the area of more sophisticated algorithms is necessary.
- Integrating an existing editor with the TBHelp Generator instead of the basic GUI we use now, will facilitate the help authoring process. Another option is adding functionality to the basic GUI and create our own editor. More research on what option to use and, if the first option is chosen, which editor and how to integrate the editor, should be done.
- Instead of creating a new HAT, it may be possible to use an existing viewer, such as JavaHelp and create a generator for it. The creation of a JavaHelp system was discussed in Section 2.3.4. The development of a UNIX-based help authoring tool which produces the JavaHelp help format should hide the administrative details of JavaHelp. More research in this area should be done.
- More research about how to integrate existing LaTeXhelp documentation automatically with ToolBus applications or other coordination architectures, is needed.

7.3 Conclusions

Our research was focussed on answering the following central question:

How can a help authoring tool be developed, that runs on top of the ToolBus software architecture?

In answering the central question, two questions arose, which we list and answer next:

1. *Is it possible to reuse existing help technology in the development of a help authoring that runs on top of the ToolBus?*
2. *Which basic requirements should a help authoring tool, built on the ToolBus, meet?*

In Chapter 2 and 3 we discussed existing help technology, to answer research question 1. We focussed on existing help authoring tools and online help formats produced by these tools. We evaluated the comparative features of four major commercial Windows help authoring tools. This comparison was made to give an overview of existing help technology and to explore the possibilities for reuse. Most online help formats are developed to run on a Windows platform. Existing platform independent online help formats are commercial products and therefore can not be reused in a ToolBus-based help authoring tool. Some help formats that are in an immature state, such as JavaHelp, offer future potential for delivering a robust platform independent system for application help and online documentation. In the absence of a UNIX-based help authoring tool, building a help authoring tool which delivers a JavaHelp format is an option to consider. More research needs to be done on this subject, see also Section 7.2.

After comparing several help formats and help authoring tools, we defined a set of basic requirements, which a HAT that runs on top of the ToolBus software architecture, should meet. These basic requirements, which are listed in 3.7, form the answer to research question 2. Because existing help formats and HAT's could not be used (Sections 4.1 and 7.1), we decided to develop TBHelp, a HAT that runs on top of the ToolBus software architecture.

TBHelp is a demonstration of a possible answer to our central research question. It is a HAT that meets most of the basic requirements we defined in Section 3.7 (See Section 7.1 for a discussion).

More research is necessary before a well-informed decision can be made on whether to continue the development of TBHelp or use an existing HAT or viewer (e.g. JavaHelp). However, in our opinion extending the functonality of TBHelp or developing a generator for JavaHelp, are currently the best solutions to our central research question.

Bibliography

[1] J.A. Bergstra and P. Klint:
The ToolBus Coordination Architecture; Proc. 1st Int. Conf. on Coordination Models and Languages, pages 75-88; Springer-Verlag, Berlin, 1996

[2] J.A. Bergstra and P. Klint:
The Discrete Time ToolBus – A software coordination architecture; Science of Computer Programming journal, volume 31, pages 205-229; July 1998

[3] R.S. Pressman:
Software Engineering: A Practitioner's Approach; McGraw-Hill; 5th edition, June 2000

[4] M.G.J. van de Brand, H.A. de Jong, P. Klint and P.A. Olivier:
Efficient Annotated Terms; Software, Practice & Experience, 30(3), 259–291, 2000

[5] P.A. Olivier and H.A. de Jong:
ATerm Library User Manual; Centrum voor Wiskunde en Informatica, 2001
`http://www.cwi.nl/projects/MetaEnv/aterm`

[6] K. Lewis:
Creating Effective JavaHelp; O'Reilly and Associates, Inc.; 1st edition, June 2000

[7] M.E. Sleeter:
Opendoc-Building Online Help for a component-Oriented Architecture; Proceedings of the 14th annual international conference on Marshaling new technological forces, pages 87-94; 1996

[8] B. Gelernter:
Help desigh challenges in network computing; Proceedings of the sixteenth annual international conference on Computer documentation, pages 184-193; 1998

[9] R. Green:
Component-based software development: implications for documentation; Proceedings on the seventeenth annual international conference on Computer documentation, pages 159-164; 1999

[10] J. Hedtke and E. Knottingham:
RoboHelp 2000 Bible; IDG Books Worldwide, Inc., 2000

[11] D. Cline:
HTML Help Tools, Software Development Online Magazine, May 1999
http://www.sdmagazine.com/articles/1999/9905/9905i/9905i.htm

[12] L. Chamberland:
Componentization of HTML-Based Online Help; Proceedings on the seventh annual international conference on Computer Documentation, pages 165-168; 1999

[13] *JavaHelp System User's Guide, JavaHelp 1.1*; Sun Microsystems, November 1999
http://www.java.sun.com

[14] David V. Granger:
Online Help Trends, moving from WinHelp to HTML-based Online Help; ForeFront, Inc., April 2001 http://www.ff.com/PDFs/HelpTrends.pdf

[15] S. J:
Help format feature matrix; Help Authoring Resource Center http://www.helpauthoring.com/

[16] Gary Conroy:
Help authoringing tools; http://techwriting.about.com/

[17] S. J:
HTML Help: The New Standard for Windows Application Help; Help Authoring Resource Center http://www.helpauthoring.com/htmlhelp/htmlhelp.htm

