Usability Evaluation of an Instrumented Version of the Harmony Internet Browser

Ingmar Egger

Usability Evaluation of an Instrumented Version of the Harmony Internet Browser

Master's Thesis

at

Graz University of Technology

submitted by

Ingmar Egger

Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

November 1997

Advisor:	o.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer
Supervisor:	Univ.Ass. Dr. Keith Andrews

Evaluierung der Benutzeroberfläche einer instrumentierten Version der Internetanwendung Harmony

Diplomarbeit

an der

Technischen Universität Graz

vorgelegt von

Ingmar Egger

Institut für Informationsverarbeitung und Computergestützte neue Medien (IICM),
Technische Universität Graz
A-8010 Graz

November 1997

Diese Arbeit ist in englischer Sprache verfaßt.

Begutachter:	o.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer
Betreuer:	Univ.Ass. Dr. Keith Andrews

Abstract

Developing easy-to-use software is one of the most difficult aspects in the software engineering process. This thesis uses a technique called software instrumentation to perform usability tests with real users. In this thesis the implementation steps which were used to produce an instrumented version of Harmony, the native Unix client of the Hyperwave web server, are shown.

This version of Harmony was tested in a field test and the whole testing phase including the data transmission process is also described. Special attention is paid to the analysis of the collected data and general trends in the working behavior, particularly the navigation habits, of the test users are presented. Critical aspects of this usability testing method conclude the thesis.

Kurzfassung

Die Entwicklung von einfach zu bedienender Software ist einer der schwierigsten Aspekte in der Softwareentwicklung. Diese Diplomarbeit beschreibt die Anwendung einer unter der Bezeichung Software Instrumentierung bekannten Methode zur Durchführung von Feldversuchen mit Testpersonen. In dieser Diplomarbeit werden die Schritte, die notwendig sind um eine instrumentierte Version von Harmony zu erstellen, beschrieben.

Diese Version von Harmony, der für Hyperwave entwickelten UNIX-Anwendung, wurde in einem Feldversuch getestet und die gesamte Testphase inklusive der Datenübertragung wird in dieser Diplomarbeit beschrieben. Besondere Aufmerksamkeit wird der Auswertung und Analyse der gesammelten Testdaten gewidmet und allgemeine Trends im Arbeitsverhalten, insbesondere im Navigationsverhalten, der Testpersonen werden aufgezeigt. Eine Betrachtung kritischer Aspekte dieser Testmethode beschließt diese Diplomarbeit.

I hereby certify that the work presented in this thesis is my own and that work performed by others is appropriately cited.

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

Acknowledgements

I am indebted to my colleagues at the IICM and IHM (the Institute for Hypermedia Systems of JOANNEUM RESEARCH), Prof. Hermann Maurer and Keith Andrews who have provided valuable help and feedback. Special mention and thanks go to: Bernhard Marschall, Michael Pichler, Josef Moser, and Hubertus Rothermann.

Last but not least, without the understanding of my family this thesis would not have been possible.

Ingmar Egger
Graz, Austria, November 1997

Credits

- Figure 3.2 was drawn by Keith Andrews, and is used with permission.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People have always had the goal to communicate with each other to get and to exchange information about various topics. Several techniques were used to accomplish this and finally the step of introducing computers in the communication process was the most important one.

One of the most significant events in this process was the birth of the Internet, the network of networks. This revolution in the process of communication started about twenty years ago. Today it is possible that computers communicate and exchange data with millions of other computers over the Internet. Due to this every user can require and get almost any information located anywhere on the Internet. One of the greatest problems people on the Internet are confronted with is the huge amount of information which is available over the net. Users have problems getting an overview about the documents of their interest, especially if the information retrieved from the available systems is not up-to-date. The next chapter, Chapter 2, deals with the Internet and presents its most important concepts.

Chapter 3 introduces the concepts of usability engineering. This chapter presents the most important methods and theoretical backgrounds of usability engineering. In this chapter, the method of software instrumentation, which is used in this thesis, is also described. The theoretical background of this usability testing technique is also explained in this chapter.

Chapter 4 discusses the influence of usability engineering in companies. The extent to which the methods of usability engineering, in particular software instrumentation, are used is also shown in this chapter. One case study performed at Microsoft and one case study done at Georgia Institute of Technology, are presented in this chapter. Both studies were chosen because the method of software instrumentation was used.

Hyperwave, a project started at Graz University of Technology, the first second generation hypermedia information system, is designed to cope with this problem of distributed information systems called disorientation or the "Lost in Hyperspace" syndrome. Hyperwave provides several important features, such as search, orientational, and navigational facilities, information structuring by using collections, and it builds up a structure of these collections, called the collection hierarchy. A very important feature is that Hyperwave stores hyperlinks separately from documents and not embedded within them, and that it supports therefore links in all kinds of documents such as MPEG video clips, images, audio, PostScript, and 3D scenes. Hyperwave uses the collection hierarchy to show the user a structural overview of the information space and to grant or deny access rights to parts of the information structure. Hyperwave and its features are described in more detail in Chapter 5.

Harmony is the Unix/X11 native client for Hyperwave. Harmony supports several navigational facilities as well as location feedback. The main part of Harmony is the Session Manager, which controls all activities in the session, communicates with the Hyperwave server, starts viewers to display the required document, and so on. Harmony supports different viewers, since it has to handle several types of documents such as text, image, audio, film, postscript and 3D scenes. All viewers share a consistent graphical user interface (GUI) in order to support the user in using them. Chapter 5 discusses also the concepts of Harmony.

Chapter 6 explains how the method of software instrumentation was implemented for Harmony. This chapter describes the whole development process of the instrumented version of Harmony. Information about various aspects including producing the instrumented version, creating the logfiles, transmitting the logfiles, and the analysis process, is given.

Chapter 7 describes the research experiment. In this chapter the test plan is presented. The testing activities, including the pilot test and the real test, are also described in this chapter. This chapter is concluded with an analysis of the process to obtain test users for the test experiment.

Chapter 8 presents the results of the analysis of the collected data. The goal of the test is to verify several presumptions, particularly in the field of navigational techniques. Among the various topics of this chapter are a detailed overview, a list of most used functions, the searching behavior of the test users, and a short analysis of error situations. The results of the analysis are also presented in graphical form.

Chapter 9 and Chapter 10 conclude the thesis with a summary of the main results and an outlook for future work and research.

Chapter 2

The Web

The Internet, the worldwide network of networks, is involved in a growing process. The Internet supports several different information systems. The Internet was established about 20 years ago and the goal was to connect the ARPAnet (Advanced Research Project Agency) and various radio and satellite networks together. The purpose of the ARPAnet was to build a network that could work even through unforeseen breakdowns of certain nodes during a nuclear attack. The existing communication was only between a source and a destination computer, but the idea was to build a network, in which every computer could talk with any other computer on the whole network. The ARPAnet was extended with services like remote login (telnet), file transfer (ftp), and electronic mail (email). The ARPAnet switched to TCP/IP to connect diverse networks together, and the new created network was called Internet.

TCP/IP - Transmission Control Protocol/Internet Protocol

The Internet is a packet switched network. The Internet Protocol IP defines the addressing and routing mechanism, puts the message data in envelopes known as packets with source and destination address. Internet addresses are unique 32-bit numbers, in four 8-bit parts, e.g.: 129.27.153.18. Names are easier to remember and gratefully the Internet uses a Domain Name System (DNS), which uses a name server to translate domain names into Internet addresses, e.g.: the domain name fiicmal01.tu-graz.ac.at stands for the Internet address 129.27.153.18. If the packets are too large, the Transmission Control Protocol TCP takes the packets, breaks them into pieces, and numbers them. The numbers are useful to control the order of the received packets. If one is missing or corrupted, it is re-transmitted. The TCP data is placed in a TCP envelope which in turn is placed inside a IP envelope. The IP packets are then routed through the Internet until they reached their destination. At the destination the IP and TCP envelopes are unpacked and the original data is reconstructed and checked (are packets out of order, corrupted, or got lost etc.).

2.1 Basic Internet Applications

Since the beginning of the Internet in 1983 four basic Internet applications exist:

- Remote Login (telnet)
  Telnet is used for logging into other computers on the Internet. The user can access all services the remote machine provides. Telnet command: telnet address-of-foreign-host e.g.: `telnet iicm.edu`

- File Transfer Protocol (ftp)
  The ftp is used to move files from one computer to another. There are two different ways to receive files from the foreign computer: ascii for text files and binary for program files. And there also exists two different access modes: anonymous, password can be the whole email address, and identified, account name and password. Ftp command: ftp address-of-the-foreign-ftpserver e.g.: `ftp iicm.edu`

- Electronic Mail (email)
  Email is the most used Internet application, since it is easy to use and much more faster than the traditional surface mail. The mail is sent from one computer to the next until it reaches its destination. Today many mail programs exist, the most popular is the mail protocol SMTP (Simple Mail Transfer Protocol). Mail has some weaknesses like security, which is on a low level and foreign people can read your own private email.

- Network News (news)
  With news you can easily find or get an answer to any kind of question you have. There exist thousands of so-called newsgroups to which you can talk to get the desired information. Newsgroups can be compared with discussion groups or bulletin boards. The newsgroups are hierarchically structured, with the broadest grouping first in the name, followed by an arbitrary number of subgroups. The USENET is a set of voluntary rules for passing and maintaining newsgroups. There exist seven major news categories: comp (computer science), news (network news), rec (hobbies, recreational activities, arts), sci (scientific research), soc (social issues), talk (debating different topics), and misc (all other topics).

2.2 First Generation Information Systems

As mentioned at the beginning the Internet consists of different information system but they all have not enough functionality to provide the power to exploit the large information and communication resources available on the Internet. Each of the existing information systems has its own special features but also weaknesses.

2.2.1 Archie

On the Internet many ftp-servers exist and for users it is difficult to find the right one to get the information they are looking for. Archie, developed at McGill University in Canada, is probably the first and most famous dictionary server to help locating files by name. Today more than 15 archie servers exist and they index over 1500 FTP servers worldwide. The content of each archie server is updated monthly.

2.2.2 WAIS

WAIS (Wide Area Information Server), developed in 1989 by Thinking Machines, Apple Computer, and Dow Jones supports a powerful search engine for full-text searching large, previously indexed databases. WAIS clients send queries to servers, display ranked results, and fetch and display desired documents. WAIS has no functionality for browsing (hyperlinks) or structuring information contents but it can be considered real search engine. An important feature of WAIS is *relevance feedback*: Parts of the search result can be taken as input for a further search and the search can be refined to find certain documents. Many Gopher and WWW servers can use WAIS to provide search functionality.

2.2.3 Gopher

Gopher was developed at the University of Minnesota in 1991 as a campus wide information system and was the first information system that includes browsing and full-text searches. The design philosophy of Gopher was to make it possible for departments and other groups at the University, to publish information on their own desktop systems and to hide the distributed nature of the server from the user using the system.

Gopher structures information as a hierarchy of menus comparable with a file system which is easy to use but has many weaknesses. For the user browsing through the menus is like browsing through a big graphical tree. In reality the user browses through a graph, from one menu the user comes to the next sub-menu and so on, but sub-menus can reside in many different places and a loop can be built. The user has difficulty to come back to previous selected menus and therefore the user can run in troubles and gets lost.

Most Gopher servers support no search facilities themselves but use WAIS as an add-on search engine. They also have no integrate hyperlinking facilities. It is based on the Client/Server Model, servers accept connections from clients and return either documents, collections of documents, references of objects on other servers, or perform searches of document collections.

2.2.4 WWW

The World-Wide-Web (WWW, W3 or “The Web”) was developed at CERN in Geneva in 1989 as an information system for the particle physis community[And96]. WWW was created by Tim Berners-Lee and Robert Cailliau as a hypertext information system. It is now a distributed hypermedia information retrieval system which provides access to a large universe of documents. Merging the techniques of hypertext (the user clicks with the mouse), information retrieval, and wide-area networking produces the WWW model.

Hypertext is not a linear text (one page after another), it consists of pieces of text (called nodes or documents), which are connected by links. Hypertext forms a information space of documents and links, following the links is like navigating through this information space. A link is displayed as a hot spot in the document, by clicking on it the user comes to the destination of the link, which can be part of a document or the document itself. Hypermedia is multimedia with hyperlinks. Documents may contain text, images, audio, film and 3D-scenes, etc. The WWW consists of three basic concepts:

- URL - Universal Resource Locator
  A URL is used to specify the location of a resource available electronically. URLs link WWW pages together and have a specific format:
  `protocol://domain:port/selectorstring`.
  The first element of the URL, protocol, is a name scheme referring to various Internet protocols. Such name schemes are:
  - `http`
  - `hyperg`
  - `gopher`
  - `mailto`
  - `ftp`
  - `news`
  - `wais`
- HTTP - Hypertext Transfer Protocol
  HTTP is an ASCII protocol atop TCP/IP and implements a fast and flexible mechanism for following references between units of distributed information. The name is often misinterpreted, it does not only transfer hypertext, also other kind of information e.g.: audio.
- HTML - Hypertext Markup Language
  HTML is the most important in WWW, it is an SGML (Structured Generalized Markup Language) encoded text format for WWW and is used to create hypertext documents with a logical structure. With HTML WWW embeds links and inline graphics in documents.

WWW structures its information in documents and links them together. Hierarchies are presented as lists of hyperlinks, and it is a distributed hypermedia system i.e. documents can comprise text, image, audio and film. WWW uses HTML to embed links within text documents, but it supports no links from other kinds of documents e.g.: links in film clips.

WWW servers have built in the *common gateway interface* technology which allows them to start application programs to produce dynamic information. However, the flexibility provided by CGI leads to various different interfaces and to a loss of unity. The problem of finding resources on the Internet became greater due to exponential growing rates of the WWW. Programs like Webcrawler, Yahoo, Lycos, Infoseek or Excite, which build centralized indexes of the content of sites and manually maintained catalogs of sites, have improved the situation somewhat.

In June 1995 a 3D file format for modeling objects and worlds on the Web was specified. This format, called the `Virtual Reality Modeling Language (VRML)` is rapidly extended.

Sun Microsystems developed a new secure object oriented programming language and introduced it under the name `JAVA`. Java executes on a virtual machine. The introduction of Java brought interactivity to the Web, in the form of downloadable miniapplications or *applets*. Before Java was introduced CGI scripts used in forms were the only possibility for some kind of interaction. Java is a secure language, since it is possible to access physical mediums like floppydisks, harddisks or printers using a JAVA applet, only with restrictions (Security Manager). Microsoft introduced `ActiveX` which is also an object oriented programming language. ActiveX can be used to develop Web controls, but it is not secure, since it is possible to access physical mediums and to start applications like Excel.

WWW is a great step forward but except hyperlinks, it has no structural facilities and links are not bidirectional. The user cannot find out which other documents point to a particular document. This is a limitation of WWW, because in using one-way links so called *dangling links* can occur when a document is deleted or moved. Like Gopher, WWW servers generally have no integrated search facilities but can use WAIS for searching. Altogether, WWW should be considered a *first generation* networked hypermedia system.

2.3 Second Generation Information Systems

Second generation networked hypermedia information systems attempt to improve some of the previously mentioned weaknesses. The greatest problem is obviously that of disorientation, also known as the "Lost in Hyperspace" syndrome. When the user browses through the information space many questions arise:

- Where am I ?
  The user sees only one current document at a time. The document can contain hyperlinks and for users it is difficult to gain an overview. They do not know which links from other documents point to the current document (incoming links) and to which other documents the hyperlinks in the current document point to (outgoing links).

  In the existing systems like WWW to look backwards or using a map is not possible. Hyperwave uses a local map to show the relation of incoming and outgoing links of the current document to other documents.

- How much information can I see about a certain topic ?
  It is difficult for a user to see how many documents and megabytes exist about a certain topic.

- How much of the information have I already seen or is left ?
  Without so called footprints or checkmarks the problem of seeing the same document again and again can occur.

- Is this information the newest one ?
  If users found information about a certain topic, they cannot be sure that the found information is really the newest one.

Hyperwave, a *second generation information system*, was designed to present answers regarding these questions. Hyperwave is described in more detail in Chapter 5.

