

Masters's Thesis in Telematics, Graz University of Technology

Institut for Information Processing and Computer Supported New Media

The Design and Implementation of the Harmony Session Manager

Jürgen Schipflinger

Advisor: O. Univ. -Prof. Dr. Dr. h. c. Hermann Maurer

21st December 1998

Supervisor: Dr. Keith Andrews

Diplomarbeit in Telematik, TU Graz

Institut für Informationsverarbeitung und Computergestützte neue Medien

The Design and Implementation of the Harmony Session Manager

Jürgen Schipflinger

Gutachter: O. Univ. -Prof. Dr. Dr. h. c. Hermann Maurer

21. Dezember 1998

Betreuer: Dr. Keith Andrews

Abstract

This thesis describes the Harmony session manager. Harmony is a graphical client and authoring tool for the Hyperwave information server. Harmony provides an intuitive interface to Hyperwave's underlying infrastructure, such as hierarchical structuring, rich metadata, sophisticated search functionality, and user management.

The Harmony session manager provides the central navigational and authoring functionality of Harmony. The collection browser visualises the hierarchical collection structure in a dynamically generated tree and provides location feedback for all visited documents. The local map visualises hyperlink structures between documents in the form of a dynamically generated graph. Simple and extended search dialogues interface to Hyperwave's powerful search facilities.

The Harmony session manager's authoring functionality includes interactive structuring, insertion, and deletion of documents, and interactive editing of document and link metadata. Individual document viewers are started by the Harmony session manager to display document contents. The session manager, in cooperation with the document viewers, supports document content editing and interactive link creation. Finally, extensibility is provided through a programmable API and user-configurable menus.

Ich versichere, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt zu haben und mich auch sonst keiner unerlaubter Hilfsmittel bedient zu haben.

Die Diplomarbeit ist in englischer Sprache verfaßt.

Acknowledgements

I would like to thank everyone who has helped me in my work and also those who gave suggestions for problem solutions encountered during the writing of this thesis. In particular I want to thank Dr. Keith Andrews and Prof Frank M. Kappe for their help with various problems and the ideas they brought into our discussions. Univ.-Prof. Dr. Hermann A. Maurer and all the people at the IHM and IICM for their support.

Index

1	INTRODUCTION.....	1
2	HYPERTEXT, MULTIMEDIA, HYPERMEDIA	3
3	NETWORK INFORMATION SYSTEMS	5
3.1	INTERNET.....	5
3.1.1	<i>TCP/IP - the Transmission Control Protocol/Internet Protocol</i>	<i>5</i>
3.2	BASIC INTERNET APPLICATIONS.....	6
3.3	ARCHIE	6
3.4	WAIS	6
3.4.1	<i>Scoring</i>	<i>7</i>
3.4.2	<i>Relevance Feedback.....</i>	<i>7</i>
3.5	GOPHER	8
3.5.1	<i>Access to Gopher.....</i>	<i>8</i>
3.5.2	<i>Browsing through Gopher.....</i>	<i>9</i>
3.6	WORLD WIDE WEB (WWW).....	9
3.6.1	<i>HTTP</i>	<i>9</i>
3.6.2	<i>URL.....</i>	<i>10</i>
3.6.3	<i>HTML.....</i>	<i>10</i>
4	HYPERWAVE	12
4.1	HYPERLINK NAVIGATION	12
4.2	OBJECT ATTRIBUTES.....	12
4.3	COLLECTION HIERARCHY	13
4.3.1	<i>Collections.....</i>	<i>14</i>
4.3.2	<i>Cluster.....</i>	<i>14</i>
4.3.3	<i>Sequence.....</i>	<i>15</i>
4.3.4	<i>MultiCluster</i>	<i>15</i>
4.3.5	<i>AlternativeCluster</i>	<i>16</i>
4.4	PURPOSES OF THE COLLECTION HIERARCHY	16
4.5	MULTILINGUALITY	17
4.5.1	<i>Multilinguality in Hyperwave.....</i>	<i>17</i>
4.6	SEARCH FACILITIES IN HYPERWAVE	18
4.6.1	<i>Full Text Search.....</i>	<i>18</i>
4.6.2	<i>Search on Attributes.....</i>	<i>18</i>
4.6.3	<i>Search Scope</i>	<i>19</i>
4.7	USER MANAGEMENT	19
4.8	HYPERWAVE MESSAGES	20
4.9	INTEROPERABILITY	20
4.10	THE ARCHITECTURE OF HYPERWAVE	21
4.10.1	<i>Full Text Server.....</i>	<i>22</i>
4.10.2	<i>The Document Server.....</i>	<i>22</i>
4.10.3	<i>The Object Server.....</i>	<i>23</i>
5	THE HARMONY SESSION MANAGER.....	27
5.1	THE ARCHITECTURE OF HARMONY.....	27
5.2	USER INTERFACE	28

5.2.1	<i>The Collection Browser</i>	29
5.2.2	<i>The Local Map</i>	33
5.2.3	<i>The Search Dialog</i>	35
5.2.4	<i>The History Browser</i>	40
5.2.5	<i>The Information Landscape</i>	41
5.2.6	<i>Server Status Browser</i>	44
5.2.7	<i>Multilinguality</i>	46
5.3	THE HARMONY DOCUMENT VIEWERS	49
5.3.1	<i>Text Viewer</i>	50
5.3.2	<i>Image Viewer</i>	51
5.3.3	<i>Film Player</i>	51
5.3.4	<i>Audio Player</i>	52
5.3.5	<i>PostScript Viewer</i>	52
5.3.6	<i>VRweb 3D Viewer</i>	53
5.4	AUTHORING WITH HARMONY.....	54
5.4.1	<i>Inserting New Collections or Clusters</i>	54
5.4.2	<i>Inserting Documents</i>	55
5.4.3	<i>Creating Remote Objects</i>	56
5.4.4	<i>Editing Documents</i>	57
5.4.5	<i>Moving and Copying Objects</i>	58
5.4.6	<i>Deleting Objects</i>	59
5.4.7	<i>Editing Object Attributes</i>	60
5.4.8	<i>Creating Links</i>	63
5.5	EXPANDABILITY	65
5.5.1	<i>Expandability of the Harmony Menu</i>	66
5.5.2	<i>The Harmony API</i>	67
6	SELECTED DETAILS OF THE IMPLEMENTATION	73
6.1	IMPLEMENTATION OF THE LOCAL MAP	73
6.1.1	<i>A Hierarchical Graph Layout Algorithm</i>	73
6.2	LOCATION FEEDBACK.....	79
7	EXTENSIONS AND FURTHER WORK.....	81
8	CONCLUDING REMARKS	82
9	BIBLIOGRAPHY	83
	APPENDIX A. HARMONY KEY BINDINGS	86
	APPENDIX B. CONFIGURING EXTERNAL TOOLS IN THE MENU	89
	APPENDIX C. SEARCH SYNTAX IN HARMONY	91
	ATTRIBUTE SEARCH.....	91
	CONTENT SEARCH	91
	APPENDIX D. THE HARMONY API	93
	THE HARMONY TOOL API	93
	HARMONY SESSION MANAGER API.....	94
	APPENDIX E. RULES FOR DOCUMENT SELECTION IN AN ALTERNATIVECLUSTER	97
	APPENDIX F. THE HARMONY ICONS	100

List of Figures

<i>Figure 1 : Linear text and hypertext.....</i>	<i>3</i>
<i>Figure 2: Node, Links and Anchors comprising a Hyperdocument.....</i>	<i>4</i>
<i>Figure 3: Page designed using HTML 4.0.....</i>	<i>11</i>
<i>Figure 4: Collection Hierarchy.....</i>	<i>13</i>
<i>Figure 5: Sequence in Hyperwave.....</i>	<i>15</i>
<i>Figure 6: Architecture of Hyperwave.....</i>	<i>22</i>
<i>Figure 7: The Harmony Collection Browser.....</i>	<i>29</i>
<i>Figure 8: Harmony Sort Order Dialog.....</i>	<i>32</i>
<i>Figure 9: Local Map options.....</i>	<i>34</i>
<i>Figure 10: The Harmony Local Map.....</i>	<i>35</i>
<i>Figure 11: The Harmony Search Dialog.....</i>	<i>36</i>
<i>Figure 12: Harmony Search with "location feedback" in the Collection Browser.....</i>	<i>37</i>
<i>Figure 13: Harmony Extended Search.....</i>	<i>39</i>
<i>Figure 14: Harmony Active Collections.....</i>	<i>39</i>
<i>Figure 15: The Harmony History Browser.....</i>	<i>41</i>
<i>Figure 16: Information Landscape Overview Map.....</i>	<i>42</i>
<i>Figure 17: The Harmony Information Landscape.....</i>	<i>43</i>
<i>Figure 18: The Harmony Information Landscape with activated link structure.....</i>	<i>43</i>
<i>Figure 19: The Harmony Server Status Browser.....</i>	<i>44</i>
<i>Figure 20: The Harmony User Connection Browser.....</i>	<i>45</i>
<i>Figure 21: Harmony Status Browser, Send Message.....</i>	<i>46</i>
<i>Figure 22: Language Preference Dialog.....</i>	<i>48</i>
<i>Figure 23: Harmony session in Japanese.....</i>	<i>49</i>
<i>Figure 24: The Harmony Text Viewer.....</i>	<i>50</i>
<i>Figure 25: The Harmony Image Viewer.....</i>	<i>51</i>
<i>Figure 26: Harmony Film Player with moving links.....</i>	<i>52</i>
<i>Figure 27: Harmony PostScript Viewer.....</i>	<i>53</i>
<i>Figure 28: The Harmony VRweb 3d Viewer.....</i>	<i>54</i>
<i>Figure 29: Insert Collection Dialog.....</i>	<i>55</i>
<i>Figure 30: Insert Dialog.....</i>	<i>56</i>
<i>Figure 31: Insert Remote Document Dialog.....</i>	<i>57</i>
<i>Figure 32: Harmony Editing Text with Emacs.....</i>	<i>58</i>
<i>Figure 33: Harmony Copy Object Dialog.....</i>	<i>58</i>
<i>Figure 34: Delete Document Dialog.....</i>	<i>59</i>
<i>Figure 35: Delete Collection Dialog with Database Object Query.....</i>	<i>60</i>
<i>Figure 36: Harmony Attributes Dialog.....</i>	<i>61</i>
<i>Figure 37: Harmony Edit Attributes Dialog.....</i>	<i>62</i>
<i>Figure 38: Harmony Link Creator, Source set.....</i>	<i>64</i>
<i>Figure 39: Harmony Collection Browser with Tools Menu.....</i>	<i>67</i>
<i>Figure 40: HyGen Vocative Link Generator.....</i>	<i>69</i>
<i>Figure 41: HyGen Glossary Generator.....</i>	<i>70</i>
<i>Figure 42: The HarSearch.....</i>	<i>71</i>
<i>Figure 43: Architecture of Harmony.....</i>	<i>27</i>
<i>Figure 44: Nomenclature for layout algorithm.....</i>	<i>74</i>
<i>Figure 45: Local Map after step one, before level optimisation.....</i>	<i>76</i>
<i>Figure 46: Local Map after step one, after level minimisation.....</i>	<i>77</i>
<i>Figure 47: Local Map after step two.....</i>	<i>78</i>
<i>Figure 48: Local Map laid out from left to right.....</i>	<i>79</i>
<i>Figure 49 : Collection Browser with no path to root collection.....</i>	<i>80</i>

Credits

Figures 1, 2 and 40 were drawn by Keith Andrews.

Figure 6 was drawn by Frank Kappe.

1 Introduction

Modern information systems contain text, graphics, sound, animation and other kinds of information. However, multimedia documents consume a great amount of (disk) space. The physical location of the information may be distributed and can be accessed over a network. Furthermore, links (references) between several documents may exist and may be presented with the original information. When peoples use such systems and browse through the information by following links, issuing searches and considering matching documents, they may become “lost in Hyperspace” [EH89]. In particular, it is hard to remember the location of information in the information universe, and therefore it is hard to refind information. A serious system should address these issues and try to avoid or at least reduce the disorientation of the user in the masses of information. It also should present the structure of the information as clearly as possible, in order to help the user to understand the information.

Hyperwave is a large, networked hypermedia system which was originally developed at the IICM (Institute for Information Processing and Computer supported new Media), an institute of the Graz University of Technology, under the original name Hyper-G. Now Hyperwave is a commercial product. People can connect to a Hyperwave server from every where over the world, assuming their computer has access to the Internet or at least to their intranet if they are behind a firewall. Access to a Hyperwave server can take place identified or anonymous, entrance to several areas of the information can be granted or denied. Documents can be structured in a collection hierarchy or linked in a hyperlink structure. Sequences of documents and a powerful search complete the rich feature list of Hyperwave.

Hyperwave also is a multimedia system and supports many types of documents. For example text, image, movie, sound, 3D and postscript as well as a generic type for other document types are managed. In Hyperwave attributes can be stored for every document in the database. This meta-data can be used to add information such as keywords or the time where the object is created or modified. As a special feature, Hyperwave supports documents in different languages. This means, that one document can be stored in the database in several different languages, and displayed in the most appropriate language version depending on the user preferences.

This thesis describes the Harmony session manager. The Harmony session manager is the original UNIX - X11 client for Hyperwave and takes advantage of all the features Hyperwave offers, to supply the user with a convenient interface to browse the stored information. The Harmony collection browser visualises the collection hierarchy in a dynamically generated tree. The collection browser is further used to provide location feedback for documents visited during a Harmony session. In the Harmony local map hyperlink structure is visualised. It is further possible to selectively add child-parent relations and inline images to the map. The level of the structure depth and the number of displayed documents can be configured. A powerful user interface to the Hyperwave search functionality completes the navigational features of the Harmony session manager. The whole user interface can be displayed in a user configured language. The interface is currently available in German, English, and Japanese.

The Harmony session manager further has rich online authoring functionality. In the collection browser documents and collections can be inserted into the collection hierarchy. Documents can be moved and copied within the hierarchy. It is further possible to delete single documents and also whole collections performing complex searches before deletion.

The Harmony session manager starts independent viewer processes to display document contents. The Harmony session manager provides linking functionality to its document viewers. Links can be created defining source and destination anchors in the several viewers. This functionality is provided for document formats without native linking capabilities.

In order to allow to develop independent modules, Harmony supports an API (Application Programmable Interface). This API can be used by the modules to access the rich features of Harmony. The API defines functions to use Harmony's navigational features. It is further possible to use Harmony to visualise and edit documents.

2 Hypertext, Multimedia, Hypermedia

Most documents stored in information systems are text documents. In printed material, like books or newspapers, the structure is strictly limited to one page after another, so-called linear text. Hypertext, used in modern information systems, comprises of many pages of text, called nodes or documents. These documents are connected by so-called hyperlinks. Figure 1 shows hypertext in contrast to linear text. While reading documents links can be followed to other documents, parts of them, or also to other parts of the same document. Following links is just like navigating through the information universe.

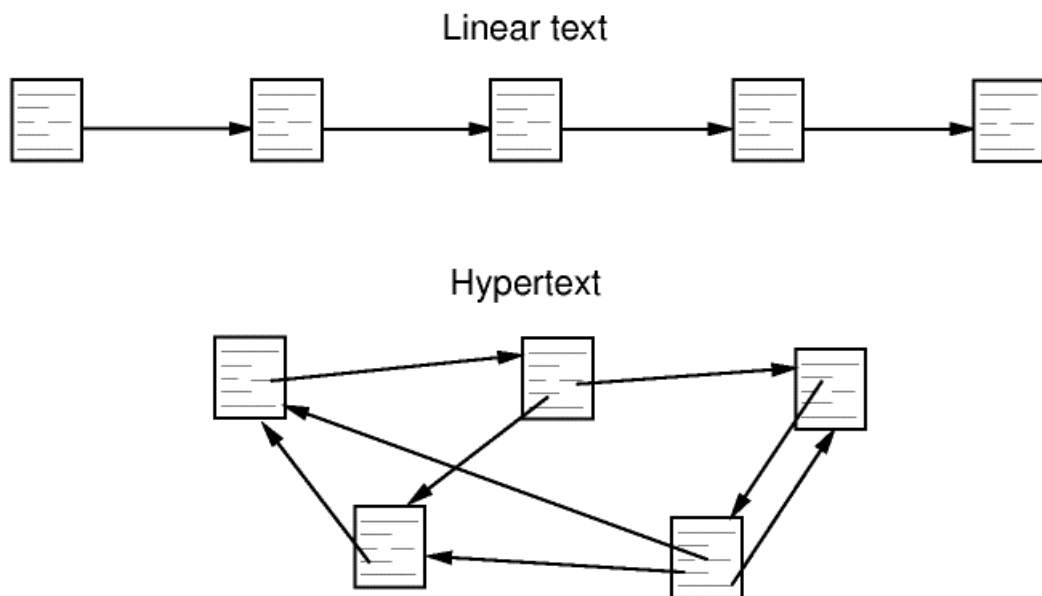


Figure 1 : Linear Text and Hypertext.

Hypermedia is the generalised form of hypertext. Links can not only point to text documents, they can also reference images, film or audio clips, so-called multimedia documents. With advanced systems and browsers it is further possible to create links in such multimedia documents. These may, for example, be areas in image documents or time slices in film clips. In this way, a hyperdocument, such as that shown in Figure 2, is constructed. Hyperlinks can also point to other services on the Internet such as FTP, Gopher, WAIS or News-resources.

Hyperdocument

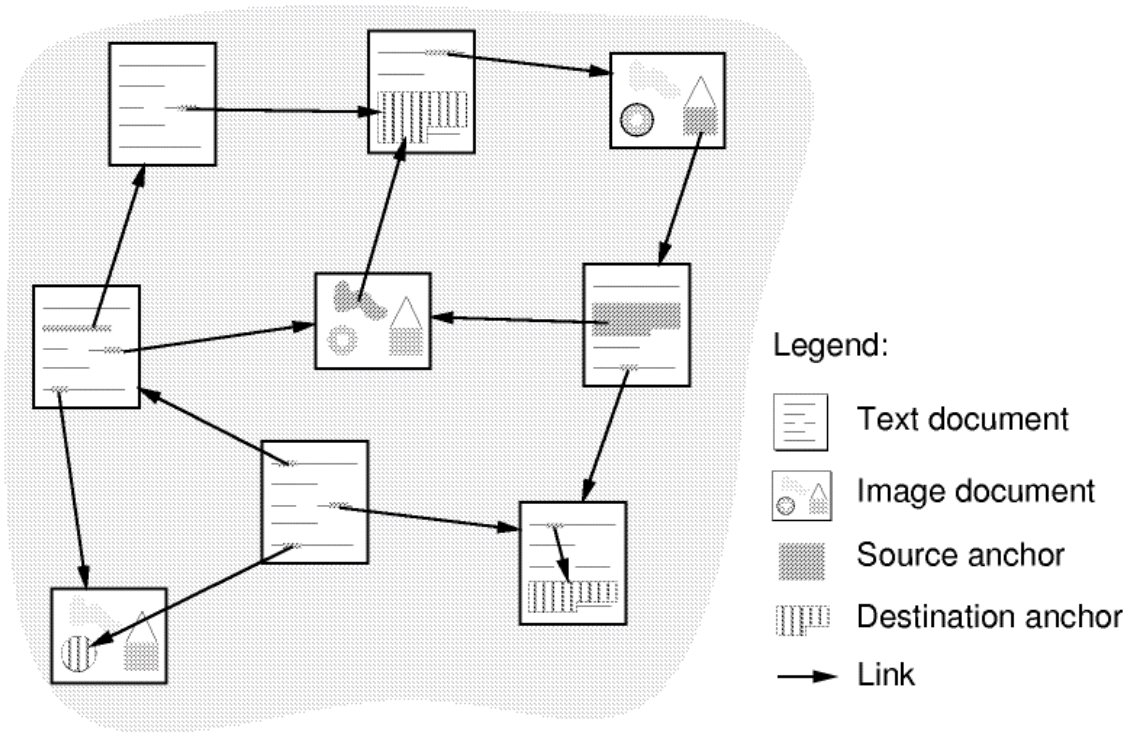


Figure 2: Node, Links and Anchors comprising a Hyperdocument.

3 Network Information Systems

From 1992 until 1996, the time when Harmony was being developed, there were several relevant information system in the Internet. There were Archie, WAIS, Gopher, WWW and Hyperwave. In the meantime the WWW has become the most important information system in the Internet. Hyperwave, due to its structure and facilities, has been targeted towards intranet use.

The following pages describe the information systems which were relevant at the time Harmony was developed.

3.1 Internet

The Internet was born about 20 years ago as a research project of the US Department of Defense's "Advanced Research Projects Agency" (ARPA). It was built to connect the ARPA and various radio and satellite networks together. A particular goal of the Internet was to build a network which is able to withstand the breakdown of several nodes during a nuclear attack. This was fulfilled by automatically rerouting messages depending on the existence of connections. The idea was to build a network where every computer can talk to any computer on the whole network. Services were added to the ARPAnet to complete it. These included remote login (telnet), file transfer (ftp), and electronic mail (email). In order to connect networks from different vendors, the Transmission Control Protocol/Internet Protocol (TCP/IP), was developed. In 1983 the whole ARPAnet was switched to this new protocol and the name Internet was born.

3.1.1 TCP/IP - the Transmission Control Protocol/Internet Protocol

The Internet is a packet switched network. The IP part of the protocol takes the message data and adds an envelope called packet. Each of these packets is provided with a source and destination address. They are routed over the Internet from one host to another.

The Transmission Control Protocol is based on the IP. It splits large messages into sequences of small packets, and numbers them. At the destination host, the packages are collected and if not in the correct sequence, reordered to build the original message.

On the Internet every host (or computer) has a unique address. These Internet addresses are 32 bit numbers, split into four 8 bit parts. A sample Internet address is 129.27.153.18. Since the Internet has to be used by humans it is inadequate to use numbers as addresses. Therefore a hierarchical naming scheme, the Domain Name System (DNS) was introduced. In this system each computer has an own human readable name. For example fiicmal01.tu-graz.ac.at is the name for the host with the address 129.27.153.18. With this naming scheme it is also possible to move services like FTP from one physical computer to another. The name where the service can be accessed stays the same, but the Internet address for the DNS name can be changed.

Each level in the DNS, the parts separated with dots, is called a domain. The top level domains (the right most) currently in use are; `com` (commercial), `edu` (educational), `gov` (government), `mil` (military), `org` (organisations) and `net` (network resources). Furthermore there exist around 150 country codes like `at` for Austria.

3.2 Basic Internet Applications

With the beginning of the Internet four basic Internet applications were introduced.

- **Remote Login** (telnet): Telnet is used to log into a remote host. It provides a basic, terminal style interface to the remote interface.
- **File Transfer Protocol** (FTP): FTP is used to transfer files from and to remote computers. There exist two modes to transfer data. ASCII to transfer text data and binary for program files. There are also two different access modes in FTP. Anonymous access and identified access with account and password. Usually FTP sites operates in anonymous mode.
- **Electronic Mail** (email): Email is the most used Internet application. It provides the possibility to send messages to other people in an easy to use and intuitive way and is much faster than surface mail and can be used from the computer at home. Email is not an “end to end” service like the previous applications. It is a “store and forward” service. Messages are sent from one computer to the next until the destination computer is reached. There exist many mail programs on the Internet. Most of them use the “Simple Mail Transfer Protocol” (SMTP). Most of the programs support features like forwarding, carbon copies, aliases and reply.
- **Network News** (news): News is a medium for people to discuss any topic on the Internet. There exists thousands of so-called news groups, dealing with nearly every topic. There are for example the groups `comp.lang.javascript` for the JavaScript language or `alt.culture.at` for discussions about Austria.

3.3 Archie

Archie was developed at the MC Gill University in Canada. It was a world wide search engine, for files public available on FTP servers. It was also possible to search in the indexes of Archie for files, which contains certain words in their description. The system of Archie was very simple. Operators of FTP servers were requested to register their server. In turn Archie contacted these servers via FTP and issued a directory listing using standard FTP commands in regular time intervals. The result was stored in local indexes and could be queried using Archie clients.

3.4 WAIS

WAIS (Wide Area Information Service) was a distributed text searching system, used in the Internet to find information in a collection of data. The WAIS server used an index built of (key)words. This index was independent of the kind of data which should be searched. It could be a collection of images as well as, typically an index for textual articles. The document data itself could reside elsewhere on the Internet.

In order to present data to people on the Internet through a WAIS server, an index had to be created by someone. For textual information, normally all words of the articles were indexed. The index could be queried by a WAIS capable client. The resulting list of matching objects was typically presented as a linear list, sorted by the score of the match. On this list further operations could be performed.

3.4.1 Scoring

Scoring was done by a number between zero and a thousand, where a thousand means the best match to the query and zero the worst. As an example to explain how this scoring was done, consider the query “University Graz and Telematik”. WAIS looked in its indexes, counts how often the word “University”, the word “Graz”, the word “and “ and the word “Telematik” appear in a document. The sum of these counts weighted on “how interesting” a word is, was used to calculate the score. After all sources were searched, a list of the document titles was displayed. This list was sorted by the score and normally truncated after about 15 to 50 entries. If a list entry was a text document, this document could be displayed by WAIS.

One problem of queries in this form was that words like “and” are used to build the score too. It may happen, that documents with many “and”s will result in a high score, also if it does not contain the words “University”, “Graz” or “Telematik” and therefore will not be of any interest.

It was further impossible to search for items added to the index after a specific time (for example last visited) and so it was hard to compute meaningful searches in growing databases with many matching documents.

3.4.2 Relevance Feedback

Nevertheless WAIS had a very powerful and useful feature, called relevance feedback. The result of a first search perhaps contained articles which perfectly matched the query but contained not enough information. WAIS could then be used to mark such articles or parts of them as relevant and extracted words for use in future searches. These searches could be performed on the same or on a different source. So with WAIS it was possible to begin with a simple search term, consider the results and use the features of WAIS to build more sophisticated queries to get the information someone was interested in.

In order to use WAIS for searches for some information, a search source (WAIS server) had to be specified. Specifying sources helps the user to find proper information by narrowing the search. The source could be one or more servers. But normally the proper servers were not known. This problem was solved with WAIS as well.

There existed a list of all public WAIS servers, which itself was a WAIS server. So this server, normally called “directory-of-servers” could be searched to find a proper list of WAIS servers to search in. The search in such a server should be more general than the real query, since the names of the servers normally did not contain words like “Telematik”. To find a proper list of servers, a broad search term had to be used to find not only specialised servers. The result of the query was a list of servers more or less matching the key words. This servers in turn could be used as source for future searches.

3.5 Gopher

Gopher [MA95] was introduced by the University of Minnesota. The goal was to build an information service with the main function to “go fer” things. This system was first aimed as a computer service, which gave several departments of the campus the possibility to offer information specific to themselves. This information should be maintained by the departments, but the several services could be accessed as if they were in one big database. Each department could have their own server and had full control over the server and the offered information.

As the next step, an application was developed which served these demands. To handle the amounts of data and to differentiate the services, information was structured in menus related to their topic. It was clear that this system could not only be used campus wide, but also world-wide. The only condition to satisfy was to connect the remote servers with one network, the Internet.

The great advantage of Gopher was that many services of the Internet, as for example FTP, WAIS, Telnet etc, were combined in one interface. All of the information, independent of type or protocol was structured in menus. The items could be menus again, data of some kind or also entry points to other Internet services just as Telnet and nevertheless other Gopher servers. But the user had not to know this details, Gopher handled this transparent and in a unique way.

3.5.1 Access to Gopher

It normally did not matter which Gopher server was accessed, but a server geographically near should be chosen. Nevertheless there existed two ways to connect to a Gopher server.

1. Public Gopher Access Sites

In order to connect to such Gopher sites, no Gopher client was necessary. The only you had to do, was to telnet to the host and login with the corresponding login name. In turn gopher automatically started and the Gopher session could begin.

Accessing gopher in this form was the simplest form, and it provided the user with all information, since Gopher not really needed a graphic terminal to operate correct. Only the interface was restricted to what a terminal offered. But they were able to give the feeling how Gopher works.

2. Gopher Clients

Another form to access Gopher was to use a Gopher client. Such clients existed for many platforms as for example UNIX, Macintosh, DOS or WINDOWS. These clients normally used the possibilities of their platform and offered a more convenient interface than the terminal based Public Gopher Access.

Independent of which platform or client was used, the information was always presented in the same form. So also when the environment had changed the information looked familiar.

3.5.2 Browsing through Gopher

3.5.2.1 Menus

All of the information in Gopher was structured in menus. A menu was typically a collection of items which were related in some manner.

The entry menu of a Gopher server normally gave you an overview about what services the server offers. Selecting one of the items led you to more specific information. This could be continued until a data item was reached. The information in the item could be displayed and, depending on the client, processed in some form, for example the information could be mailed to someone.

For access to other Gopher servers in general a menu item like “Other Gopher and Information server” existed somewhere in the menu structure. This menu item contained lists of other servers, structured possibly on location or content. But for the user, access to other servers was transparent and equal to access to local menu items or data.

3.6 World Wide Web (WWW)

The World Wide Web (WWW or W3) was originally developed at CERN in Geneva in 1989 as an information system for the particle physics community. WWW is a distributed, heterogeneous hypermedia information system. Hypertext documents are linked together with hyperlinks to build a large information universe. Hyperlinks are presented in browsers as “hot spots” which can be activated by clicking on them and in turn leading to related information. But not only textual information is available, also multimedia documents such as images, sound or movies can be accessed via the WWW.

The WWW consists of three key specifications. HTTP, URL and HTML described in the next sections.

3.6.1 HTTP

HTTP (Hypertext Transfer Protocol) is an ASCII protocol to define the communication between a WWW server and client. The client sends a request message to the server which replies with a respond message. The whole protocol is simple and mainly defined to transfer documents from a server to a client.

In the first version of HTTP, for every transaction an Internet connection was opened and closed after the end of the transaction. With version 1.1 of HTTP there exists the possibility to request more than one data item without closing the connection. This is especially useful for text documents with embedded inline images. The browser can transfer the text and the included images without having to reconnect, which leads to a speedup for the whole document transfer.

Sometimes it is desirable to set preferences for browsing a Web server, such as the preferred language or the last visited page. HTTP supports a mechanism, cookies where such information can be stored. Typically the server sets the information in the cookie. The next time the browser connects to the server, the cookie is presented and the server can react in an appropriate way. Nevertheless there exists no way for the server to initiate data transfer from the server to the client.