[18] *ActiveX Controls*; Microsoft Corporation, 2000 http://www.microsoft.com/com/tech/ActiveX.asp

[19] David Rubinstein:
Help is on the way from ForeFront; Software Development Times, October 2000 http://www.ff.com

[20] Richard M. Stallman:
GNU Emacs Manual; iUniverse.Com, Inc., September 2000 http://www.gnu.org/manual/emacs-20.3/emacs.html

[21] *Texinfo: The GNU Documentation Format* http://www.gnu.org/manual/texinfo-4.0/texinfo.html

[22] Stephen J. Hegner, Paul Mc Kevitt and Peter Norvi: *Intelligent Help Systems for Unix*; Kluwer Academic Publishers, April 2001

[23] Valerie Quercia and Tim O'Reilly:
Volume 3M: X Window System User's Guide; O'Reilly and Associates, Inc., 2nd Edition January 1993

[24] *About eHelp Corporation*; eHelp Corporation, 2001 http://www.ehelp.com/about/

[25] *Doc-To-Help Professional 2000 Overview*; Wextech Systems, 2001 http://www.wextech.com/prd2hp4overview.htm

[26] *Virtual Media Suite 2000*; Virtual Media Technology, 2000 `http://www.vmtech.com/`

[27] *NetHelp Overview*; Netscape Communications Corporation, 1997 `http://home.netscape.com/eng/help/`

[28] *HyperHelp: the online help system for Unix and Linux applications*; Bristol Technology, January 2000 `http://www.bristol.com/hyperhelp/`

[29] Kevin A. Siegel:
Essentials of RoboHELP Classic 9 IconLogic; December 1, 2000

[30] David Farkes, Scott Boggan and Joe Welinske:
Developing Online Help for Windows 95 Solutions Publishing; March 15, 1999

[31] T.M. Duffy, J.E Palmer and B. Mehlenbacher:
Online help: design and evaluation Norwood, New Jersey, Ablex Publishing Company; 1992

[32] P. H. Westendorp:
Product Design, Interface Design and Help Design College Vormgeving van Visuele Informatie, Technische Universiteit Delft, 1999

[33] A. van Deursen, B. Elsinga, P. Klint and R. Tolido:
From Legacy to Component: Software Renovation in Three Steps CWI, University of Amsterdam, Cap Gemini Institute; Cap Gemini, 2000

[34] *Webglimpse - Search Engine Software*; July 2001 `http://webglimpse.net/`

[35] *ForeHTML Pro: Features for Developing ForeFront's InterHelp* ForeFront, Incorporated, April 2001 `http://www.ff.com/Products/ForeHTMLP/InterHelp.htm`

[36] *WWW Search engine software* The ht://dig group, 2001 `http://www.htdig.org`

[37] *FreeFind Search Engine* FreeFind, 2001 `http://www.freefind.com`

[38] The Inktomi Corporation, 2001 `http://www.inktomi.com/products/search/`

[39] N. Plat:
Search Engines for Intranets LLRX Web Journal, August 1998 `http://www.llrx.com/features/nina.htm`

[40] Elliotte Rusty Harold: *Xml Bible*;Hungry Minds, Inc; 2nd edition, June 2001

[41] Scott Boggan: *XML: What Do Help Authors Need to Know? Part 1* Win-Writers, Training for Software User Assistance, July 2001 `http://www.winwriters.com/9906sb.htm`

[42] Scott Boggan: *The need for an XML standard for web-based Help*;Help for the Web Initiative, June 2000 `http://www.help4web.org/publication/s_boggan.htm`

[43] *HelpBreeze JavaHelp Edition: A powerful WYSIWYG editor for HTML base help*; Solutionsoft, 2001 `http://www.solutionsoft.com/javahelp.htm`

[44] Helmut Kopka, Patrick W. Daly:
A Guide to LaTeX: Document Preperation for Beginners and Advanced Users. Addison-Wesley Pub Co, 3rd Edition; February 1999

Appendix A

Installation Guide

This appendix describes the required steps needed for implementing the ToolBus HelpTool. First you will need to install a server for running several scripts and making helpfiles accessible from the web. A server that is relatively easy to install is the Apache HTTP server. More on the Apache server and its installation can be found at `http://httpd.apache.org/`. The Toolbus HelpTool uses the webspider *webglimpse* to service search requests on help information generated by TBHelp. Webglimpse in combination with *glimpse*, can work to index and search your own "corner" of the web, defined by the links you place on a starting page. More on glimpse, webglimpse and their installation, can be found at `http://webglimpse.net/download.html`. The TBHT is a ToolBus application. The ToolBus is a novel software architecture intended for the flexible development of cooperating, distributed applications. The ToolBus package together with its Installation steps can be found at `http://www.cwi.nl/projects/MetaEnv/toolbus/`.

Below we describe where to get TBHelp and how to install it.

We assume that a server, glimpse, webglimpse and the ToolBus package have all been correctly installed. Get the newest version of the TBHelp HAT. Unzip and extract with *gunzip TBHelp<version>.tar.gz* and *tar xvf TBHelp<version>.tar*. Directory *TBHelp<version>* contain a file *README*, which describe the installation procedure for TBHelp.

Once TBHelp has been installed, the TBHelp Generator can be started with *toolbus tbh.tb* and the *GenViewer* program belonging to the TBHelp Viewer, can be started with *java GenViewer*.