Chapter 3

Usability Engineering

3.1 What is Usability ?

This chapter is based on Jakob Nielsen's excellent book[Nie93]. Nielsen[Nie93] gives a very practical approach to usability engineering. As computer vendors started dealing with the user's needs, the most frequently used term was *user friendly* to describe systems. It shows however that this term was not appropriate, due to several reasons. First, users do not need machines, which are *friendly* to them, but they need machines which support, them in their daily business. Second, a system, which is *friendly* to one user may feel very tedious to another.

Usability itself is a narrow concern compared to the larger issue of system acceptability. As shown in Figure 3.1, the overall acceptability of a computer system is a combination of its social acceptability and its practical acceptability. Social acceptability depends very much on the single user and might vary from one user to another. Practical acceptability describes various categories including cost, support, reliability, compatibility with existing systems and usefulness. Usefulness is the issue of whether the system can be used to achieve a desired goal. Usefulness can be split further into utility and usability. Utility is the question if the functionality of the system in principle can do what is needed, whereas usability is the question of how well users can use that functionality.

Usability applies to all aspects of a system with which humans might interact. Nearly every computer feature has user interface components, even a data transfer facility between two computers will normally include an interface to trouble-shoot the link when something goes wrong.

3.1.1 What Does Usability Affect ?

As defined in Dumas/Redish[DR93] usability is an attribute of the entire package that makes up a product - the hardware, software, menus, icons, messages, manuals, quick reference, online help and training. Due to changes in technology it is useful to consider these product pieces as a whole and not as separate entities. Regarding this concept products now

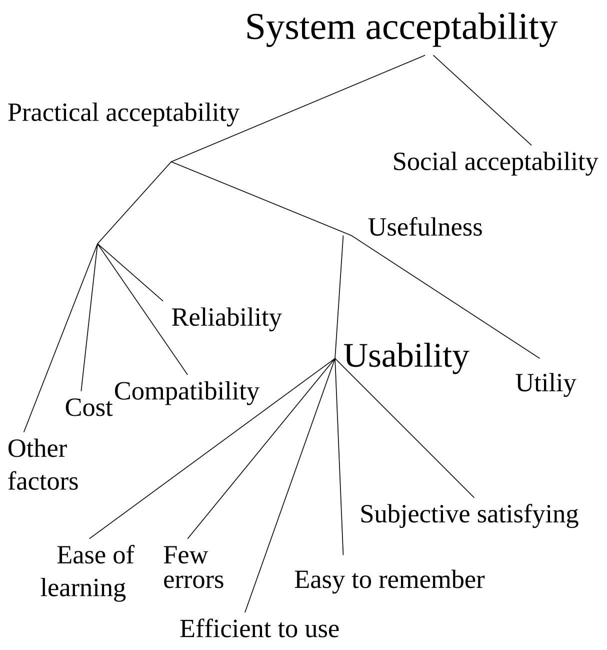


Figure 3.1: Model of system attributes.

communicate with users in many ways. The system communicates with the users through the interface icons, menus, messages, prompts and the way these elements are organized, as well as print manuals, tutorials and online help. So to comply with the usability concept all parts of the product as described above must be consistent and work together to be usable.

3.2 Definitions of Usability

Usability means that people who use a product can do so quickly and easily to accomplish their own tasks. One has to maintain the following points to achieve this approach:

- Usability means focusing on users
  - To develop a usable product, you have to know, understand, and work with people who represent potential users of the product. No one can substitute them.
- People use products to be productive
  - To develop a usable product, you have to understand users' performance goals and to know the users' jobs and tasks.
- Performance goals are:
  - Time it takes to do what the user want

  - Number of steps the user have to go through
  - Success the user have in predicting the right action to take

- Users are busy people trying to accomplish tasks
  - People connect usability with productivity, because no one gets paid for time spent just sitting at a computer and therefore people's tolerance for time spent in learning and using hardware and software tools is very low.

- Users decide when a product is easy to use
  - Users, not designers or developers, determine when a product is easy to use.

To develop usable products, you have to understand how much time and effort typical users are willing to spend figuring out how to do a task with the product. It is very important that one realizes that usability is not a single, one-dimensional property of a user interface. Even that usability has multiple components it is associated with the five usability attributes listed and discussed below. These five usability attributes are defined to make usability measurable.

Usability is typically measured by having a number of test users, selected as representative to the intended user population as possible, use the system to perform a specified set of tasks, though it can also be measured by having real users in the field perform tasks they are doing anyway.

- Learnability

  Learnability is the most fundamental usability attribute, since most systems need to be easy to learn, and since the first experience most people have with a new system is that of learning to use it. Therefore the system should be easy to learn so that the user can start getting some work done with the system really soon.
  Nielsen[Nie93] uses a graph to visualize learnability, this graph shows a curve, which is called learning curve.
  As shown in Figure 3.2, the learnability curve represents a continuous series of improved users performance and not a *learned/notlearned* distinction. Systems that are easy to learn have a steep incline for the first part of the learning curve and allow users to reach a reasonable level of usage proficiency within a short time. Generally all user interfaces have learning curves that start out with the user being able to do nothing. Exceptions are so called walk-up-and-use systems. A disadvantage of the learning curve approach is probably, that it does not apply to cases where users are transferring skills from previous systems.
  Learnability can be measured very easy. To measure learnability one has to get a few users, who have not used the system before, ask them to perform some tasks and to record the time it takes these users to reach a specified level of proficiency in using the system. However, these users should be representative of the intended users of

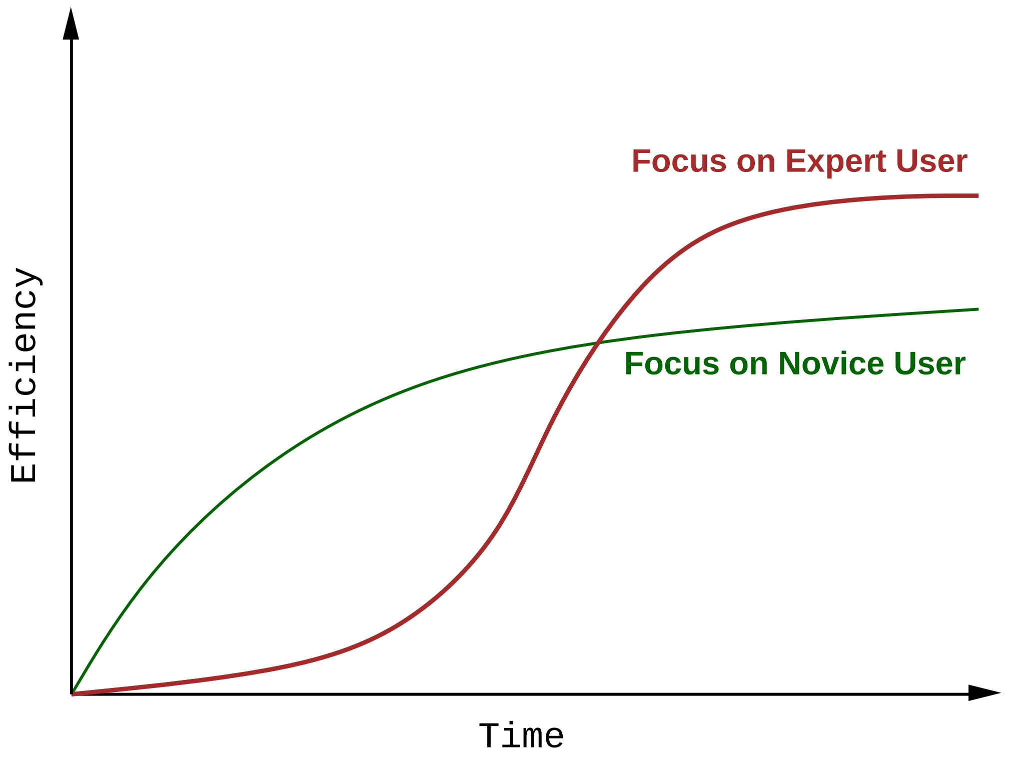


Figure 3.2: Learning curve of a hypothetical system.

the system.

There are several ways to express the specified level of proficiency

- Successful task completion
  A user has reached the defined level if he or she is able to complete a certain task successfully.
- Completing a set of tasks
  The users need to be able to complete a set of tasks in a certain, minimum time before one considers them as having "learned" the system.

- Efficiency
  Efficiency means, that the user is capable of reaching a high level of productivity, once he or she has learned using the system. Depending on the tasks the user wants to perform, it could be possible that it is not necessary for the user to gain expert level performance in a relatively short time. However, the complexity of some systems is so high, that it takes more than a few hours to reach such expert level performance. A typical way to measure efficiency is to get a set of users, with some kind of experience in using the system, and to measure the time it takes these users to perform some typical tasks. The experience these users should have can be defined through the time they spent using the system.

- Memorability
  A user, who was not using a system, which he or she already had used and learned, for a longer period of time, should be able to start working with the system without having to learn it from the start. A tool to create annual reports can be such a system. Since it will not be used much often from the user it is necessary that the user interface of the tool is easy to remember.
  Nielsen[Nie93] mentions two ways to measure interface memorability.

  - A standard user test with casual users
    In this method a standard user test is performed with casual users, who have been away for an amount of time from the system, and the time they need to perform some typical test tasks is measured.
  - A memory test
    In this method a memory test is conducted with users after they finish a test session with the system and they are asked to explain the effect of various commands or to name the command, draw the icon, that does a certain thing. The score for memorability is the number of correct answers from the users.

  It came out that the first method is more representative, even if the memory test is easier to carry out.

- Errors
  The term error can be defined as any action that does not lead to the desired result. Errors can be divided into

  - Catastrophic Errors
    These are errors which often destroy the work of the user. Such errors **must not** occur, since it is very difficult to recover from.
  - Minor Errors
    These errors can be corrected immediately by the user and only slow down the transaction time of the user.

  A method to measure the error rate of a system is to count the number of such actions. This counting is done while the user is using the system for a specified task.The goal of each computer system should be to minimize the number of errors, that a user can make when he or she is using the system.

- Satisfaction
  An important goal for each system is that the users should be satisfied when they are using it. In the area of entertainment, like computing games, the usability attribute satisfaction is a very important aspect, since the user spends lots of time using these applications.

  There are some methods to measure subjective satisfaction

  - Psychological measurements
    This approach uses EEG, but it is inappropriate for usability engineering, since the users are often too nervous.

- Interviews
  In this method test users are asked for their subjective satisfaction. For analysis purposes it is necessary to average the replies of multiple users. To ensure consistency written questionnaires are given to the users.
- Questionnaires
  Nielsen[Nie93] shows two types of subjective satisfaction questionnaires. These questionnaires are typically very short. One possibility is to use Likert scales, the other is to use semantic differential scales. For a Likert scale, the questionnaire postulates some statement (e.g. "It was very easy to learn how to use this system") and asks the user to rate their degree of agreement with the statement. Typically 1-5 or 1-7 rating scales are used. If you use a 1-5 rating scale in this case, the reply options will be typically

  1 = strongly disagree
  2 = partly disagree
  3 = neither agree nor disagree
  4 = partly agree
  5 = strongly agree

  A semantic differential scale lists two opposite terms along some dimension (e.g.very easy to learn vs. very hard to learn) and asks the user to place the system on the most appropriate rating along the dimension. It is normally best to keep the questionnaire short in order to maximize the response rate. When rating scales are used, one needs an anchor or baseline to calibrate the scale before it is possible to assess the results.
- Collect data after release
  Data that is gathered through measuring the extend that users choose to use the system over any available alternative systems.

Categories of Users

The tasks of the users and their individual characteristics and differences are two of the most important usability issues.

Nielsen [Nie93] shows a cube model as a way to visualize different experiences of users. As shown in Figure 3.3, this model is a three dimensional model consisting of three main dimensions along which the experience of the users difference. These are:

- Experience with the system
- Experience with computers in general
- Experience with the task domain

Users are normally considers to be either novices or experts or somewhere in-between. The transition from novice to expert user of a system often follows a learning curve.

Almost all systems of some complexity have so many features and so many uses that any given user only makes extensive use of a small subset. Thus, even an "expert" user may be quite novice with respect to many parts of the system not normally used by that user. Therefore expert users still need access to help systems for those parts of the interface that they do not use often, and they will benefit from increased learnability of these features.

However, one can find some other factors to distinguished users such as

- Age
- Gender
- Differences in spatial memory and reasoning abilities
- Preferred learning style

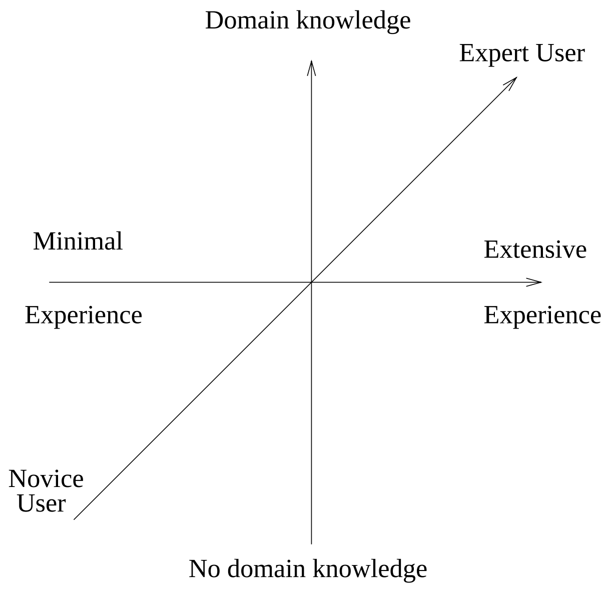


Figure 3.3: Three dimensional cube model.

The main goal of each system designer should be to design the user interface so that it is usable for as many people as possible. Therefore, systems which claim to be usable provide two sets of menus, one for novice users and one for expert users. Other concepts are to provide shortcuts and the usage of a terminology that is understandable not only for domain specialists.

3.3 The Usability Engineering Lifecycle

The most important aspect in considering usability engineering is not to see it as an action to fix up the interface which takes place before the release of a product. Rather, usability engineering should be considered as a set of activities that take place during the development of the product and are ideally started in the early stages before the user interface has even been designed. If there are plans for future releases of a product it is therefore important to follow up the release of the product with field studies of its actual use. The lifecycle model means that one should not start as soon as possible with the design, but to do usability activities just before design is started. The main advantage of such an early begin is that it will not be necessary to change the design to comply with usability recommendations. Usability work that is done before starting to design may make it possible to avoid developing features that are not necessary. Several of these activities like market research or product planning process may even be performed by marketing groups. The lifecycle model will be described in more detail now.

Nielsen[Nie93] suggests the following list of usability activities.

1. **Know the User**
   The most important step in the usability process is to study the intended users and use of a product. The principle "know the user", which is the most basic of all usability guidelines, is often difficult for the developers to follow, since they have sometimes problems to get access to users. Grudin[Gru90][Gru91a][Gru91b] shows a few obstacles to such access, including

   - Nearly every development company wants to protect its developers from being known to customers, since customers may bypass established technical support organizations, and call developers directly, sidetracking them from their main job.
   - Sales representatives often fear to let anybody else from the company talk to "their" customers, fearing that the developers or usability people may offend the customer or create dissatisfaction with the current generation of products.
   - User organizations only making users available for a short time, either because they are highly paid executives or because they are unionized and dislike being studied.

   Since there exist no easy answers how to get access to users, much time is wasted by arguing over what users might be like or what they may want to do. It is necessary to know the class of people who will be using the system. Sometimes it is possible to identify individual users through

   - Work experience
   - Educational level
   - Age
   - Previous computer experience
   - Reading and language skills
   - Work environment
   - Social context

2. **Task Analysis**
   It is very important to identify the tasks users perform and even want to perform with the system.
   A practical way to get this information is to interview users. Information can also be gathered through viewing concrete examples of the work products of the users. It is often very useful to observe users doing real tasks. In this case information can be collected by encouraging the user to explain what he or she is doing. This approach is similar to the thinking aloud method.
   Nielsen[Nie93] defines the result of a task analysis is a list consisting of

   - All the things users want to perform with the system
   - All the information they will need to achieve these goals
   - The steps that need to be performed and the interdependencies between these steps
   - All the various outcomes and reports that need to be produced
   - The criteria used to determine the quality and acceptability of these results
   - The communication needs of the users as they exchange information with others while performing the task or preparing to do so

3. **Functional Analysis**
   It is also important to analyze what is it that really needs to be done, and what are procedures which can, and perhaps should, be changed[Sch88].
   For example, initial observations of people reading printed manuals could show them frequently turning pages to move through the document. A naive implementation of online documentation could take this observation to indicate the need for faster paging or scrolling mechanisms. A functional analysis would show that the cause of this behavior might be that users want to find specific information, but they have a hard time in locating the correct pages [ERG$^+$89].
   However, it is a good idea to coordinate functional analysis with task analysis.