Modern browsers support a technique called “push technology”. Here the browser contacts the server depending on a user-defined schedule, to query the server for information changed since the last visit. If the information has changed, the appropriate pages are downloaded to a local file system. The user can in turn browse this local information in a fast way, or after the computer has been disconnected from the Internet.

3.6.2 URL

The URL (Uniform Resource Locator) is used in the WWW to uniquely identify pages on the Internet. But URL’s also can point to other Internet services such as FTP, Telnet, News, Gopher or WAIS. URL’s for the respective services are shown below.

- **HTTP:** http://host:port/selectorstring
- **Hyperwave:** hyperg://host:port/selectorstring
- **FTP:** ftp://host/filename
- **TELNET:** telnet://user@host
- **News:** news://newsgroup/article
- **Gopher:** gopher://host:port/selectorstring
- **WAIS:** wais://host:port/selectorstring

3.6.3 HTML

HTML (Hypertext Markup Language) [BCG94] is the most used document format in the WWW. It defines in an SGML (Structured Generalized Markup Language) [Gol90] like form, the structures in a document. In HTML so-called tags mark headings, paragraphs and text to be displayed in a bold font, for example.

```
<HTML>
<HEAD>
<TITLE>The title of the document</TITLE>
</HEAD>
<BODY>
<H1>The heading of the document</H1>
<P>The first paragraph in the document</P>
and some <B>bold</B> text.
</BODY>
</HTML>
```

Until version 4.0 of HTML there was no possibility to define the exact look and feel of the document. There was only mark-up to set some parts of the document to be displayed emphasised, but what exact emphasised means, was a matter of the browser. HTML 4.0 now also defines styles for the document.

This is done with CSS (Cascading Style Sheets). With CSS it is possible to define the exact look of a document and it’s content. Some of the parameters which can be set are.

- **FONT:** For fonts the size, font family, weight or colour can be specified

- **(Text)Blocks:** For text the used margin, border, padding, or line spacing can be specified.

With HTML 4.0 it is not only possible to define the look of elements on the page, it is also possible to set the position of text or blocks of text. Figure 3 shows a HTML page created utilising the capabilities of HTML 4.0, CSS and JavaScript. The drawback for the moment of CSS and positioned text is that there exists no browser which supports the full standard.

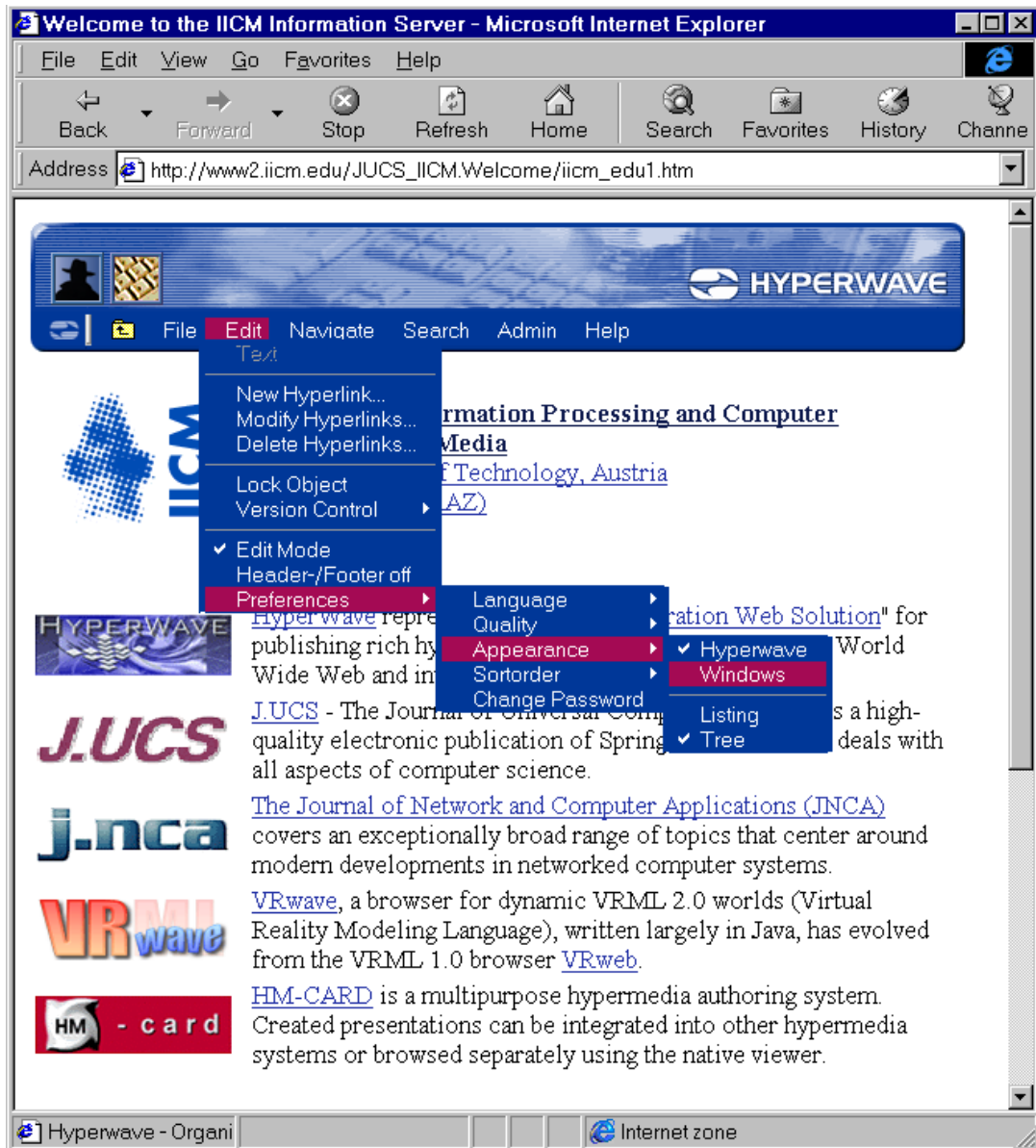


Figure 3: Page designed using HTML 4.0.

4 Hyperwave

Hyperwave [Mau92][KM93] is a large, networked hypermedia system which was originally developed at the IICM (Institute for Information Processing and Computer supported new Media), an institute of the Graz University of Technology, under the original name Hyper-G. Now Hyperwave is commercial product.

Hyperwave combines the features of WAIS, Gopher and World Wide Web as well as some additional features to offer the user a convenient way to find information. Hyperlink navigation, hierarchical structuring of information, sequences and various search facilities are provided as standard.

4.1 Hyperlink Navigation

In Hyperwave the user has the possibility to navigate, by simply browsing through the information and following hyperlinks to other documents. This is done by activating a link by clicking on a “hot spot”, a so-called source anchor. In turn, a certain document, where the link is connected to is displayed. The destination may be a collection, cluster or a whole document, but it is also possible to point to an area in a destination object, the destination anchor. In addition, Hyperwave has the possibility to follow links in the reverse direction. In other words, it can answer the question, “which other documents refer to the current document”. This feature can be used by a client to present a list of documents which refer to the current document or display a map of the “link environment” around the current document. It is also useful for maintaining database consistency.

Browsing through information rather than searching for it is an easy way to have a look at information and has already successfully been used in a number of information systems, for example WWW, and is indeed very well suited for applications such as encyclopaedias, user manuals, computer based teaching material, presentations, on-line help systems and the like. In Hyperwave links are supported from and to text, images, movie, sound, postscript and VRML documents. Nevertheless, navigation in this manner raises some problems when applied to large amount of data. This effects has already been discussed in earlier sections (“lost in Hyperspace” for the user, and “spaghetti links” for the author). In order to reduce such problems Hyperwave provides some additional structuring and navigation facilities. In Hyperwave all documents are members of collections. Collections group related documents in different information areas. Additionally all documents are automatically searchable by there content or a set of attributes. Guided tours in Hyperwave, called sequences, are a further facility to present information in a structured way.

4.2 Object Attributes

In Hyperwave all documents have an assigned set of attributes. These attributes describes the document just like the entries in the card files in a library. Some of the attributes are pre-defined by Hyperwave such as Title, Keyword, Author, TimeCreated and some more. Other

attributes are used by the system to identify and manage the information stored in the Hyperwave server. The Rights attribute, for example, defines a set of users or a group which can access information. TimeCreated and TimeModified are created and updated by the system to allow finding information created or modified in, after or before some given time.

For the administrator it is further possible to define custom attributes to fit the requirements of the particular information system. These attributes can be used to visualise the documents in lists and are also used to search for specific documents. The search mechanism will be described later. For a detailed list of Hyperwave attributes and their values refer to [Mau96].

4.3 Collection Hierarchy

Orthogonal to the hyperlink structure every Hyperwave document is contained in one or more collections. These collections themselves are members of other collections. Inserting documents and collections into one or more parent collections leads to a collection structure as for example shown in Figure 1. Since it is possible to insert the same document in more than one collection, the hierarchy is not simple a tree, but in general an acyclic directed graph of information.

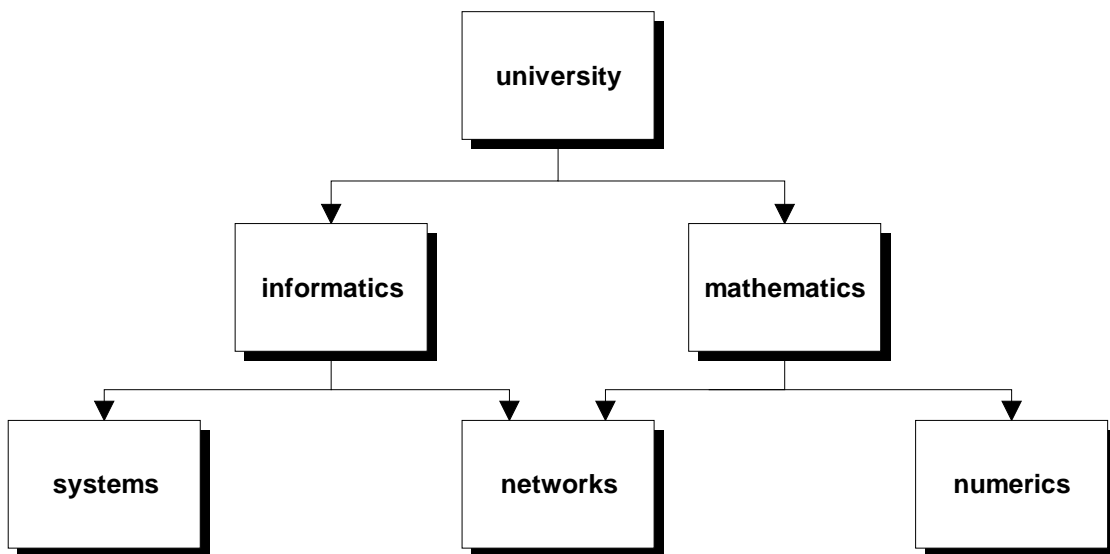


Figure 4: Collection Hierarchy.

Some of the collections in the Hyperwave server have a special meaning. The nomenclature used for these collections is as follows.

Root Collection

The topmost collection in the hierarchy is a server-generated collection, the so-called root collection. All objects belong to the root collection .

Hyperroot

If the server is in the so-called distributed mode, there is one further collection above the root collection, the Hyperroot. This collection contains as artificial members all Hyperwave servers which are known by the system, the server pool.

Home Collection

Every Hyperwave user may have their own home collection. This home collection can be compared with home directories in a UNIX [Gil92] environment. Data of personal interest as well as links to other collections and documents on the Hyperwave server, and also links to information systems can be placed into the home collection. In this form the home collection can also be seen as a “Hot List” or “Bookmarks” stored on the server. Not only a sequential list of documents can be built. Documents and links can also be placed in sub-collections and in this form built a hierarchy of Bookmarks. For unidentified users, a default home collection is used typically the root collection of the server.

The current implementation of Hyperwave defines several types of collections with different behaviour when visualised.

4.3.1 Collections

Ordinary collections as described above are used to structure information. When visited typically all documents and collections are presented as an ordered list of items. If one of the documents contains the special attribute ContentHint the behaviour of the collection can be changed. The document with the value “CollectionHead” for this attribute is opened if the collection is entered. The order of the items in which they appear in the list may be defined statically by the “Sortorder” attribute in the parent collection, or dynamically by the client. Sorting is based on attributes of the collection members such as Title, Author, Sequence number, CreationTime and other attributes.

4.3.2 Cluster

A cluster is a collection with the attribute:

```
CollectionType=Cluster
```

When a cluster is visited all or some of its members are visited/displayed too, dependent on the type of the members. This collection type serves two purposes.

1. **Multimedia documents:** A cluster may contain different types of documents, which are visualised at the same time. For example a text document and a image document are shown, while a film document is played.
2. **Multilanguage documents:** If a cluster contains one document in different languages, the document in the most appropriate language for the user is visualised.

Multimedia and multilanguage documents can be combined in one cluster and so form multilanguage multimedia documents. The exact rules for the treatment of clusters can be found in Appendix E.

4.3.3 Sequence

A sequence is a collection with the attribute:

```
CollectionType=Sequence
```

A sequence collection is like a guided tour prepared by an expert. A set of documents with an predefined visiting order comprises the tour. The tour is taken by following automatically generated “next” and “previous” interface elements. Several tours may contain the same document and therefore when meeting this document, the user has the decision which tour to follow. Also every time there is the ability to use one of the other navigation methods. Sequences are typically used to present a compilation of highlights about a particular topic, very much like taking a guided tour of the sights of a city. They are also useful for preparing hypermedia presentations of existing material. An example of a sequence is shown in Figure 5.



Figure 5: Sequence in Hyperwave.

4.3.4 MultiCluster

A MultiCluster is a collection with the attribute:

```
CollectionType=MultiCluster
```

A MultiCluster is a document consisting of many sub-documents, which are concatenated together to form the whole document. A document containing a list of addresses and phone numbers of the employees of a company can be implemented using a MultiCluster. Each address entry is maintained by an employee in a separate document with appropriate access rights. All the files are inserted into the MultiCluster. When opening the MultiCluster, all the documents are concatenated together to comprise a single list. The order in which the files appear can be determined by the Sortorder attribute set on the MultiCluster.

4.3.5 AlternativeCluster

An AlternativeCluster is a collection with the attribute:

```
CollectionType=AlternativeCluster
```

An AlternativeCluster is a collection of documents from which one or zero documents are chosen by certain criteria. The behaviour of such a cluster is more like that of a simple document than a collection. This makes it possible to make inline image links to AlternativeClusters. The settings for how documents should be chosen can be set in the user's user record in the server. It is also possible to set default values in the configuration files for the server. Two typical applications of an AlternativeCluster are:

1. To allow a choice between documents which have different levels of quality but the same type (MimeType). A typical application are images with different resolutions or colour depth which represent the same content, but consume different amounts of space. A user with low network bandwidth is able to receive an image with less quality in a reasonable amount of time. Others which are locally connected to the server (intranet) will choose images with full resolution and colour depth.
2. To allow a choice between documents in different representations which do not have the same type (MimeType)
Sometimes it is necessary to offer information in different data formats or media types. For instance, an announcement may be available in PostScript, HTML, sound or even video. For a user it is possible to set a preferred document type.

When the user accesses an AlternativeCluster, one of its members is chosen by the specified criteria. In the Attribute PrefMimeTypes of the user's user record, a list of preferred MimeTypes with corresponding qualities can be specified. The exact rules for document selection in an AlternativeCluster are described in Appendix E

4.4 Purposes of the Collection Hierarchy

Documents or better the information stored in documents, can be structured using the collection types introduced in the previous section. The hierarchy defined by these collections serves the following purposes.

- **Navigation:** A user may navigate through the information using the collection hierarchy. When entering the server an initial list of collections and documents is presented. Users visit a collection by opening it, and all its sub-collections and documents are displayed.

This step is continued until the necessary information is found. In this way the user gets an overview of which and how much information is available in a collection. In turn the user can decide to have a closer look at a specific part by visiting further collections or documents. It also helps the user to keep track the position of the information, since every document is contained in at least one collection. With this structure it is possible to determine the position of a document or collection in the collection hierarchy. The probability to “become lost in Hyperspace” is reduced. In Figure 8 the implementation of this kind of navigation in the Harmony collection browser is shown. Navigation is not restricted to this navigation metaphor. At every point Hyperwave users may use search facilities, hyper-navigation or sequences, to retrieve the appropriate information.

- **Access Rights:** Hyperwave is a multi-user system and can be accessed by many users. Assigning access rights to objects allows the restriction of read and/or write operations for parts of the information to authorised users or user groups. A collection is not displayed in collection listings, if there is no right to access the collection. This mechanism is necessary when Hyperwave should be used for “secret” and public data on the same server. Hiding data by assigning access rights is also appropriate for private data which other users should not see.
- **Search Scope:** By activating or deactivating collections, search operations can be performed on certain parts of the information hierarchy, and in this way reduce the matches of a search operation. Only members of “active” collections and members of them are returned as the result of the search. For example, by not including collections with dictionaries when knowing the meaning of the search term, can drastically reduce the number of found items.
- **Access Limitations:** By limiting the simultaneous access to collections to a specified count of users, it is possible to simulate library functionality. In this form only a certain number of users can access the members of a collection, for example a dictionary or an encyclopaedia, at the same time. If too few licences are available, more have to be obtained, by buying further licences of the book from the distributor.

4.5 Multilinguality

Users typically prefer to read documents written in their native language. An information system therefore ideally should support a mechanism for multilinguality of documents. This means that the system should be able to store documents in different language but with the same content in an easy to use and intuitive way. The system should also be configurable to retrieve documents in a user-specified language if available. If the document is not available in the preferred language, some fallbacks should be defined. Hyperwave meets these requirements in the following way.

4.5.1 Multilinguality in Hyperwave

In Hyperwave each object is defined by a set of attributes. One of the attributes is the Title attribute, which is a language-dependent attribute. A language shortcut defining the language of the title is added to beginning of the title value. Valid prefixes include:

- en for English
- ge for German

- jp for Japanese

The language-dependent title attribute can be utilised by a collection browser. Every collection and document can be visualised using the title attribute. The language of the title can be chosen from those available according to a user-provided language priority list.

A similar mechanism can be applied to display the content of a document in a preferred language. In Hyperwave this functionality is covered by a special collection, the Cluster. Clusters may contain documents in several languages but with the same content. A document is language-dependent if it has only one language-dependent title attribute. A client can select one of the members of a cluster considering the language information in the title attribute and display it.

4.6 Search Facilities in Hyperwave

Documents which are inserted into a Hyperwave server are automatically searchable. Just after insertion the document can be found using the Hyperwave search mechanisms.

There exist two basic methods for indexing and searching documents in Hyperwave.

4.6.1 Full Text Search

Text documents are automatically entered into a full text index upon insertion into the database. This means that every text document and documents with extractable text is examined and every word is put into a full text index. Full text queries can then be performed on these documents. Hyperwave offers two full text search engines.

- The native Hyperwave full text engine. This engine supports fuzzy boolean queries and WAIS-like nearest-neighbour searches based on the vector space model. For more information about full text search refer to [Fas93]. Supported document types are text documents in the formats HTML and HTF. For other document types there exists a method to configure an external filter to extract the words to be indexed.
- The Verity Search 97 engine. Verity supports many types of documents, including HTML and many Windows Office formats like Word and Excel. Image documents are indexed by the comment stored in the files and the publishing format PDF and many more are supported. For more information about Verity Search 97 engine refer to [VER98].

4.6.2 Search on Attributes

A second method to search for documents in Hyperwave is to search on the attributes of a document. Every object stored in Hyperwave has assigned a set of attributes. Some of these attributes are for system use only, but many of them are also used for searching. Title, Author, Keywords, Name, Modification Time and Creation Time are some of the attributes automatically indexed by the server .

If the system-defined searchable attributes are not enough to fulfil search requirements, the administrator of the server can configure some freely definable attributes to be indexed by the server. With this extension it is possible to create personalised servers with specialised search capabilities.

The Hyperwave server in turn offers a rich set of functions to search for these attributes. Search values for one attribute can be combined with boolean operators, as can search on several attributes. Typical queries might be:

“Give me all documents with *Hyperwave* and *Graz* in the title”

“Give me all object with *multimedia* in the keyword(s) where the author is *Nielson*”

“Give me all *images* with keyword *Dali*”

“Give me all documents which have been created between *98/01/01* and *98/01/31*”.

4.6.3 Search Scope

The above introduced search capabilities offer a wide range for searches performed on one server but Hyperwave support further facilities for searching.

Distributed Search

The search can be expanded to search across several Hyperwave servers. In this case, the server where the search is initiated broadcasts the search parameters to the other servers involved. The search is also performed in the local database if requested and the result collected. In the meantime the remote servers perform the same search in parallel. When each search has finished the result is transferred to the requesting server. When all remote servers have submitted their results, the combined list of matching objects presented as one search result.

With this distributed search mechanism bandwidth and computing time is saved. Bandwidth because only the search parameters are transmitted to the remote servers, and only the list of matching documents is returned. In contrast, search engines on the WWW have to periodically request the document, and if changed transmit it, in order to index the document. User time is saved because the search is performed in parallel at all servers at the same time.

For all the above presented search facilities the search scope for both fulltext search and attribute search can be defined in a flexible way. The search can be performed on

- The local Hyperwave server. All documents on the local server are possible candidates.
- A single collection on a single Hyperwave server. Only documents contained in the collection or a sub-collection are found.
- A set of collections on a number of Hyperwave servers.
- A number of Hyperwave servers in their entirety.

4.7 User Management

Hyperwave is not only a simple document storage and retrieval system, Hyperwave also can store and manage user information. Data records of a user are stored as Hyperwave objects, the user object. The information stored includes:

- Login name
- Passwords
- Auto identification hosts
- Group membership (see below)

- Accounts

Several users can form a user group which is also handled by Hyperwave. User groups help managing access rights in systems with many users. For a more detailed discussion of users and groups see [Win95]. Hyperwave also can facilitate external user databases for identification and authentication of users. Up to now, gateways to LDAP, X.500, and Windows NT user database exist.

The major purpose of users is to login as a user. This is necessary to control access to documents and collections. Every object in Hyperwave is owned by one user. The owner of an object, and also the privileged “system” user can regulate access to this object. By assigning read, write and delete rights for one or more users and one or more groups.

It is also possible to assign a “Home collection” to a user. This home collection can be compared with the “home directory” in an UNIX system. It serves the same purposes. Users can store private data in their home collections.

The documents stored in the home collection can also be used like folders with bookmarks. When these bookmarks are further grouped, for example by category, and stored in different collections, the home collection or a sub-collection of it, forms a hierarchical tree of links to documents of the user’s interest.

4.8 Hyperwave Messages

In contrast to HTTP, the protocol between Hyperwave and its clients is connection-oriented. A client initiates a session with one server. All further operations are sent to this server. This means that the server is aware of all clients currently connected, and if they are identified, of the users or at least their synonyms.

In this manner the Hyperwave server and the connected clients have the possibility to send messages from the server to the client and vice versa. Such messages could be used to:

- Pass notes between two or more clients or users.
- Send broadcast messages for updates or changes in a collection.
- Send a warning message about an upcoming server shutdown.

Harmony used these messages to build a chat system. See Section 5.2.6 for more details.

4.9 Interoperability

Interoperability is a very important feature of information systems. Much information is already stored in other legacy systems. Transparent interoperability improves the usability of an information system, since the whole mass of data can be accessed through one interface.

Hyperwave has gateways to Gopher, WAIS, WWW and FTP in order to retrieve information from them. Hyperwave servers are able to store and handle objects which operates as an entry point to the different information systems or also as pointer to remote documents. Clients do not have to know the details of the external protocols, they only have to connect the Hyperwave server, which in turn handles the request, translates the protocol and passes the retrieved data to the client. Hyperwave also has a built-in cache which stores recently received documents from other Hyperwave servers or from Gopher, WAIS, WWW or FTP servers in the local document cache. When requested again by a client, the data is transmitted from the

local store to reduce access time and save bandwidth. For Gopher servers the menu structure is transformed into a Hyperwave collection hierarchy. WAIS queries are treated as Hyperwave queries and WWW documents as if they are stored within Hyperwave.

When Hyperwave is accessed by WWW clients, a HTML document is created dynamically for each collection. This document contains links to sub-collections and documents. For all document types except text and image documents, the hyperlink information is lost. However, the hyperlink information can be accessed by dynamically generated HTML pages. The Document Reference Pages contains links to documents which refer to the current document, or are referred by the current document. For Hyperwave clients, documents stored in the WWW look like normal objects stored in the Hyperwave server.

4.10 The Architecture of Hyperwave

Due to the demand of many concurrent users and the large number of documents, the client / server model is used in Hyperwave. In contrast to terminal-based systems, where every keystroke have to be sent to the host, and also all of the computations, formatting or scrolling text, entering keywords, decoding information etc., has to be done at the host, these actions are performed by the client in the client / server model. This model has some principle advantages:

- The response time to keystrokes is normally small, since they are processed in the client and in turn reasonable feedback (time) for user actions can be achieved. This is very important for user interfaces, where response times to actions over 0.1 seconds are confusing for the user, and therefore normally not accepted.
- Bandwidth is saved, because keystrokes or mouse movement do not have to be transmitted to the (remote) host. Only if new data is required, is it fetched from the server, usually in large blocks. Transmitting large blocks causes less overhead than small packages would.
- The individual capabilities of the local hardware and software environment (for example windows, icons, true colour displays, graphic accelerators, “unorthodox” in- and output devices etc.) can be used by the client. In terminal based systems the smallest common denominator of the capabilities have to be used.
- When many users are connected to the host at the same time, in the client/server model, the necessary computing work is shared among several computers. This results in small server machines. The server does not have to be a “super computer”.

Figure 6 shows the architecture of an Hyperwave server. The Hyperwave server contains three different servers. The full text, document and object servers, but for a client appears to be a single server.

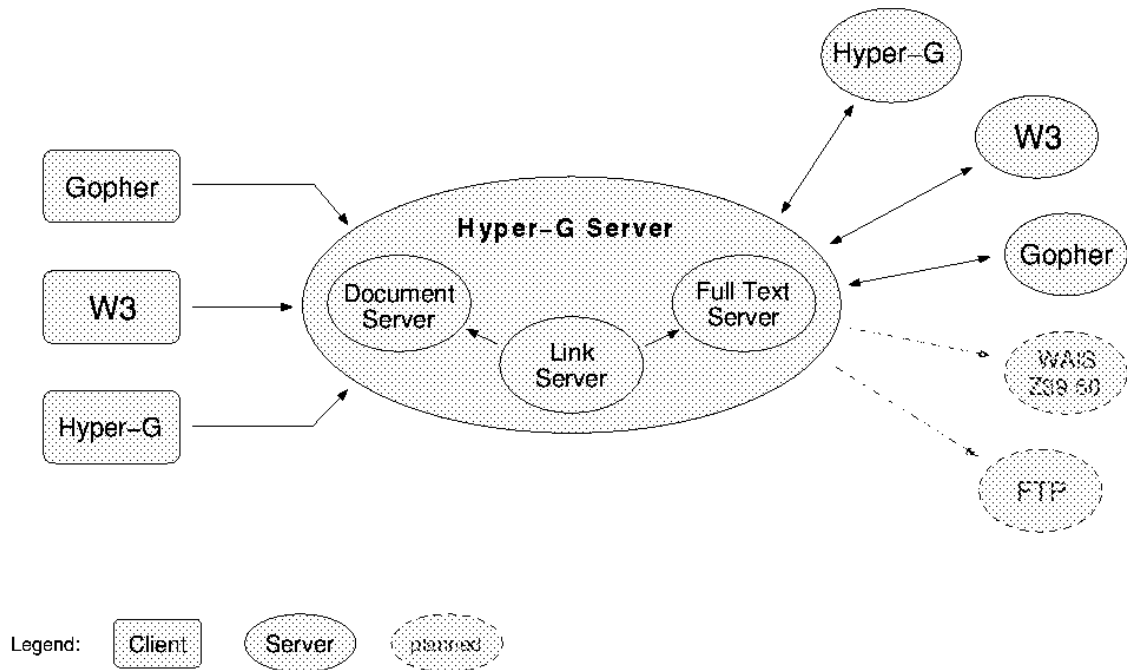


Figure 6: Architecture of Hyperwave.

4.10.1 Full Text Server

The full text server maintains inverted indexes of keywords. These keywords are collected from the textual contents of documents during the insertion process. The words are of course removed from the indexes if a document is deleted. For every new language, a separate index is maintained. In all languages a set of words which are not indexed is defined. These words, the stop list, are frequently used words, and unsuitable for searching.

For documents types where Hyperwave does not know how to extract words, external filters can be configured. These filters extract words from the document which are then added to the indexes. External filters can also be used to index documents in "strange" languages. In Japanese documents, for example words are not separated with spaces, and therefore hard to identify and extract.

The native full text server is able to handle fuzzy boolean queries and WAIS-like nearest-neighbour searches based on the vector space model. For more information about the full text server see [Fas93].

It is also possible to configure the full text server to use the Verity Search 97 [VER98] engine. Verity is able to process and index about 200 document types, but at the moment is not available on all platforms.

4.10.2 The Document Server

The document server stores all local documents. It processes the transfer of documents to the client. The document server is also responsible for receiving documents from clients and storing them in the local file system. It also fetches and caches files from remote servers.

When a client requests documents, the document server checks whether the data is in the local server and if so, transmits it to the client. Otherwise, if the document is stored on a remote server, the responsible server is contacted and the data is fetched. The incoming data is

forwarded to the client as well as stored in the local document cache. Thus, often used documents are held in the local cache and on further requests transmitted directly from the cache.

4.10.3 The Object Server

The object server is the most visible part of Hyperwave. Clients connect the object server in order to retrieve object descriptions, fetch children, perform queries or initiate the transfer of documents. The object server stores descriptions of documents, links, anchors, collections, users, sequences, remote servers etc., in an object-oriented manner. Descriptions consist of several attributes and are stored in a database. For more information on Hyperwave objects see [Kap91] [Mau96]. Relations between the objects are also stored in the database. Typical relations are:

- Which objects belong to a collection (child parent-relation).
- Which source anchors are connected to which destination anchors or documents.
- Which source anchors are part of which document.

As the object server has knowledge about these relations, it is able to maintain the consistency of the database and the relations between the documents. For example, it can remove source and destination anchors when the corresponding document is removed.

In this manner “dangling links” in the database are prevented, or more exactly, are known by the server and for viewing purposes are not shown. It is further possible to automatically maintain and generate “next” and “previous” elements, when a document is deleted from or is inserted into a sequence. For an example see Figure 5.

The Link Database

In Hyperwave links are strictly separated from documents. The link information of hypertext documents is extracted during the insertion into the database. Links in documents, where links are not part of the format specification can be created and deleted using Hyperwave and Harmony. The link information is stored as an object in its own right and is maintained by the object server. All Hyperwave mechanisms, such as access rights, custom attributes, and attribute search can also be used with links. The advantages of a link database include:

- Links can be created and modified in documents, even if the document itself is not modifiable. For example, when documents are stored on a CD-ROM, or the document resides on another write-protected medium. It is further possible to create public or private notes or annotations to documents where no write access is possible.
- A second advantage of separate link storage is the ability to support bi-directional links. In contrast to “normal” links, links in Hyperwave can be followed and tracked in both directions. Unlike traditional web servers, Hyperwave therefore has capabilities to answer questions like:
 - Which documents refer the current document?
 - Are there annotations to the current documents?

Some problems can be solved with this model, which is hardly possible in systems with uni-directional links.

1. Whenever a document is deleted, the system is able to generate information about other documents referring to the document in question. Hence references to to-be-deleted documents can automatically be identified and if requested removed from the link structure, thus maintaining link integrity. This is especially important in large multi-user systems, where the person deleting a document can not be expected to know all the links to the document. The same problem occurs when a document is automatically removed or inaccessible because of its expiration date. A system with bi-directional links may flag or delete links to such documents automatically.
2. Bi-directional links allow advanced user interfaces to draw local maps like the one shown in Figure 8. Documents and references between them are visualised. Such maps can also be created without bi-directional link databases. However, every referenced document has to be fetched and parsed for links, which is time consuming wastes bandwidth. With the help of a link database this can be done using a single request do the database.
3. A further advantage of the storage of link objects in a separate database is the ability to assign keywords or titles to link objects, or more exact to source or destination anchors. These attributes are, as for all other Hyperwave objects, indexed and searchable.
4. Link objects can also be members of collections and are visualised when visiting the collection. In this manner, different entry points into one large document can be defined and put together in a collection as a kind of index.
5. Links in Hyperwave can also have assigned access rights. This means when documents are visited by two users with different access rights some links may not be visible for one of the users.
6. It is possible to create typed links in Hyperwave. A link in Hyperwave therefore not only is a reference to another object, it also specifies the type of reference. A link, or rather the document behind the link, can be an annotation to part of a document or also a question about the document. This information is also used in the Harmony local map to restrict the map to special types of link.

Object ID's

One of the major task of the object server is to assign identifiers to every object in the database. It has to ensure that no ID is reused for a second object. The Object ID is used as an identifier for the object when specifying objects for operations.

With the mechanism of ID's one problem arises. Objects on different servers may use the same IDs and the uniqueness of IDs between server boundaries can not be guaranteed. To distinguish between objects with the same ID but residing on different servers a second ID, the server ID, is added to the object ID when accessed by a client. These two ID's are concatenated to one server-embracing ID. The server ID is built from the IP address of the birthplace

server. An object on the local server has the server ID zero. With this Global Object ID, every Hyperwave object spread over the world is identifiable unambiguously.

The descriptions of the documents stored in the object server consists of several attributes, for example author, title, keywords, creation time, modification time, access rights, etc. Some of these attributes are assigned by the user. Others, such as creation time and modification time are automatically generated and maintained by the server, when a document is inserted or modified.

A description exists for every document in the database and when an attribute search is performed, the queries are based on these descriptions. Every kind of document is searchable, as long as the author assigns proper attributes. For example, image documents may have assigned keywords like “museum” and will be found by an attribute query for “museum”.

The object server supports complex boolean queries on different attribute values, as well as boolean combinations for several attributes. Typical queries may be “Give me all documents with Hyperwave in the title, created by user fkappe and created between 98/01/01 and 98/6/12” or “Give me all images with Dali and Escher in the keywords”.

In contrast to Gopher where special “search items” have to be created for searches, in Hyperwave the client and in turn the user decides where and how to search. The search is independent of the information the user is currently browsing. The client only has to provide an interface for the search.

Searches can be performed by the object server on different areas in the server:

- The search can be performed on one specific collection, for example a dictionary. When one, or as described later, more collections are named to search in, all members of these collections are searched. When a child is itself a collection, its members are also searched recursively.
- The search scope can be widened to include collections on remote Hyperwave servers. When searches on remote servers are involved, the queries are sent to the remote servers and the search actions are performed in parallel and independently from each other.

Access Rights

In the most general case, a Hyperwave server can be accessed from everywhere by anyone. In general not every kind of information is public information. Some areas or documents like reports or some other data should only be accessible by particular users or groups of users. In Hyperwave this problem can be solved by assigning access rights to objects.

Rights can be granted to single users or to user groups just as is done in the UNIX file system. One user may belong to one or more user groups. In Hyperwave access rights can be assigned to collections, documents and links. Read and write rights are handled separately. It is also possible to assign different rights for operations like copy or move. With this rights schema a document can be readable but the user is not allowed to modify it. The right to see a collection enables a client to display the collection, but this does not imply that all members of the collection are accessible. In other words disabling a collection by assigning access rights only disables the navigation into this collection. If members of this collection are found in an alternative way, the access right for the object is checked independently of the right of the containing collection. This behaviour may be more clear if it is taken in account, that it is possible for one object to be contained in more than one collection. In this way the appearance,

especially the number of visible members, of a collection can vary when accessed by different users.

Hyperwave has built-in user management. Information about users and groups is stored in objects handled by the object server. Stored information include the user name, login name, group membership, and so on. It is further possible to utilise external user databases. Information in the external databases is used to identify a person on the server. User management is described in detail in Section 4.7.

Licensing

A further feature of the object server is the licensing mechanism. This mechanism is used to grant access to a particular part of the information only to a certain number of users at the same time. When a user visits such a collection and the configured limit of users is not already reached, the user has the right to visit all members of the collection. However, when the limit is reached, all further access to the information by other users is denied. After a specifiable time of inactivity, the right to access information is taken away. Inactivity in this sense means that no (sub)documents or (sub)collections have been requested. If so the counter for simultaneous users accessing the information is decremented and another user can have access to the information tree.

This mechanism is analogous to borrowing a book in a library. When a book is frequently requested, one copy would not be enough. The library has to buy further copies to satisfy demand. The same can be implemented using Hyperwave. When someone buys further licenses, the limit counter for simultaneous users is increased, but no further (disk) space is needed. In this form publishing houses can sell books in electronic form online on the web, which than can be read using the Internet, but with limited access.

Note that since version 4.0 of Hyperwave, licensing is no-longer supported using the WWW gateway. With Harmony this mechanism is still available, but Harmony is also no longer part of the distributed Hyperwave package.

Accounting

In Hyperwave every document can have an assigned price. When a user requests a document with an assigned price, the price of the document is subtracted from the account of the user. If the user's account is too low, a warning message is displayed, and access to the document is denied. The account of the user is stored in the user record in the Hyperwave server and can only be set by a system administrator.

Accounting can be used to offer billable information in a publicly accessible server. Particular information can only be accessed if the user has enough credit in the account.

5 The Harmony Session Manager

In the first part of this chapter, the architecture of Harmony is described. In the second part of this chapter, the user interface of the Harmony Session Manager is presented. Navigation in the collection hierarchy using the collection browser is described. The local map is presented as a tool to retrieve and visualise relationship between documents. It is explained how Harmony facilitates the rich search capabilities of Hyperwave. It is further shown how Harmony treats multilinguality, using multilingual user interfaces and displaying documents in different languages. In addition, a short description of the Harmony document viewers is included.

5.1 The Architecture of Harmony

From the technical point of view, Harmony is divided into two software parts: the session manager, and the document viewers. The session manager and the document viewers are implemented as separate processes. The communication between each individual viewer and the session manager is done via the Harmony native Document Viewer Protocol DVP. Figure 7 shows the Harmony software architecture.

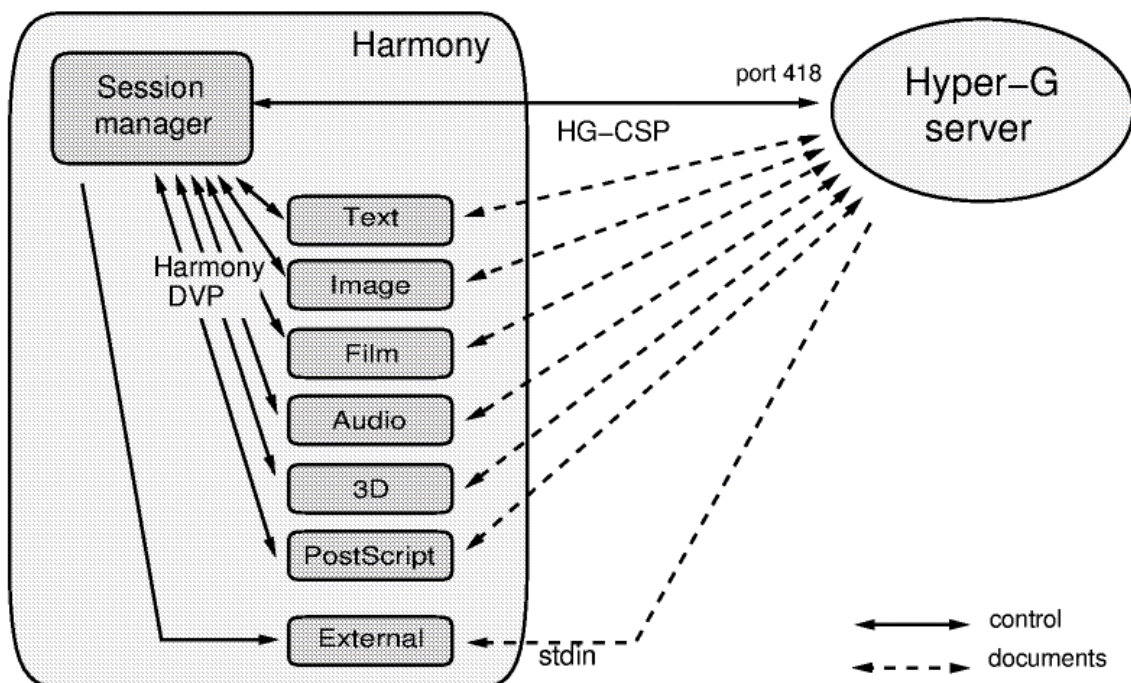


Figure 7: Architecture of Harmony.

Session Manager

The session manager process is responsible for every kind of transaction with the Hyperwave server. At start-up it initiates the connection to the server, and in turn all operations with the server are carried out over this connection. Further all object, collection and structure specific operations and visualisations are handled by the session manager. For example, the collection browser and also the local map are part of the session manager process.

When the user initiates the display of a document, the session manager starts the appropriate document viewer, and forces the viewer to retrieve the document data and to visualise the information. On the activation of a link, the viewer sends the request to the session manager, which handles all necessary operations.

Document Viewer

For each document type a separate viewer process is started by the session manager. When a document has to be displayed, the viewer retrieves the document data over a separate connection. When received, the data is visualised by the viewer. The viewer further adds link to the display, if any are present.

The separation of viewers and the session manager in different processes has some advantages compared to a single process concept.

- New kinds of documents can be added by simply implementing a new viewer with the standard communication functions.
- An old version of a viewer can be updated by replacing the program binary, without influencing other viewers or the session manager.
- When loading documents, each viewer handles its own data transmission. In the meantime, the user can continue browsing on the server, or may peruse at other documents. The only process which is busy is the one document viewer. All other viewer processes and the session manager are free and ready to be used, the user does not have to wait until all document data is transmitted.
- On multi-processor computers, different processes can be handled by different processors, which can drastically improve the performance. This is especially of interest for movie documents, which are particularly computing intensive, or also for 3D documents.

5.2 User Interface

Harmony, for the user a unified user interface, consists internally of two major parts. The session manager is responsible for establishing a connection to the Hyperwave server. All transactions with the database are carried out over this connection. The session manager is further responsible for presenting interfaces for navigation in the collection hierarchy, for visualising relationship between documents and for performing searches.

The Harmony document viewers display all kinds of documents, visualise links within documents, provide aids to create such links, and also edit documents.

For the user these two parts are indistinguishable. All interface elements, both in the different viewers and the session manager, adopt a consistent look and feel. For example, in all the viewers, menu items with the same meaning are placed at the same position in the menu structure, and links are chosen and activated with the same interactions. Also user interface elements, such as buttons or status bars are placed in the same position within the windows. This gives the user the feeling of a single integrated interface. For further readings refer to [MAN95] [And95].

5.2.1 The Collection Browser

The Harmony collection browser uses the structure information from Hyperwave's collection hierarchy and visualises this information in the form of a tree. The user has the ability to navigate through the hierarchy in a simple and intuitive way. Figure 8 shows a collection browser with an opened collection hierarchy.

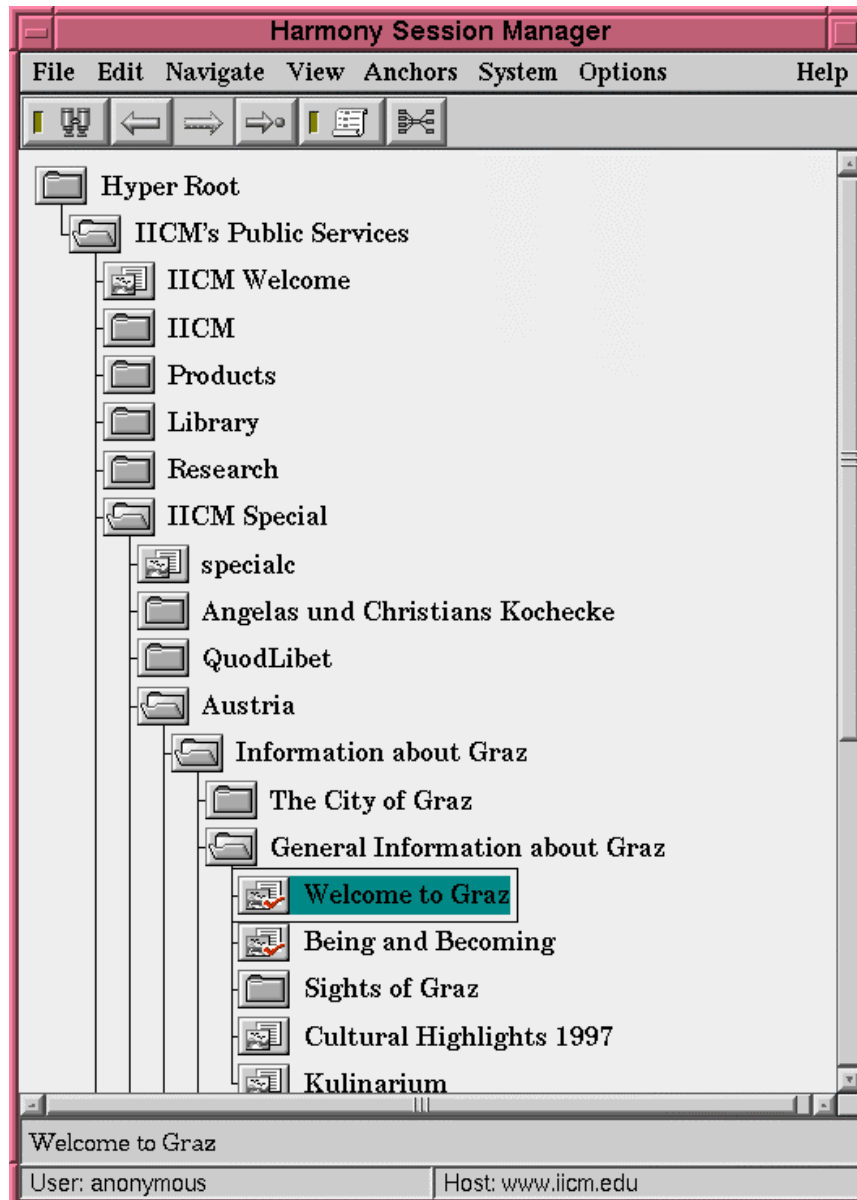


Figure 8: The Harmony Collection Browser.

At the beginning of a Hyperwave session, after the connection to the server has been established, the Harmony session manager maps the collection browser window. As an initial state, the path from the Hyper-Root to the users home collection is opened, and the members of the home collection are displayed.

Here for the first time, a special feature of Hyperwave is used. When connecting to the server only the home collection of the user is known. Since Hyperwave also has information about the parents collections of the home collection, and in turn the further parent collections,

a path to the Hyper-Root can be determined and displayed in the tree. The position of the user's home collection in the collection hierarchy on the server is clearly determined.

In Hyperwave it is possible for an object to have more than one parent collection. How this problem is solved in Harmony is described in Section 6.2.

The displayed tree visualises the hierarchy of the collections. To indicate the different levels, members of a collection are indented relative to this parent collection. In order to clarify the relation between collections and their members, lines (edges) between the members and their parent are drawn.

Now the user can start to browse through the collection hierarchy. If there is a collection of interest, this collection can be activated and in turn the members are displayed in the tree. To indicate that all members are shown, the collection icon is changed from closed to open. Two behaviours of the collection browser can be chosen.

1. **Full display:** In this mode the user has full control over which collections to open and which to close. When a collection is opened all members are added to the view. The user can close this collection by again double clicking on it. All open sub-collections are collapsed and only the selected collection remains in the tree. In other words, the whole subtree, beginning after the collection to be closed, is removed from the tree, and the rest of the tree is rearranged to close the gap. The view can be cleaned by double clicking on the open Hyper-Root.

This mode is useful when more than one collection should be displayed at the same time. For example to compare two collections, or copy (move) on collection to another. However, too many open collections may cause in extensive use of the scroll bars, and therefore become wearisome.

2. **Partial display:** This mode has not found his way into Harmony, nevertheless it is mentioned here.

In this mode only the members of the current collection are displayed. The user can open further sub-collection or one of the collections in the path. Every time a new collection is opened, the previous open collection is closed (the members are removed from the view) and only the path to the current collection and as well their members are displayed.

The number of collections displayed at the same time remains foreseeable, since only one collection is open. (Unless a collection contains many members).

This mode may be used for browsing through the information. Only the members of the current collection are displayed and accessible. The user also do not lose control where a document or collection is located in the collection hierarchy. The probability to "become lost in hyperspace" remains low.

As an additional navigation help, a function called "home" is provided. When home is activated, the user's home collection is shown and its members are added to the tree.

Visualisation of Documents

In the above described form the user can navigate in the hierarchy utilising the structure provided by the author. When a document of interest is reached, it can be visualised in the same way collections are opened and closed, by double clicking.

For text, image, movie, sound, postscript and 3D documents Harmony native viewers are available and are started on demand. For every type of document, a new window is opened

and the document is visualised in this new window. All further documents of the same type are visualised in the same window. The Harmony native viewers are described in Section 5.3.

Hyperwave and Harmony also support a generic document type. For such generic documents an external program and the program parameter can be specified. The viewers for generic documents can be configured by the user. It is also possible for the author to define a particular program as the viewer for the document using an object attribute.

Generic documents are used for unsupported document types, such user-defined data with an application-dependent viewer.

Visualisation of Clusters

Multimedia documents which are composed of more than one document are stored in Hyperwave in so-called clusters. A cluster is a special kind of collection. It contains a set of documents, and possibly further clusters. When visited, one or more members of the cluster are visualised at the same time according to some rules.

For each visualised document a separate viewer is opened. For example a movie, some descriptive text, and images of the actors can be implemented using a cluster.

Next time a document is visualised only the appropriate viewers remain on the screen and all other, no longer necessary viewers, are unmapped. An exception to this behaviour is, when the user specially want to “pin” the current document on the screen. In this case, the current documents are held and all further documents are displayed in a new viewer window. The comparison of two documents (also two different parts of the same document) is easy in this fashion.

It is also possible that the user is only interested in which documents are members of the cluster. To check this, the cluster can be opened by a special function, called “show children”. This function applied to a cluster displays the members of the cluster as for a collection. All operations can now be performed on a particular member of the cluster, for example removing it or simply viewing it.

To indicate, that a document or cluster has already been seen, the icon in front of the title is marked with a tick. This is especially useful for documents which belong to more than one parent collection.

Sort Order

The members of a collection are displayed in the tree in a specific order, depending on the Sortorder attribute. There are two different types of sort order. One can be specified by the author of the collection, the other by the user as a parameter in Harmony.

1. **Collection-specific sort order:** For every collection, a sort order can be specified as an attribute of the collection. This attribute is stored in the database and is valid for all users who access this collection. When Harmony visits collections with a Sortorder attribute, other settings are overruled, and the members are arranged as the attribute prescribes. This form of sort order is very useful for collections where the author wants to state the sort order.

A possible collection with an own sort order would be the collection: “calendar of events“, a list of dates of theatre playings, where the members might be sorted by the “expiry date” attribute.

2. **User-specific sort order:** For all collections with no pre-defined sort order, the user can specify the preferred sort order.

The sort operation is performed on attributes of the objects and is very flexible. Since all objects can have attributes the sort operation is applicable to all kind of objects. Harmony can currently sort the objects according to the following attributes.

- Title
- Author
- Creation Time
- Expiry Time
- Sequence Number
- Name
- Open
- Score
- Document Type

The Sequence attribute is especially defined for the sort operation. It is assigned to objects by the author and is simply a number. Sequence numbers are especially useful when sort on other attributes is not unique or not applicable, for example pictures which should not be sorted on title or other attributes, but on the content. This problem can be served by assigning simple sequence numbers.

Sort attributes can be chained together and individual attributes can be decreasing or increasing. One imaginable sort order could be: Sort first by title and second by decreasing creation time.

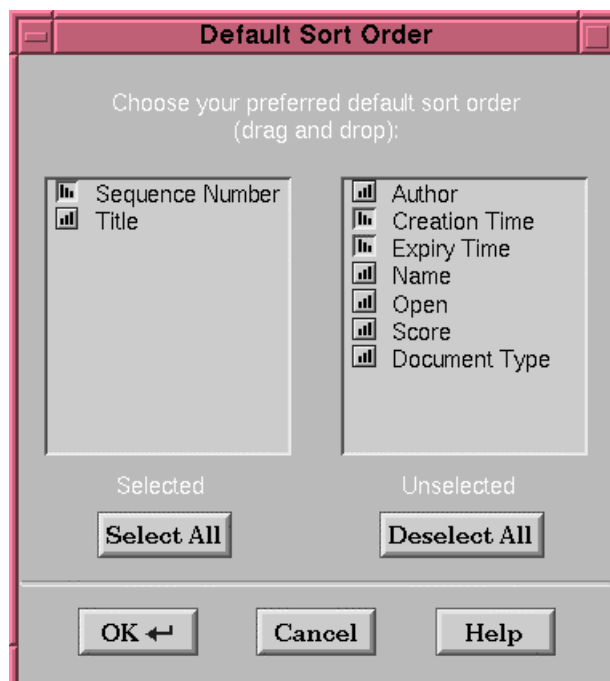




Figure 9: Harmony Sort Order Dialog.

In Harmony the dialog shown in Figure 9 is used to select the desired sort order. The user can select attributes of a list of possible attributes. To flip an attribute from selected to unse-

lected or back, the user double clicks on an item. Reordering entries within a list and also between the two lists is possible by selecting an item with the mouse, dragging it to the desired position in the list, and dropping it. The sort direction is indicated by an  or  icon before the attribute, for increasing or decreasing order respectively. This can be changed by simply clicking on the icon. When satisfied the user presses ok and the new sort order is applied.

Location Feedback

The previously listed features describe methods for navigating within the collection browser, and changing the appearance. The collection browser also provides navigational feedback. Every time a document is visualised in a Harmony native viewer, any hyperlinks are visualised in an appropriate form. If interested in a link, the user has the possibility to follow it by activating the link and in turn the destination of the link is displayed.

Applying this kind of navigation raises one serious problem. When activating links, reading the destination document, following further links, stepping back some documents and again following links, it is hard to remember the “location” of the document. This effect is known as the “lost in hyperspace syndrome”. In Harmony and Hyperwave this effect is reduced. When a link is activated the destination document is visualised in the collection browser. If the document is not already in the displayed tree, the document is added to the view. To do this, a path to the root collection is determined, and displayed in the tree. This mechanism is called *location feedback*.

With this mechanism it is easy to locate and remember the position of the document in the server. It is also possible to open the parent collection and display the other documents in the collection. Since all collections on the path to the root can be opened as well, it is possible to gain an overview of other related documents. There are two different scenarios depending on the type of destination of a link:

1. **The link destination is a document:** In this case the destination document is displayed and if a destination anchor is specified, the respective part of the document is highlighted. If the destination is not already visible in the viewer, the document is scrolled. In the Collection Browser, the document is added at its position in the collection hierarchy.
2. **The link destination is a collection:** In this case the destination collection is displayed in the collection browser and all members are added to the view.

5.2.2 The Local Map

In addition to hierarchical structuring of documents in collections, associative hyperlinks are used to create relations between documents. With the help of the link structure the user can navigate by activating, and in turn following links. One of the problems which with this form of navigation is to find all information linked to the current document. When one link is followed a new document is opened with further links in it. To display further documents referred to by the first document it is necessary to step back and activate the next link.

In Harmony a method to help the user gain an overview of the link structure is provided, the Harmony local map. In Hyperwave the link structure is defined orthogonal to the collection hierarchy. As mentioned earlier, links in Hyperwave are bi-directional and are stored in an separated link database. With this capability the database can find documents linked to a specific document without needing to parse the document. A link is only a relation in the da-

tabase and thus it is easy and fast to ask which documents refer to a specific object and also to find documents which are referred to by a document.

In a large information system there are often many hundreds or thousands of documents. Displaying a map of the whole link information for all documents would generate an enormous map. In Harmony, the local map contains documents linked in some manner to the document of current interest.

The local map is generated for the document last activated. All documents which refer to or can be reached by a link are displayed. Every document which was already visited is displayed with an check mark in front of it. By default, two levels of referencing and referenced documents are displayed. The number of levels of incoming and outgoing links can be configured interactively. Hyperwave not only supports simple hyperlinks, there exist also links to inline images, and links to texture information in 3D scenes. Annotations are a special type of link which is supported in Hyperwave. These types of links can be selectively added or removed from the display in the local map as shown in Figure 10.

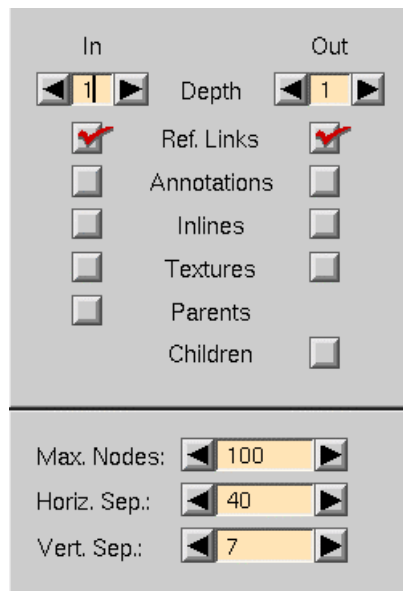


Figure 10: Local Map options.

Every object in Hyperwave has at least one parent and if the object is a collection zero or more children. The display of parent-child relations can also be enabled or disabled in the local map. A typical local map is shown in Figure 11.

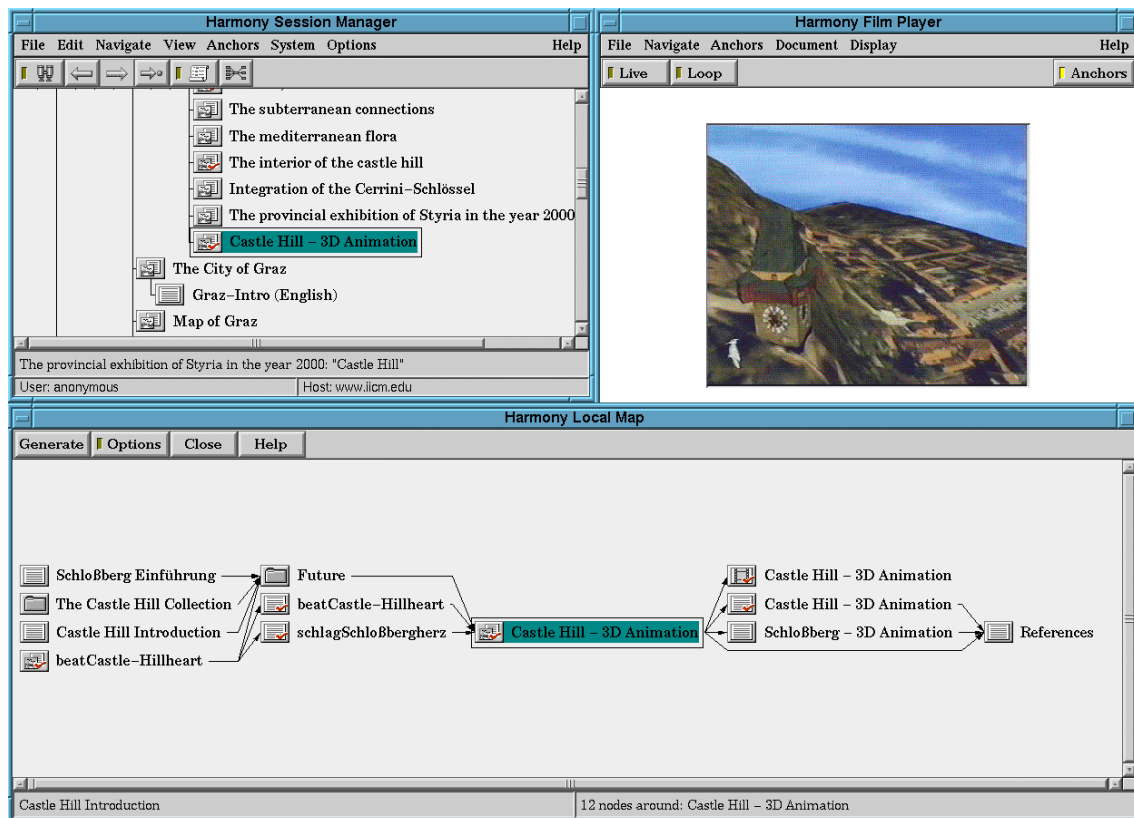


Figure 11: The Harmony Local Map.