4. **Evolution of the User**

   Using a system changes the users and as they change they will use the system in new ways. Carroll and Rosson [CR91] named this phenomenon as the *coevolution of tasks and artifacts*.
   A typical change is that users become experts after some time and they want interaction shortcuts.
   It is therefore really important not to design a system just for the way users will use the system in the first short period after its release.

5. **Competitive Analysis**
   Sometimes it is desirable to analyze existing products heuristically according to established usability guidelines and to perform empirical user tests with these products. Since a competing product is already fully implemented and can be tested very easily. The result of such a process is a list of guidelines for the new product. The main advantage is that testing of fully implemented competitive products reveals guidelines that should be avoided as well as those that should be used.

6. **Setting Usability Goals**
After a task analysis is done usability goals have to be set. A proper approach is to specify several different levels of performance for each usability attribute [WBH88].
One practical way to do this is using a usability goal line, which represents the range of specification levels for one usability goal. Such a goal line is shown in Figure 3.4. A big advantage of such a goal line is that the interpretation is very simple.

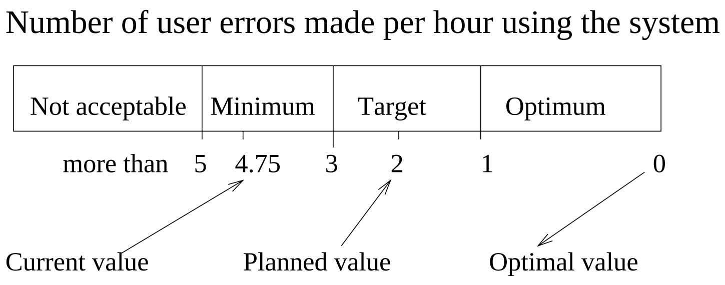


Figure 3.4: An example of a usability goal line.

For new products it can be hard to specify usability goals, as for existing products, since in this case usability goals are defined to reach an improvement of the new product in contrast to the existing one.
A possible approach for a new product could be to define a set of sample tasks and ask several usability specialists about their opinion how it will take users to perform these tasks.

7. **Iterative Design**
After an interface is implemented it will usually be evaluated. This will often lead to design revisions. However, such a revision may even introduce new usability problems and this is a reason for combining iterative design and evaluation.
It is not always a good idea to concentrate on only one usability parameter to improve, since this can have bad influence on the other parameters.
The best concept is probably to evaluate design ideas by trying them out in concrete designs and then to perform heuristic analysis. It is really important to recognize that this process has to be repeated in an iterative way as soon usability problems are encountered and understood.

8. **Parallel Design**
In this method several designers make different designs and the goal is to explore different design alternatives before a single approach is selected that will be developed in further detail and subjected to more detailed usability activities. The parallel design concept is depicted in Figure 3.5. Typically three to four designers are involved in such a parallel design process.
It is very important that the designers work independently and that they did not discuss their designs with each other until their interface designs are completed.

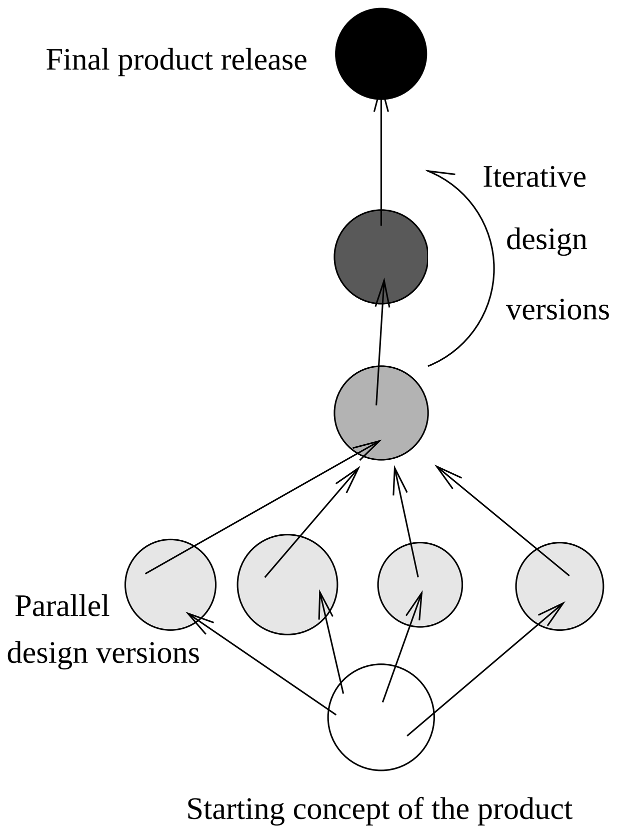


Figure 3.5: Parallel Design Concept.

The parallel design method is not cost-extensive, since most of the design ideas are not being implemented and it is a cheap way to explore several design alternatives. There is financial benefit from using the parallel design method, since several design approaches are explored at the same time and the time-to-market for the product decreases.

9. **Participatory Design**
   In this method several users are involved in the design process through regular meetings between the design team and the users. These users are called subject matter experts or SMEs.
   One important aspect is that users are not designers. However, it is essential to recognize that participatory design is not simply asking users what they need or what they want, since they often do not exactly know it.
   Nielsen[Nie93] suggests that these concrete designs should be presented to the user in form of prototypes, if available or paper mock-ups or even a few screen designs. In larger projects there is a need of refreshing the pool of users, since after some meetings these users did not represent real users anymore, they tend to "think like designers".

10. **Coordinated Design of the Total Interface**

All elements including the total user interface, the documentation, the online documentation and tutorial videos as well should be consistent.
Consistency is not just measured at one single point, but it has influence on successive product releases and even on total product families.
To achieve consistency of the total interface it is necessary to have some centralized authority for each developed product. Interface standards are even an important approach to achieve consistency.

11. **Apply Guidelines and Heuristic Analysis**
Guidelines are a list of principles for user interface design. There are several different kinds of guidelines:

    - General guidelines applicable to all user interfaces
    Such guidelines inform the user about the system state and actions
    - Category-specific guidelines for the kind of system being developed
    These guidelines refine general guidelines and are more specific.
    - Product-specific guidelines for the individual product
    Such guidelines refer to icons,icon shapes or object shapes, which are typical for the product.

    There exist standards as well and the difference between standards and guidelines is that a standard specifies how the interface should appear to the users and guidelines provides advice about the usability characteristics of the interface.

12. **Prototyping**
Prototyping means to develop something that can be tested with real users. The main advantage is that the time and costs can be saved with prototyping. Nielsen[Nie93] shows two techniques used for prototyping which are called **vertical prototyping** and **horizontal prototyping**.

    - **Vertical Prototyping**, which means that the number of features is reduced and the result is a narrow system that includes in-depth functionality, but only for a few selected features.
    - **Horizontal Prototyping**, which means that the level of functionality is reduced and the result is a surface layer that includes the entire user interface to a full-featured system but with no underlying functionality.

    These two techniques are shown in Figure 3.6. Prototypes can be used for a special form of participatory design called *interactive prototyping*, where the prototype is modified on the fly as the test user comments its weak spots.

    Another form of prototypes are scenarios which are produced by reducing the level of functionality and the number of features as well. However, these scenarios are not very realistic, because the user cannot move through the system freely.

Nielsen[Nie93] refers to a scenario as a description of an individual user using a specific set of computer facilities to achieve a specific outcome under specified circumstances over a certain time interval.

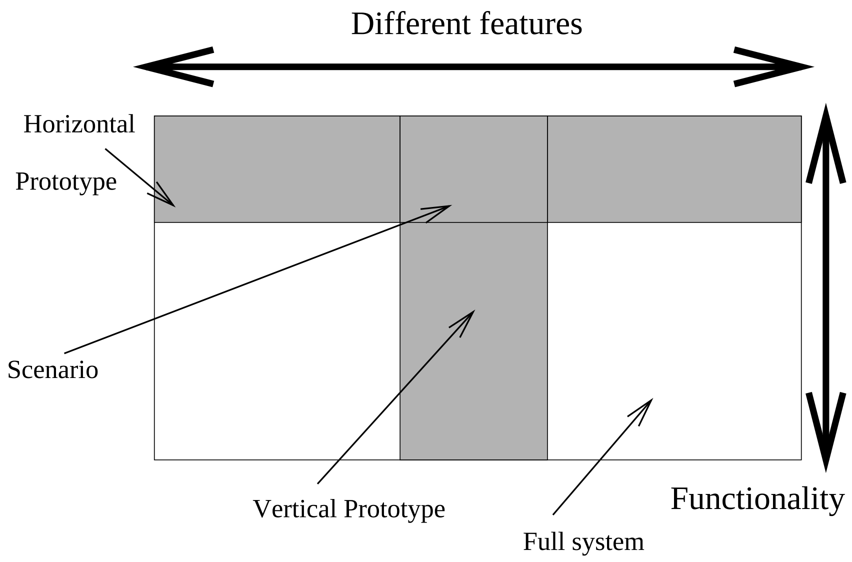


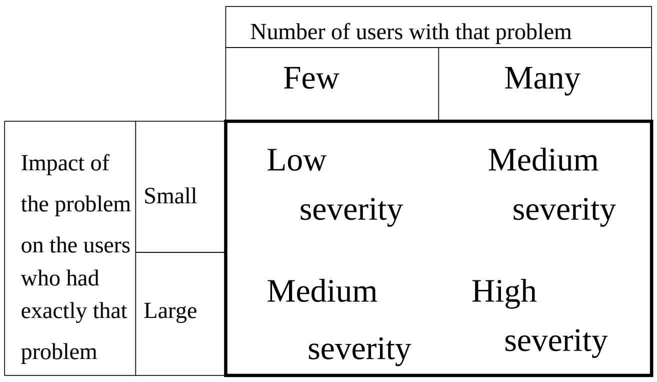
Figure 3.6: The two dimensions of Prototyping.

There a few techniques which can be used to produce prototypes and it will be a good idea to combine several of the techniques listed below to develop several prototypes:

- Place less emphasis on the efficiency of the implementation
- Accept less reliable or poorer quality code
- Use simplified algorithms that cannot handle all the special cases that normally require large programming effort to get right.
- Use a human expert operating behind the scenes to take over certain computer operations that would be too difficult to program. These technique is called The Wizard of Oz.
- Use a different computer system than the eventual target platform.
- Use low-fidelity media that are not as elaborate as the final interface but still represent the essential nature of the interaction.
- Use fake data and other content
- Use paper mock-ups instead of a running computer system. These paper mock-ups have the advantage that they can be shown to larger groups on overhead projectors and used in conditions where computers may not be available, such as in conference rooms.
- Rely on a completely imaginary prototype where the experimenter describes a possible interface to the user orally. These technique is called *Forward Scenario Simulation*.

13. **Empirical Testing**
The result of each evaluation is a list of the usability problems in the interface. Since often one does not want to fix all the problems, it is a good idea to give priorities to the problems. A way of quantizing these priorities is to collect severity ratings. Nielsen[Nie93] shows a proper approach to *severity rating*

0 = this is not a usability problem at all
1 = cosmetic problem only
– *need not to be fixed unless extra time is available on project*
2 = minor usability problem
– *fixing this should be given low priority*
3 = major usability problem
– *important to fix, should be given high priority*
4 = usability catastrophe
– *imperative to fix this before product can be released*



		Number of users with that problem	
		Few	Many
Impact of the problem on the users who had exactly that problem	Small	Low severity	Medium severity
	Large	Medium severity	High severity

Figure 3.7: Table to estimate the severity of usability problems.

As shown in Figure 3.7, there are some factors which can have influence on severity

- The number of users, who run into that problem
- The extent to which those user are hurt by the problem
- Is the problem only one for the first time or will it persist

14. **Collect Feedback from Field Use**
After the release of a product it is important to collect usability data for the next version and for future products. It is a fact that this data cannot be collected in laboratory studies.

Methods that are used for this kind of studies are

- Interviews

- Questionnaires
- Observal Studies
- Instrumented Software
  Since follow-up studies are concerning with the usability of an existing system, logging data from instrumented versions of the software becomes especially valuable for its ability to indicate how the software is being used across a variety of tasks.

3.4 Usability Heuristics

Usability heuristics are basic characteristics of usable interfaces. These principles could and should be used for a usability inspection of an interface. Nielsen[Nie93] discusses the principles shown below.

- **Simple and Natural Dialogue**
  User interfaces should be as simple as possible. The main goal should be to match the tasks of the user in as natural a way as possible. This means that mapping between computer concepts and user concepts becomes as simple as possible. The result of this effort should be that the navigation through the interface is minimized for the user.

  [Ric91] and [Tra91] set up a few guidelines in using colors in screen design:

  - Do not over-do it
  - Make sure that the interface can be used without colors (color blindness)
  - Try do use color only to categorize, differentiate, and highlight, not to give information, especially quantitative information

  Based on proper task analysis, it is often possible to identify the information that is really important to users. This principle is often referred to as *Less is more*. The *Less is more* rule does not just apply to the information contents of screens but also to the choice of features and interaction mechanisms for a program.

- **Speak the Users' Language**
  The terminology in user interfaces should be based on the language of the users and not on system-oriented terms. Dialogues should as far as possible be in the natural language of the user.

  A good concept to design a user-oriented dialogue is to aim at good mappings between computer display of information and the conceptual model used by the user. But such mappings are not always easy to discover. One way is to perform a task analysis, talk with users, observe them and build up an understanding of the users and their domain.

Some of the commonly used techniques to understand the users' model are:

- Ordered Recall
    - Users are allowed to freely associate and mention as many concepts as they can think of, with concepts that are mentioned close together assumed to be associated in the mind of the users
- Card Sorting
    - Each concept is written on a card and the user sorts the cards into piles
- Paired Similarity Ratings
    - Users are given a questionnaire listing all possible pairs of concepts and asked to rate their similarity [MS88]
- Metaphors
    - Icons, like a thrash can, are used in graphical interfaces for specific operations. However, a problem is that not all metaphors are meaningful to all cultures.

- **Minimize User Memory Load**
  Computers should minimize the need of remembering for the user as much as possible. For people it is much easier to recognize something that is shown to them. It will be much harder for them if they have to recall the same information from memory without help.
  But the computer should not display too much information at once following the principle *Less is more.*

  Nielsen[Nie93] suggests that whenever users are asked to provide input, the system should describe the required format and, if possible,provide an example of legal and sensible input, such as a default value. It is often a good idea to fill the input field itself with a default value (e.g.: the actual time) and to let the user modify this value. To display legal ranges of input values will be also a good idea.

- **Consistency**
  This concept means that if users know that the same command or the same action will always have the same effect, they will feel more confident in using the system. The effect of consistency will be that the users will be encouraged to try out new parts of the system.

  The same information should be presented in the same location on all screens and dialog boxes and it should be formatted in the same way to facilitate recognition. However, consistency is not only a question of screen design, it also includes considerations of the task and functionality structure of the system.

- **Feedback**
  It is essential that the user is continuously informed by the system about what it is doing and how it is interpreting the input of the user. The system should not wait until an error situation has occurred to give feedback, but instead also provide positive or even partial feedback as information becomes available.

  Normally the response times of the system should be as fast as possible. [Mil68] and [CRM91] found that the basic response times are:

  – 0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except displaying the result.

  – 1.0 second is about the limit for the user's flow of thought to stay uninterrupted,even though the user will notice the delay, but no special feedback is necessary during delays between 0.1 and 1.0 second.

  – 10 seconds is about the limit for keeping the attention of the user focused on the dialogue. For longer response times the system has to provide feedback, especially if the response times are highly variable.

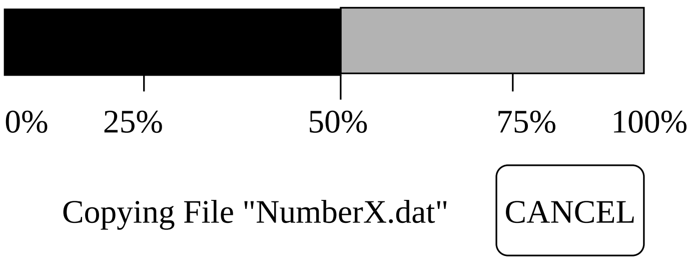


Figure 3.8: Example for a percent-done indicator.

  Mechanisms which are used in case that the response time is greater than 10 seconds are

  – Percent-done progress indicators, as shown in Figure 3.8

  – Graphic progress bars

  – Progress indicators in form of a spinning ball, a busy flee flying over the screen or dots printed on a status line

- **Clearly Marked Exits**
  Users make errors no matter what else is done to improve the user interface. One goal of the interface should therefore be to make it as easy as possible to recover from these errors. The system should offer the user an easy way out of as many situations as possible.

Regarding this principle all dialog boxes and system states should have a cancel button or other escape facility to bring the user back to the previous state. If a system provides an undo facility this command should be available throughout the system since users quickly learn to rely on it.

Finally, the various exit and undo mechanisms should be made visible in the interface and not hidden in "cryptic" key commands.

- **Shortcuts**
  For experienced users it should be possible to perform frequently used operations especially fast, using dialogue shortcuts, also known as accelerators.

  A good feature which is similar to shortcuts is called *type-ahead*, which means that the user can typing the next input before the computer is ready to accept it. It can be dangerous to allow type-ahead and click-ahead features in all circumstances, especially in critical situations.

- **Good Error Messages**
  Error situations are critical for usability, because they represent situations where the user is in trouble and will be unable to use the system to achieve the desired goal. Error messages should follow four simple rules [Shn82]:

  - They should be phrased in clear language, avoid obscure codes and it should be possible for the user to understand the error message without having to refer to manuals or code dictionaries. If it is necessary to include internal, system-oriented information or codes in the error message, this information should be given at the end of an otherwise human-readable error message and be combined with constructive advice such as *Report this information to your systems manager to get help.*
  - They should be precise rather than vague or general.
  - They should constructively help the user solve the problem. One useful way to generate such constructive error messages is by guessing at what the user really meant to say.
  - Error messages should be polite and not intimidating or put the blame explicitly on the user, like the classic one *ILLEGAL USER ACTION, JOB ABORTED.* Error messages should definitely avoid abusive terms like fatal, illegal and so forth.

  Systems should also provide good error recovery. The user should have the possibility to undo erroneous commands and to edit and reissue previous commands.

- **Prevent Errors**
  Better than to have good error messages would be to avoid the error situation in the first place. Every time the user is asked to spell out something, there is a risk of spelling errors, so selecting a filename from a menu rather than typing it in is a simple

way to redesign a system to eliminate an entire category of errors.

User errors can be classified because of

1. Their frequency
2. Their serious consequences

These errors can be found through user testing or by logging errors as they occur during field use of the system.
Potential risks of errors are modes. A classic example of modes comes from early text editors, which had separate insert and edit modes. By showing states clearly and distinctly to the user, a designer can follow the principle of *providing feedback*, and therefore reduce the risk of an error.
A good idea is to avoid having too similar commands.

- **Help and Documentation**
  The best situation will be that a system is so easy to use that no further help or documentation is needed. The user interface should be as self-explaining as possible, except for walk-up-and-use systems.

  Regular users of a system may want documentation to reach higher levels of expertise. However, one of the most important aspects of documentation is that the most users simply do not read manuals. [Wri83][Wri91] showed that users go through three stages if they interact with manuals

  – Searching - Locate information relevant to a specific need, using the index or the overview map
  – Understanding - the information
  – Applying - Carry out a procedure as described in the documentation

  The sections of the manual should be as self-contained as possible and include a variety of examples.
  There exist three different levels of documentation and it is not useful to provide a system with too much documentation.

  – Short reference cards and/or job aids [Rei88]
  – Tutorial and/or introductory manuals for learners
  – Traditional reference manuals for expert users

- **Heuristic Evaluation**
  Usability guidelines are used in this method to inspect the user interface. Heuristic evaluation means to have a small set of evaluators who examine the interface and judge its compliance with recognized usability principles called *heuristics*.
  The evaluator goes through the interface several times, at least two times, and inspect the various dialogue elements and compares them with a list recognized usability

principles. This method can be used if the user interface exists on paper only.
Heuristic evaluation can also be performed using a technique which is called *pluralistic usability walk-through*. In this technique representative users, product developers and usability specialists evaluate the user interface.

3.5 Usability Testing

Performing tests with real users is the most important usability engineering method. There are a few methods to perform such user tests. Before such a test is started it is very important to clarify the purpose of the test since it will have great influence on the kind of testing to be done. First one has to distinguish if a formative or summative evaluation should be conducted.
Nielsen[Nie93] describes two evaluation methods:

- **Formative Evaluation**
  This method is done in order to help improve the interface as part of an iterative design process. The main goal of formative evaluation is to learn which detailed aspects of the interface are good and bad, and how the design can be improved. The thinking-aloud test is a typical formative evaluation.

- **Summative Evaluation**
  This method aims at assessing the overall quality of an interface for use in deciding between two alternatives or as a part of competitive analysis to learn how good the competition really is. A measurement test is a typical summative evaluation.

3.5.1 Test Plan

Before starting a usability test a test plan should be written. The test plan should also include a budget for the test. A typical test plan addresses the following issues [Nie93]:

- What do you want to achieve ? (Test goal)
- Where and when will the test take place ?
- How long is each test session expected to take ?
- What computer support will be needed for the test ?
- What software needs to be ready for the test ?
- What should the state of the system be at the start of the test ?
- What should the system response times be ?
- Who will serve as experimenters for the test ?
- Who are the test users going to be, and how are you going to get hold of them ?

- How many test users are needed ?
- What test tasks will the users be asked to perform ?
- What criteria will be used to determine when the users have finished each of the test tasks correctly ?
- What user aids, like manuals, online-help, will be made available to test users ?
- To what extent will the experimenter be allowed to help the users during the test ?
- What data is going to be collected, and how will it be analyzed once it has been collected ?
- What will the criterion be for pronouncing the interface a success ?

3.5.2 Pilot Test

At the begin of the testing phase the test procedure should be tried out on a few pilot subjects. Normally one or two pilot subjects are enough, except for large tests. At least one pilot subject should come from the intended user population. The other pilot subjects can be people who are easily available to the experimenter.
Pilot tests can cause changes to

- Questionnaires used for subjective satisfaction rating
- Instructions for some test tasks
- Planned time for test tasks

3.5.3 Test Users

It can be a problem to get test users, who should be as representative as possible of the intended users of the system.
Sometimes it is possible to identify the exact individuals who will be using the system. In this case it will be easy to find test users, even though it may get some difficulties to get them to spend their time on user testing instead of doing their primary work. Users can be recruited from temporary employment agencies, or students from local universities.

Nielsen[Nie93] shows two ways to use usability testing in order to compare the usability of two or more systems.

- Between-subject testing
  - Each test user participates in a single test session. But the main problem is the huge variety of user skills and therefore it can be necessary to have a very large number of test user in each condition. Even detaching users to the various groups can be *dangerous* since users who volunteer early are likely to be different from users who volunteer late. The best solution to this problem is to use random assignment.

- Within-subject testing
  - All the test users get to use all the systems that are being tested. This method has the great disadvantage that the test users cannot be considered as novice users when they approach the other system, they have *learned* from using the first system. This is called *transfer of skill* and takes place between systems. Users are divided into groups to control this effect.

3.5.4 Test Experimenters

A real important decision is to choose an experimenter who is in charge of running the test. There are a few features an experimenter must have knowledge of [Nie93]:

- Extensive knowledge of the application and its user interface
- System knowledge to understand what the users do as they perform tasks with the system
- Ability to handle system crashes otherwise a programmer should be available

One way to get experimenters with a high degree of system knowledge is to use the designers of the system themselves as evaluators. However, there are a few ethical considerations for user testing:

- **Before the test:**
  - Have everything ready before the user shows up
  - Emphasize that it is the system that is being tested, not the user
  - Acknowledge that the software is new and untested, and may have problems
  - Let users know that they can stop at any time
  - Explain any recording, keystroke logging, or other monitoring that is used
  - Tell the user that the test results will be kept completely confidential
  - Make sure that you have answered all the questions the user might have before processing
- **During the test:**
  - Try to give the user an early success experience
  - Hand out the tasks one at a time
  - Keep a relaxed atmosphere in the test room, serve coffee and/or have breaks
  - Avoid disruptions,close the door and post a sign on it and disable telephone
  - Never indicate in any way that the user is making mistakes or is too slow
  - Minimize the number of observers at the test
  - Do not allow the user's management to observe the test

  - If necessary, have the experimenter stop the test if it becomes too unpleasant

- **After the test:**
  - End by stating that the user has helped you find areas of improvement
  - Never report results in such a way that individual users can be identified
  - Only show videotapes outside the usability group with the explicit permission of the user

For each test one has to design test tasks and the basic rule for them is that they should be chosen to be as representative as possible of the users to which the system will eventually be put in the field. The test users should be given the test tasks in a written form. Good sources for test tasks are

- Task Analysis
- Information from logging frequencies of use of commands in running systems
- Field Observation

3.5.5 Stages of a Usability Test

A usability test typically has four stages [Nie93]:

1. **Preparation**
   The experimenter should make sure that
   - The test room is ready for the experiment
   - The computer system is in a start state that was specified in the test plan
   - All test materials,instructions, and questionnaires are available

2. **Introduction**
   The experimenter welcomes the test user and gives a short explanation of the purpose of the test. The experimenter should have a checklist which should cover following aspects [Nie93]:
   - The purpose of the test is to evaluate the software and not the user.
   - The test results will be used to improve the user interface.
   - The system is confidential and should not be discussed with others.
   - A statement that participation in the test is voluntary and that the user may stop at any time.
   - The test results will be kept confidential and not shown to anybody in a form where the individual test user can be identified.
   - An explanation of any video or audio recording that may taking place and in case of videotaping that the face of the user will not be visible.

- The user is welcome to ask questions, but the experimenter will not answer most of them during the test, since the goal of the test is to see if the system can be used without outside help.
- Specific instructions for the kind of the experiment that is being worked with (Think Aloud)
- Invitation to the user to ask clarifying questions regarding the experimental procedure before the test starts.

3. **Running the test**
   The experimenter should not express any personal opinions. The experimenter should not help the test user except that the user is getting unhappy with the situation.

4. **Debriefing**
   After the test, the user is debriefed and is asked to fill in any subjective satisfaction questionnaires. During these questionnaires the users are asked for any comments or suggestions for improvement of the system.

3.5.6 Measure Usability

In the usability engineering lifecycle it is very important to see which usability goals have been met. Usability measurement studies form also the basis for comparing competing products. The measurement process is always done by having a group of test users perform a predefined set of test tasks while collecting time and error data.

Typical quantifiable usability measurements are:

- The time users take to complete a specific task
- The number of tasks of various kinds that can be completed within a given time limit
- The ratio between successful interactions and errors
- The time spent recovering from errors
- The number of user errors
- The number of immediately subsequent erroneous actions
- The number of commands or other features that were utilized by the user
- The number of commands or other features that were never used by the user
- The number of system features the user can remember during a debriefing after the test
- The frequency of use of the manuals and/or help system,and the time spent using these system elements
- How frequently the manual and/or help system solved the problem of the user

- The proportion of user statements during the test that were positive versus critical toward the system
- The number of times the user expresses clear frustration or joy
- The proportion of users who say that they would prefer using the system over some specified competitor
- The number of times the user had to work around an unsolvable problem
- The proportion of users using effective working strategies compared to users who use inefficient ones
- The amount of "dead" time when the user is not interacting with the system (response-time delay versus thinking-time delay)
- The number of times the user is sidetracked from focusing on the real task

3.5.7 Usability Testing Methods

1. **Thinking Aloud**
   This method has traditionally been used as a psychological research method. A thinking-aloud test involves having a test subject use the system while continuously thinking out loud. Through the verbalization of their thoughts, test users enable the experimenter to understand how they view the computer system.

   The main disadvantage of the thinking-aloud method is that it does not fit very well to most types of performance measurement and its strength is the wealth of qualitative data it can collect from a fairly small number of users.
   A real great problem is that thinking out loud seems very unnatural to most people, and some test users, especially expert users, have great difficulties in keeping up a steady stream of comments as they use a system. Therefore performance measurements are probably less representative of the regular working speed of the users.

   However, one of the great advantages of the thinking-aloud method is that one can collect informal comments about small irritants that would not show up in other forms of testing.

2. **Constructive Interaction**
   Constructive interaction involves having two test users use a system together which is also known as *discovery learning.*
   A disadvantage of the method is that the users may have different strategies for learning and using computers.

3. **Retrospective Testing**
   If a videotape has been made of a user test session, it becomes possible to collect additionally information by having the user review the recording. The comments

of the users while reviewing the tape are sometimes more extensive than comments during the test. It is of course possible for the experimenter to stop the tape and to question the user in more detail without fearing to interfere with the test.

4. **Coaching Method**
   The experimenter in this case is called "coach". During a coaching study, the test user is allowed to ask any system-related question of an expert coach who will answer to the best of his or her ability.

5. **Usability Kiosks**
   That means to put a user interface on display in a heavily trafficked area in order to collect comments from users and other people passing by. This method is also known as hallway method.

3.5.8 Where to Conduct User Testing

Many user tests take place in specially equipped usability laboratories. Nielsen[Nie93] shows that a permanent usability laboratory has a few advantages, like

- Fewer scheduling problems
- Other groups will not be disturbed
- The test can be observed by more people

Sometimes companies have portable usability laboratories to be more flexible. The equipment needed includes a camcorder, a notepad, may be a laptop to run the software that is being tested, and two microphones. With this equipment it is possible to conduct user tests in any office.

3.5.9 Other Usability Assessment Methods

There are a few other usability methods that can be used to gather additional usability data.

1. **Observation**
   This method involves visiting one or more users and then doing as little as possible in order not to interfere with their work. The goal of such an observation is to become invisible to the user so that they will perform their work in the same way they normally do.

2. **Questionnaires and Interviews**
   These methods are indirect methods and used to measure the subjective satisfaction of the users in an objective way. Questionnaires and interviews did not study the user interface itself, they try to get information by asking the users of their opinions about the user interface.

The advantage of questionnaires is that they can be administered without the need to have any other people present beside the user answering the questions. Interviews require more usability staff time since there is a need for an interviewer who has to read the questions to the respondent and the interviewer has to record the answers instead of being filled in by the respondent. Interviews are more flexible, since the interviewer may explain difficult questions in more depth and interviews can be more free-form (the interviewer can ask additional questions which are not on the script).

3. **Focus Groups**
   In a focus group, about six to nine users are brought together to discuss new concepts and identify issues over a period of about two hours. Each group is run by a moderator.

4. **Logging Actual Use (Instrumentation)**
   This is the approach taken in instrumenting the Harmony user interface for this study. Logging means that the computer automatically collect statistics about the detailed use of the system. Logging can be used to collect information about field use of a system after release. This method is especially useful, since it shows how users perform their actual work and it is easy to automatically collect data from a large number of users working under different circumstances.
   An interface logfile contains statistics about the frequency with which each user has used each feature in the program and the frequency with which various events of interest such as error messages have occurred. Such statistics can be used to optimize frequently used features and to improve and to make features more accessible to users that are rarely used. It may also be possible to completely remove such features from the system.
   Logging is usually achieved either by instrumenting low-level parts of the system software, such as keyboard and mouse drivers or by modifying the software of interest. The latter approach is much preferred, since it makes it easier to log events of interest. Once a system is instrumented it is easy to collect data over extended periods of time. This methods raises some privacy concerns that can normally be addressed by explaining to users that only summary statistics are being collected and that results will only be reported in a form where the individual users cannot be identified. However, users should be informed when interaction logging is going on and that they should be able to disable the log if they so desire.
   The major advantage of this method is that interfering with the users did not happen in any way. The major disadvantage is that the logging data shows only what the users did, but not why they did it. It is possible to combine the logging method with other usability methods such as interviews.

5. **User Feedback**
   User feedback can be gathered from installed systems and has several advantages:

   - It is initiated by the users, so it shows their immediate and pressing concerns
   - It is an ongoing process, so feedback will be received without any special efforts to collect it
   - It will quickly show any changes in the needs,circumstances or opinions of the users,since new feedback will be received whenever such changes occur.

User feedback is gained through reply cards, network newsgroups, bulletin boards or beta test sites.