The local map can be used for navigation. When double clicking a document it is displayed in the proper viewer. As a result of a single click the location of the document in the collection hierarchy is displayed in the collection browser (location feedback) and all parents in a path to the “root collection” are added. A new local map around the new current document can be generated. In this form it is possible to trace back the way someone has come and it is also possible to gain an insight about of how much information is linked to a particular document.

Information displayed in the local map is not only of interest when browsing, it is also helpful during editing the information. An author wanting to delete a document can generate a local map to find out if other documents are referring to it. When there are such documents the author can decide to continue deleting the document or not. A further possibility would be to inform the owners of these documents, although mechanisms exist in Hyperwave to handle links to no-longer existent documents.

For an in-depth description of the algorithms used to generate the layout of the local map, refer to Section 6.1.

5.2.3 The Search Dialog

One of the most important features of Hyperwave is a powerful, built-in search engine. Hyperwave offers a wide range of search possibilities. These are boolean searches on attributes of an object, and also a *full text* search with fuzzy boolean queries and WAIS-like nearest-neighbour searches. Both the search on attributes and the full text search can be constrained to selected collections or be extended to include more than one server. Figure 12 shows the Harmony search dialog.



Figure 12: The Harmony Search Dialog.

The search offered by Harmony facilitates all of the Hyperwave search capabilities. The easiest way to use the Harmony search is to enter one word in the “Search for” field and press the “Search” button. With this method, the entered word is search in the selected³ collection on the local server. The attributes searched for are the Title and the Keyword attribute by default.

If there are matching objects, these are displayed as a list placed in an area below the search input fields. In this list a user-definable set of the object attributes are displayed. Further an icon representing the document type is placed in front of each matching object, as shown in Figure 13.

Full Text Search

When an object was found through a full text search, it has an associated score. This score is an indicator of how well the document matches the search parameters. A score of 100% is the best match, a score of 0% the least. Documents found by an attribute search are automatically assigned a score of 100%. The score of 100% is normally not the score returned by the fulltext server. A typical value is about 25%, but Harmony by default normalises the scores to values between 0 and 100%. This behaviour can also be turned off and the native values displayed. The objects in the result list are sorted by default by the score, for documents with the same score additionally by the Title attribute.

Location Feedback

The user now can scan the result. For further consideration a single click on an item of interest activates location feedback for the selected object. In the collection browser the selected matching object is scrolled into view. If it was not already present in the collection listing, the whole path from the object up to an already visited object is opened and displayed. With this feature it is possible to pre-categorise items by the collection they are contained in.

³ The selected collection is the last object clicked on. If this object was not a collection, this option is disabled.

The document “TABLE” found in a sub-collection of “MS Dynamic HTML/ JavaScript Documentation” has another meaning than found in a sub-collection of “Langenscheidts Taschenwörterbuch Englisch”. Figure 13 shows the result of a search for “table” with location feedback in the collection browser.

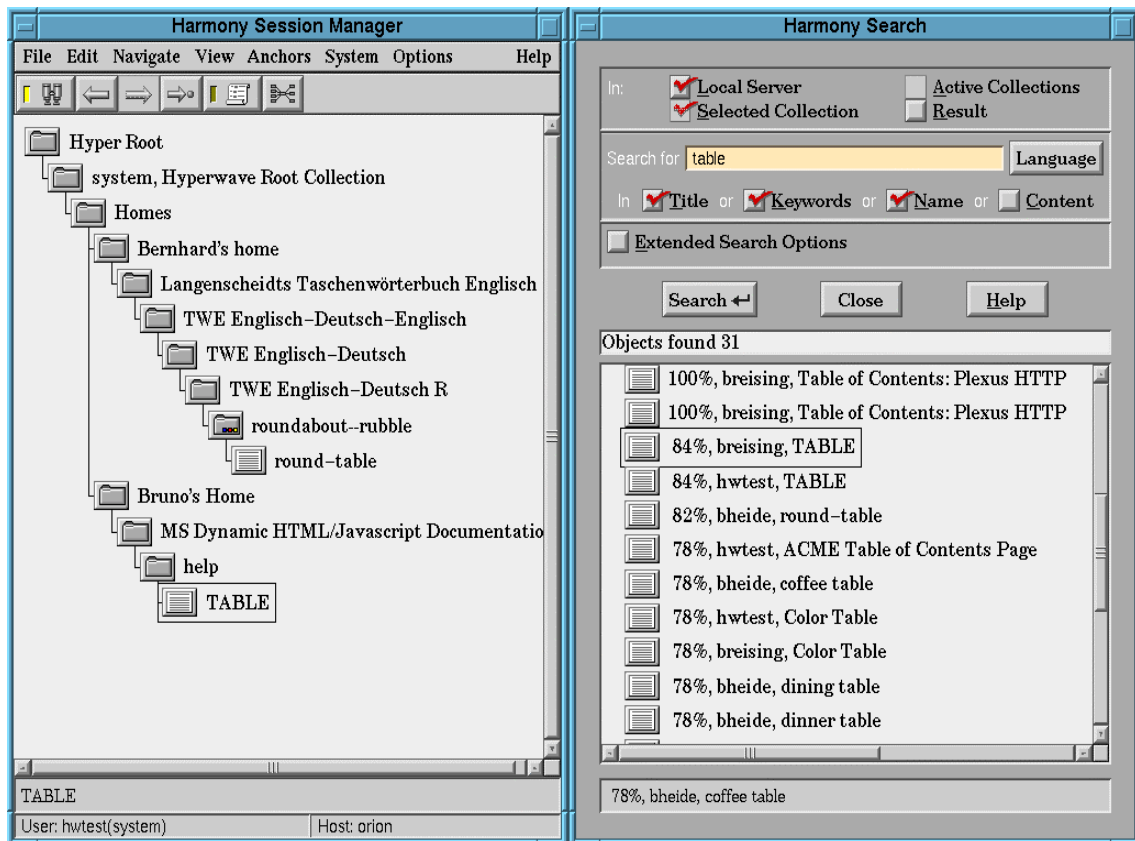


Figure 13: Harmony Search with "location feedback" in the Collection Browser.

When an object looks promising, it is of course possible to view it. By double clicking on it, the object is displayed in the appropriate viewer.

If none of the found objects is satisfying or there are too many results, the search can be refined. Unlike systems such as the WWW, it is not necessary to step back to the search page. It is just possible to modify the previous search parameter and do the search again.

Boolean Operators

One possibility to refine the search parameters is to enter more than one word in the search field. Words separated with the space character are assumed to have the boolean “AND” operator applied. This means that all of the entered words have to be contained in the object to match the query. The “AND” operator can be written as “&”, “&&” and also as the word “and”.

It is also possible to combine words with the boolean “OR” operator, written as “|”, “||” and the word “or”. When combined with the “OR” operator only one of the entered words has to be contained in the object to match the query.

Both the “AND” and the “OR” operator as well as parenthesis “(“, “)” can be combined to form complex boolean queries. For a closer look at the search syntax, refer to Appendix C.

Searched Attributes

If this doesn't lead to satisfactory search results, it is further possible to change the set of attributes to search in. Possibilities are the attributes Title, Keyword, and Name. The content switch activates the full text search of the Hyperwave server.

Search Scope

Not only search words can be altered, the scope of the search is also adjustable. Combinations of the following areas are allowed:

- **The local server:** If this option is turned on the search is performed in the whole local server, where the local server is the server Harmony is connected to.
- **The selected collection:** If this option is turned on, the search is performed in the selected collection. A document only matches if it is a member of this collection or one of its sub-collections.
- **The active collections:** If this option is turned on, the search is performed in all active collections. How to activate collections is described below.

All the above listed options can be used additive. In other words, it is possible to select any combination of them. If "selected collection" and "active collections" are switched on, the search is performed in both the selected and the active collections.

Extended Search

For more sophisticated searches it is further possible to switch on the extended search options. In this mode the search can be extended to further attributes such as Author or Modified.

- Author can be used to find only documents created by the specified author.
- Modified is used to search for objects created or modified before or after a specified date, and also within a range of dates. Date entries can be entered as absolute dates in the form "year/month/day hour:mins:secs" or as a relative date in the form -ndays -nhours -nmins where n is a number

A date entered in the last format is translated by Harmony to a date n days, hours and mins before the actual date and time. This can be used to search for objects inserted or modified in the last n days. Figure 14 shows a search dialog with extended search enabled.

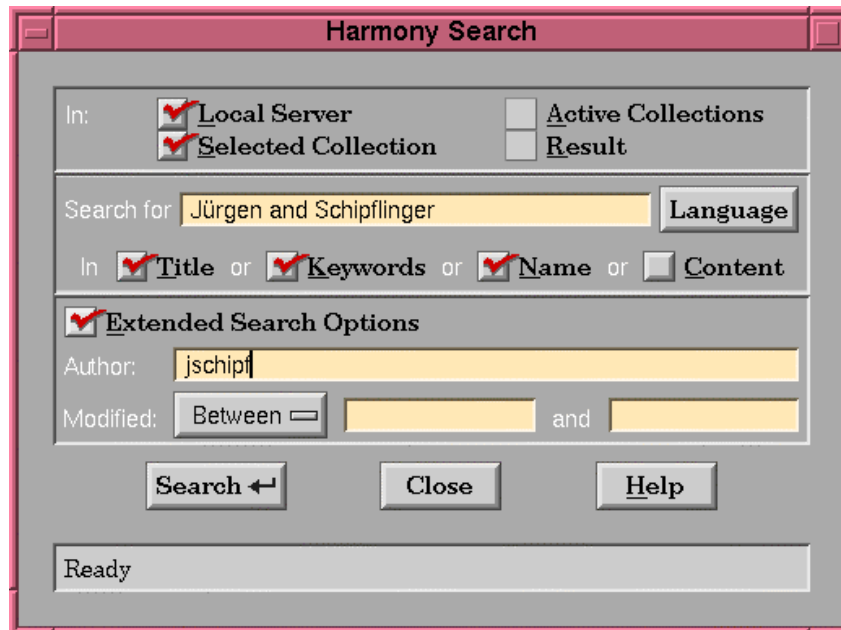


Figure 14: Harmony Extended Search.

These search parameters can also be used to filter the number of found objects. When a search returns too many matches, an additional entered Author or Modified parameter can drastically improve the result.

Active Collections

As mentioned earlier, search can be performed on activated collections. In Harmony collections can be added to or removed from the list of active collections in the Harmony active collections as shown in Figure 15.

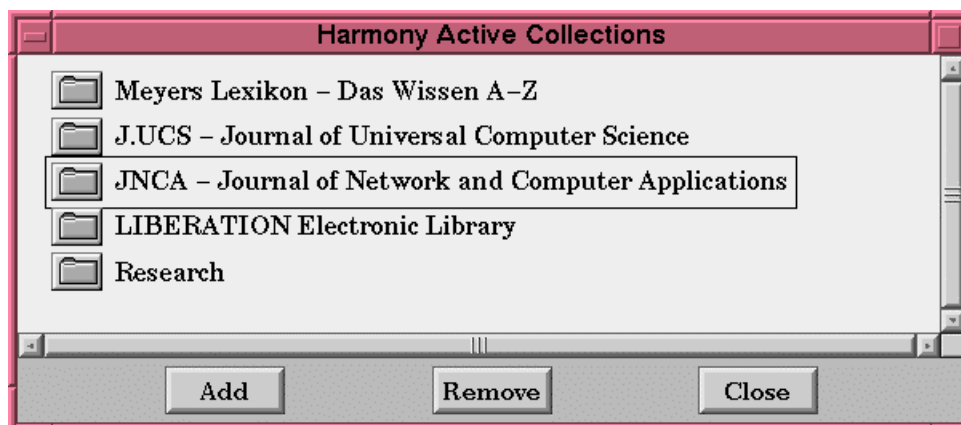


Figure 15: Harmony Active Collections.

The active collections simply displays a list of the activated collections. It provides buttons to add and remove collections from the “activated list”. Collections can be activated by browsing to them using the collection browser or any of the other browsing method and pressing the “add” button. When the remove button is pressed, the selected collection is discarded from the

list. In the active browser “location feedback” is also available, as already discussed in earlier sections.

Search in Result

A further option in the Harmony search is described here at the end of the search section, the option to search in the result set. In contrast to previous options the search is only performed on the already found objects. With this feature it is possible to reduce the number of matches successively by applying selective queries on the objects in the current search result list.

Every search operation is recorded in the Harmony History. With the search objects in the History it is possible step back to an earlier performed search in the Harmony session. This is particularly useful, when search parameter alternation has not lead to the desired result.

5.2.4 The History Browser

Users typically browse through the information universe using the collection browser, hyperlink navigation, the local map, or navigate to documents found with the help of searches.

The other possibility is to use the Harmony history browser. During navigation all visited collections, documents and also performed searches, are recorded in the history list. This recorded history can be utilised using the Back and Forward buttons in the collection browser window. Pressing Back or Forward displays the appropriate object from the history list.

An alternative way to use the History functionality is to map the History window as shown in Figure 16 and use its features. In the History window all recorded stages are displayed as a chronological list. The user can scroll through this list and directly chose the items of interest. For further information each document is marked with a time stamp, showing when it was last visited. This is useful since sometimes it is easier to remember the time when a document was seen (just before I went to lunch) than the position in the history.

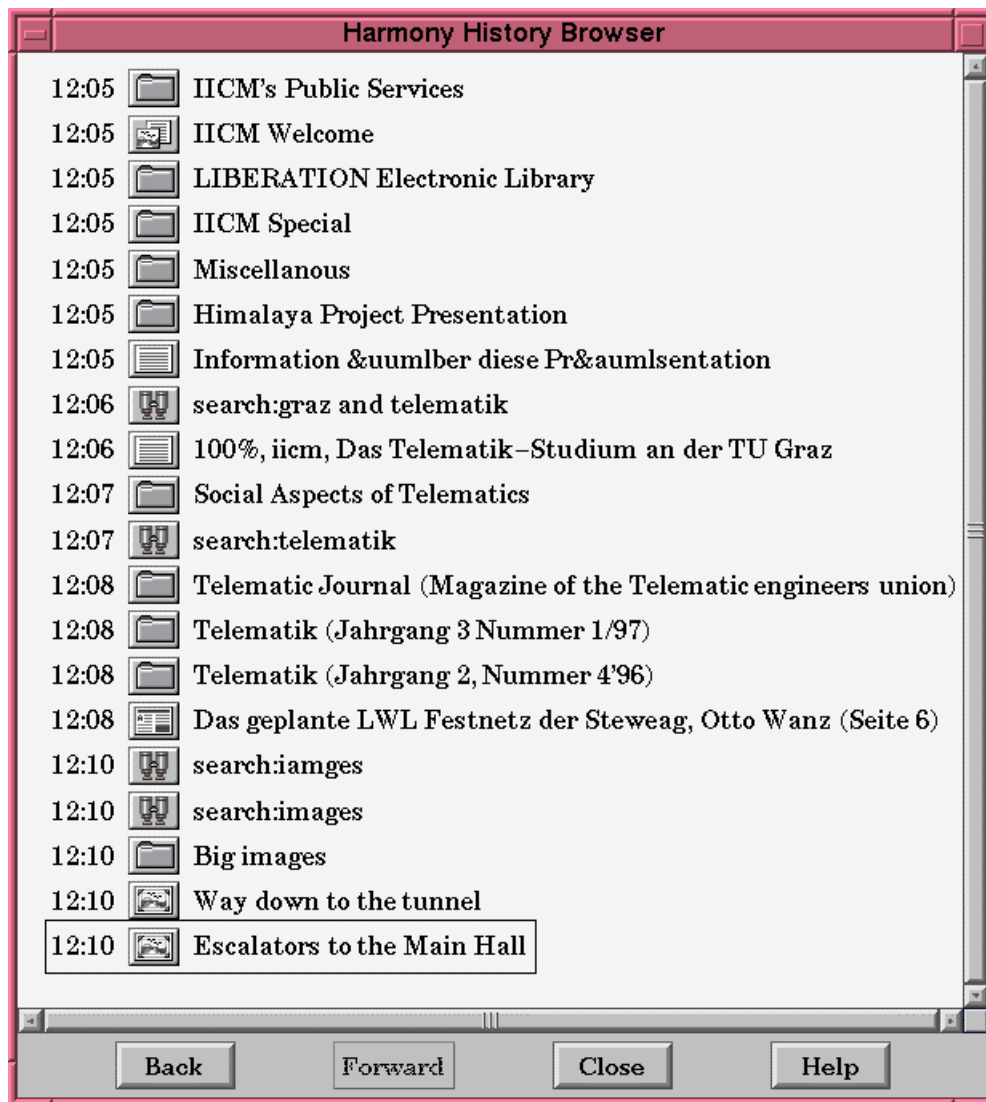


Figure 16: The Harmony History Browser.

If the document is a document managed by Hyperwave, it is also possible to single click on the item and the document's position in the collection hierarchy is shown in the collection browser with the path to the root collection opened. When a collection is chosen out of the history, it is shown in the collection browser and if not already, their children are displayed. When a search is recorded, all relevant attributes are stored in the History entry, including the query text, the active collections, the selected collection and of course the activated attributes and scopes. When a search is activated from the history these parameters are used to perform the search again.

5.2.5 The Information Landscape

The Harmony information landscape displays the information structure in a Hyperwave server in a three-dimensional manner as shown in Figure 18. Collections are visualised as pedestals on a plane. Documents in a collection are three-dimensional objects placed on the pedestals. The size of the document's is optionally mapped to the height of the objects.

The user can fly over the information landscape and can activate a document by double clicking on the document representation. The documents are then displayed in the Harmony viewers. It is further possible to open and close collections by double-clicking on them.

An overview map can be activated to show the structure from a view point above the scene. This map is especially useful to gain an overview, of where documents are on the server. Figure 17 shows an Overview map. Large rectangles indicate collections or clusters with many documents.

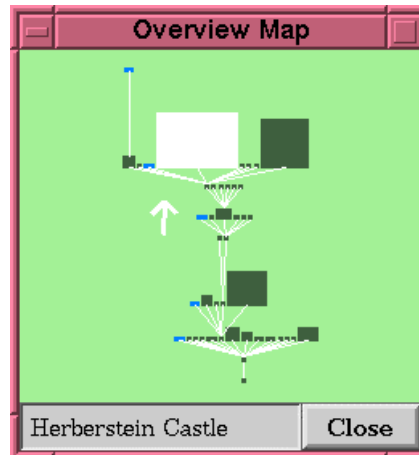


Figure 17: Information Landscape Overview Map.

The hyperlink structure can also be visualised using the information landscape. Linked documents are placed orthogonal to the collection structure above and beyond the corresponding object. An information landscape with activated link structure is shown in Figure 19.

All Harmony specific features are also available. Selecting an object in the landscape updates the collection hierarchy in the collection browser and vice versa. Location feedback is implemented in the information landscape by “flying” the user automatically to the selected object. For further information about the information landscape refer to [Eyl95][Wol96].

5.2.6 Server Status Browser

The Harmony server status browser displays current status information of the Hyperwave server which Harmony is connected to. The information displayed includes the server name, local time, data transfer statistics, up times and so on. The displayed information is updated every 60 seconds by default, but can be changed to any other interval. Figure 20 shows the Harmony server status browser.

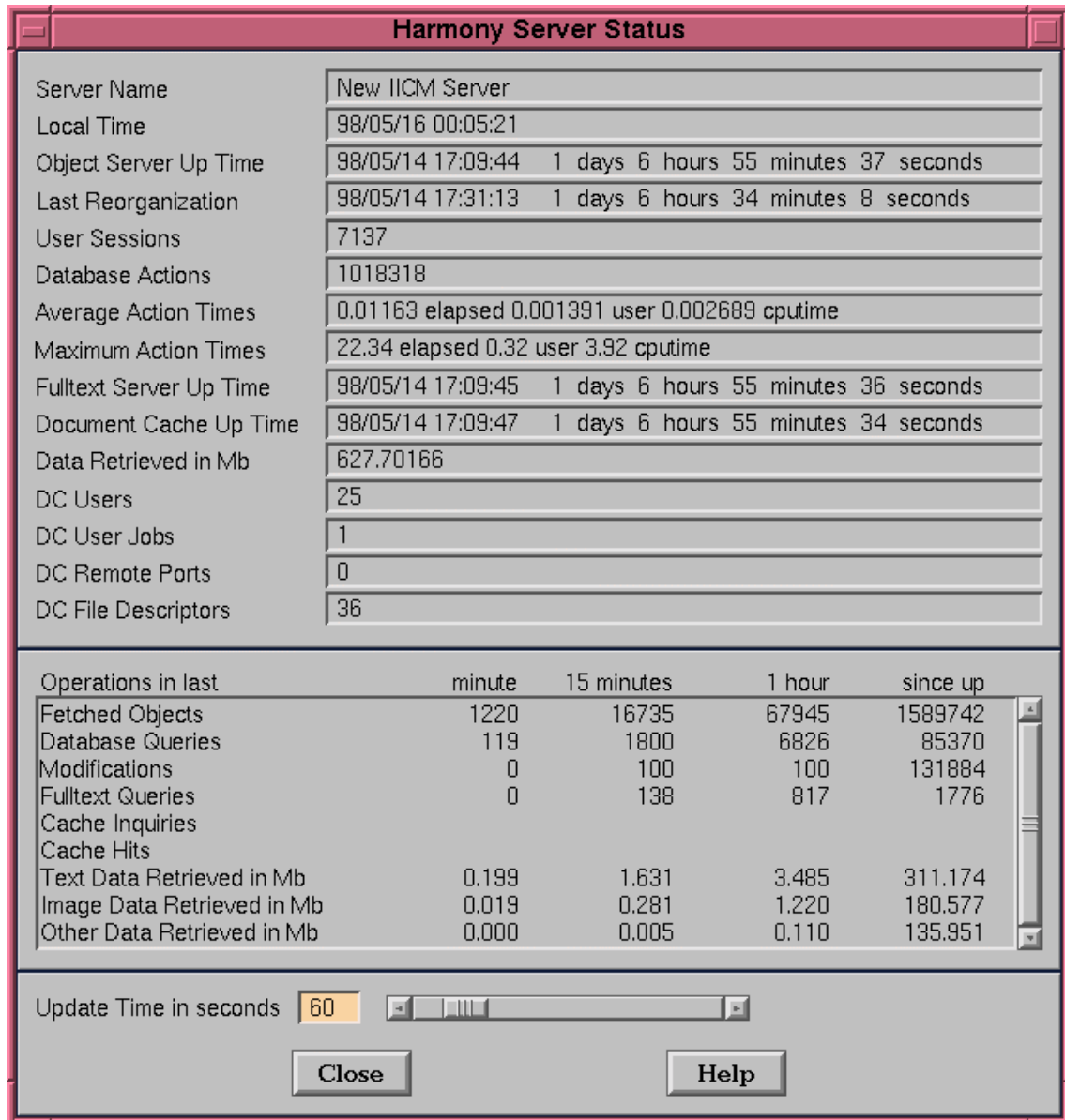


Figure 20: The Harmony Server Status Browser.

From the status browser it is possible to open the Harmony user connection browser. This function maps a new window, which contains a list of currently connected users. This list is also updated periodically with the same interval as the status browser. The user's own connection is marked with an asterisk as for example, *jschopf in Figure 21. The user now can select one or more users out of the list and start an online conference. Figure 22 shows the Harmony status browser with a chat session between some users. In this window the upper

area is used to enter messages which than are sent to the selected users. The lower area is used to display the history of the conference. All sent and received messages are recorded with information about sent messages and involved users.

This feature can also be used to send information to all connected people. It is, for example, used by the system to inform about an upcoming server shutdown. If the Message window is not already opened, a pop up window informs the user about the available message.

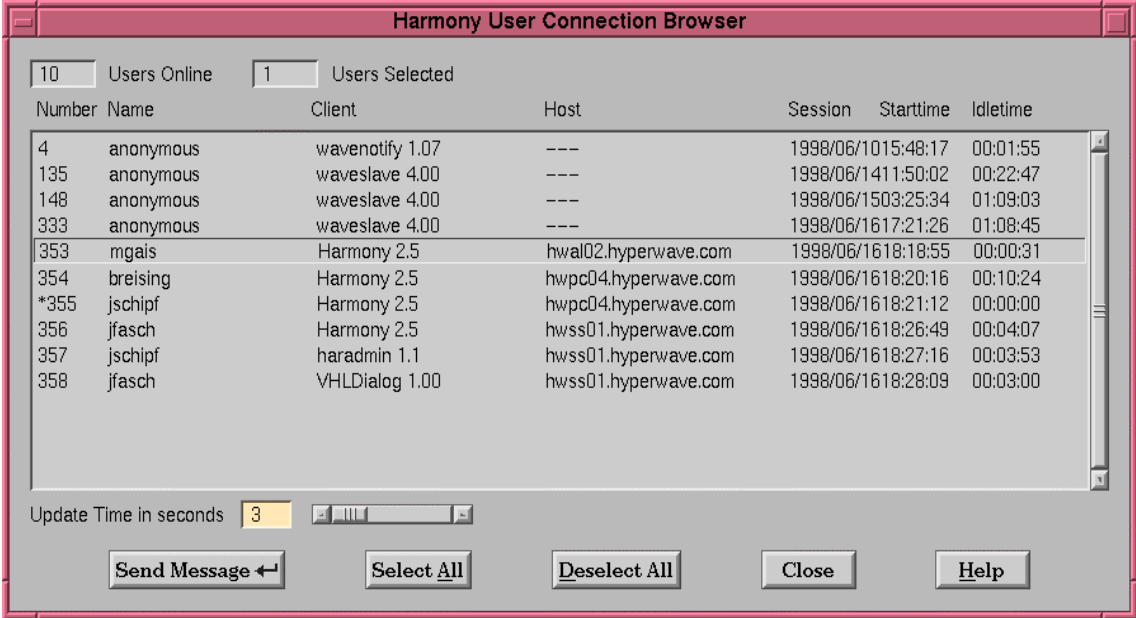


Figure 21: The Harmony User Connection Browser.

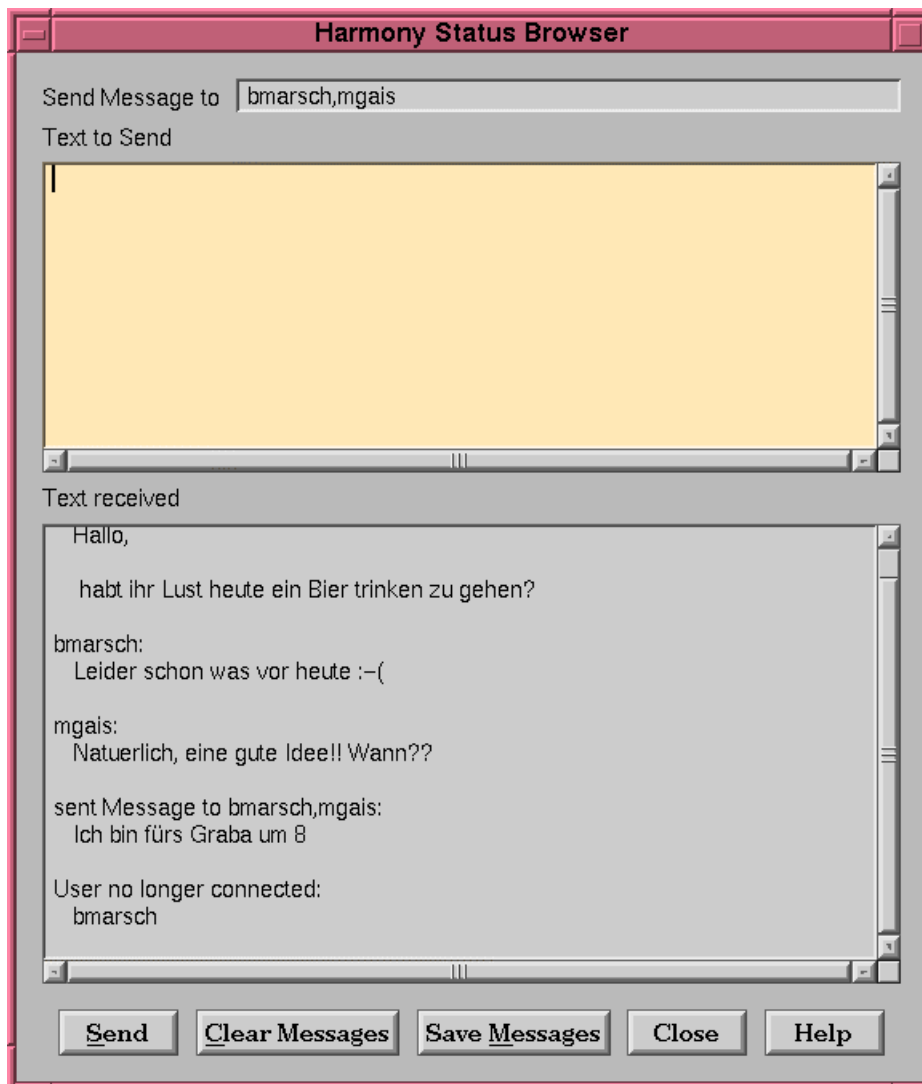


Figure 22: Harmony Status Browser, Send Message.

5.2.7 Multilinguality

There are two major areas, where Harmony handles information in more than one language, multilingual documents and a multilingual user interface.

5.2.7.1 Multilingual Documents

The system should support multilinguality for the information stored in the database. Just as user guides for electrical equipment, household utensils or cars, which are delivered typically in different languages, Hyperwave and Harmony support documents in different languages as a built-in functionality. The language which is chosen can be specified by the user. This functionality is achieved with the help of so-called clusters.

A cluster is a special type of collection, where the members of the collection to be visualised are selected according to some rules. When a cluster is visited and it contains documents in different languages, the one in the currently selected language is visualised. This mechanism is not only applicable to text documents, it may also be used for images (for example, if the legend is in different languages), sound in the form of speech or also movies. To indicate

the language of a document the title of the document is preceded by a special language shortcut and is stored in the title attribute of the database object. Some shortcuts for languages are for example “en:” for English, “ge:” for German or “jp:” for Italian.

Since clusters are not only used to implement multilinguality, but also to implement multi-media documents the above mechanism is extended to support all types of documents. The precise rule for document selection within a cluster is:

- All language-independent documents are visualised. Language-independent documents are indicated by more than one title attribute.
- For language-dependent documents the one in the currently selected language is visualised, if available. This is applied to every document type. (Document types are Text, Image, sound etc.) Language dependency is indicated by only having one title attribute.
- Cluster: a cluster may contain recursive other clusters. This is useful when more than one language-dependent documents of the same type should be visualised together.

In other words, when Harmony visits a cluster, it first displays all language-independent documents, for example a digital movie. Next it looks for language-dependent documents, such as for example the text describing the content of the movie. If there is a document in the current language and of a not already displayed type, Harmony visualises it.

Further documents of an already displayed type are ignored. This feature can be used as a kind of rudimentary version control, since sort order is also taken into account in clusters. So if the sort order of the cluster is the sequence number and the sequence attributes are set according to the version number, the latest version of the document is chosen. Older versions are not displayed but are available and if necessary, viewable.

When Harmony reaches a sub-cluster, the above described procedure is repeated. This may be useful for example when some images should be displayed at the same time. In the above example this can be used to display photos of the actors of the movie.

5.2.7.2 Multilingual User Interface

In Harmony not only the documents are displayed in the chosen language, also the whole user interface is available in multiple languages. Elements like buttons, menus, text in dialogs, and the representation of database objects are in the chosen language. The language can be changed by the user interactively. The dialog for choosing the preferred languages is shown in Figure 23.

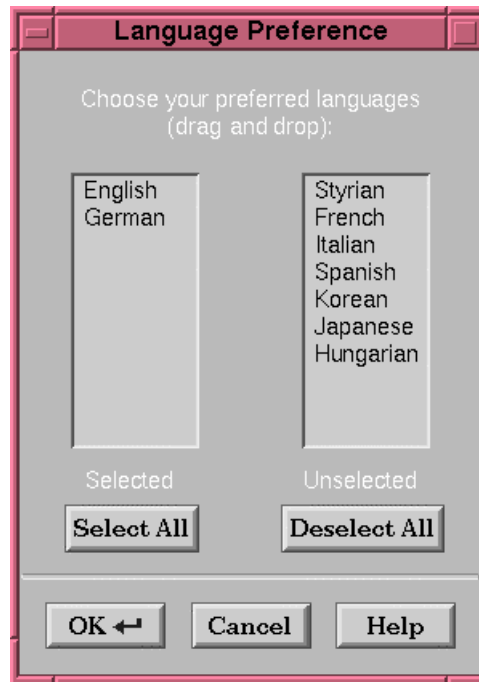


Figure 23: Language Preference Dialog.

The user can specify an ordered list of preferred languages. The languages can be toggled from chosen to unchosen and back by double-clicking on a language. Reordering the entries within a list (and also between the two lists) is also possible by selecting an item with the mouse, dragging it to the desired position in the list, and dropping it. When the lists are ordered as preferred the user presses OK and the user interface changes to the preferred language and documents are displayed in the new language, if available. Figure 24 shows Harmony with Japanese as user interface language and a Japanese document shown in the Text Viewer.

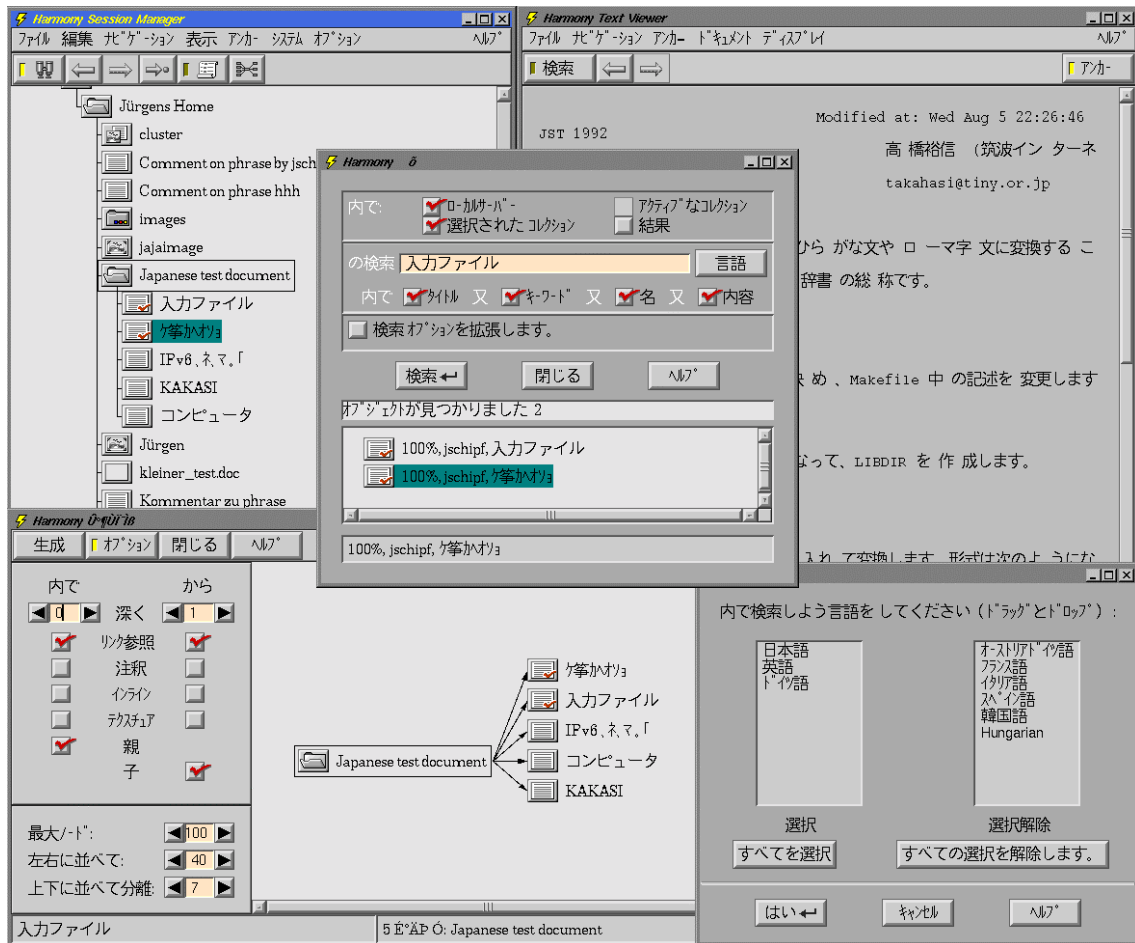


Figure 24: Harmony session in Japanese.

As depicted above, an ordered list of languages can be chosen. When Harmony decides which document should be displayed, the following rules are applied.

If there is a document in the first language in the preference list, this document is chosen. If there is no such document, the next language in the list is tested. This step is continued until a document is found or there is no further language in the list. If there is no document found with the preceding procedure, an arbitrary document, the first in the list retrieved from the server, is chosen for display.

5.3 The Harmony Document Viewers

The Harmony document viewers run as separate programs. Nevertheless they are integrated into Harmony as a whole, and all have a consistent user interface and the same menu structure, as far as possible.

All viewers support an API to send commands both from the session manager to the viewer and vice versa. The whole connection to the Hyperwave server, fetching objects, anchors and document data is done via the session manager. All native viewers support interactive link creation. The Session Manager - viewer API functions include:

- view document
- map/unmap window
- terminate

- functions for location feedback
- functions for link creation

All viewers support hyperlinks, even though some of the displayed formats have no built-in Hyperlinks. This is possible since links in Hyperwave are stored as separate objects, independent of the format. This link information is used by the viewers to overlay the viewed document with this links. Hence Harmony can add link functionality to formats without inherent hyperlink capability.

All viewers also support interactive Hyperlink creation. Link creation is treated in detail in Section 5.4.8. Due to the modularity of this system, all viewers can be developed independently.

5.3.1 Text Viewer

The Harmony text viewer as shown in Figure 25 is able to view text in HTML, plain text, and Hyperwave's original format HTF. Inline images in GIF, PNG, JPEG, and TIFF formats are supported. Marked text can be set as source or destination anchor. For more information about the text viewer refer to [Gai94].

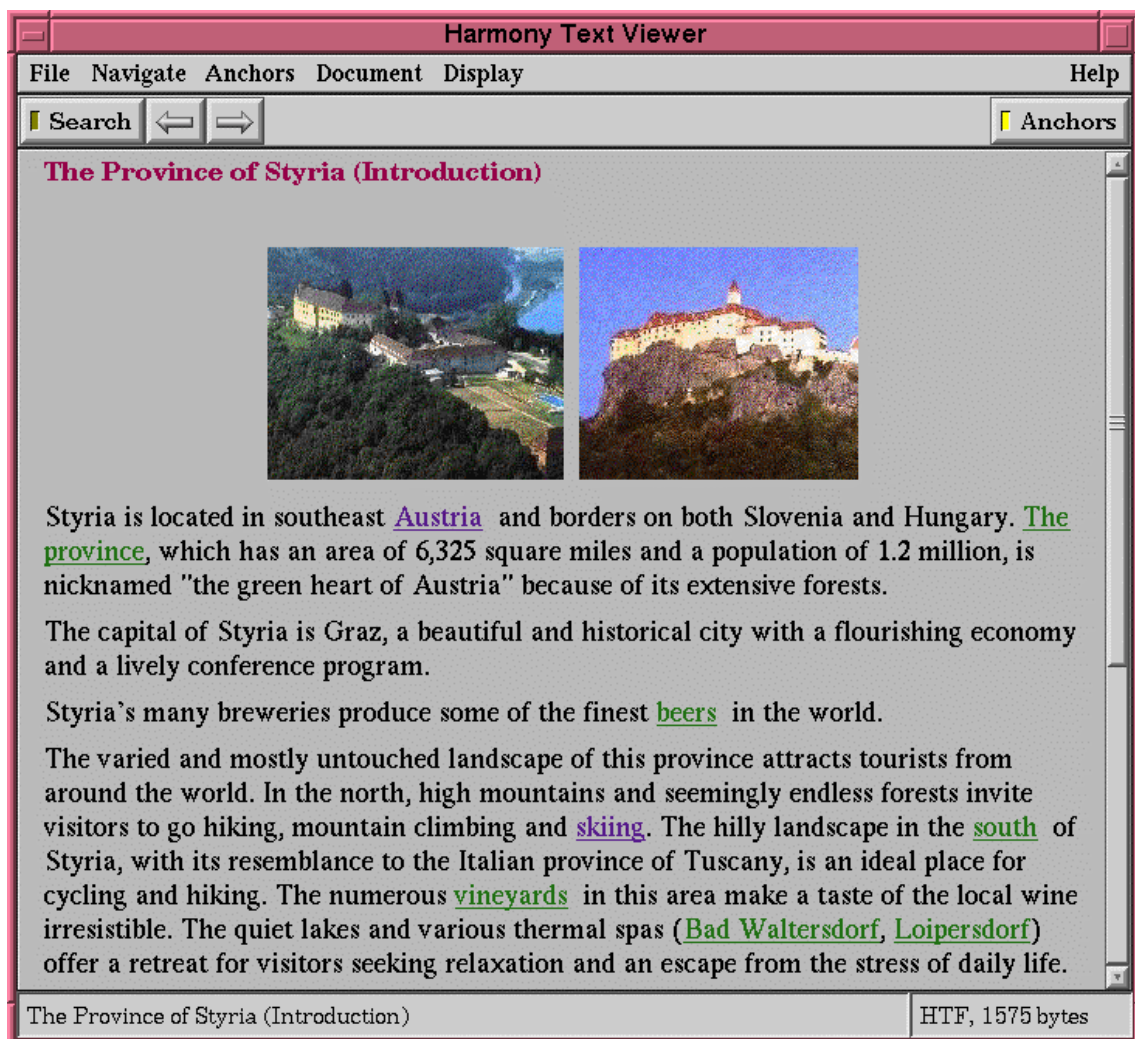


Figure 25: The Harmony Text Viewer.

5.3.2 Image Viewer

The Harmony image viewer as shown in Figure 26 is able to view images in the formats GIF, JPEG, TIFF and PNG. It support links from and to areas of the image. Anchors can be created with the shapes:

- Circle
- Ellipse
- Rectangle
- Polygon

For more information about the image viewer refer to [PiP96].



Figure 26: The Harmony Image Viewer.

5.3.3 Film Player

The Harmony film player as shown in Figure 27 is able to play movies in the format MPEG-1. It support links from and to time slices of the movie. As a special feature of this player it is also possible to create and view links to areas of the current image. This area can move over time. For more information about the film player refer to [Mar95].

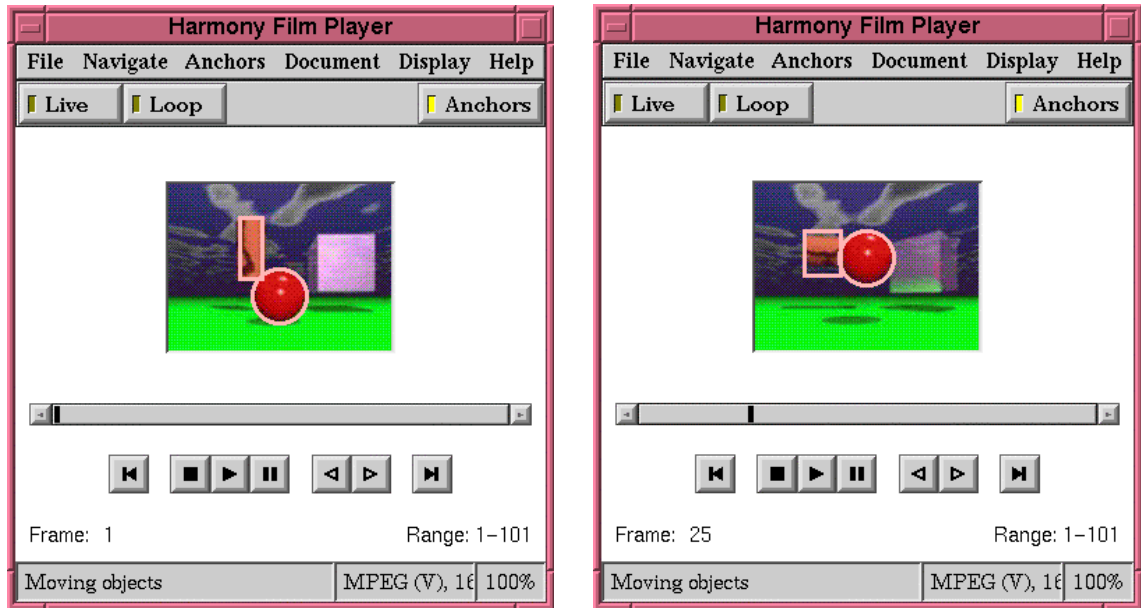


Figure 27: Harmony Film Player with moving links.

5.3.4 Audio Player

The Harmony audio player is able to play audio clips in several formats. To play these formats it uses external programs. It supports links to time slices of the audio stream. For more information about the audio player refer to [Gei97].

5.3.5 PostScript Viewer

The Harmony PostScript viewer as shown in Figure 28 is able to view PostScript documents. It uses ghostscript as render engine and displays the output. It support links from and to areas of the PostScript document.

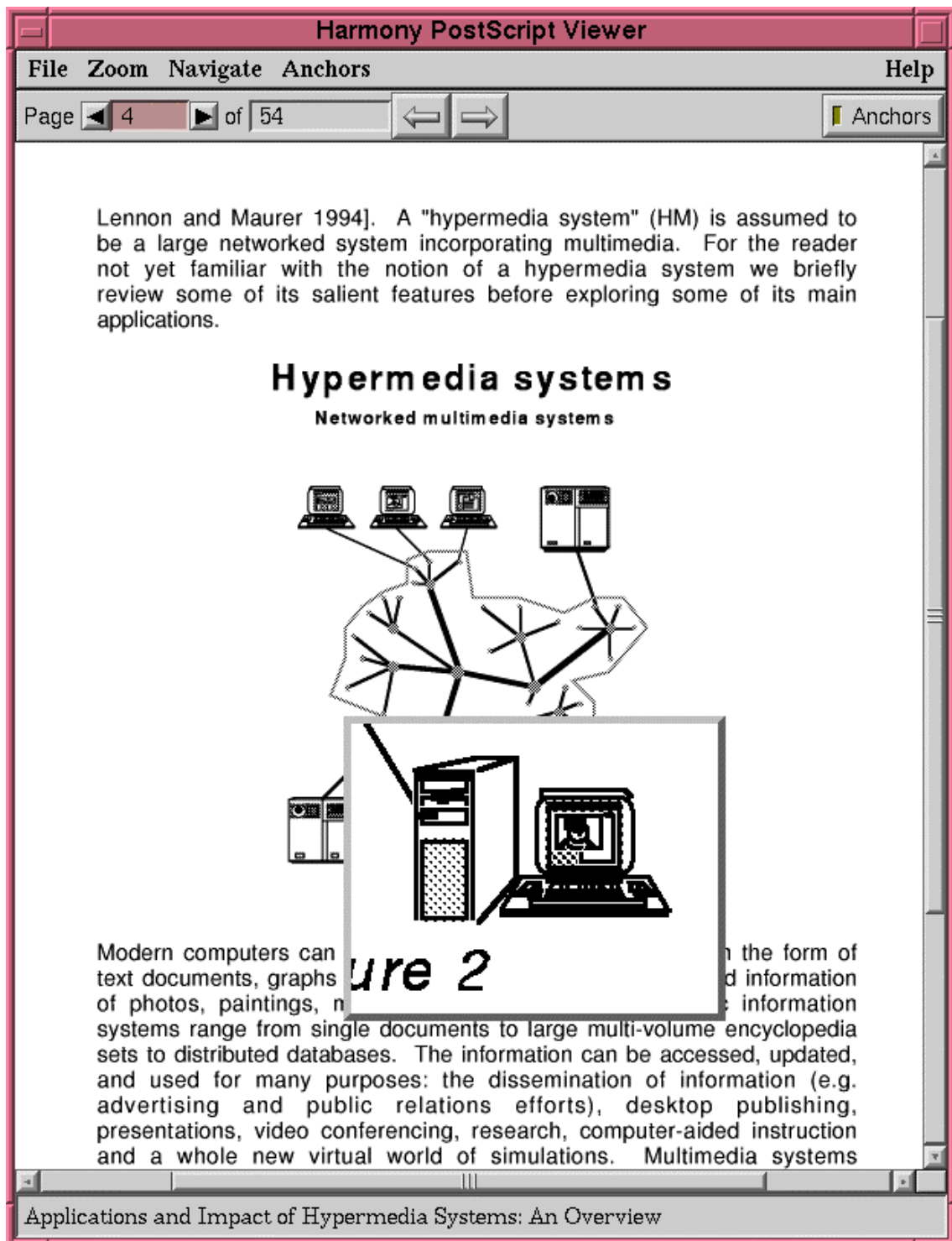


Figure 28: Harmony PostScript Viewer.

5.3.6 VRweb 3D Viewer

The Harmony VRweb 3D Viewer as shown in Figure 29 is able to view scenes in the formats VRML and SDF. Several navigation methods are provided. Objects of the scene can be defined as source anchors and viewpoint into the scene as destination. For more information about the VRweb 3D viewer refer to [Pic93].

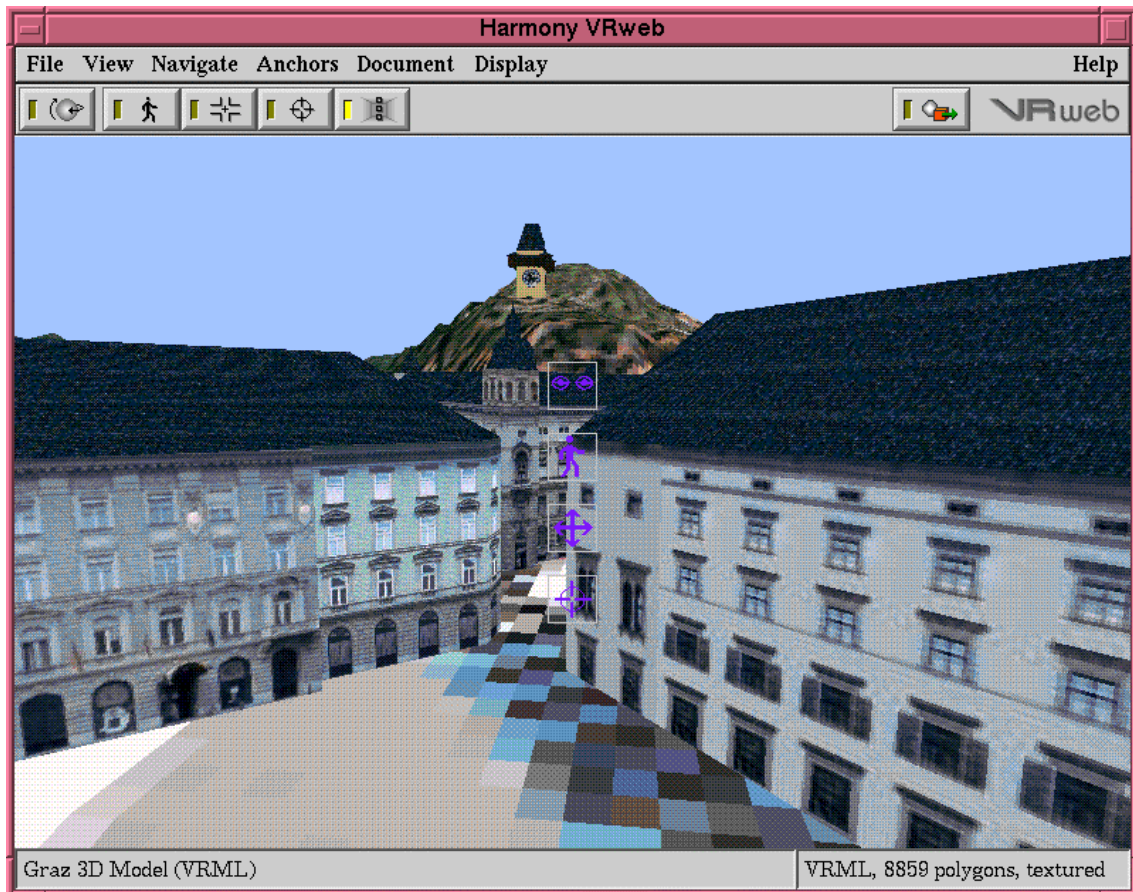


Figure 29: The Harmony VRweb 3d Viewer.

5.4 Authoring with Harmony

In Harmony it is possible to manipulate the content and the structure of the data stored in a Hyperwave server. The user can insert new objects into a Hyperwave server. In order to manage the structure of the information, there are facilities to copy, move or delete objects. Harmony supports editing of object attributes and also the content of documents. Functionality is available to create hyperlinks from and to documents and collections. The user can also create annotations to documents or collections.

In order to author a Hyperwave server the user needs the appropriate access rights. Therefore editing is normally only possible for identified users. The right to edit is checked by the Hyperwave server.

5.4.1 Inserting New Collections or Clusters

In order to insert a new collection or cluster, the user has to navigate in the collection browser to the collection or cluster where the new object is to be inserted. When the target collection is reached the user can open the Harmony insert dialog. This dialog is used to insert collections, to upload documents from the file system and to create new documents from scratch. Figure 30 shows the insert collection dialog.

When the insert dialog is opened for the first time, the field “insert into collection” is pre-set to the collection currently selected in the collection browser. If the dialog has already been mapped earlier it is also possible to press the “Current” button to set the target collection.

Next the “collection” or “cluster” from the “New” listbox has to be selected. In the dialog it is necessary to enter a title. Title attributes are used in the collection browser to visualise the object. The name attribute is also required for collections. The value of the name attribute will appear in the URL for the new object, therefore an intuitive word should be entered. Values for the “Keywords” and “Rights” fields can be entered, but are not required.

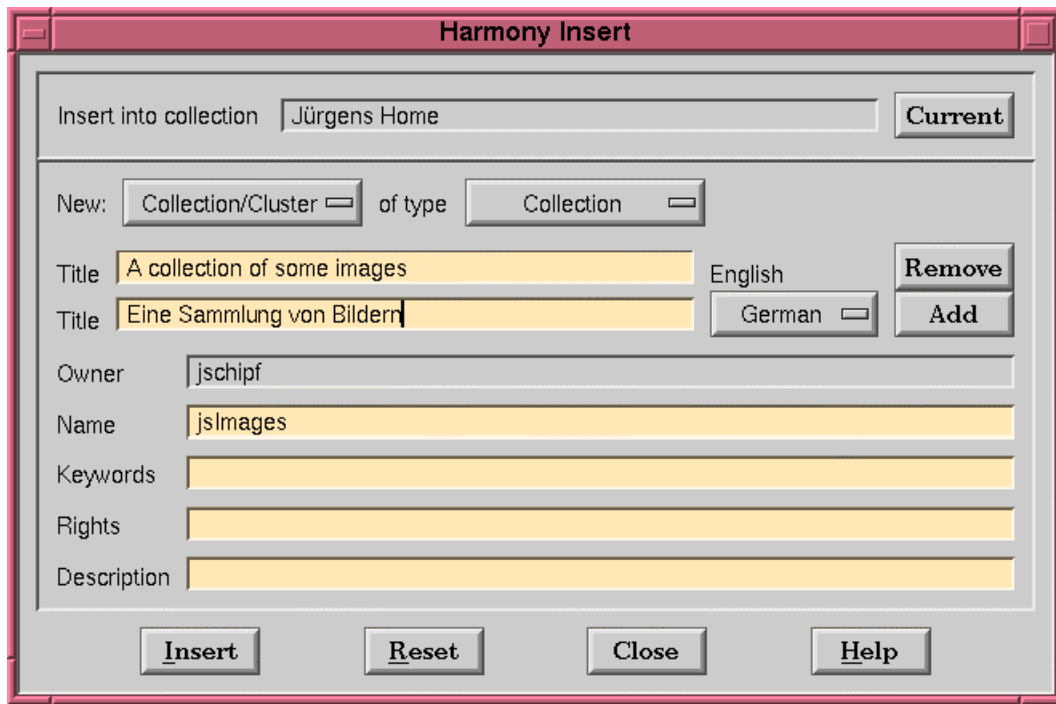


Figure 30: Insert Collection Dialog.

A click on “Insert” creates a new object in the database. The collection browser is updated to reflect the new information structure. Now it is possible to alter the text for Title, Name or other input fields, and insert further collections or also other object types.

5.4.2 Inserting Documents

Similar to inserting collections, the user has to navigate to the target collection, activate the Harmony insert dialog and click, if not already set, the “Current” button to set the target collection. From the “New” listbox the kind of document which should be inserted, is chosen. Figure 31 shows the Insert Dialog with document type “Image” selected.

The user now has to fill out “Title” and optional “Keywords” and “Rights”. In contrast to collection insertion, a “file choose” element is added to the dialog. Here the user has to enter the local file name of the document, or choose it from the file list.

When all necessary input elements are filled out, the user clicks on “Insert” and the document is inserted into the Hyperwave server. The collection browser is updated to show the new document as a member of the current collection.

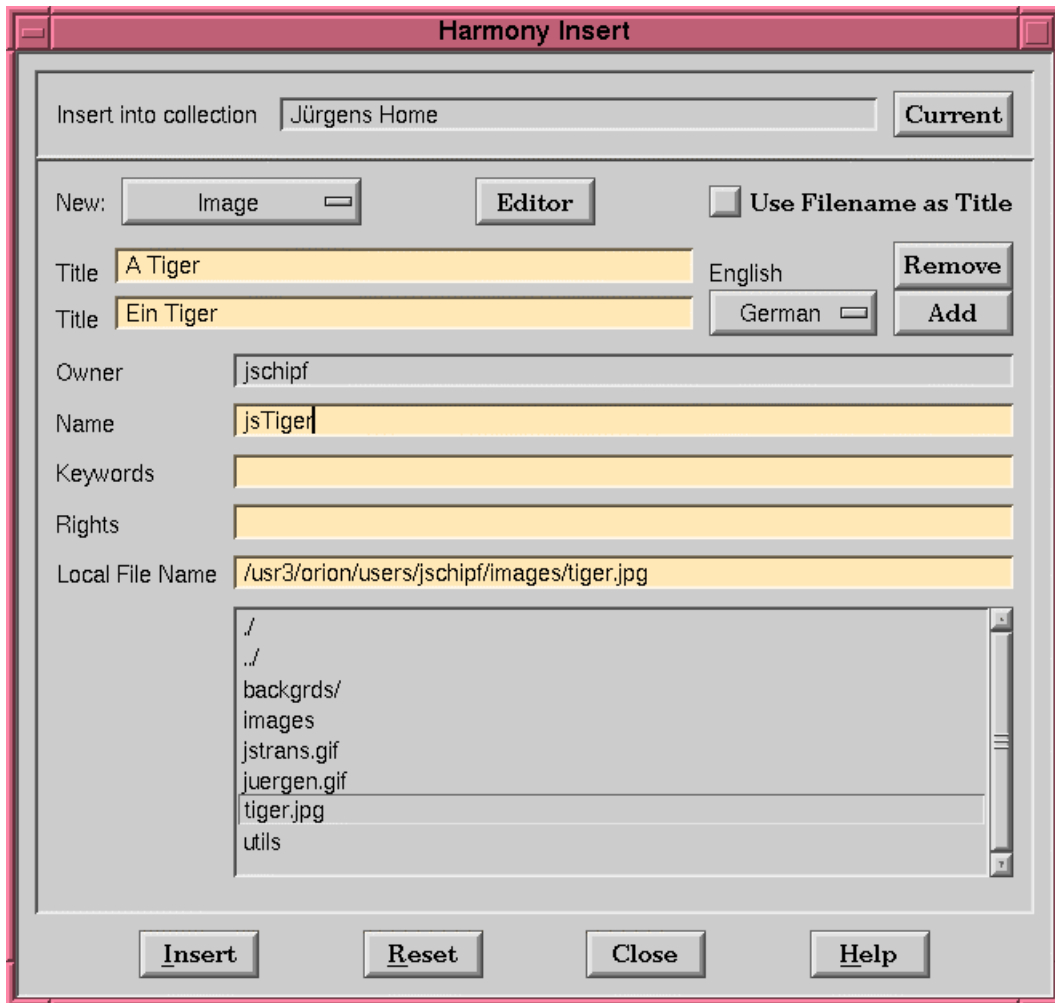


Figure 31: Insert Dialog.