Chapter 4

Usability in Practice

4.1 Usability Engineering at Companies

Usability has to be seen as a corporate identity factor of entire product families. For a company that sells software or other products on the open market, the usability of each product will contribute to the general reputation of the company as a quality supplier. Just a single product with poor usability can cause severe damage to the sales of the entire product family. This very important observation was recognized by the most companies and therefore each company tried to establish a usability engineering program. One of several of such usability engineering programs is presented now, namely the usability approach taken at Microsoft.

4.2 The Usability Approach at Microsoft

As described in [Wik94] Microsoft realized very early that ease of use do have great influence on selling rates of products. A usability group and several usability laboratories were established. Product development teams at Microsoft are typically composed of several subteams with specific goals, including

- Marketing
  Marketing aims to define a new product, or a new release, based on features that will compete and sell.
- Program Management
  The goal of the program management team is to design the interface, that part of the software, the end-user sees.
- Development
  The development team, the programmers, write code to create the specified product.
- Testing
  Testing, also called quality assurance, ensures that the code performs as specified.

- User Education
  The goal of the user education team is to produce the print and on-line documentation and training that teaches users how to use the product and presents details about the features of the product.

Before the foundation of the usability group in the mid 1980s, each team based its work primarily on "traditional" sources of information about end users. Marketing used demographic data based on feature-driven buyer profiles, like surveys, focus groups, and customer feedback cards. The interface design team worked using the data of the marketing group and from their "working" audience profile consisting of assumptions about when and how people would use the product. Development team and quality assurance team built an audience profile using sources like those listened above instinct and experience, reviews in the trades, anecdotal information from the field, and information from other developers.

In the mid 1980s a movement from feature-centered design to user-centered design started. User-Centered design means that the teams now include information about how users work, think and solve problems in their product planning. However, not all of this information comes from the usability group. The product management team has several techniques to help them focus on the users, including instrumented versions and beta testing. The role of the usability group is to provide additional information to the user information the teams already have. This is done through "formal" usability testing at all stages of product development. Product development can be divided into three main phases:

- Planning
  - During the product planning phase, usability specialists can do at least several types of exploratory usability testing: with users only, with products of competitors, on previous versions of the product, with documentation, and with prototype software. Usability data can impact the product specification, the marketing strategy, and the documentation plan.
- Coding
  - During the coding phase, the usability team can test iteratively with product prototypes and with documentation versions. This testing can impact the evolution of the new product (paper mock-ups, prototypes, code).
- Testing
  - During the testing phase, usability testing can confirm the ease-of-use of the "debug" versions of the software and documentation (Alpha and Beta versions), impacting both the final product and future versions.

Wiklund[Wik94] reports that the Usability Group at Microsoft was founded 1988 by Mary Dieli, and one of its goals was to work as user advocates. The usability group offers a lot of different services from ergonomic testing, laboratory testing to team training in usability. Several teams like marketing people or developers consume these services on regular basis.

4.2.1 Case Study: Term Tool

This study dealt with a tool to collect inexpensive data, which is called Term Tool. For the usability group at Microsoft it was one of the goals to create tools to collect useful data at little cost.

Term Tool helps to generate user vocabularies - lists of colloquial terms that users use to describe common tasks. In graphical user interfaces menus are very hard to design for the designer because the designer had to communicate the usefulness of a function in a very small amount of space - a single word or a short phrase. Traditional usability tests are not a very good way to get colloquial terms from users. The words or phrases a user generates to represent a task are often a useful byproduct of a usability test. But it is time consuming and expensive to test enough users to generate a representative sample of terms that would improve users performance with help in a meaningful way.

Term Tool is a Visual Basic application that gives the possibility to collect such colloquial terms from users. It graphically represents tasks as before and after pictures and users are asked to generate four terms in response to the question like "To change from this (Before) to this (After), what would you call it ?" The responses were written to a text file and collected and collated in Microsoft Excel.

In this study Term Tool was used to support on-line help indexes. It was not enough to simply generate a list of possible terms and include them in the index, since the usability team wanted to know whether including terms in an index would improve the performance of the users at finding information.
To test Term Tool 25 Microsoft Excel tasks were presented to 20 new spreadsheet users. These test users were selected from the Microsoft usability database. An instrumented version of Microsoft Excel 4.0 was used to collect the 30 most commonly used commands. Tasks involving these commands were created.
The users in this part of the task generated 168 different "colloquial" terms. The highest number of terms for one command was 11 and the lowest number was 2. If a term was generated by at least 5 users it was included in the index with the appropriate page number and a "see also" reference.

Ten test users were given the Microsoft Excel 4.0 User's Guide index and ten users were given the index with the user terms added to the index. The users had to find the page number for te task displayed on the screen and each correct page number was counted as a success otherwise as an error. The result showed that the users using the augmented index had remarkably more success than the others.

Therefore Term Tool is an effective and inexpensive way of generating user vocabularies and a useful way to improve indexes, both print and on-line.

4.3 Usability Engineering at Universities

A lot of universities do have usability engineering programs like Tufts University or Virginia Tech and some of them work together with companies like Microsoft, American Airlines, Digital and others. These agreements between universities and companies mean that undergraduate and graduate student interns work for the usability groups of the companies as usability specialists. Besides usability people of the universities conduct usability studies for research purposes.

One such study which characterizes browsing strategies in the World-Wide Web is presented here[CP95].

4.3.1 Case Study: XMosaic

This study was conducted at Georgia Institute of Technology and captured client-side user events of the XMosaic Web-Browser of NCSA. The goal of the study was to determine actual user behavior from log file analysis in order to understand user navigation strategies. Log file analysis also yields design and usability guidelines for WWW pages, sites and browsers. The study was conducted for a three week period at Georgia Institute of Technology. In contrast to the case studies at Microsoft,this study used interns as test users like Computing staff, faculty and student populations of the Georgia Institute of Technology.

Cove and Walsh[CW88] show in a study the three browsing strategies that are most popular by Web users:

- Search browsing (directed search where the goal is known)
- General purpose browsing (consulting sources that have a high likelihood of items of interest)
- Serendipitous browsing (purely random)

Due this approach it is possible to distinguish between browsing as a method of completing a task and open ended browsing with no particular goal in mind. But it should be also clear that browsing and searching are not mutually exclusive activities. Therefore it is one of the most difficult tasks to design WWW pages that support both the browser and the searcher. Since this is not always possible, there is a need to balance this problem. The first step to do this is to determine what strategies are being used by the population.

In this study all events generated by consenting Computing staff, faculty and student populations of the Georgia Institute of Technology, who operate NCSA's XMosaic running Sun OS 4.1.3 should be captured. A version of XMosaic was coded to trap all user interface level events. This activity is known as Software Instrumentation.

The computing environment consisted of over 250 Sun OS 4.1.3 machines connected via a LAN. All captured events were processed and forwarded to a secure disk to minimize the

risk of data loss resulting from network and/or system failures.
It was also important to have a meaningful representation of the data of user events. Such a representation provides not only a clear understanding of the extent and functionality of the interface, but also a clear extraction of task specific data during analysis. The events were recorded according to the User Interface Design Environment guidelines for task representation. Because of this, all actions were classified in three levels:

- Application Action (high-level task, Open File)
- Interface Action (mid-level task, select item from pull-down menu)
- Interface Technique (low-level task, Mouse Click)

An entry in the log file looks like:

```
      Aug 3 00:23:11 foo.cc.gatech.edu uel: 775887872
  123 1 Mouse Navigate Anchor::http://www.somewhere/
```

This means, that a user clicked on a hyperlink in the document window that pointed to `http://www.somewhere/` the user is identified as participant number 123 and the event was generated from machine `foo.gatech.edu on August 3rd`.

When the user started the test a consent window was displayed that informed the user of the experimental procedures employed as well as of their rights as human subjects. The intent of the consent window was both informative and to minimize the "Big Brother" effect. This consent window appeared the first time XMosaic was executed by each user during the sample period. The majority of the users chose to participate in the study.

XMosaic was selected since in 1994, when the study was conducted, it was the most popular Web client. The orginial log file continued over 43000 events, with each record uniquely identifiable by user id and time of occurrence. Since users will often leave XMosaic running for extended periods of time without interacting with it determining session boundaries was necessary. To do this, all events that occurred over 25.5 minutes apart were delineated as a new session.

Document requests were distinguished by protocol and it came out that the most popular method to explore the Hyperspace was to use hyperlinks (52%). The next methods in order of popularity were "Back", "OpenURL", "Hotlist", "Forward", "Open Local", "Home Document" and "Window History". This indicated that users typically did not know the location of documents a priori, or relied on other heuristics to navigate to a specific document. Most users did not select items in the hotlist and window history and therefore it seems that they either preferred using "Go_To" or did not know how to employ this interface technique.
All menu items have corresponding keyboard events, but only 4272 events were executed via the keyboard. This may be due to the lack of display of keyboard equivalents next to menu items. One result of the analysis was that users rarely traverse more than two layers in the hypertext structure before returning to an entry point. One navigation method that was also used often was use of homepages as indexes to interesting places.

Chapter 5

Hyperwave and Harmony

5.1 Hyperwave

Hyper-G now named Hyperwave is an extension of WWW[Mau96]. Hyperwave servers provide WWW pages with lots of additional functionality. Hyperwave assures compatibility across various user sites. This compatibility means that link consistency within servers and across server boundaries can be assured and searches in a combination of parts of different Hyperwave databases become possible.

Conventional WWW viewers like Mosaic, Netscape Navigator, Microsoft Internet Explorer, work great with Hyperwave.
Hyperwave has a general annotation facility and "scripting support". This concepts enable the use of the Web not only for information presentation purposes, but also for structured discussions between groups of persons. As mentioned before, Hyperwave supports bidirectional linking and automatic link consistency, which means that an error like *Object cannot be located* cannot occur. Hyperwave has also orthogonal hierarchical structuring and composition facilities and integrated indexing and search facilities.

Basic Concepts

The Hyperwave Data Model contains structuring elements beyond the primitive node-link model.

- Collections
- Clusters
- Sequences

The most important of the elements is the **collection**. A collection is a composite object, which contains documents or other collections. Since every document or collection must be a member of at least on collection, except the root collection, a collection hierarchy is created. This hierarchy is a tree structure, which is called a *directed acyclic graph.*

The concept is extended by two subclasses of a collection, **clusters** and **sequences**. A cluster is a composite document consisting of other documents. If the user selects a cluster, all members of the cluster are visualized together. Clusters are especially useful in the creation of multilingual documents. If such a cluster is selected, all language-independent members of the cluster, these are objects which have more than one `Title` attribute, are visualized and from the remaining language-dependent members, one per document type is chosen, based on the language preferences of the user, and visualized.

A sequence is an ordered list of sub-collections or documents. The members of a sequence are displayed one after the other defined by the `SortOrder` attribute of the sequence and possibly the `Sequence` numbers of the members. The user perceives the members of the sequence as nodes connected by "next" and "previous" links, which are generated dynamically during browsing.

Searching

Searching is fully integrated with browsing and it is a very powerful feature that every Hyperwave document, including documents, links, collections, can be searched for.

Hyperlinks

Hyperwave maintains a separated link database. Links are bidirectional , which means that it is possible to navigate backwards from destination to source. This concept provides link consistency.

Hyperwave is a multi-user and multi-author system, since users may also contribute material, like annotations or personal home-collections, to the information base, using their standard Hyperwave client.

Most users will access Hyperwave using Web browsers like Netscape or Internet Explorer. In this case, a Hyperwave server behaves like a "standard" Web server with some special features. However, accessing a Hyperwave server with a "native" Hyperwave client, users are offered additional navigation aids and other features. In this case, users may also modify the information on the server, if they have appropriate access permissions. The Hyperwave clients are also compatible with "standard" Web servers.

Hyperwave is a *second generation hypermedia system* which uses tightly coupled structuring, search and linking facilities, and sophisticated user interface metaphors. Hyperwave is available for software download from `http://www.hyperwave.de` and also on CD-ROM.

Features

Hyperwave Information Server offers a lot of interesting and very useful features. Some of these sophisticated features are:

- User and group management dialogs support automatic home collection generation when inserting a new user.
- Deletion of collection trees.
- Rights Wizard to specify object access rights.
- Boolean operators in searches.
- Step by step search by restricting searches on results of previous search operations.
- Documentation is available in HTML format and help texts in German language.
- A group of Hyperwave servers may share a common user database and maintain link consistency within the server pool. A single Hyperwave Server can be defined for user and group management, so that a registered user is able to log on every server of the server pool.
- HTML Parser for parsing HTML files during document upload.
- CGI in PLACE Templates.
- A version of wavemaster, the HTTP interface, with Secure Socket Layer 128-bit encryption support.
- Version control to save and retrieve all versions of a document ever created. The version control ensures that documents are secure while editing and review within workgroups.
- Online Link Editing with the Link Wizard to create and edit hyperlinks between HTML documents without the need of an editor or HTML authoring tool. There is need for editing the document itself.
- Fulltext searching and indexing of more than 150 different document formats including HTML, plain text, MS Word, MS Excel and Adobe PDF.
- Annotations can be made to single words, phrases or parts of a document. Every Hyperwave user can attach such annotations to documents, even if the user does not have write permission to the documents themselves.
- Additional Navigation Aids like the Hyperwave Explorer applet, which embodies similar navigation functionality as the Microsoft Windows filesystem Explorer. A Local Map applet displays a map of hyperlinks and parent/children relations for a given document. The Local Map is also available as an HTML-based document reference screen for WWW clients without Java functionality.

- External User Identification gateway which supports the WindowsNT 4.0 server's user database.
- SQL Databases are supported.
- Common Logfile Format for using generic statistics programs to analyze the logfiles of a Hyperwave server.
- Support frames in HTML documents.

Hyperwave, an innovative information server that solves many of the problems associated with electronic publishing on the Internet and intranets, won BYTE Magazine's third annual **Best of Show** award at CeBIT '97.

5.2 Harmony

Harmony is one of the "native" Hyperwave clients. Harmony is the Hyperwave client and authoring tool for Unix platforms under XWindows[And96]. Harmony is mainly used for authoring, searching and browsing through the information on a Hyperwave server, and communicating with other users.

The main Harmony window is the Harmony Session Manager. The Harmony Session Manager is mainly used for navigation. The collection browser is the main window of the Harmony Session Manager. The collection browser provides intuitive hierarchical navigation through Hyperwave collection structures, which are displayed in form of icons. Icons are also used to represent clusters, documents and anchors. Hierarchical navigation through Hyperwave structures is made as intuitive as possible, since only a limited set of visible choices at each level is presented to the users. The Harmony Session Manager provides:

- Navigational Functions
  These functions, like `Open, Close, Children, Home, GoTo, etc.`, are used for navigation in the collection browser.
- Informal Functions
  These functions, like `Attributes, ShowHost, ShowCount`, provide additional information about Hyperwave objects.

A toolbar which provides access to the most frequently used functions and extensive keyboard support are also available.
Harmony has native document viewers for text, image, film, audio, PostScript and 3D scenes. It is also possible to configure Harmony to start external viewers. However, the whole hyperlink facilities are not available in this case. Any kind of media may contain hyperlinks in Hyperwave. Therefore Harmony provides hyperlink facilities in each of its document viewers. All viewers share a number of common navigational facilities like `hold, attributes, back, forward, references, annotations and parents`.

An important feature to visualize relationships between objects in Hyperwave is Harmony's Local Map. The Local Map generates a graphic visualization of a document's hyperlink relationships and it is also possible to display annotations or parent-child (collection membership) relationships.

Harmony embodies the sophisticated search facilities of Hyperwave. Hyperwave has fully integrated attribute and content search mechanisms. However, searches are not restricted to objects on the local Hyperwave server.

Information Landscape

A three-dimensional representation of the collection structure of a Hyperwave server is available through the Harmony Information Landscape. This three-dimensional representation is an alternative of the two-dimensional collection browser view. However, much of the functionality of the Harmony Session Manager is also available in the Harmony Information Landscape.

Future Trends and Development

Since Hyper-G was commercialized and a WWW gateway was established, most users use a WWW client like Netscape. Therefore the development process of Harmony was finished and the gateway functionality was extended with Java applets. These applets use many concepts and functionality of Harmony, like the implementation of the Local Map applet in the newest release of Hyperwave Information Server Release 2.5.

Chapter 6

Instrumenting Harmony

An instrumented version of Harmony, the Unix-based Hyperwave client, was tested in a field study.