In Harmony it is not only possible to upload documents from the file system, it is also possible to create new documents. When the insert dialog is mapped, and “Text” or “Image” is selected an “Editor” button is added to the dialog. When the user clicks on this button, the “file choose” element is removed. Instead the document viewer for the chosen type of document is activated and also the configured editor is started. The user has now the possibility to edit the new document. In the edit mode the document viewer provides a “Preview” button. When the user click on this button, the new document is displayed in the viewer. After the content of the new document is completed, a press on “Insert” in the insert dialog takes the new document and uploads it to the Hyperwave server.

5.4.3 Creating Remote Objects

In Hyperwave links to documents on remote servers like WWW, Gopher and FTP servers can be stored. This objects are so-called remote objects. To insert a remote object the steps are similar to those for inserting collections. When the insert dialog is mapped select “WWW”, “FTP”, “Telnet” or “WAIS”. Figure 32 shows the insert Dialog with new “WWW” selected. After entering “Title”, “Keywords” and Rights it is necessary to enter the URL of the remote document. When the user clicks on “Insert” the remote object is inserted into the Hyperwave server.

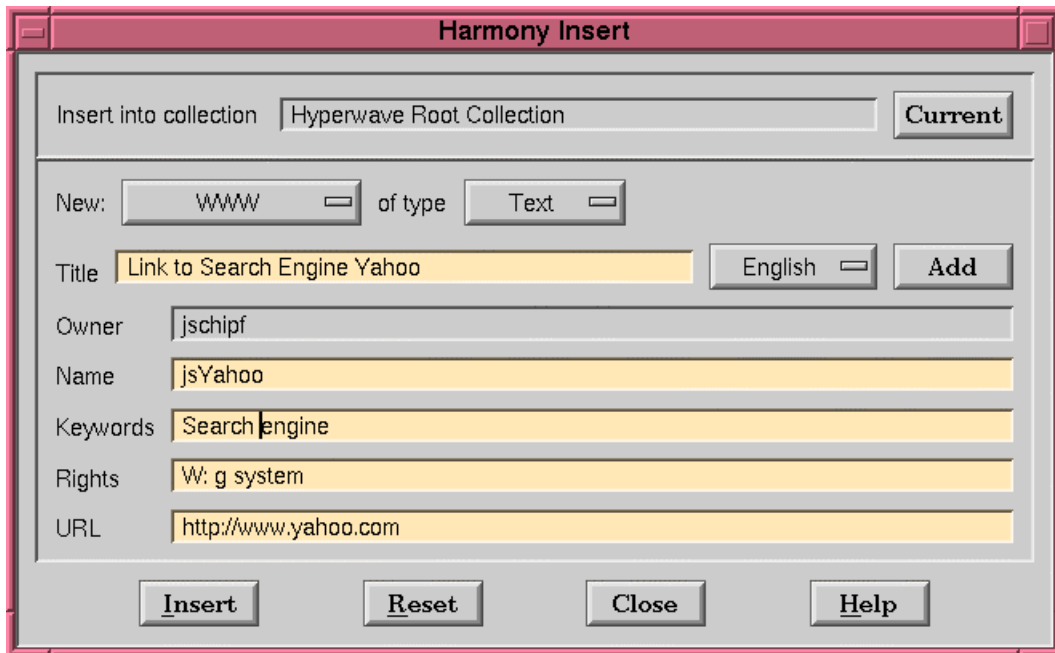


Figure 32: Insert Remote Document Dialog.

5.4.4 Editing Documents

Document editing is at the moment only implemented for text and image documents. In order to edit a document the user has to display the document in the viewer. In the viewer the menu entry “Edit” in the “File” menu starts editing. The Harmony document viewer in turn writes a temporary file and starts an external editor to modify the content. Figure 33 shows a text document being edited with Emacs started from the Harmony text viewer.

While editing, the user can save the document and click on “Preview” in the viewer. The viewer rereads the file and displays the new document. This is especially useful for HTML documents. Here the user can check if the HTML code written behaves as expected.

When all changes are done, the user saves the document and exits from the viewer. After confirmation dialog, the document is uploaded to the Hyperwave server, and the changes are committed to the database.

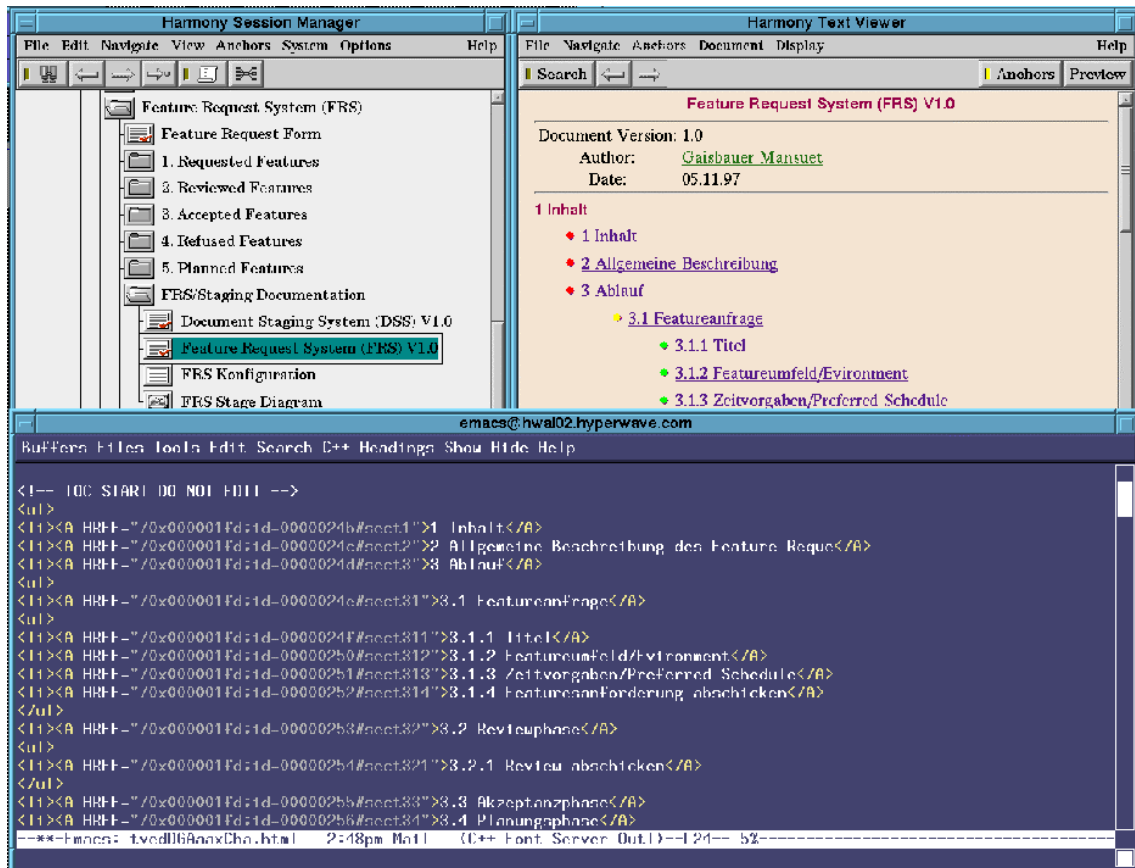


Figure 33: Harmony Editing Text with Emacs.

5.4.5 Moving and Copying Objects

Documents on a Hyperwave server are structured in collections and sub-collections. Harmony provides facilities to manage this structure. It is not only possible to move documents or collections to other collections, it is also possible to copy the objects. In Hyperwave copy is similar to the “ln” command in UNIX. The data of the object is not physically copied, but symbolically linked into a further collection.

In order to move or copy an object, the user has to select the object in the collection browser. Selection of the menu entry “Move” or “Copy” in the “Edit” menu opens the Harmony Move/Copy Dialog, as shown in Figure 34. In this dialog the user has to enter the name or object id of the target collection and to click “Move” or “Copy”.

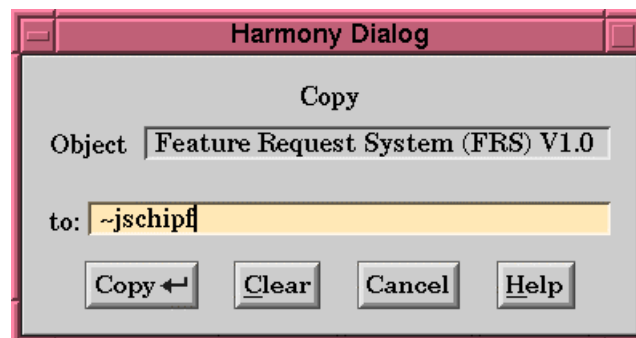


Figure 34: Harmony Copy Object Dialog.

5.4.6 Deleting Objects

To delete an object, the user has to select the object and activate the “Delete” item from the “Edit” menu. After this the Harmony delete dialog as shown in Figure 35 is opened.

As stated earlier an object can belong to more than one collection. If the object to be deleted is only a member of one collection it is physically deleted. If it is a member of more than one collection, the behaviour of the delete operation depends on settings in the delete dialog. If the checkbox “Delete all Occurrences” is checked, the object is deleted physically. If not checked, the object is only removed from the current collection.

In the case where the object is deleted physically, the user can decide whether all incoming and/or outgoing links should also be deleted.

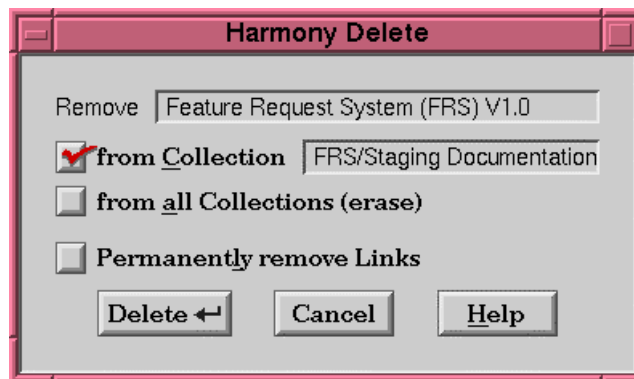


Figure 35: Delete Document Dialog.

The dialog for deleting collections is slightly different to that for deleting documents as shown in Figure 36. When deleting a collection, all immediate members and also all members of sub-collections (recursively) are deleted.

Therefore there exist two further input elements in the delete dialog for collections. One is the “With Confirmation” checkbox. If checked the user is asked for every object to be deleted, to confirm the delete operation.

The second is the “matching the HYPER-G DATABASE OBJECT QUERY” checkbox. If checked an input field is added to the dialog. In this field the user can specify a “Database query”. Only objects which match this query are deleted. This can be applied to data where only objects with specified attributes should be deleted. For example objects containing a particular phrase in their Title.

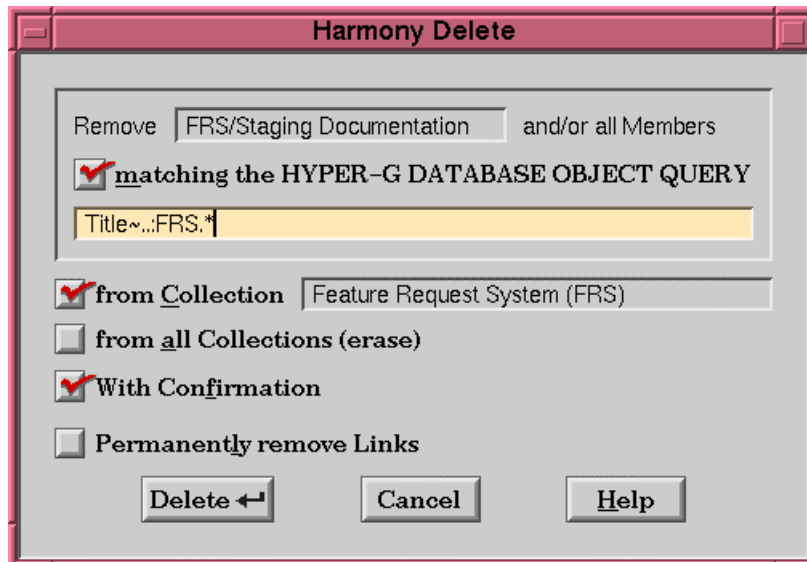


Figure 36: Delete Collection Dialog with Database Object Query.

5.4.7 Editing Object Attributes

Every object handled by a Hyperwave server has assigned several attributes. These attributes are stored in a separate database, separated from the object data. In Harmony attributes can be viewed and modified (given appropriate access rights) by the user.

In order to edit object attributes, the user navigates to the object. A click with the right mouse button on the object opens the Harmony attributes dialog. If the user has the right to edit the object attributes, an “Edit” button is visible at the bottom of the dialog. Figure 37 shows an attributes dialog.

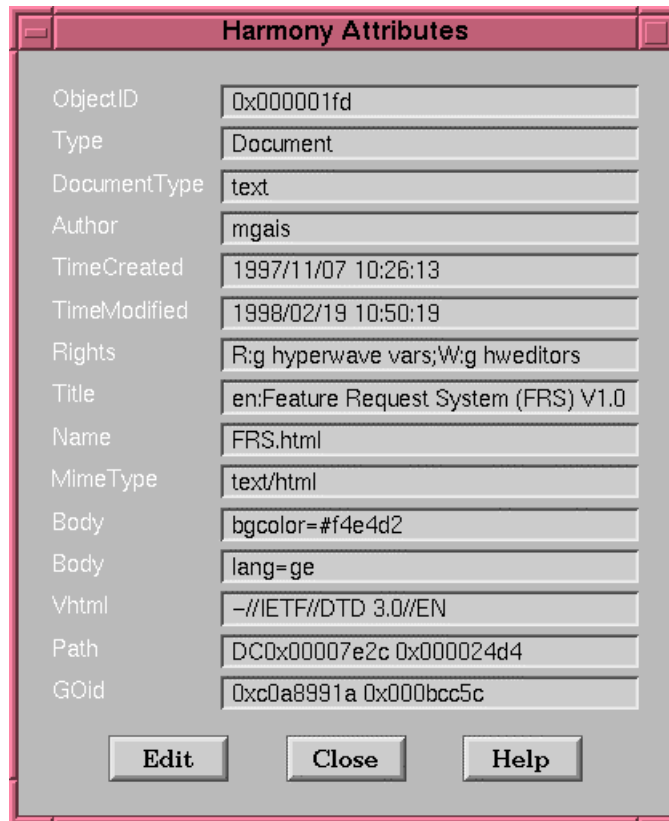


Figure 37: Harmony Attributes Dialog.

After a click on the “Edit” button, the dialog changes as shown in Figure 38. Attributes which are not editable are arranged at the top of the dialog, editable attributes grouped at the bottom. Attributes which are editable by the user are displayed in an edit text field. User editable fields include:

- Title
- Keywords
- Name
- TimeOpen
- TimeExpire

and many more.

Some of the attributes are handled by the Hyperwave server internally and therefore are not editable. These attributes are displayed for information only. System handled attributes include:

- ObjectID
 - Type
 - DocumentType
 - CollectionType
 - Subdocs
- and many more.

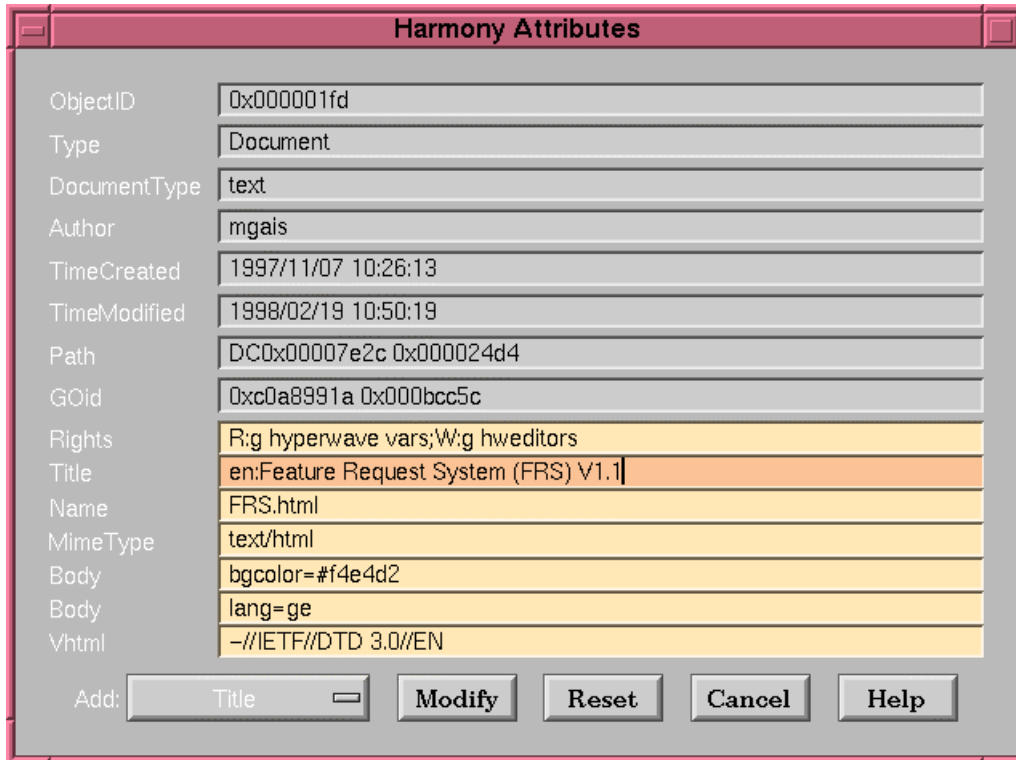


Figure 38: Harmony Edit Attributes Dialog.

Modify Attribute Values

The user modifies one or more attribute values, by changing the text in the edit field. When a change is made, the background colour of the edit field changes to indicate the modification.

Adding a New Attribute

To add a new attribute the user can select the desired attribute from the “add” listbox. A line with the attribute name and an input field is added to the dialog. The user can enter the new value in the input field.

Deleting Attributes

If an attribute is to be deleted, the text in the edit field has to be removed. Attribute fields with no value are removed from the object on insertion.

After all modifications have been done, the user can click the “Modify” button. All changes are committed to the database and the dialog changes into view mode. If the object is displayed in a browser and the modification influences the visualisation in the browser (for example a changed title) all appearances are updated as well.

5.4.8 Creating Links

One of the most important feature in Harmony is the facility for interactive link creation. A link in the sense of Hyperwave consist of a hot spot, the source anchor and a destination. The source anchor is a specially marked area in the document. Double clicking on the source anchor displays the destination object of the link. The destination may be a collection, cluster or a whole document. It is also possible to point to an area in a destination object, the destination anchor.

For Hyperwave the anchor itself is simply an object, stored separately from the document data. This means the anchor has attributes like all other objects. An anchor can therefore have for example a Title, Author or Rights attribute. An advantage of the anchor object is that anchors can be defined in document formats which do not support linking inherit in their format specification.

In Hyperwave links are supported for text, film, image, sound, 3D and PostScript documents. The position of the anchor in the document is stored as an attribute of the anchor. The visual representation of source and destination anchors is different in the different document types:

- In text documents anchors are a sequence of characters or an inline image.
- In film documents simple anchors are time slices of the whole area of the film. However anchors can also be rectangular or circular areas in the current viewed scene of the film. These areas can change their size and position over time. So for example, a moving car or a person can be the source and indeed, the destination of a hyperlink.
- In image documents anchors are rectangles, circles, ellipses or polygons.
- In sound documents anchors are time slices.
- In 3D documents source anchors can be objects or object groups, destination anchors are camera positions with an orientation.
- In PostScript documents anchors are rectangular areas of the document, or a given page.

A detailed description of possible link positions can be found in [Mau96].

In all viewers anchors can overlap each other and also an anchor can be a subset of another anchor. For example in a map of Europe, the whole of Europe can be a link to information to Europe, and as subset the countries and the capitals may also have such links.

5.4.8.1 Interactive Link Creation

To define a link, four steps are necessary: define a source anchor, define a destination anchor, assign anchor attributes and create the link with the given settings. The source and destination anchor of a link can be defined interactively using Harmony's native viewers.

Source Anchor

To define the source anchor, open the document in the viewer and mark the area for the link. In the text viewer for example, this is done by marking the corresponding text with the mouse.

Next, clicking “Define as Source” in the “Anchors” menu of the viewer opens the Harmony link creator as shown in Figure 39. Parameters for the source anchor are set by Harmony according to the area marked previously.

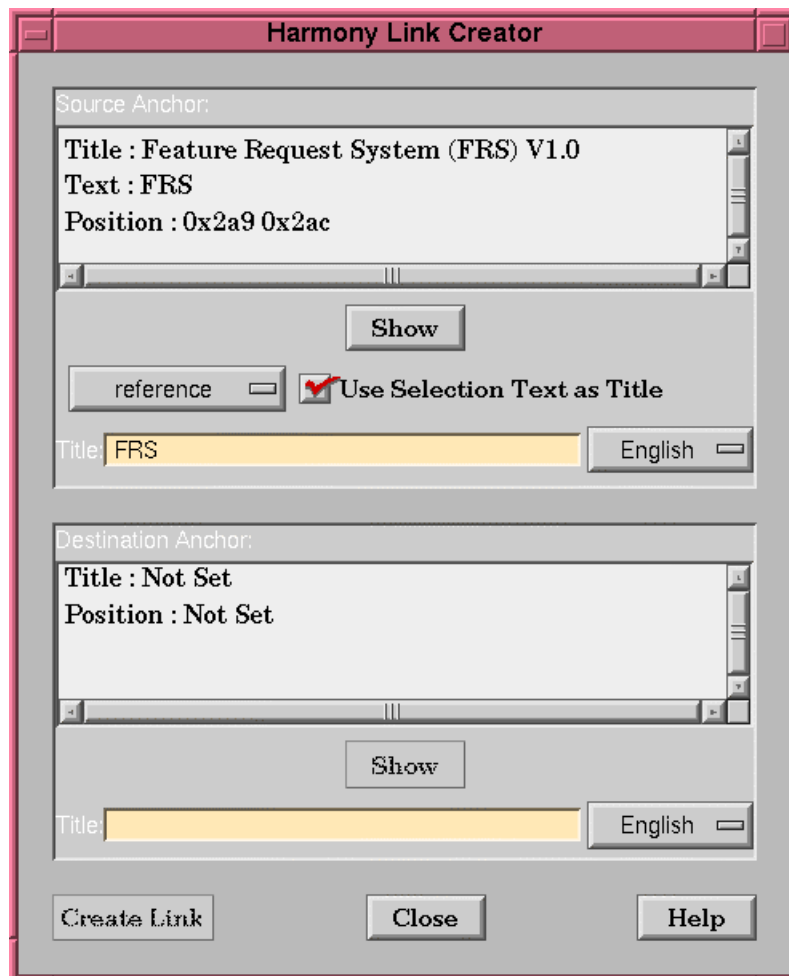


Figure 39: Harmony Link Creator, Source set.

Destination Anchor

Defining the destination anchor in the viewer is done the same way as defining a source anchor. The only difference is that the destination anchor can also be a whole document (Use Default Destination). To set a collection, cluster or a whole document as the destination anchor, it is also possible to use the collection browser. When the object is selected, “Define As Destination” from the “Anchors” menu of the collection browser also sets the destination anchor.

Link Attributes

One of the possible attributes for an anchor is the Title attribute. To assign a title to an anchor is useful in Hyperwave since anchors can also be found by a search. It is further possible to insert the anchor as member of a collection. In this case, the anchor is displayed in the collection browser with an anchor sign contained in the document icon. When activating the anchor the corresponding position is highlighted in the document.

Link Creation

After entering some optional attributes for the anchor object, a click on “Create Link” inserts the source and destination anchor into the Hyperwave server. All involved viewers are updated to visualise the newly created link.

Inline Images

In Harmony it is possible to interactively insert inline images into text documents. In the text document the place where the inline image should be inserted has to be selected. The inline image will be inserted at the beginning of the selection. The function “Define As Inline Source” from the “Anchors” menu of the viewer sets the position in the link creator dialog. The inline image itself has to be defined as the destination of the link. After inserting the link, the inline image is visualised in the text viewer.

There are several advantages to this link creator concept. Since link information is presented in the creator in a separate window, all navigational features of Harmony can be used to find the counterpart of a link. For example, it is possible to use the Harmony search and define one of the found documents as the destination of a link.

In order to create links to the same destination, the destination has only to be set once. Only the source anchor has to be defined for every link. For example several occurrences of the same inline image in the same or different documents.

Furthermore, it is not necessary to first set the source and then the destination. The user can first define the destination. This is very useful, when during browsing, a document is found to which another document should refer.

5.5 Expandability

Harmony is a very powerful tool to insert information into Hyperwave server and manipulate the stored information. It offers possibilities to create and rearrange the collection hierarchy. Harmony supports the editing of documents and gives the capability to create hyperlinks from and to documents and collections. The relationship between documents can be visualised and messages can be sent from one user to another.

One drawback of the rich features of Harmony is the resulting code size. To implement more and more features within one large program is very difficult. It is better to implement some features in separate modules, which can be utilised by Harmony. Such modules can be implemented independently from each other, and the coding task can be done by different users.

A further advantage of separated modules is easier distribution. The main part of Harmony can be kept small. If some additional features are needed by a user, the appropriate module can be downloaded on demand.

Harmony offers two methods to integrate other software and Harmony modules. One of the possibilities is to configure the Menu of the session manager to start external programs. Harmony further has an API (application programmable interface) which can be used to get information on the current state of the session manager and also to control the session manager and the document viewers.

5.5.1 Expandability of the Harmony Menu

In the Harmony session manager, it is possible to configure the menu of the collection browser to start external programs. The menu where new items are appended is a submenu of the “File” menu. When external tools are configured, the menu entry “Tools” appear and contain new items.

Just starting programs is not enough for tools which want to operate in conjunction with Harmony. It is also necessary to supply some status information and parameters along with starting the tools, such as:

- host and port Harmony is connected to
- the currently logged in user
- information on the current object
- parameters to request interactive input from the user
- information about the Harmony API (see below)

For a more detailed description on how to configure the menu see Appendix B. There you will find a complete list of parameters which can be passed to external programs. Figure 40 shows the Harmony collection browser with an extended menu.

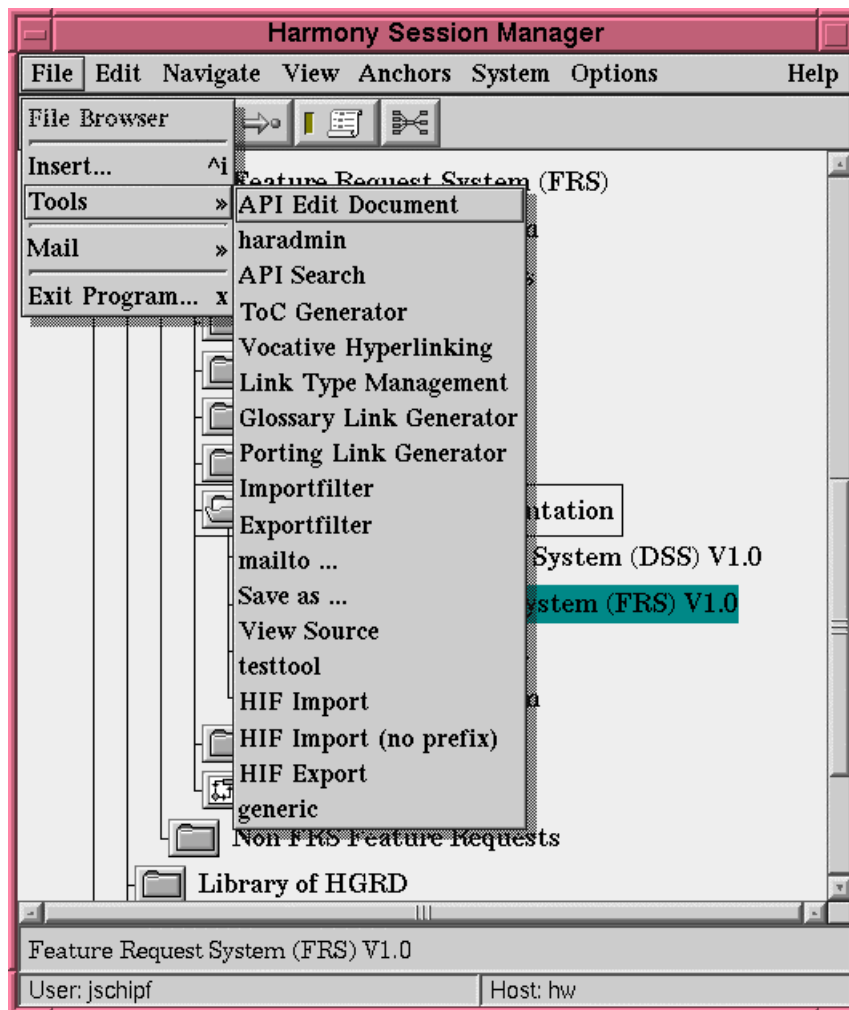


Figure 40: Harmony Collection Browser with Tools Menu.

5.5.2 The Harmony API

In the previous section a method was presented to start programs from within Harmony with the help of the menu. However, simply starting a program is not enough. The tool may want to use the capabilities of Harmony. The Harmony collection browser already has the capability to visualise and browse the structure of the Hyperwave server. The display and editing of documents is already implemented in Harmony.

There is no sense in re-implementing these features in external programs. To support the tools with the capabilities of Harmony, an API (application programmable interface) was implemented. With the help of this API tools can be seamlessly integrated into the environment of a Harmony session. External programs can use the API to communicate with Harmony. The API offers several types of interaction with the tools:

- There exist functions to get information about the current object in the collection browser. The attributes of this object can be requested and processed.
- It is possible for the tool to subscribe to state changes of Harmony. When such a change occurs, the tool is notified. One of these state changes is the navigation of the user in the collection browser. Every new currently selected object is reported to the tool. A possible

usage of this feature is to enable and disable editing buttons with respect to the document type currently selected.

- There exist API calls to open and close collections in the collection browser.
- Documents can be visualised in the appropriate viewers.
- The Search Browser can be opened and search parameters set and retrieved.
- Tools can initiate and stop editing operations in the different viewers. It is also possible to create links with and without interaction with the user.
- Not only can the tools initiate functions and operations, Harmony can also control the tools. Harmony can set the language for the user interface, map and unmap, and iconify and deiconify the tools.

For a more detailed description of all the API functions see Appendix D.

HyGen

A set of tools, which are implemented utilising the Harmony API, is known as HyGen (Hyperlink generation). HyGen offers semiautomatic generation of hyperlinks including, table of contents generation, a vocative link generator as shown in Figure 41 and a glossary generator as shown in Figure 42. These tools were implemented for the European Space Agency.

HarSearch

A further tool utilising the Harmony API [Rod97]. This tool extends the search functionality of the Harmony Sessionmanager with an improved user interface as shown in Figure 43.

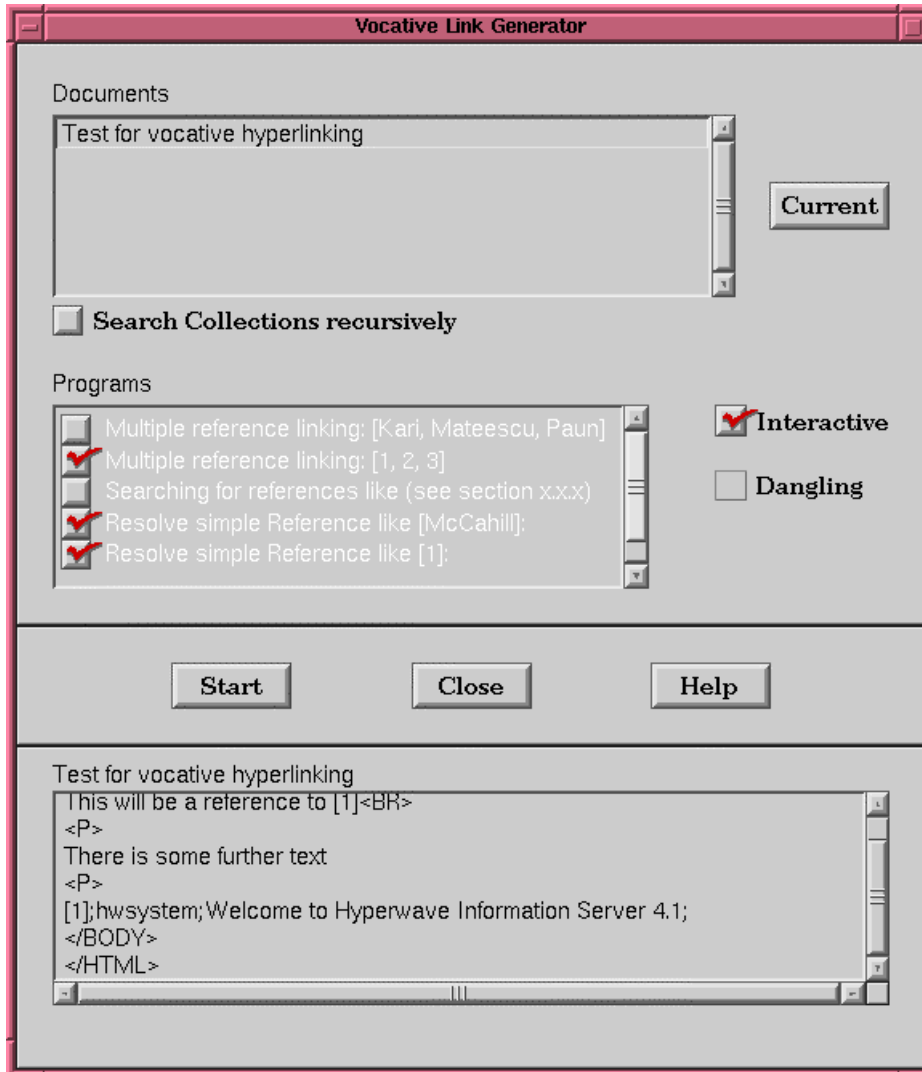


Figure 41: HyGen Vocative Link Generator.

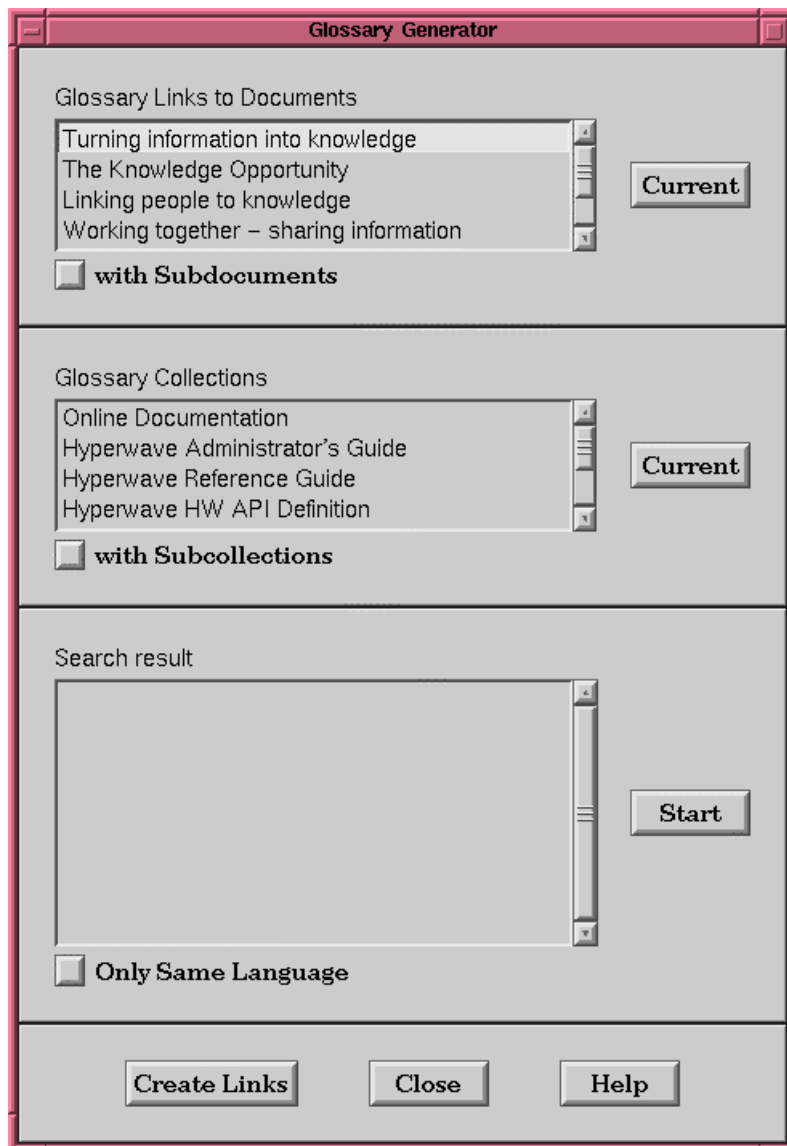


Figure 42: HyGen Glossary Generator.

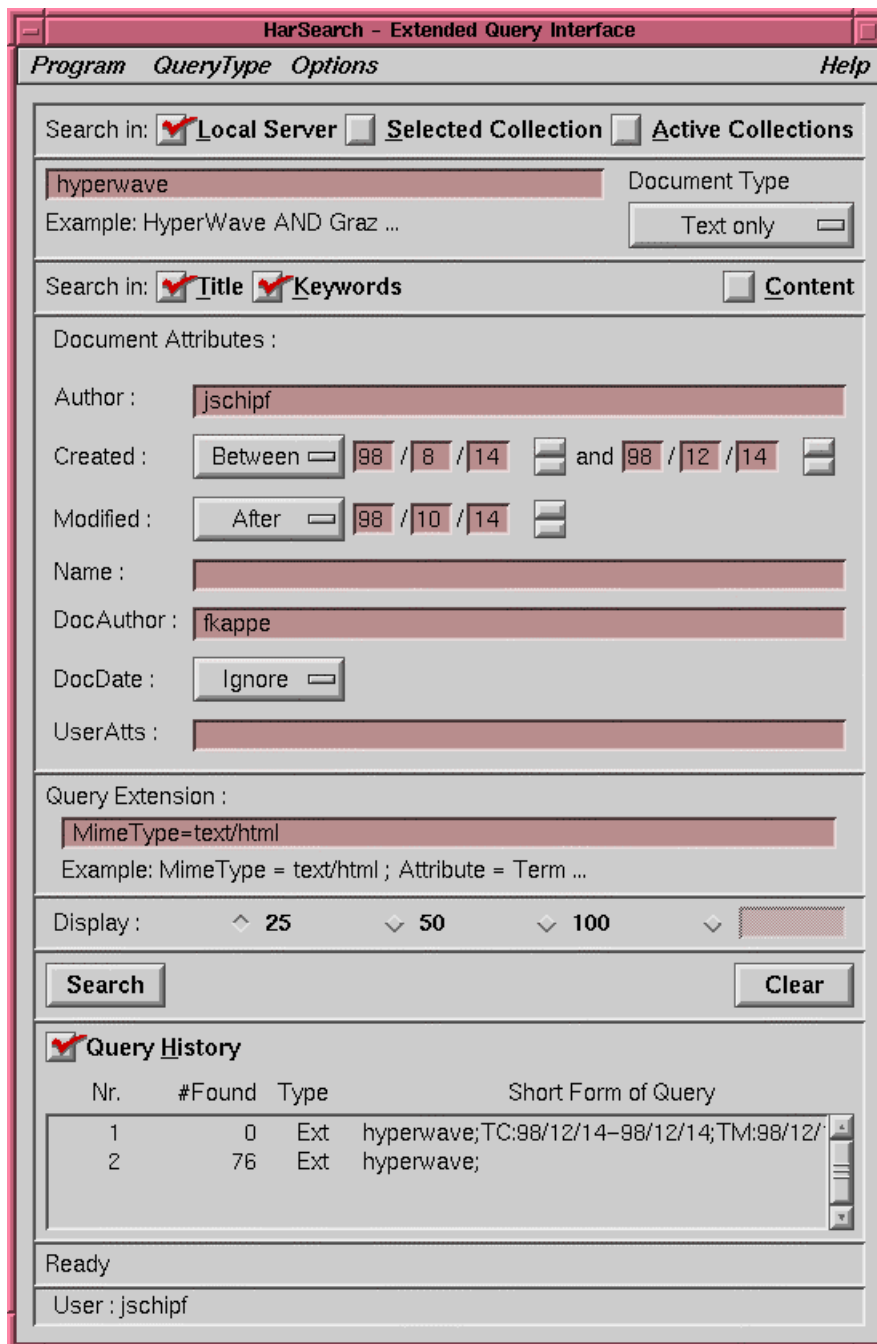


Figure 43: The HarSearch.

6 Selected Details of the Implementation

Harmony is implemented in C++ [Str91]. The user interface is built using the Interviews toolkit [LCI91] [LVC89]. In this chapter, particularly interesting details of the implementation are presented.

6.1 Implementation of the Local Map

The local map is a tool which generates a map of the structural information around a document as described in Section 5.2.2. This structural information may be parent-child relations or hyperlink relations. The structure built from this information, can be regarded as a cyclic directed graph. Documents are the nodes and links and relations form the edges. Since a link always points from one document to another the graph is directed. The graph may be cyclic as well, because a document A can have a link to document B, B to C and C again can have a link to document A. Document B already can have a link back to document A and therefore every edge can also be bi-directional. When the information structure is laid out, there are several issues which improve the readability and usability of the displayed graph.

1. The graph should have a hierarchical structure. These hierarchy is defined by the direction of the links. Document A which has a link to document B is the predecessor of document B.
2. When laid out, edges should have as few crossings as possible. If there are too many crossings, it is difficult to follow edges from one document to another. It is also difficult to find dependencies between several documents.
3. Since the local map is generated dynamically and on demand, the layout time should be short.

With the above restrictions, wire routing algorithms as used in VLSI design [BP83] are not suitable for this problem. In the design of chips run-time performance is of no matter. The quality of the layout is the most important issue, since the quality of routing directly affects the space and time performance of the resulting chip.

For dynamic and interactive layouts, layout quality can be neglected with respect to run-time performance. Users are willing to accept an imperfect layout, but they will not accept long computing times. Waiting too long contradicts the aim of the local map to help users find information and orient themselves. If a single layout of a local map takes more time than navigating around in order to find the same information, the local map makes no sense.

6.1.1 A Hierarchical Graph Layout Algorithm

A more capable algorithm for the above described problem and applied with some changes in Harmony, is a graph layout algorithm originally presented by Sugiyama [STT81]. This algorithm was introduced to visualise and understand hierarchical system structures.

One problem of Sugiyama's algorithm is the layout of cycles. In the algorithm cycles are collapsed to one node with the name of all nodes building the cycle. Since hypertext structure information which should be presented in the local map possibly include cycles, the algorithm should be adapted to handle cycles too. Some adjustments to Sugiyama's algorithm, which solved the above mentioned problem, were presented in [RDM87]. With some further adjustments, this algorithm was applied in the Harmony local map.

Nomenclature

The nomenclature used in this section is illustrated by the example in Figure 44. A node and an edge are defined in the usual way for directed graphs. In the local map a node is a document and an edge is a link. The nodes that can be reached from a given node by following the directed edges leaving the node are called the successors of the node. The nodes from which a given node can be reached by following directed edges are called the predecessors of the node.

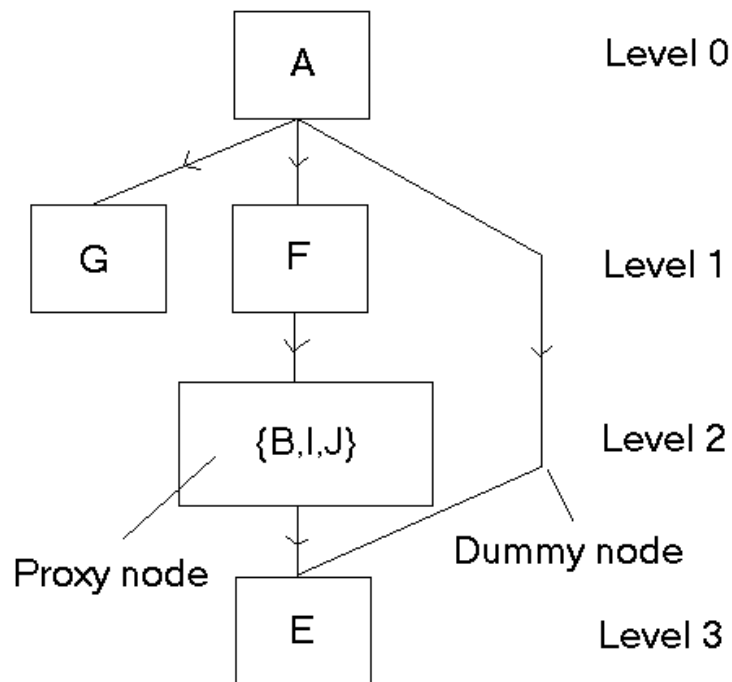


Figure 44: Nomenclature for layout algorithm.

A level is a discrete up/down position. They are numbered from top to bottom, starting with level 0 at the top of the graph. In a strict hierarchy, each node will have all ancestors at a lower level and all descendants at higher levels.

Edges that span more than one level are called long edges. A dummy node is added at each intermediate level to indices where the edge crosses the level. A long edge can change direction only at the positions of the dummy nodes. A dummy node has no size or label but is otherwise treated like a normal node by the layout algorithm.

A cycle is a set of nodes such that there is a closed directed path from each node in the set to itself that passes through the other nodes in the set. It is not possible to lay out the nodes in a cycle hierarchically, so that the original Sugiyama algorithm collapsed all the members of the cycle to a single node, called a proxy. The algorithm treated a proxy as single node and labelled it with the names of all the node in the cycle. In Figure 44, {B,I,J} is a proxy replacing the cycle $B \rightarrow I \rightarrow J \rightarrow B$.

Algorithm

The hierarchical layout algorithm has three phases. The first phase assigns nodes to levels. The second phase sorts the nodes on each level to minimise the number of edge crossings. The third and final phase fine tunes the positioning of nodes and routing of edges to make the layout easier to understand.

Phase One:

As first step of the algorithm, the graph is examined for cycles. Cycles are found by doing a depth-first search. Whenever a node is found to be its own descendant, a cycle is indicated. Cycles are broken by temporarily reversing the edge that completed the cycle and proceeding with the algorithm. When the graph is displayed, the reversed edge is drawn in the reverse direction (i.e. in the correct direction). The nodes connected by the edge appear on the levels that they would have occupied if the edge had been directed downwards.

The second step is to compute the transitive closure of all successors of nodes to determine which nodes have no successors, and therefore belong at the bottom of the graph. Nodes are assigned to levels by assigning nodes with no successors to the current level, by removing them and their edges from the closure, and performing the same operation on the next level up the graph until all nodes have been assigned to a level.

The third step to assigning levels is to break up long edges that connect nodes that are more than one level apart. These long edges are broken into segments, each of which goes between adjacent levels, and dummy nodes are inserted at the intervening levels. Dummy nodes and the segments of a long edge are treated by the algorithm as normal nodes and edges. The reason for segmenting the edges and creating dummy nodes is to allow the edges to bend at each level thereby providing flexibility in positioning of the two nodes connected to the long edge and permitting a more compact graph layout.

Figure 45 shows the layout of the graph after the completion of phase one of the algorithm. One problem of this algorithm is, that nodes like the nodes “up” or “help” in Figure 45 are placed at the bottom of the graph causing long edges, which makes the graph hard to read.

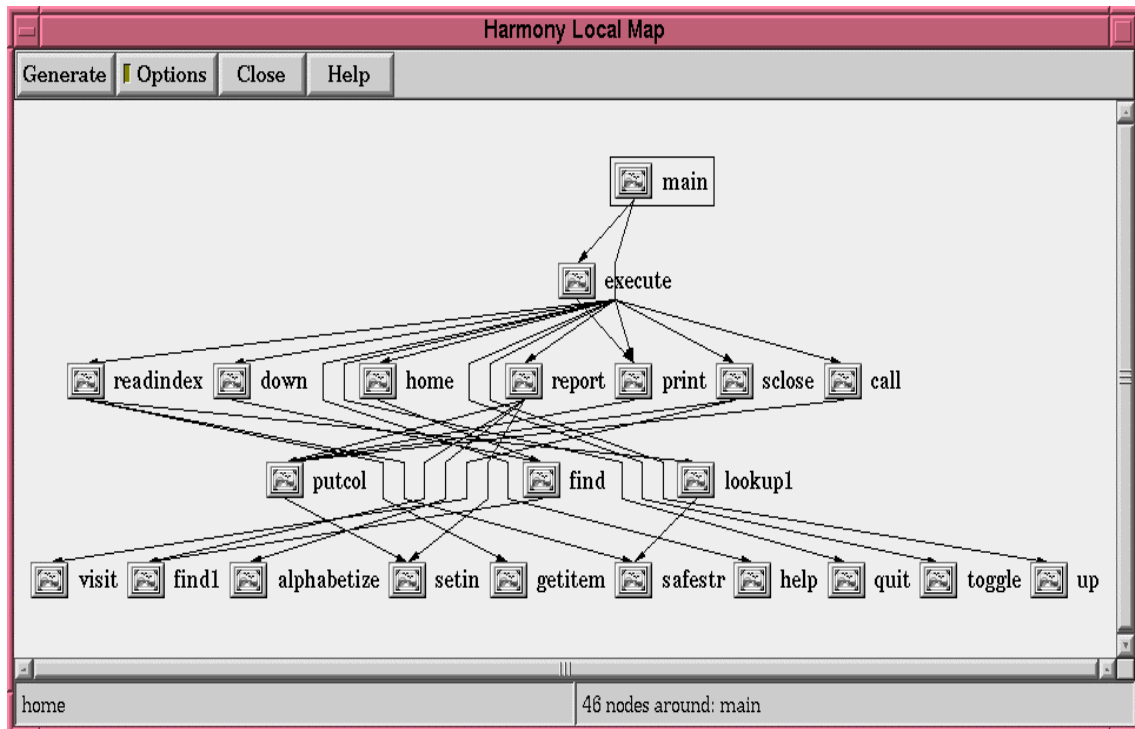


Figure 45: Local Map after step one, before level optimisation.

The algorithm used in Harmony solved this problem, performing an additional step between step two and three of phase one. This additional step minimises the level for each node in the graph. The algorithm used is as follows.

Nodes are sorted by increasing level. For each node in the graph, the minimum of the levels of all descendants of the node is computed. If the minimum + 1 is smaller than the current assigned level, the level minimum + 1 is assigned to the node. If no descendants are present the old level is preserved. The algorithm continues with the next nodes. Figure 46 shows the same graph as in Figure 45 with shortened levels.

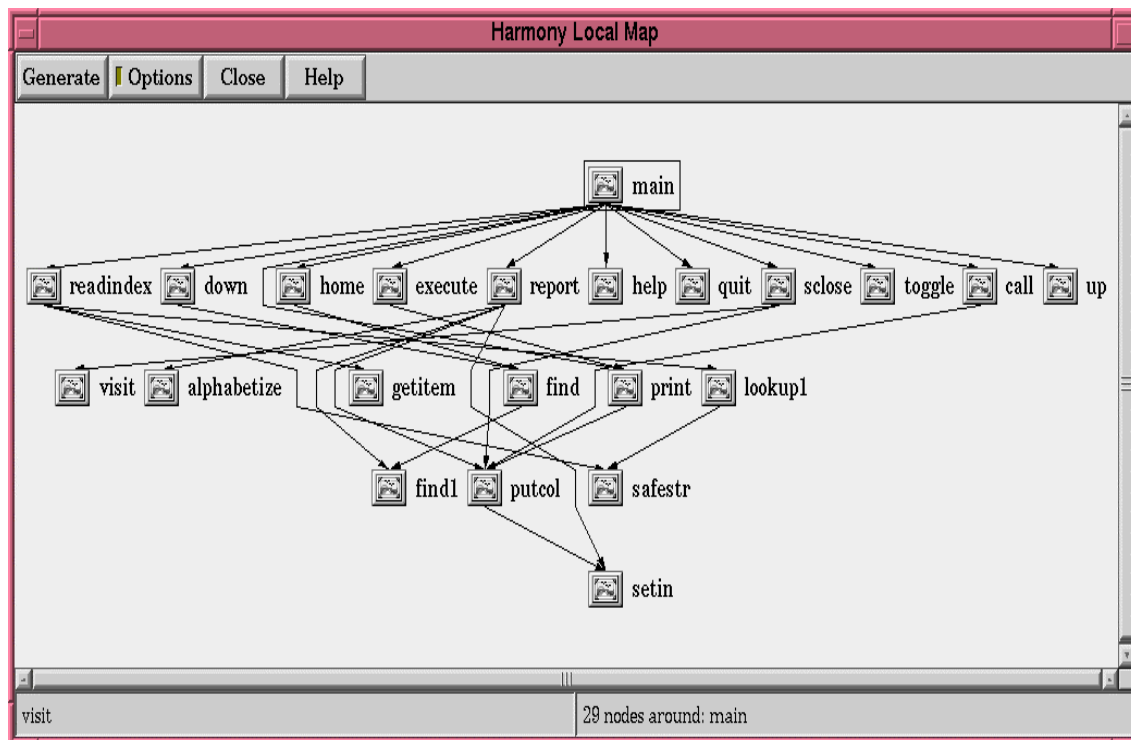


Figure 46: Local Map after step one, after level minimisation.

Phase Two

The second phase of the algorithm makes multiple passes over the graph, changing the order of nodes in each level to minimise the number of edge crossings. The first pass begins at the top level of the graph and moves down. The second pass begins at the bottom level and moves up. Subsequent passes alternate between top-down and bottom-up passes until the algorithm terminates.

A metric that estimates the best horizontal position of each node, called the barycentre, is used to order the nodes on a level. The up-barycentre (down-barycentre) of a node is the average position of its immediate predecessors (successors) including dummy nodes on the previous (next) level.

The first pass (downward) sorts by down-barycentre. The second pass (upward) by down-barycentre. The third and subsequent passes sort by the average of the up- and down-barycentres of a node.

The second phase of the algorithm terminates when all edge crossings have been eliminated or after a fixed number of passes have been made over the graph. Figure 47 shows the graph after the second phase. Notice that the number of crossings has been reduced.

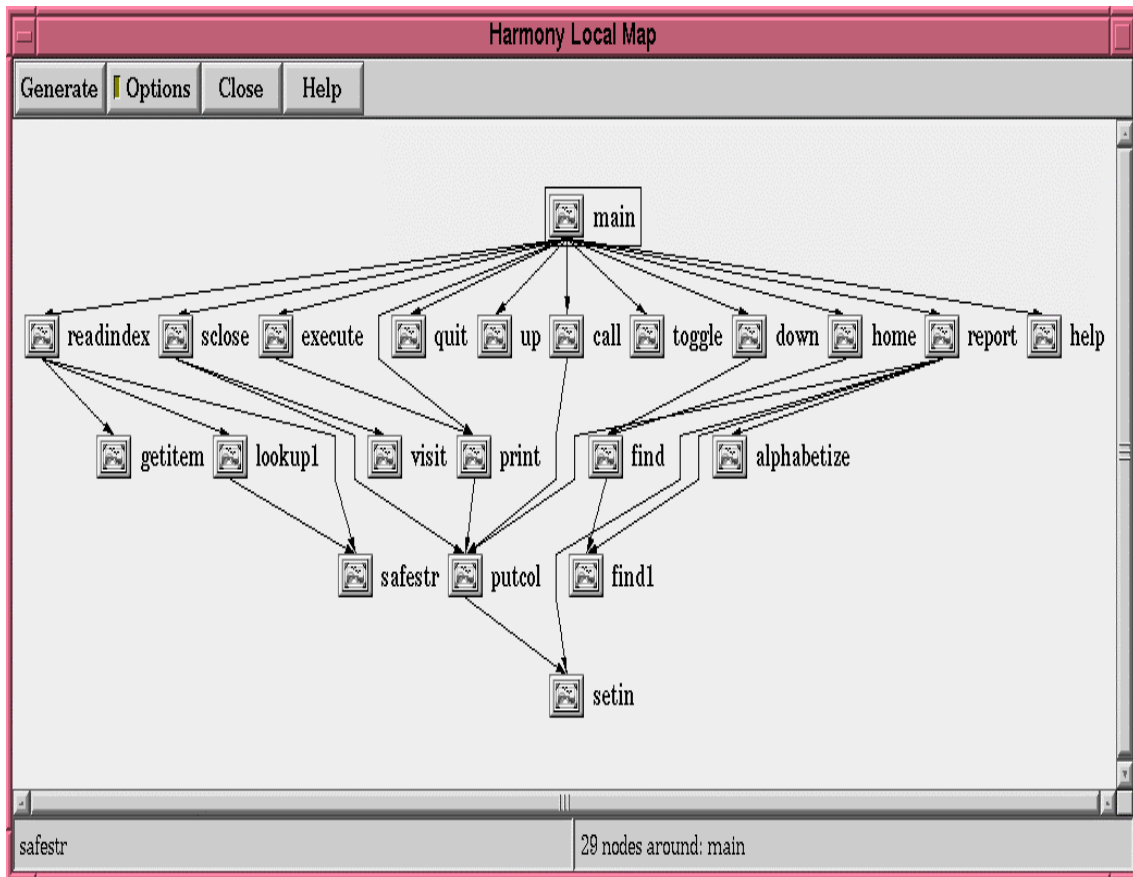


Figure 47: Local Map after step two.

One problem occurs in the above described crossing minimisation phase. The algorithm does not eliminate certain edge crossings because the level-sorting heuristic considers only pairs of levels. Optimisations needing consideration of more than two levels are not done. The current algorithm only uses local information (i.e. the positions of immediate predecessors and successors) when sorting nodes on a level. The heuristic looks for a local but not a global minimum.

This problem is not solved but reduced in the algorithm used for the Harmony local map through an additional step before the beginning of the second phase. In Phase one, after all nodes have assigned levels, an initial placement is applied. First, in the lowest level all nodes are placed with a unique distance. Starting with this level, the children of nodes are placed close to the predecessor. This leads in a starting layout with all nodes connected as far as possible in one line.

Phase Three

The third phase of the algorithm prepares the graph for visualisation. Upto now all nodes have had a size of zero. Real nodes typically have an icon and some text and hence a size. The positions of the nodes have to be adjusted to not overlap each other. It is also necessary to assign distance between the different levels, taking into account the size of the nodes.

When visualising the graph, the first attempt was to arrange the nodes from top to bottom. In other words to place the node with the level zero at the top. All further levels are placed below the preceding level. The typical structure displayed in a local map for Hyperwave has many nodes in one level. This nodes represent collections or documents which typically have long titles. Arranging them non overlapping leads to a very wide map. In order to view this

map, extensive use of the scrollbar is necessary. Therefore, the node arrangement was changed to place the nodes from left to right. Figure 48 shows a local map laid out from left to right. In contrast to Figure 47 the required space is reduced.

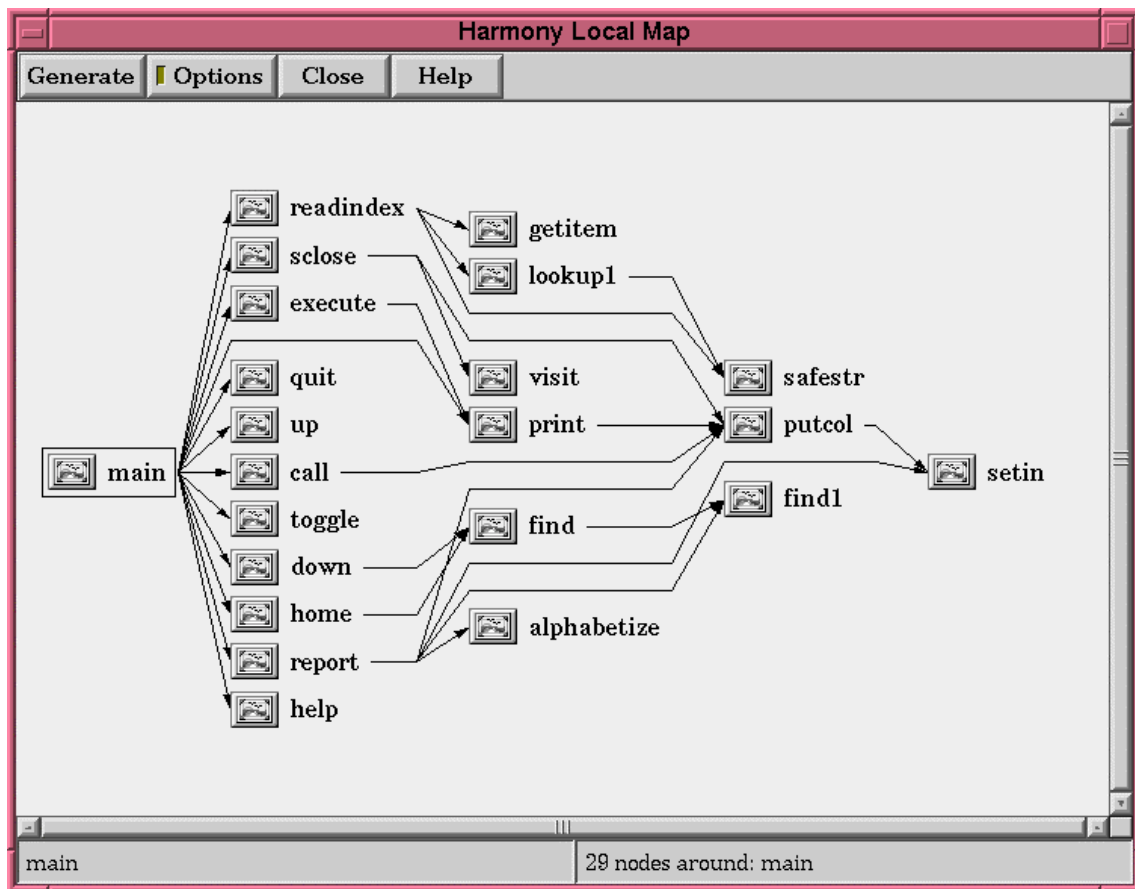


Figure 48: Local Map laid out from left to right.