6.1 Software Instrumentation

A formal definition of instrumentation should be given first. Instrumented versions of software are specially augmented versions that create separate logfiles to record every mouse click and keystroke of a test user as well as the time when the action happened. The logfiles can be ASCII files or binary files. In the first case even test users are able to read the logfiles, whereas in the second case an additional tool is necessary to view the logfiles. It is possible for analysis purposes to reconstruct and understand not only what tasks a user accomplished, but how the user carried them out. For example, when the user chose the keyboard in preference to the menus or Toolbar to perform a given task.

Usability data, gathered through such instrumented software can be used for task analysis as well.

6.1.1 Software Instrumentation with Harmony

Harmony, the Unix-based Hyperwave client for X11, communicates with the Hyperwave server through the Session Manager who starts further processes, like Text Viewer, and provides navigational aids. The Harmony package consists of several viewers and some additional tools

- Session Manager
- Text Viewer
- Image Viewer
- Film Player

- 3D Scene Viewer
- HarAdmin Tool
- EditAPI Tool

The viewers and tools listed above were all instrumented. These viewers and tools are written in C++ and InterViews, which provides the GUI elements like buttons, dialogs, message boxes or menu structures.To achieve the necessary level of instrumentation a new class was developed. This class provides member functions for manipulating the logfiles and for data transmission. InterViews had to be extended to provide information about which interaction technique the test user chose. Available interaction techniques are

- Mouse
- Keyboard
- Menu
- Button

Finally, data defines for almost every event were set. These data defines and the interaction technique as well as a timestamp are written to the logfile. The logfile has ASCII format which means that it can be viewed without a special viewer.

A header file was created which contains the data defines. A symbolic notation fur such a define is:

```
<DEFINE>::="#define" | <NAME OF THE DEFINE>
     | """ | <STRING TO BE WRITTEN TO THE LOGFILE> | """
```

An example for such a define is:
```
#define n_localmap "Navigate_LocalMap "
```

If the test user executes the "LocalMap" command the corresponding entry in the logfile might be:
```
key Navigate_LocalMap Tue 2 Mar 1997 10:33:20
```
This means that the test user had selected the "LocalMap" command through the keyboard on Tuesday 2nd March

The selection of the ASCII format for the logfiles was the result of the following considerations.

- The name of the user or any personal information must not be written to the logfile
- Any system information, like password, user-id, must not be written to the logfile
- The user should be able to read the logfile using zcat whenever he or she wants

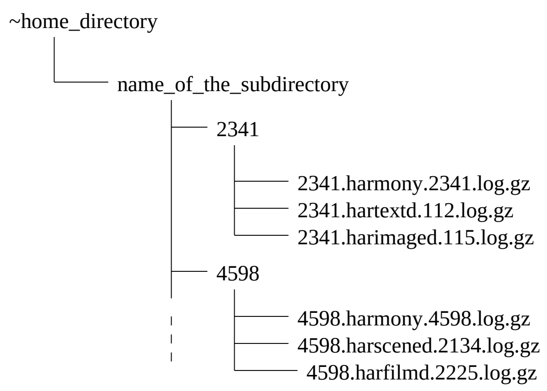


Figure 6.1: Disk structure created by the instrumented version of Harmony.

- It is simpler to compare two or more logfiles, since they are human-readable
- Since there is no violation of user privacy as mentioned above, there is no need to encode the logfiles
- Logfile analysis is easier, since there is no need to decode the logfiles (only parse the strings)

Once the test user has finished a Harmony session, the generated logfile is saved in a subdirectory. This subdirectory is named with the process-id and is generated in a subdirectory of the user's home directory named `.harmony-log`. Then the logfile is compressed. Depending what software the user has installed on his or her machine `gzip` or `compress` is used for compression. A logfile is generated for each viewer started during the Harmony session. This depends on the type of documents that are provided by the Hyper-G server and that the user works with.

A typical filename consists of the session-id, the name of the viewer or tool, the process-id and the suffix `gz` or `Z`, depending on the utility that is used to compress the logfile `gzip` or `compress` :

```
1211.harmony.1211.log.gz or 1211.harmony.1211.log.Z
```

If a session does not terminate correctly this suffix does not exist, since the compression utility was not executed.Then the filename looks like:

```
1211.harmony.1211.log
```

After a typical Harmony session the disk structure could be as shown in Figure 6.1. For each session a new subdirectory, named with the session-id is created in `.harmony-log`.

6.2 Data Submission

One of the most important and difficult tasks is the transmission of the test data after the test is finished.
It is not enough to create an instrumented version of a software without planning how to submit the data after the test. Since each test user generates `his or her` own logfiles, data had to be transferred over the network to the IICM in Graz for analysis purposes. To achieve this a set of Perl-scripts was developed.

- prepare.pl
- client.pl
- sendlogs.pl
- server.pl

The scripts and their functionality are discussed here.

6.2.1 Prepare.Pl

The script `prepare.pl` works as follows. It is automatically invoked, when the test period ends. This means, if the test user starts Harmony after the test period has ended the script `prepare.pl` is started first. The script gets the path where the logfiles are stored and creates a tar archive, using the tar command this command is normally available in each Unix system. After that the tar file is compressed either with gzip or with compress and finally `uuencoded`. If this process was successful, the individual logfiles are removed from the disk. The result of the script `prepare.pl` is a single file in the home directory of the user. This file is called

```
xxx.logfiles.tar.gz.uue or xxx.logfiles.tar.Z.uue
```

where `xxx` stands for the username.

6.2.2 Client.Pl

The script `client.pl` is created to carry out the transfer of the logfiles to the IICM in Graz. There are two versions of this script. One version communicates with the script `server.pl` which is running on a machine at the IICM and the other version works with email. The reason for this strategy was to minimize the potential risk of data loss during transmission. Data loss can occur because of Internet bandwidth or firewalls. Therefore the version with email is used by test users outside the IICM, whereas the other version is used by the test users at the IICM.

Test users came from

- Germany

- Australia
- Japan
- Italy
- Macau

The main task of both versions is to transfer the tar file that was created from the script `prepare.pl` to the IICM in Graz.

The version that is used at the IICM communicates with the script `server.pl` which is also running on a machine at the IICM. The communication is done via TCP/IP and the script `server.pl` is able to handle more than one connection. When the connection is established the file name and size are transmitted first and then the data. The script `server.pl` counts the file size and if the transmitted file size and the received number of bytes are identical, the number of bytes received are sent back to the script `client.pl`. The script `client.pl` compares this value with the file size and if it is identical the connection is closed. Otherwise the transmission process is started once more. If the data transmission was successful the file is removed and a file named `.transmission-info` is created in the subdirectory `.harmony-log` of the home directory of the user.

The other version of the script `client.pl`, which works with email, uses the mail command to send the tar file to the IICM. The command is used

```
   mail iegger@iicm.edu < xxx.logfiles.tar.gz.uue
or mail iegger@iicm.edu < xxx.logfiles.tar.Z.uue
```

After sending is done a file named `.transmission-info` is created in the subdirectory `.harmony-log` of the home directory of the user.

6.2.3 Sendlogs.Pl

The script `sendlogs.pl` works like the version of the script `client.pl` which works with email. Except the script sendlogs.pl can be started manually by the user to transfer the logfiles. This is useful in case something goes wrong with data transmission.

6.2.4 Server.Pl

The script `server.pl` runs on a machine at the IICM in Graz and listens to a specific port for connections. The script `server.pl` works with the script `client.pl` as described above. The script `server.pl` accepts connections via TCP/IP and stores the transmitted data in a file in a subdirectory of a disk on a machine at the IICM. The name of the created file is called

```
   xxx.logfiles.tar.gz.uue or xxx.logfiles.tar.Z.uue
```

where `xxx` stands for the username.

6.3 Data Analysis

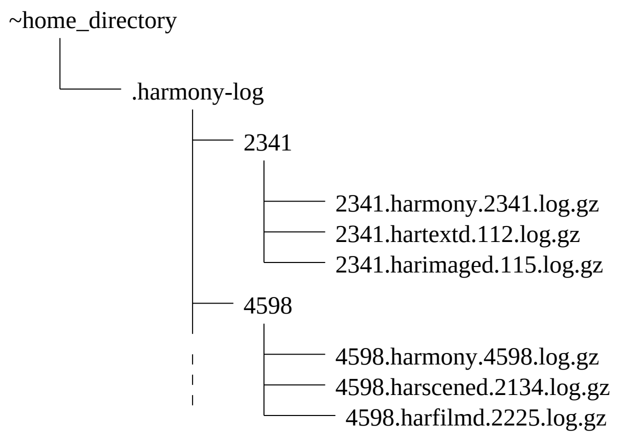


Figure 6.2: Disk structure created after `uudecoding` and `uncompressing` the received tar files.

The next goal after the usability data has been received at the IICM, whether through email or via client-server, is to analyze the data. A Perl-script was created to simplify the analysis process.

The main goals of the analysis are to reveal

- Typical patterns of use
- Most commonly used features of Harmony
- How often do users search
- What are the most common ways to navigate
- Number of hyperlinks followed
- Comparison of searching, hierarchy navigation and hyperlinks
- Number of error messages encountered
- Patterns including error messages
- Actions taken after getting error messages

The answers to these questions are intended to focus work on improving and fine-tuning the user interface of Harmony. A Perl script was created to support the analysis. The functionality of this Perl script is described here. As explained before, one usability data file per test user is created and transmitted to the IICM in Graz. These files are collected and stored in a separate subdirectory. During the analysis process these files are `uudecoded`, `uncompressed` and the tar file extracted.

The result of these activities could be as shown in Figure 6.2. The used structure is the same as the users have. After the unpacking process is finished the real analysis process is started. This is done by executing the Perl script. The script gets the `name_of_the_subdirectory` and starts to read the logfiles. During this activity the number of actions, corrupted logfiles, number of viewers started, and other information are extracted from the logfiles. The result of the script consists of a set of HTML files, which can be viewed with any Web Browser. The script generates a graphical interpretation of the data as well. However, the interpretation of the results of the Perl script is done manually.

Chapter 7

Research Experiment

After the instrumenting phase was concluded some *pre-pilot tests* were started. This was an iterative process, because after one or more viewers were instrumented, the new version was installed and new testing was done. These tests were especially valuable, since the tests show

- The stability of the instrumented version of Harmony
- The correctness of the creation process of the logfiles
- The functionality of the data transmission process
- The amount of data to expect from each user

Since these testing activities were done to become familiar with the method of software instrumentation and evaluation, the main goal of these activities was to create an instrumental version of Harmony, which run stable enough for further testing purposes. As remarked in a previous chapter it is very important to find test users. The test users for these *pre-pilot tests* were members and students of the IICM in Graz. There were some reasons for this selection:

- Availability
  Since all test users in this phase are colleagues or freelance staff no organization was needed to recruit them.
- Feedback
  In case of software bugs informal feedback could be gathered very fast.
- No Restrictions
  There was no need to insist, that the test users do not publish any details of the tested software.

For data transmission the Perl scripts `client.pl` and `server.pl` were used. Since the transmission occurred only between computers situated at the IICM in Graz, there were no problems due to network bandwidth or firewalls.

7.1 Test Plan

Nielsen [Nie93] gives a good example of a typical test plan. The test plan for this test addresses the following issues:

- What do you want to achieve ? (Test goal)

  The goal of the usability test is to reveal typical patterns of use, most commonly used features, navigation techniques.

- Where and when will the test take place ?

  The test users, spread over the world, will participate in the test via the Internet. Each test user, who participates in the test, can work in his or her own working environment, only the computer of the user must be connected to the Internet.

- How long is each test session expected to take ?

  The test will run over a period of 30 days and therefore more than one test session will happen. It is not possible to estimate how long the individual test sessions will take.

- What computer support will be needed for the test ?

  Each test user needs a computer which is connected to the Internet, since the users will be contacted with email.

- What software needs to be ready for the test ?

  The instrumented version of Harmony must be available per anonymous FTP from the IICM in Graz. The script `server.pl` must be installed and running on a machine at the IICM in Graz. The test users need to have Perl version 5.003 or higher installed on their machines. Finally, the test users must have UNIX or a UNIX clone like Linux, operating system installed on their machine.

- Who will serve as experimenters for the test ?

  No experimenters are needed, since the goal of this method is to collect usability data from the users without interfering them.

- Who are the test users going to be, and how are you going to get hold of them ?

  There are three different types of tests.

  - Pre-pilot test: Members and students of the IICM in Graz

  - Pilot test: Two persons, both known from postings to the newsgroup `comp.infosystems.hyperg`
  - Real test: Users contacted through the Newsgroup `comp.infosystems.hyperg`, WWW mail archive servers, Hyperwave Support Group

- How many test users are needed ?

  Nielsen [Nie93] suggests at least 2 test users for the pilot test, one of them should come from the pool of the intended user population, and at least 20 test users for the real test.

- What test tasks will the users be asked to perform ?

  No test tasks will be designed and given to the users, since data should be gathered about the *real* working with the software.

- What user aids, like manuals, online-help, will be made available to test users ?

  Only a Readme file with instructions, how to install and run the instrumental version will be given to the users. Users can use the online-help, if such help is available from the system.

- What data is going to be collected, and how will it be analyzed once it has been collected ?

  A data file will contain several lines of user actions during the session. Each user action consists of the type of action, in which way the action was executed and a timestamp. The analysis will be done with a Perl script, which creates tables and graphical interpretations of the user actions, to see typical patterns of use, most used functions and error situations.

7.2 Pilot Test

Since pilot testing is done to ensure that the test procedure is correct, this phase is one of the most important activities in the whole testing process. Typically one or two users are needed for a pilot test, but this number can increase if a larger test is planned. In this case two test users, both from outside the IICM, were selected.

In this phase of the testing process the *pre-pilot tests* were finished and an instrumented version was made available via FTP to the two test users. The two intended test users were contacted via email and asked for their participation. Both were interested in participating in the test. After that a set of instructions were given the test users via email. These instructions mentioned how to install and run the new version. A Readme file was included in the package as well. This file contained information about:

- Installation and instrumentation
- Running the test
- Data transmission

For the pilot test the test users were asked to set the period of the test phase to 14 days. The instrumentation had to be enabled with an environment variable. This variable was disabled per default. The instrumented version of Harmony was equipped with a set of message boxes to inform the user and to minimize the big brother is watching you effect [Nie93]. If the instrumentation was not enabled none of these message boxes appeared on the screen of the user and of course no logfiles were created. Due to this concept there was no need to develop a separate version for users, who did not want to participate in such a test, since the instrumentation was disabled per default. However, the instrumented version was simultaneously the newest version of Harmony.

Both users outside the IICM used the Perl scripts `client.pl` and `server.pl` for data transmission purposes. Since one of them came from outside of Austria there were problems in transmitting data to the IICM in Graz. These problems were due to network bandwidth. However, the script `prepare.pl`, which was invoked automatically at the end of the test period and run local on the machine of the user, worked properly and created the tar archive, which contained the logfiles of the user. Therefore it was possible for the user to send this tar archive manually via email to the IICM in Graz. However, people from the IICM in Graz participated in this pilot test as well.

7.3 Revised Test Plan

The test plan was revised after the pilot test phase was finished. The revised test plan looked like:

- What do you want to achieve ? (Test goal)

  The goal of the usability test is to reveal typical patterns of use, most commonly used features, navigation techniques.

- Where and when will the test take place ?

  The test users, spread over the world, will participate in the test via the Internet. Each test user, who participates in the test, can work in his or her own working environment, only the computer of the user must be connected to the Internet.

- How long is each test session expected to take ?

  The test will run over a period of 30 days and therefore more than one test session will happen. It is not possible to estimate how long the individual test sessions will take.

- What computer support will be needed for the test ?

  Each test user needs a computer which is connected to the Internet, since the users will be contacted with email.

- What software needs to be ready for the test ?

  The instrumented version of Harmony must be available per anonymous FTP from the IICM in Graz. The test users need to have Perl version 5.003 or higher installed on their machines. Finally, the test users must have UNIX or a UNIX clone like Linux, operating system installed on their machine. `UUencode` must be installed on the system of the user. The data transmission will be done with email.

- Who will serve as experimenters for the test ?

  No experimenters are needed, since the goal of this method is to collect usability data from the users without interfering them.

- Who are the test users going to be, and how are you going to get hold of them ?

  There are three different types of tests.

  - Pre-pilot test: Members and students of the IICM in Graz
  - Pilot test: Two persons, both known from postings to the newsgroup `comp.infosystems.hyperg`
  - Real test: Users contacted through the Newsgroup `comp.infosystems.hyperg`, WWW mail archive servers, Hyperwave Support Group

- How many test users are needed ?

  Nielsen [Nie93] suggests at least 2 test users for the pilot test, one of them should come from the pool of the intended user population, and at least 20 test users for the real test.

- What test tasks will the users be asked to perform ?

  No test tasks will be designed and given to the users, since data should be gathered about the *real* working with the software.

- What user aids, like manuals, online-help, will be made available to test users ?

  Only a Readme file with instructions, how to install and run the instrumental version will be given to the users. Users can use the online-help, if such help is available from the system.

- What data is going to be collected, and how will it be analyzed once it has been collected ?

  A data file will contain several lines of user actions during the session. Each user action consists of the type of action, in which way the action was executed and a timestamp. The analysis will be done with a Perl script, which creates tables and graphical interpretations of the user actions, to see typical patterns of use, most used functions and error situations.

7.4 Real Test

After the pilot test phase was finished the real test was started. A 30 day test period was planned for the real test. As a consequence of the pilot test, the Perl scripts, which were needed for data transmission, were changed. The reasons for these changes were:

- The amount of data of each test user in the pilot test was between 50 and 150 kbytes.
- Most of the mail demons should be able to handle such amount of data.
- Potential problems with network bandwidth and firewalls.

The Perl scripts `client.pl` and `sendlogs.pl` were changed and the following command was used in these scripts:

```
   mail iegger@iicm.edu < xxx.logfiles.tar.gz.uue
or mail iegger@iicm.edu < xxx.logfiles.tar.Z.uue
```

As a result of these changes, the test user had to be informed, when his or her logfiles were received at the IICM in Graz. After the correctness of the received logfiles was checked an email was sent to the test user and he or she was asked to remove the tar archive which existed local in his or her home directory.

The first step in the real test phase was to look for test users. [Nie93] suggests at least 20 test users for this test method. [Wik94] reports, that Microsoft sports a fully established user database and have contacts to agencies, which provide people who are interested in participating in such usability tests. Since no established user database was available at the IICM, there was a need for other sources for test users. Possibilities to find test users were:

- The newsgroup `comp.infosystems.hyperg`
- Mail archives at WWW Servers
- Through the Hyperwave technical support staff

After some days a list consisting of 67 potential test users was established. An email was sent to all of them asking for their participation in the test. Some of the people of the list were representatives of groups of users and therefore the number of potential test users was much greater than the named 67. A two step strategy was developed to get test users for the real test. Two emails were sent to the potential test users with a waiting phase of two weeks between the two mails. After a waiting phase of four weeks the second phase was finished and as a result 9 people of the list (13.43%) chose to participate in the test. Finally 22 people of the list gave simply no reply to the mail. Among the 36 people of the list, who gave negative replies were 8 persons with no valid email. This can happen, if someone leaves a company or an institute. A possible explanation for the difference between the first and second phase is, that some people were engaged in other projects or were on holiday and therefore catched the email too late or simply deleted the first email. Therefore the number of replies increased in the second phase. See Table 7.1 for detailed information over the individual phases. This user behavior is also displayed in Figure 7.1.

The rest of the people of the list apologized and gave several reasons for not participating in the test:

- Data on Hyperwave servers can be manipulated with other tools like Amadeus, which is the client for Microsoft Windows
- A WWW gateway exists and therefore it is possible to use Netscape Navigator for browsing a Hyperwave server
- Some said that they simply had no time to install and use the new instrumented version of Harmony

See Table 7.2 for detailed information over the individual reasons for not participating in the usability test. Several users gave answers which included more than one reason for not participating. Therefore their responses fall into more than one category of reasons. The category *Host or user unknown* was included in the negative responses even if it could be counted as no reply. This was done, since it was definitely clear that these people would not participate in the usability test. The column called *Responses* shows how much of the users gave an answer which included the corresponding reason. These reasons are also shown in Figure 7.2.

The test participants were asked to download the new instrumental version of Harmony from the FTP server of the IICM in Graz and detailed instructions for the installation of the new version were also given to them. Since the test period was set to 30 days per default, they needed not set this variable manually. If a user used Harmony for instance 40 days after the first time, the test period has expired and the logfiles were transmitted. However, there is no guarantee that the test period of each test user is exactly 30 days, since it depends very much on the individual user when Harmony is started again, eg. 31 days after the first usage or 40 days.

However, the amount of data of each test user depends on the extent to which the software is used from the user.

Phase	*Users*	*Responses*					
		positive		negative		no reply	
First call	67	2	2,98%	9	13,44%	56	83,58%
Second call	67	7	10,44%	27	40,29%	33	49,25%
Summary	67	9	13,43%	36	53,73%	22	32,83%

Table 7.1: User responses during the two inquiry phases.

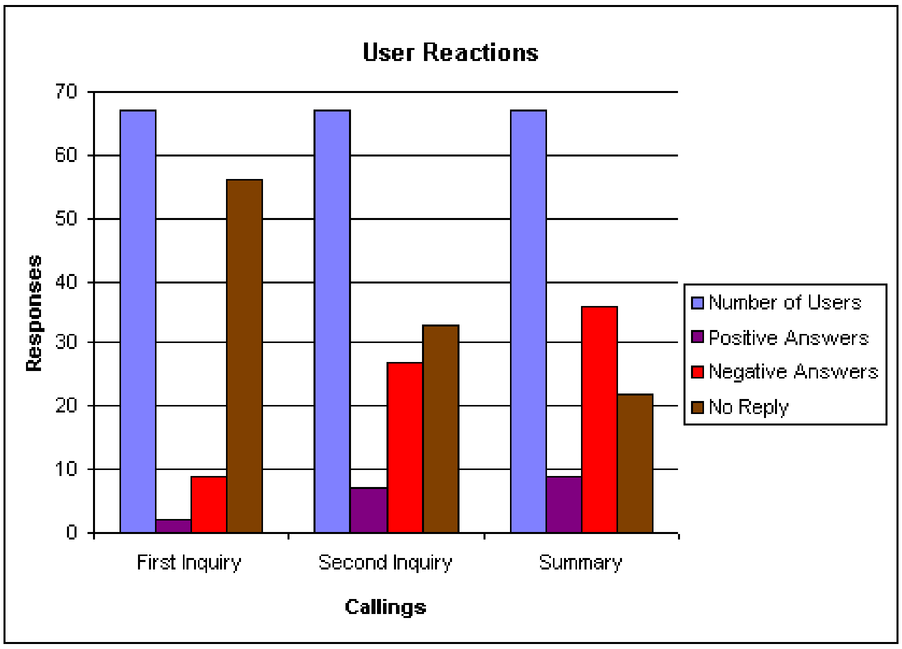


Figure 7.1: User reactions in two inquiry phases.

Reasons	*Responses*	
No Hyper-G or Harmony user left	5	10,64%
Use Wavemaster and Netscape	9	19,15%
Use Amadeus	4	8,51%
Use other Tools	4	8,51%
Platform not supported	1	2,13%
No time available	7	14,89%
Use shell scripts like hginfo	1	2,13%
No work on regular basis	7	14,89%
Use other Web Servers	1	2,13%
Host or user unknown	8	17,02%

Table 7.2: User responses during the two inquiry phases.

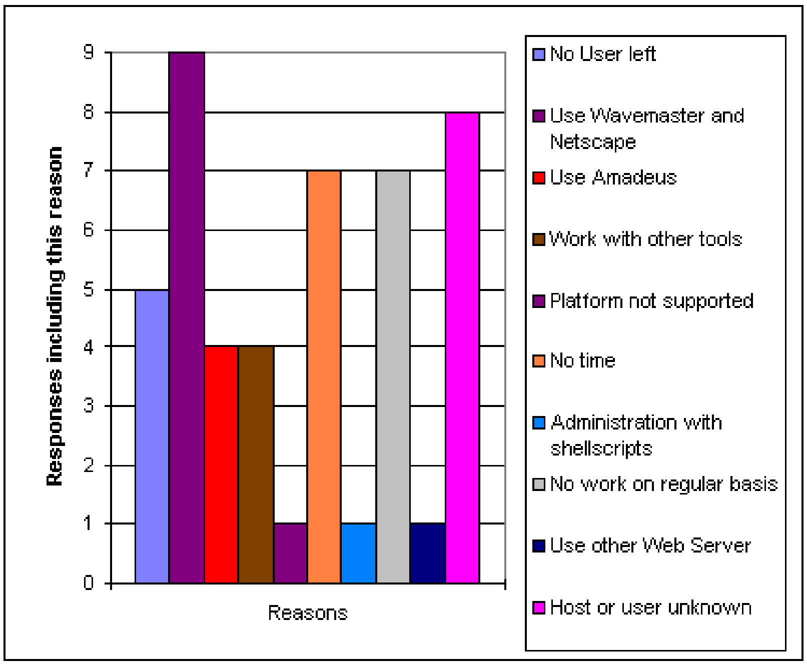


Figure 7.2: Reasons for not participating in the usability test.

Chapter 8

Results and Analysis

After the test phase was finished the analysis process was started. As mentioned before, a Perl script was written to support the analysis. However, the results, produced with the Perl script, must be interpreted manually for a detailed analysis.
For analysis purposes, the test users were divided into two main groups.

- Pilot test users
  The two users who participated in the pilot test.

- Real test users
  The test users who participated in the real test which run for a period of 30 days. This includes also users, who are members or students of the IICM and supported the testing process.

General trends of the working behavior of each of these user groups were analyzed and compared. During this analysis the group of real test users was not divided into users, who were associated with the IICM and users, who were not. The reason for this approach was that both groups participated in the test which run for a period of 30 days. In the first part of this chapter the results of the pilot test users are analyzed. The second part deals with the test results of the real test users.

8.1 Pilot Test

In this section the real test users are not considered. The following tables represent only the behavior of the two pilot test users.

8.1.1 Overall Statistics

Table 8.1, 8.2, and 8.3, show the number of recorded logfiles, testing period and usage of the instrumented viewers.

Logfiles	*Empty*	*Sessions*	*Unfinished*	*Actions*	*Time*
63	2	61	0	375	545146 sec

Table 8.1: Summary of all recorded logfiles of the pilot test.

Start of the test	*End of the test*
Mon 30 Dec 1997 14:56:47	Sat 15 Feb 1997 19:58:24

Table 8.2: Testing period of the pilot test.

The pilot test users executed 375 actions during the test period which means about 6 actions per session. It is an interesting fact, that all logfiles of the pilot test users were finished.

Table 8.3 displays all instrumented viewers including viewers, which were not used during the test. It can be seen that the pilot test users only used the *Session Manager, Text Viewer* and the *Image Viewer*. It is remarkable that two sessions with the *Image Viewer* were done but that in both sessions no command was issued. Probably, the viewer was used in this case only to display images. It is obvious that the pilot test users used only the *SessionManager, TextViewer* and the *ImageViewer*.

8.1.2 Used Functions

As shown in Table 8.4, the pilot test users used *14,57%* of all possible commands and a reason for that behavior might be that these test users used the instrumented version of Harmony only for a period of 14 days.

8.1.3 Searching Behavior of the Pilot Test Users

Harmony offers three ways to browse a Hyperwave server. One way is to use the search facilities like the search dialog. Other possibilities are to use hyperlinks or to use the collection tree. Regarding this, the habits of the test users were analyzed.
The search dialog of Harmony is logically divided into three parts.

- First part
  - Whole Database
  - Selected Collections
  - Active Collections
  - Result
- Second part
  - Title

Viewers	*Sessions*	*Time*	*Actions*
SessionManager	36	322745 sec	360
TextViewer	25	213727 sec	15
ImageViewer	2	8674 sec	0
3D Viewer	0	0 sec	0
FilmPlayer	0	0 sec	0
HarAdminTool	0	0 sec	0
EditAPITool	0	0 sec	0

Table 8.3: Viewers and their usage during the pilot test.

Used Actions	*Possible Actions*
22 14,57%	151

Table 8.4: Relation between used functions and not used functions during the pilot test.

  - Keyword
  - Name
  - Content

- Third part
  - Extended

This structure is used in Table 8.6. In Table 8.7 the usage of the search dialog of Harmony is analyzed in more detail.

It is remarkable that search operations over the whole database were performed not very often by the pilot test users. An explanation could be that the pilot test users were more familiar with the data lying on the server and the server structure itself. Therefore the pilot test users reduced the area for the search operations.

Table 8.7 shows that the pilot test users used several different combinations of parameters for search operations. It is also possible to change the language which is used for the search operation, but the pilot test users have not used this possibility.

8.2 Real Test

In the following sections the working behavior of the real test users is analyzed. However, a potential problem of the analysis was that there was no information available which datasets the test users were using (many links, no links, well-structured documents). Since it was too complicate to get dumps of all the server data, the collected test data represents only the interface components which were used by the test users.

Navigation		*Editing*		*Administration*		*Various Commands*	
11	22%	5	21,74%	2	9,52%	4	7,02%
Possible actions							
50		23		21		57	

Table 8.5: Used functions divided into the four main categories - pilot test.

Option	*Usage*
Whole Database	1
Selected Collections	29
Active Collections	0
Result	1
In Title	21
In Keyword	21
In Name	0
In Content	13
Extended	0

Table 8.6: Usage of the various options of the Search dialog of Harmony - pilot test.

8.2.1 Overall Statistics

First of all, an overview of the usage of Harmony and its viewers will be given. Table 8.8 shows the number of logfiles produced during the test period. The number of empty logfiles and unfinished Harmony sessions are also displayed in Table 8.8. Unfinished sessions are logfiles, which do not contain a line starting with `EndSession SessionManager`. This can happen when Harmony crashes. Table 8.8 shows also the number of actions executed during the test period and the amount of time spent working with Harmony. The real test users executed 18128 actions during the test period. Since 1158 sessions with Harmony were analyzed, about 16 actions per session were performed by the real test users. It is remarkable that 1,72% of the logfiles of the real test users were unfinished.

Table 8.10 displays all instrumented viewers including viewers, which were not used by the real test users. It can be possible that not all viewers were used from all users, since not all servers have the same structure and documents might not be available for all viewers. However, the real test users did make use of the commands offered from the *Image Viewer* as well. It is also noticeable that the additional tools, which were also instrumented, were not frequently used. The real test users preferred the *HarAdminTool* and did not use the *EditAPITool* very often.

Option	*Frequency*
Sel. Coll. + Title + Keyword	15
Sel. Coll. + Content	9
Sel. Coll. + Title + Keyword + Content	5
Whole Database + Result + Title + Keyword	1

Table 8.7: Search operations with parameters performed by the pilot test users.

Logfiles	*Empty*	*Sessions*	*Unfinished*	*Actions*	*Time*
1220	62	1158	21	18128	7782183 sec

Table 8.8: Summary of all recorded logfiles of the real test.

8.2.2 Functions Used and Not Used

Table 8.11 shows that *60,26%* of the commands, which are offered from the user interface of the program, were used by the test users, who participated in the real test over 30 days. A reason for this can be, that there is probably different data on the used Hyperwave servers, eg. on one server might be no data for the `Film Player`, whereas on another server such data might be available. Since the number of possible commands includes all instrumented viewers, it is understandable that less than *100%* of the 151 possible commands were used.

8.2.3 Most Used Features

The following Table 8.12, shows a list of 15 most frequently used commands. It is obvious that the first commands on the list are navigational commands, which emphasizes the usage of Harmony. It is a striking feature that the search command is not in the top five.
A noticeable fact is that the most used command was the `Navigate_Open_collection` command. It is also obvious that editing operations were done with commands from the `File` and `Edit` menus. It can be seen that the possibilities of the collection tree like `Open_collection, Open_document`, were used rather frequently. The `Activate_Hyperlink` command, which is mainly used to follow hyperlinks in documents, was used more often than the `Navigate_Search` command. Therefore the ranking as displayed in Table 8.12, implies that pilot and real test users used the software mainly to browse and search the server. However, the problem of this analysis is that it is simply not known which data is maintained on the used servers. If the documents lying on the server contain lots of hyperlinks the

Start of the test	*End of the test*
Tue 01 Apr 1997 08:35:30	Wed 28 May 1997 17:04:18

Table 8.9: Testing period of the real test.