6.2 Location Feedback

In Harmony every time an object is visualised in the collection browser, location feedback is provided using the tree. Location feedback is also provided when a link in a document viewer is activated, or an object is found using the Harmony search. If, for example, a matching document in the Harmony search result list is single clicked with the mouse, the document is visualised in the collection browser. If the document is not already in the displayed tree, the document is added to the view. To do this, a path to the root collection is determined, and displayed in the tree.

When determining the path, some problems have to be taken into account. In Hyperwave, documents can be contained in more than one collection or cluster. Furthermore, some of the collections on the way up the hierarchy may be not accessible by the user. The algorithm used to solve this problems is as follows.

When the current object is not in the tree get all parent collections of the objects. If one of the parent collections is already displayed in the tree use found path and display it in the collection browser. If none of the parents is already displayed in the tree add all parent collections to a stack. Take the first parent from the stack and continue the algorithm with this collection.

This algorithm is similar to performing a depth-first search for already displayed collections and seems to lead to an arbitrary path. This effect is reduced in Harmony due to an internal object attribute cache. Every displayed object is added to this cache. When parents of an document are queried, it is first tested if the object is already in the cache. If so, the cached objects are added to the parent list. All other objects are appended to this list. During the above described algorithm already visited objects are considered first, and therefore the found path typically matches already visited objects.

A problem arises, if none of the potential paths can be followed to the root collection. This may happen if the user has less access permissions. In this case, Harmony displays a dotted line for the edge to the root collection as shown in Figure 49.

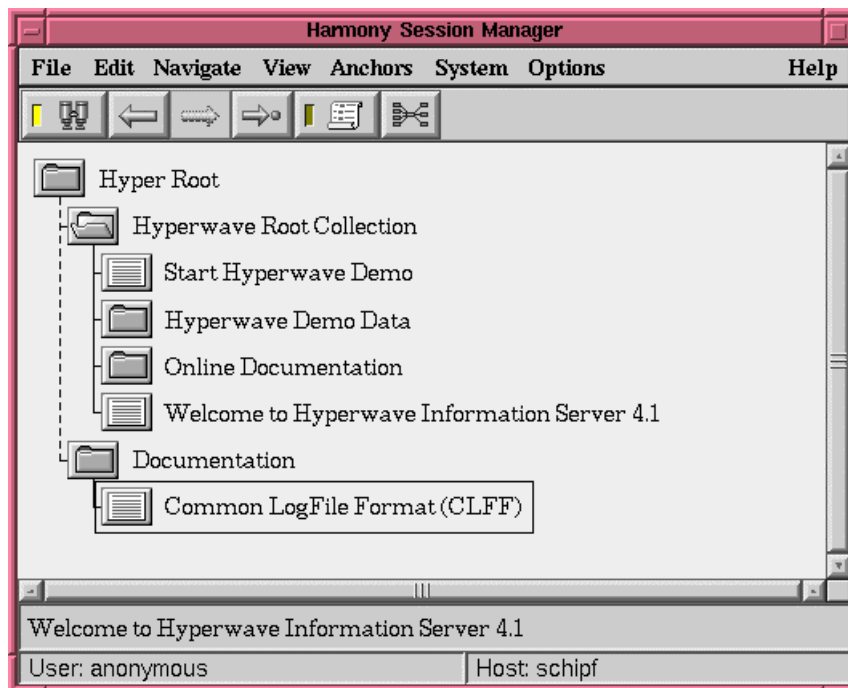


Figure 49 : Collection Browser with no path to root collection.

7 Extensions and Further Work

The Harmony project has now finished and development stopped. This chapter nevertheless describes possible extensions to Harmony, especially improvements to the user interface.

One of the most important features not already implemented is drag and drop. At the moment all operations, such as move and copy are done by selecting an object, opening a dialog and entering the destination. A more intuitive approach would be to move and copy objects with drag and drop, but this requires extending the underlying InterViews user interface toolkit.

A further inconvenience, which leads to unnecessary work, is that it is not possible to select more than one object at the same time. In order to move four objects the same operation has to be done four times. The solution to this problem is clear. Harmony should support multi selection of objects.

In the collection browser a second display mode, the partial display mode could be implemented. In this mode not all collections and documents are visualised, only the current collection, their children and all collections up to the rootcollection. The advantage of this mode would be the reduced amount of clutter, since much fewer objects are displayed.

In the local map it is possible to display the link structure around a document. If there are more than one level of incoming and outgoing links, it is nearly impossible to display the whole map without scrolling. As solution for this would be to implement the so-called fisheye views [SB92] [Fur86]. A fisheye view displays the same information as the local map. The document in the centre, the current object, is displayed in more detail than objects at some distance from the focus. This leads to a view similar to looking at the local map through a magnifying glass. The distance of the objects to the focus can be determined in different ways. The simplest one is to measure the distance in co-ordinates of the original layout. A second way would be to count the steps necessary to go from the focus to the object, following the edges of the graph. The fisheye algorithms are already implemented in Harmony but have not found their way into the user interface. One of the major problems was the scaling of the objects. An algorithm to choose a font according to the current size of the object would have to be implemented.

8 Concluding Remarks

Hyperwave is an object-oriented document management system which can be accessed over the Internet. Documents are stored in collections and clusters, and form a hierarchical information structure. Every object inserted into Hyperwave has additional meta information, the Hyperwave object attributes. These attributes can be used for visualisation or search operations. Hyperwave further provides a powerful built-in search engine for both attributes and the content of documents. Access to documents and collections can be restricted or granted to users and user groups, facilitating the built in user management or external user databases.

Harmony is the UNIX X-11 viewer for Hyperwave and consists of two parts. The Harmony viewers, and the Harmony session manager.

The Harmony session manager visualises the collection structure of Hyperwave in the collection browser, using a dynamically generated hierarchical tree. As a special feature of the Harmony session manager, location feed feedback is provided for every document found through searches, or activated using hyperlink navigation. The document hyperlink structure and also child-parent relations are clearly presented in the local map. A powerful user interface to the Hyperwave search is provided within the session manager.

The Harmony session manager further has rich online authoring functionality. It is possible to insert documents and collections into the Hyperwave server and also to delete them. Documents can be moved or linked within the collection hierarchy, restructuring the information. The attributes of documents can be viewed and modified. Hyperlinks can be edited online using the session manager, by selecting source and destination of a link using the mouse in the Harmony viewers, thereby facilitating the session manager's navigational features.

With the Harmony session manager API it is possible to extend the session manager with external programs, which can use the session manager for functions like search, hierarchical navigation or attribute and document editing. The Harmony session manager further provides a messaging system, where users can chat with each other or system messages from the Hyperwave server are displayed.

The Harmony session manager provides linking capabilities for its document viewers. Document viewers exist for many common document formats, all with the capability to visualise hyperlink information. Hyperlinks are even supported for document formats with no native hyperlink functionality, such as movies, audio or PostScript.

Since the commercialisation of Hyperwave, development of Harmony has finished. Harmony has been superseded by a WWW interface to Hyperwave, which can be accessed using standard web browsers. Up to now many features of Harmony have been re-implemented in the WWW interface or replaced using external viewers and editors. Nevertheless, some features of Harmony, such as time-dependent links in movies are still unique.

9 Bibliography

- [And95] Andrews K.: "Visualising cyberspace: Information visualisation in the Harmony internet browser". In Proc. of IEEE Symposium on Information Visualization (InfoVis95), pages 97-104, Atlanta, Georgia.
- [BCG94] Bernes-Le T., Cailliau R., Luotonen A., Nielson H. F., Secret A.: "Hyper-Text Markup Language (HTML)". Available at <http://www.w3.org/pub/WWW/Markup/Markup.html>.
- [BP83] Burnstein M., Pelavin R.: "Hierarchical wire routing". IEEE Trans. Computer-aided Design of Integrated Circuits & Systems. CAD-2, Pages 223-234, 1983.
- [DH95] Dalitz W. Heyer G.: "Hyper-G: Das Internet-Informationssystem der 2. Generation". dpunkt, Germany, October 1995.
- [Eyl95] Eyl M.: "The Harmony Information Landscape Interactive, Three- Dimensional Navigation Through an Information Space". Master's thesis, Graz University of Technology, October 1995.
- [EH89] Edwards D. M., Hardman L.: "Lost in Hyperspace: Cognitive Mapping and Navigation in a Hypertext Environment". In MCALEESE R. (editor), Hypertext: Theory into Practice, Pages 105-125, Blackwell Scientific Publications Ltd., 1989.
- [Fas93] Faschingbauer H.: "Volltextsuche in grossen Hypermediasystemen". Master's thesis, Graz University of Technology, 1993.
- [KM93] Kappe F., Maurer H.: "Hyper-G: Ein verteiltes informationssystem und seine anwendungen". In Proc. Hochschul-Computer-Forum '93, October 1993.
- [Fur86] Furnas G. W.: "Generalised fisheye views" In Proc. ACM SIGCHI 86 Conf. on Human Factors in Computing Systems, Pages 16-23, 1986
- [Gai94] Gaisbauer M. J.: "Darstellung von SGML-codierten Hypertextdocument mit einem objektorientierten Interfacetoolkit". Master's thesis, Graz University of Technology, January 1994.
- [Gei97] Geiger G.: "A Digital Audio Player for the Hyperwave Internet Server". Master's thesis, Graz University of Technology, 1993.
- [Gil92] Gilly D.: "Unix in a Nutshell". O'Reilly & Associates, Inc, Sebastopol, California, June 1992.
- [Gol90] Goldfarb C. F.: "The SGML Handbook". Oxford University Press, 1990.
- [Kap91] Kappe F.: "Aspects of a Modern Multi-Media Information System". PhD thesis, Graz University of Technology, June 1991.

- [Kro94] Krol E.: "The whole Internet: User's Guide and Catalog". O'Reilly & Associates, second edition, April 1994.
- [LCI91] Linton M. A., Calder P. R., Interrante J. A., Tang S., Vlissides J. M.: "InterViews Reference Manual Version 3.0.1". Stanford University, October 1991
- [LVC89] Linton M. A., Vlissides J. M., Calder P.R.: "Composing User Interfaces with InterViews". IEEE Computer, 22(2): Pages 8-22, February 1998.
- [MA95] McCahill M. P., Anklesaria F.X.: "Evolution of Internet Gopher". Journal of Universal Computer Science, 1(4): Pages 235-246, April 1995.
- [MAN95] Mayerhover V., Andrews K.: "Harmony user guide: Version 1.2", Technical Report, IICM, Graz University of Technology.
- [Mar95] Marschall B.: "Integration of digital video into distributed hypermedia systems", Master's thesis, Graz University of Technology, 1995.
- [Mau92] Maurer H.: "Why Hypermedia Systems are Important". In Proc. ICCAL '92, Pages 1-15, LNCS 602, Springer Pub. Co., 1992.
- [Mau96] Maurer H.: "Hyper-G now Hyperwave. The next Generation Web Solution". Addison-Wesley, 1996
- [Pic93] Pichler M.: "Interactive Browsing of 3D Scenes in Hypermedia: The Hyper-G 3D Viewer". Master's thesis, Graz University of Technology, October 1993
- [PiP96] Pichler P.: "Integration von digital Bildern in verteilte Hypermedia-Systeme". Master's thesis, Graz University of Technology, 1996.
- [RDM87] Lawrence A. R., Michael D., Messinger E., Meyer C., Spirakis C., Tuan A.: "A Browser for Directed Graphs". Software-Practice and Experience, Vol. 17(1), Pages 61-76, January 1987
- [Rod97] Rodiga A.: "The HarSearch Similarity Map: Visualising Search Result Sets Using a Maximum Similarity Spanning Tree". Master's thesis, Graz University of Technology, May 1997
- [SB92] Sarkar M. Brown M. H.: "Graphical Fisheye views of Graphs". CHI, Pages 83-91, May 1992
- [Str91] Stroustrup B.: "The C++ Programming Language". Addison-Wesley, Reading, Massachusetts, second edition, 1991
- [STT81] Sugiyama K., Tagawa S. Toda M. "Methods for visual understanding of hierarchical system structures". IEEE Trans. on Sys. Man, and Cyb. SMC-1. Pages 109-125 1981
- [VER98] "Developer's Kit". Verity, Inc., Sunnydale, California, 1998.
- [Win95] Windisch C.: "HarAdmin: A Graphical Tool for Hyper-G Server Administration". Master's thesis, Graz University of Technology, November 1995.
- [Wol96] Wolf P.: "Three-dimensional information visualisation: The Harmony Information Landscape", Master's thesis, Graz University of Technology,

1996.

Appendix A. Harmony Key Bindings

Key bindings in the session manager

Key	Function
x	Exit program (with confirm dialog)
^x	Exit program (without confirm dialog)
Tab	Next focus (used in dialogs and search)
Return	Open/close collection, display collection head, display documents
+,right	Open collections, go to first child, display collection head, display documents
-	Close collection/cluster
c	Show/hide children of collection/cluster, do not display collection head
Space, n	When current document/collection/cluster not seen display, else go to next in list and display. Collection head is displayed.
p	When current document/collection/cluster not seen display, else go to previous in list and display. collection head is displayed.
^c	Close all collections, open path to the local server
^l	Delete links
Up/down	Select previous/next object
^up/^down	Scroll one line up/down
Shift up/ Shift down	Select previous/next object (also object with no title)
Page up/ page down Alt v/^v	Scroll one page up/down
Left	Select parent collection
a	Show object attributes
e	Edit object attributes
^e	Edit multiple object attributes dialog
i	Map identify dialog
s	Map/unmap search dialog

b	One step back in history
f	One step forward in history
Shift l (L)	Generate local map
Shift r (R)	Show references
Shift a (A)	Show annotations
Shift p (P)	Show parents
^a	Annotate object
h	Map/unmap history window
Shift h (H)	Go to home
g	Go to object
^d, delete	Remove objects dialog
^i, insert	Insert object into collection
^s	Display database status
^u	Display users online
Shift m (M)	Move object
Shift c (C)	Copy object
Alt s	Display the hostname where the object is located

Key bindings in the search dialog

Key	Function
<ESC>	Close search dialog
Alt e	Toggle extended search
Alt l	Toggle local server
Alt a	Toggle active collections
Alt s	Toggle selected collections
Alt t	Toggle in title
Alt k	Toggle in keyword
Alt c	Toggle in content
Return	Start search when focus on search input

Key bindings in the status browser

Key	Function
<ESC>	Close the status browser

ENTER	Map the send message window
Alt a	Select all users
Alt d	Unselect all users

Key bindings in the send message dialog

Key	Function
<ESC>	Close the send message dialog
Alt s	Send the message
Alt c	Clear the message windows
Alt m	Save the content of the received messages window

Key bindings in the insert dialog

Key	Function
<ESC>	Close the insert dialog
Alt i	Insert new object
Alt r	Reset dialog
Alt h	Show online help

Appendix B. Configuring External Tools in the Menu

A user-configurable menu for starting external programs is integrated in the session manager menu. The following X11 resources are used to control the menu.

Harmony*tools	tool1 tool2 tooln
Harmony*Tool.tool1.commandline	Commandline
Harmony*Tool.tool1.menuentry	Menuentry
Harmony*Tool.tool2.commandline	Commandline
....	

Whereby:

Menuentry	String appearing in the File/Tools menu
commandline	command line to execute

The following variables are supported in the command line and are substituted with their value.

\$user	The users current login name
\$host	The current Hyperwave host Harmony is connected to
\$port	Port of the current connection to the Hyperwave server
\$localhost	Host where Harmony is running
\$apiport	Port of the API interface
\$filename	A file browser is displayed and \$filename is replaced by the chosen file
\$filenames	A file browser with multi-select is displayed and \$filenames is replaced by the chosen files
\$title	Title of the current object
\$id	ID of the current object
\$goid	Global object ID of the current object
\$name	Name of the current object
\$url	URL of the current object

\$display	Display Harmony is running on
\$arg0	Is substituted by the file name of the temporary file containing document data
\$interactive	A dialog is displayed to let the user enter the value which replaces this variable. For each occurrence of this variable a new dialog is presented. No variable substitution is performed on the input string.
?(term)	Term is displayed before the input field e.g.?(Enter Prefix:)
?[term]	Same as?(term) but command line is displayed as well
\$generic	Has the same function as \$interactive but also does variable substitution in the entered string.

Appendix C. Search Syntax in Harmony

Attribute Search

Entering a single search term (for example 'foo') will find all objects (documents, collections, destination anchors) with the word 'foo' in the title or as a keyword. The search is not case-sensitive. National character sets can be entered as 8-bit codes (ISO Latin 1). The output is a list of matching objects, sorted in decreasing order of score.

Multiple search terms are ANDed together, for example 'foo bar' will match only objects with both 'foo' and 'bar'. The OR operator is '|', for example 'foo | bar' will match all objects having either 'foo' or 'bar'. These operators may be combined, but without brackets. There is no NOT operator.

Prefix search is supported, for example 'foo*' matches anything containing a word starting with 'foo'. You may use the AND and OR operators as above (for example 'uni* california').

Due to a (deliberate) limitation in the server, you cannot use '*' to match all objects.

Content Search

Content or full text search performs searches on the content of text documents on one or more Hyperwave servers. The Hyperwave full text engine is able to perform a combination of ranked, boolean, weighted, fuzzy, and nearest-neighbour searches.

Query Syntax

Literals are enclosed in single apostrophes ('). Expressions in square brackets [] are optional, braces {} denote 0 or more occurrences, and a vertical bar | denotes alternatives:

```
expr      ::= term {orop term}

term      ::= factor {andop factor}

factor    ::= node [ '{float}' ]

node      ::= word {word} | '('expr')'

orop      ::= '||' | '|'

andop     ::= '&&' [optionlist] | '&' [optionlist]
```

```
optionlist ::= '[' option { ',' option } ']'  
option     ::= 'F' | 'f'
```

Examples:

1. "computer"

A single search term returns a ranked list of matching documents normalised such that the top-ranking document(s) in the set score 100%. The ranking is computed based on the number of times the word appears in the document, where it appears (title, headings, and subheadings count more), and the size of the document.

As the Hyperwave full text engine employs a stemmer, the above example would also match documents containing "computers", "computing", "compute", and the like. Certain very common words such as "is", "was", "the", etc. (so-called stop words) are not indexed and cannot be searched for. The stemmer and stop list are language-dependent and are currently available for English only. The search is not case-sensitive.

2. "tell me about computer conferences" (nearest neighbour)

The first three words are stop words and are ignored. The system will search for documents containing the other two words. Matching documents are ranked according to their score for these two words. Note that documents still match (but perhaps with a lower score), even if they only contain one of the two words.

3. "computer & conference" (boolean search)

Sometimes you want to specify that matching documents really contain all the search terms. The AND operator (& or &&) accomplishes this. You may also use the OR operator (| or ||) and parentheses to construct arbitrary boolean expressions. There is no NOT operator.

4. "computer &[f] conferencing" (fuzzy boolean search)

The 'f' makes the AND operator a fuzzy AND, i.e. it will still match documents containing only one of "computer" and "conferencing", but with a lower ranking.

5. "computer &[f] conferencing{3.5}" (weighted boolean search)

Similar to above, but allows you to make "conferencing" 3.5 times as important as "computer" in the ranking.

Appendix D. The Harmony API

This appendix provides a list with short descriptions of the API functions for both the API tool and the session manager API.

The Harmony Tool API

The following functions may be called by the Harmony session manager. The API tool should react to the calls in the described way.

void iconify ()	Iconify the window(s)
void deiconify ()	Deiconify the window(s)
void moveto (float, float)	Move to window to the given co-ordinates
void resize (float, float)	Resize the window to the to the given dimensions
void map ()	Map the window. (create a screen representation)
void unmap ()	Unmap the window. In difference to the iconify call as response to this call all screen representations should be removed.
void raise ()	Bring the window to the top of all other windows.
void lower ()	Lower the window with respect to other windows on the desktop.
void terminate ()	This function is called if the session manager wants the tool to be terminated.
void setLanguage (HgLanguage::Language, const RString& langprefs)	The session manager calls this function to change the user interface language of the tool. langprefs is a list of languages (en: ge: ...) separated by “ “.
void notify (Notify::NotifyEnum)	The tool is notified by this function, if some events the tool is interested in occurred in the session manager. (the events are set by the SMAPIclient member setNotifyMask)
void waitForNotify (BitSet& mask, int allowother=1)	This is not a RPC function, it cannot be called by the session manager. Instead, it is called by the tool to wait for some specified events. mask is the mask which the tool wants to wait for. When allowother is true other notifies from the session manager are reported as normal noti-

	fies.
cancelEdit (long refno)	The session manager calls this function to notify the tool that the edit process with the number refno is no-longer active.
dataForDoc (const RString& anchors, const RString& host, int port, long refno)	The edit process with the number refno has finished. The data (text) can be fetched at the host and port. The anchors of the document are stored in anchors. The fetched data may be pre-pended by the Hyperwave header.
reset (const RString& type)	When this function is called the tool should be reset to the initial statement (till now contains no data)
HgViewer::VwError error ()	Returns the error state of the HgViewer.
static boolean isFatal (APIError::APIErrorEnum)	Informs if the error is fatal.
static RString errorDescription (ApiError::APIErrorEnum error, HgLanguage::Language = HgLanguage::Default)	Returns the error description of error in the requested language.

Harmony Session Manager API

The following functions may be called by the API tools. The session manager will respond in the way described below.

int update(const RString& object)	Updates the displayed structure surrounding the Hyperwave object in the collection browser and the local map.						
int updateConfiguration()	Forces the session manager and the viewers to re-read the their configuration files.						
int setSourceAnchor(const RString& anchordesc)							
int setDestinationAnchor(const RString& anchordesc)	<p>Activated the Link Creation Dialog of the session manager and fills in the appropriate information (source or destination information of the hyperlink).</p> <p>Achordesc should be a Hyperwave object with the additional fields</p> <table border="1"> <tr> <td>Position</td> <td>The position “entire document” defines the document itself</td> </tr> <tr> <td>SelectionText</td> <td>The text within the selection</td> </tr> <tr> <td>LinkType</td> <td></td> </tr> </table>	Position	The position “entire document” defines the document itself	SelectionText	The text within the selection	LinkType	
Position	The position “entire document” defines the document itself						
SelectionText	The text within the selection						
LinkType							

int getLinkCreationSettings(RString& source, RString& dest)	Gets the information contained in the Link Creation Dialog. The fields in source and dest are	
	DocumentID	The id of the document which contains the anchor
	Title	When a title is given to the anchor this field is present. It contains the language string.
	LinkType	The linktype (for referential links the field is empty)
	Position	The position field
setSearchSettings(const RString& searchsettings)	Set the fields in the search browser. searchdesc contains the following fields. An absent field leaves the old value.	
	Text	Searchtext
	Author	
	ModifiedAfter	
	ModifiedBefore	
	LocalServer	[true false]
	SelectedCollection	[true false]
	Title	[true false]
	Keyword	[true false]
	Content	[true false]
	Extended	[true false]
	ActiveCollections	[true false]
	Result	[true false]
	Modified	[after before between]
	SearchImmediately	[true false]
LanguagePrefs	Is a list of languages (en: ...) separated by “ “	
int getCurrentObjects(RString& objects)	Returns the currently selected objects (selected in the collection browser. Objects are separated by “\n\n”.	
int getCurrentObjectsWithParents(int depth, RString& objects)	Fetches the selected objects together with their parents up to a specified depth. Objects are separated by “\n\n”. The groups object and parents are separated by “\n\n\n”.	
int getDestObject(const RString& srcobject, RString& destobject)	Fetches the destination object for a hyperlink. The srcobject should have at least the “GOid=0x... 0x...” field.	
int getDestAnchor(const RString& srcobject, RString& destobject)	Fetches the destination object for a source anchor. The srcobject should have at least the “GOid=0x... 0x...” field.	

int showObject(const RString& object, const RString& position)	Displays the specified object in the appropriate viewer. An optional position parameter forces the viewer to show the requested part of the document. The object should have at least the “GOid=0x... 0x...” field.
int showHelp(const RString& name)	Displays the specified object in the appropriate viewer. Name is the name of the object to be displayed (without Name=)
int setNotifyMask(ToolAPI* const Bit-Set&)	Informs the session manager about events a tool is interested in. (on these events the member function notify() of the ToolAPIServer/Client classpair is called)
int lockOperation(ToolApi*, const Bit-Set&)	Locks session manager API functions according to a given mask. Locked functions can only be used by the tool which activated the lock. Requests from other tools for these functions are rejected.
int getAuthenticationRecor(RString& identRec)	Retrieves the information necessary to identify a user at a Hyperwave server. identrec contains the two fields
	User The user
	Password The password is in readable form
int getServerRecord(RString& server-Rec)	Retrieves all information necessary to establish a connection to the Hyperwave server the session manager is talking to. serverRec contains the two fields
	Host The Hyperwave host
	Port The Hyperwave port
int newDoc(ToolApi*, const RString& objrep, APIEditorType::APIEditorTypeEnum typ, long refno&)	This function initiates an edit process in the session manager. The returned number refno is used for coming function calls to identify the edit process.
void cancelEdit(long refno)	This function is called when the tool wants to cancel the edit process with the number refno.
void terminted(ToolAPI*)	This function is called when the tool terminated.
APIError::APIErrorEnum error()	Returns the error state.
static RString errorDescription(APIError::ApiErrorEnum error, HgLanguage::Language language= HgLanguage::Default)	Returns the error description of the error in the requested language.

Appendix E. Rules for Document Selection in an AlternativeCluster

An AlternativeCluster is a collection with the attribute:

```
CollectionType=AlternativeCluster
```

When the user accesses an AlternativeCluster, one of its members is chosen for display by the specified criteria. In the Attribute PrefMimeTypes of the user's user record, a list of preferred MimeTypes with corresponding qualities can be specified.

```
PrefMimeTypes=  
  MimeTypes[";Quality=number"]*(,MimeTypes[";Quality=number"])
```

For example:

- PrefMimeTypes=text, image, application/postscript
This means that the user prefers documents of type text to images, and images to postscript, and postscript to any other type.
- PrefMimeTypes=image;Quality=50
This means that images are preferred to any other type and if more than one image is available the one with the quality nearest to 50.

Rules for choosing the document

The following rules are applied in turn.

1. If the AlternativeCluster has a MimeTypes set then only documents of this MimeTypes are taken into account.
2. These documents are compared to the PrefMimeTypes from left to right. Documents with the best matching MimeTypes are kept for further processing.
3. From these documents (all with the same MimeTypes) the ones with the smallest difference in quality compared to the quality given with the matching MimeTypes in the PrefMimeTypes field are selected.
4. If the resulting set contains more than one document, the one with the best matching language is chosen.
5. AlternativeClusters can be nested.

Assigning quality to an object

The quality of a document is represented by the attribute:

```
Quality=number ; 1 is lowest, 100 highest quality
```


Documents without a Quality attribute have a default quality of 100.

Examples

1. Example of an AlternativeCluster (with documents of the same MimeType)

```
AlternativeCluster a1 (MimeType=image)
|
|-Image i1 (Quality=100)
|-Image i2 (Quality= 50)
|-Image i3 (Quality= 25)
|-Image i4 (Quality=  1)
|-Text  t1 (Quality= 12)
```

In this example, text t1 will never be picked because of the basic MimeType of Rule 1. Depending on the quality the user has specified for MimeType image, a different image will be chosen for display. For instance:

```
PrefMimeTypes=*/*;Quality=45    # image i2 would be chosen
PrefMimeTypes=*/*                # image i1 would be chosen
(assuming Quality=100)
PrefMimeTypes=text;Quality=30,image;Quality=5 # image i4
would be chosen
```

2. Example of an AlternativeCluster (different MimeTypes)

```
AlternativeCluster a2
|
|-Image i1 (Quality=100)
|-Image i2 (Quality= 50)
|-Text  t1 (Quality=100)
|-Text  t2 (Quality= 50)
```

With the following user settings:

```
PrefMimeTypes=text,image          # t1 is chosen
PrefMimeTypes=image;Quality=35,text;Quality=100 # i2 is chosen
```

3. Example of an AlternativeCluster (different languages)

```
AlternativeCluster a3
|
|-Text t1e (Quality=100, Language=english)
|-Text t1g (Quality=100, Language=german)
|-Text t2e (Quality= 50, Language=english)
|-Text t2g (Quality= 50, Language=german)
```

With the following user settings:

```
PrefMimeType=text;Quality=90    # t1e is chosen for english  
t1g for german
```

Appendix F. The Harmony Icons

In all Harmony browsers which display a lists of documents, the visualisation is done with a document icon and the document title. The icon indicates the type of the document. When the object is an anchor, the icon indicates the type of the destination document. To visualise the anchor, a green arrow is added to icon. When the document has been visited earlier in a session, a red checkmark is additionally added. Table 1 shows the icons used in Harmony.

Object Type	Icon	Icon state seen	Anchor to	Anchor to, seen
Text				
Image				
Film				
Postscript				
Audio				
Scene				
CGI				
Generic				
Telnet				
Collection				
Collection Open				
Sequence				
Sequence Open				
Cluster				
Multi Cluster				
Alternative Cluster				
Search				

Table 1. Harmony Icons