Viewers	*Sessions*	*Time*	*Actions*
Real test users			
SessionManager	453	4140788 sec	14702
TextViewer	708	3116431 sec	2687
ImageViewer	39	363345 sec	67
3D Viewer	1	25 sec	0
FilmPlayer	0	0 sec	0
HarAdminTool	13	160408 sec	65
EditAPITool	6	1186 sec	15

Table 8.10: Viewers and their usage during the real test.

Used Actions	*Possible Actions*
91 60,26%	151

Table 8.11: Relation between used functions and not used functions during the real test.

usage of the `Activate_Hyperlink` command may probably increase. Therefore the ranking of Table 8.12 depends very much on the content of the server.

8.2.4 Detailed Classification of User Actions

Interface techniques used to execute actions

This subsection refers mainly to actions used for navigational purposes. Therefore the interface techniques used to execute these actions were analyzed. As shown in Figure 8.2 and displayed in Table 8.14, the actions `Open_collection` and `Open_document` and in Figure 8.4, the command `ActivateHyperlink` were mainly executed with mouseclicks. However, there is a possibility to start these commands with menu commands and keystrokes. This behavior of the test users could be estimated in Figure 8.3, Figure 8.5 and Figure 8.6 as well. Actions to move through the collection hierarchy like `Children, Goto, Home` or `CloseAll`, were mainly done using keyboard shortcuts. This suggests that these shortcuts are very intuitive. It can also be seen, that the test users used buttons to start actions, if such buttons were available. Examples are user actions like `Back, Forward, Search`. However, as depicted in Figure 8.4 the `LocalMap` was started only via keyboard shortcut and the button which was also available was not used. Regarding the search and history functionality as shown in Figure 8.5 and Figure 8.6, it can be seen that these actions were started with the buttons, but that the provided keyboard shortcut was also used.

Action Categories

The whole set of functions, which could be recorded with the instrumented version of Harmony, was divided into four main categories. This approach was taken since the goal of the analysis was to evaluate the working behavior of the test users. The most common

Ranking	*Type of Action*
1	Navigate_Open_collection
2	Navigate_Open_document
3	File_Edit
4	Navigate_Close
5	Anchors_Activate_Hyperlink
6	Navigate_Goto
7	Navigate_Children
8	System_Identify
9	Navigate_Search
10	Anchors_Define_As_Source
11	Edit_Attributes
12	Anchors_Define_As_Destination
13	Edit_Delete
14	Edit_Move
15	Edit_Copy

Table 8.12: Most commonly used commands.

ways to work with this kind of software are to use its navigational facilities or to use it as an authoring tool. Therefore to gain a more detailed view the following categorization was done.

- Navigation
  All user actions dealing with navigational commands like searching, following hyperlinks or browsing the collection tree. As shown in Figure 8.7, editing commands were the main part of actions, which were done by the real test users. Only **48%** of all user actions dealt with navigational commands. This fact indicates that the real test users used Harmony rather for editing tasks than for browsing or searching tasks.

  Figure 8.8, displays the navigation habits of the real test users. It shows that commands, which are dealing with the collection tree, like `Open_collection` or `Open_document`, and which can be executed with mouseclicks, were done more frequent than search commands or commands, dealing with hyperlinks. This fact is also depicted in Figure 8.2. A non linear scaling was used in Figure 8.2 to provide a more detailed view. This was done because of the great range of values, e.g. the command `Navigate_Open_collection` was used **6844** times whereas commands like `Navigate_Forward` were used only **8** times.

  These navigation habits of the test users suggest, that the concept of a graphical display of the server in this kind of tree structure, is very intuitive for the users. Probably, the fact that this structure is always displayed and therefore easy and quickly available, contributes as well to this behavior of the users. Table 8.14 implies that the number of search operations compared with hyperlink navigation and collection tree

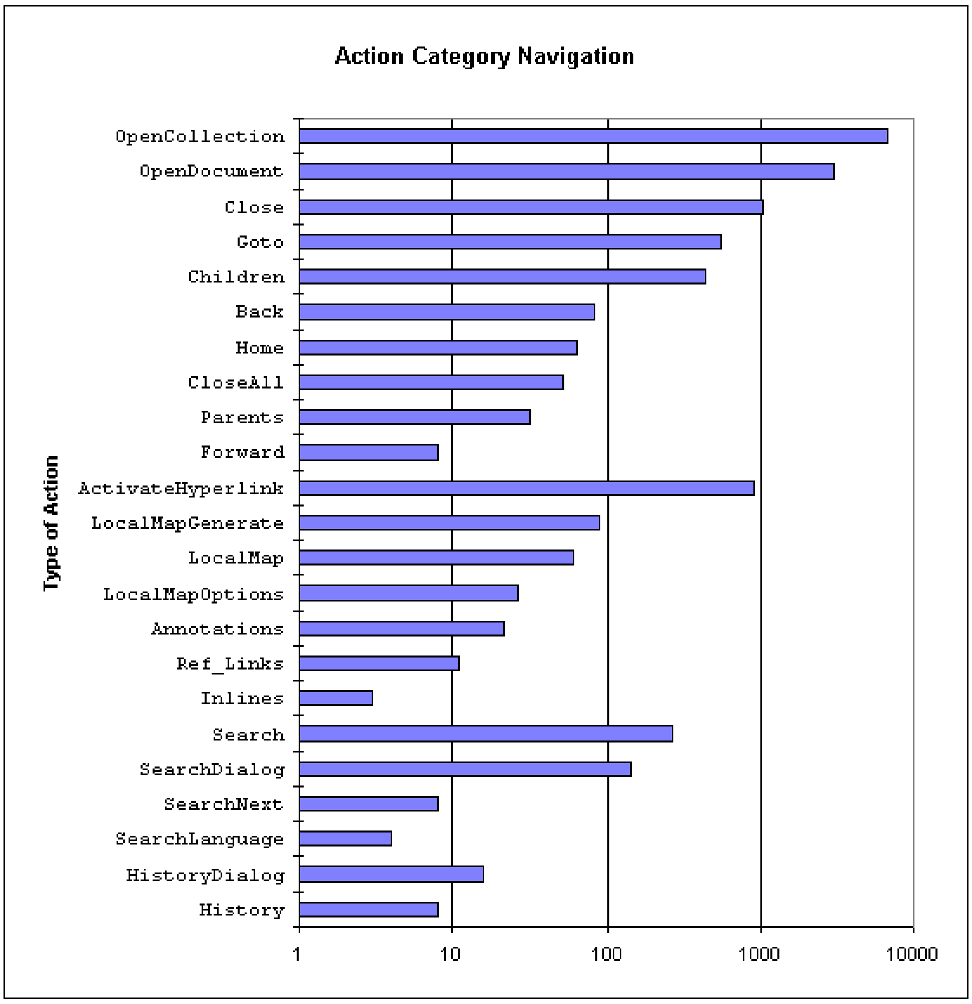


Figure 8.1: Distribution of user actions for the category Navigation.

navigation was relatively small. However, the analysis shows that the majority of the test users used the collection tree for browsing the server.

- Editing
  User actions used to work with documents like edit, move, delete documents and the *EditAPITool*, which is designed to work with documents in such a way. Table 8.15 shows that the real test users used editing commands like `File_Edit, Edit_Delete, Edit_Move` or `Edit_Copy` more often than the functions of the *EditAPITool*.

- Administration
  Commands of this category are system commands like identify and the *HarAdminTool*, which is used for user management. Table 8.16 displays these actions in more detail. As explained the functions of the *HarAdminTool* were not used very often. The most popular action in this category was the `System_Identify` command.

- Various Commands
  These are additional commands which are used to set various options. All actions displayed in Table 8.17 are commands which were used only to adjust the user interface. However, the `Clear_Cache` command was mainly executed by the system itself.

Relation between possible actions and used actions regarding the action categories

Table 8.13 shows the relation between the possible actions and the number of actions which were used by the test users. The number of possible actions represents the number of actions which can be protocolled with the instrumented version of Harmony. However, there are some actions, like pressing individual buttons in dialogs, which for technical reasons were not instrumented.

Navigation	*Editing*	*Administration*	*Various Commands*
24 48%	17 73,92%	14 66,67%	19 33,34%
Possible actions			
50	23	21	57

Table 8.13: Used functions divided into the four main categories.

The editing features of the interface were used rather frequently from the real test users. It can be seen that the users used more of the editing commands (*73,92%*) than of the navigational commands. However, the EditAPITool, which was designed for editing purposes, was not used very often.

8.2.5 Searching Behavior of the Real Test Users

The structure as shown in section 8.1.3, is used in Table 8.6. In Table 8.7 the usage of the search dialog of Harmony is analyzed in more detail. So called non-terminated sessions are also displayed in the table to show the general searching behavior of the test users. Non terminated sessions are Harmony sessions, which could not be finished correctly. In such cases, logfiles are produced as well. The options *Local Server, Selected Collections, In Title, In Keyword* are set per default. It can be seen that the real test users have not used the *Active Collections* and *In Name* options. Fulltext search, in Table 8.18 named as *In Content*, was used as well as the search for title. The option *Extended* was used 35 times to start a more detailed searching operation. However, it is obvious that some of the users simply had not changed the options in the search dialog since the number of search operations with the options *Selected Collections, In Title* and *In Keyword* is reasonably high. It is a conspicuous fact, that the real test users changed the predefined search options for nearly every search operation.

The language which is used for the search operation can also be changed with a button in the search dialog. The real test users preferred:

Type of Action	*DoubleClick*	*Button*	*Menu*	*Key*		*Total*
Collection Tree						
Navigate_Open_collection	6528	0	7	307		6844
Navigate_Open_document	2967	0	1	29		2997
Navigate_Close	0	0	0	1021		1021
Navigate_Goto	0	7	108	436		551
Navigate_Children	0	0	0	433		433
Navigate_Back	0	75	5	2		82
Navigate_Home	0	0	15	48		63
Navigate_CloseAll	0	0	26	25		51
Navigate_Parents	0	0	5	26		31
Navigate_Forward	0	7	0	1		8
Hyperlinks						
Anchors_Activate_Hyperlink	912	0	1	0		913
Navigate_LocalMapGenerate	0	82	0	6		88
Navigate_LocalMap	0	0	0	60		60
Navigate_LocalMapOptions	0	26	0	0		26
Navigate_Annotations	0	0	8	13		21
Navigate_Ref_Links	0	0	10	1		11
Navigate_Inlines	0	0	3	0		3
Search						
Navigate_Search	0	172	0	94		266
Navigate_SearchDialog	0	69	2	70		141
Navigate_Search_Next	0	8	0	0		8
Navigate_Search_Language	0	4	0	0		4
History						
Navigate_HistoryDialog	0	11	1	4		16
Navigate_History	0	0	8	0		8

Table 8.14: Details of all actions used for navigational purposes.

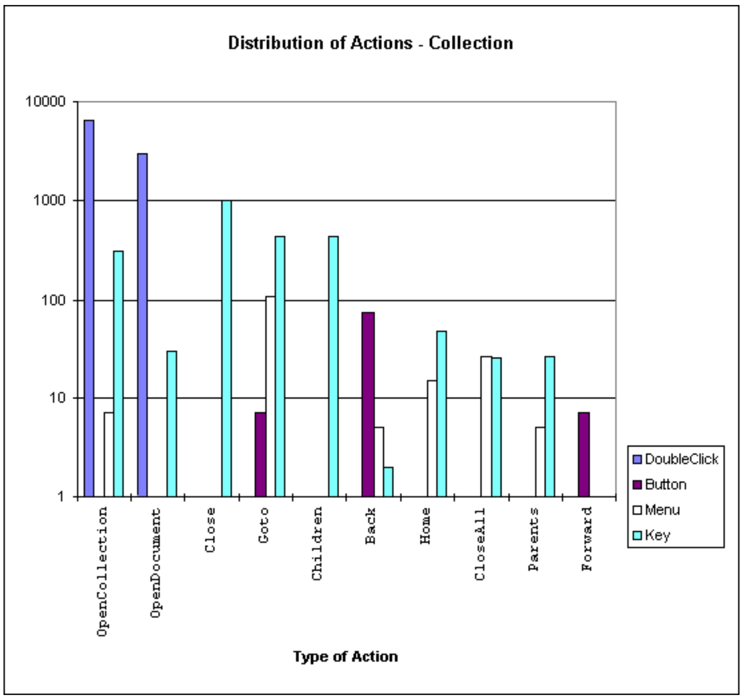


Figure 8.2: Distribution of user actions for the category Collection.

Type of Action	*DoubleClick*	*Button*	*Menu*	*Key*		*Total*
File_Edit	0	0	1083	0		1083
Anchors_Define_As_Source	0	0	252	0		252
Edit_Attributes	0	0	0	238		238
Anchors_Define_As_Destination	0	0	209	0		209
Edit_Delete	0	0	0	203		203
Edit_Move	0	0	0	193		193
Edit_Copy	0	0	57	99		156
Anchors_Define_As_Inline_Source	0	0	87	0		87
Anchors_Delete	0	0	73	1		74
File_Insert	0	0	0	54		54
File_Save_As	0	0	27	0		27
APITool_Quit	0	6	0	0		6
APITool_Edit	0	5	0	0		5
APITool_Insert	0	3	0	0		3
Anchors_Use_Default_Destination	0	0	2	0		2
APITool_Cancel	0	1	0	0		1

Table 8.15: Details of all actions used for editing purposes.

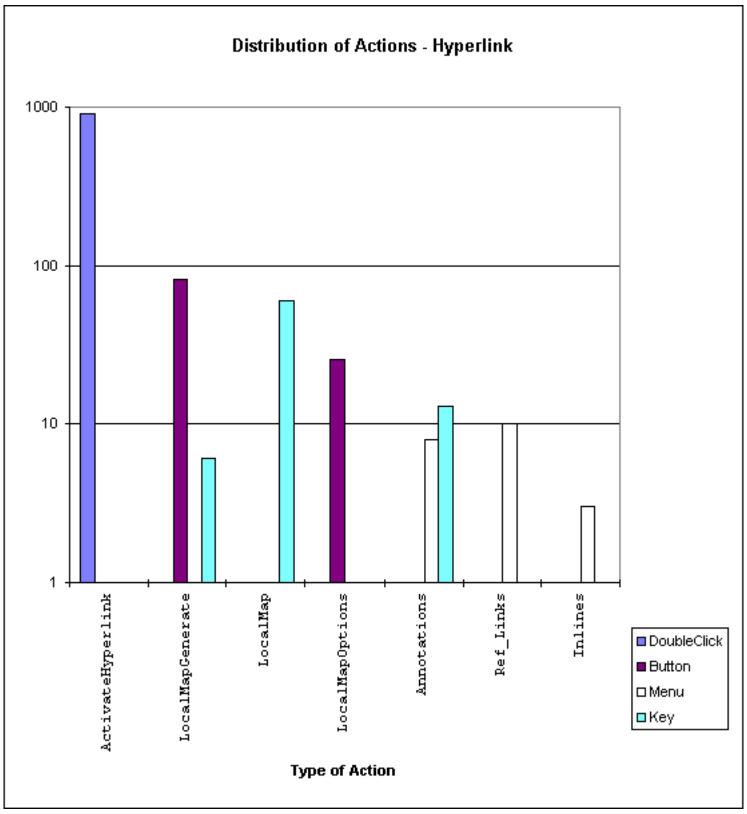


Figure 8.3: Distribution of user actions for the category Hyperlinks.

Type of Action	*DoubleClick*	*Button*	*Menu*	*Key*		*Total*
System_Identify	0	0	201	196		397
System_Status	0	0	18	6		24
System_StatusBrowser_SendMessage	0	19	0	0		19
HarAdmin_Options_Users	0	15	0	0		15
HarAdmin_Edit_Users	0	11	0	0		11
HarAdmin_Users_Identify	0	10	0	0		10
HarAdmin_ShowAll_Users	0	8	0	0		8
HarAdmin_New_Users	0	6	0	0		6
HarAdmin_Options_Groups	0	6	0	0		6
HarAdmin_New_Groups	0	3	0	0		3
HarAdmin_Edit_Groups	0	3	0	0		3
HarAdmin_Show_Groups	0	3	0	0		3
System_Change_Password	0	0	1	0		1
System_StatusBrowser_DeselectAll	0	1	0	0		1

Table 8.16: Details of all actions dealing with system and administrative concerns.

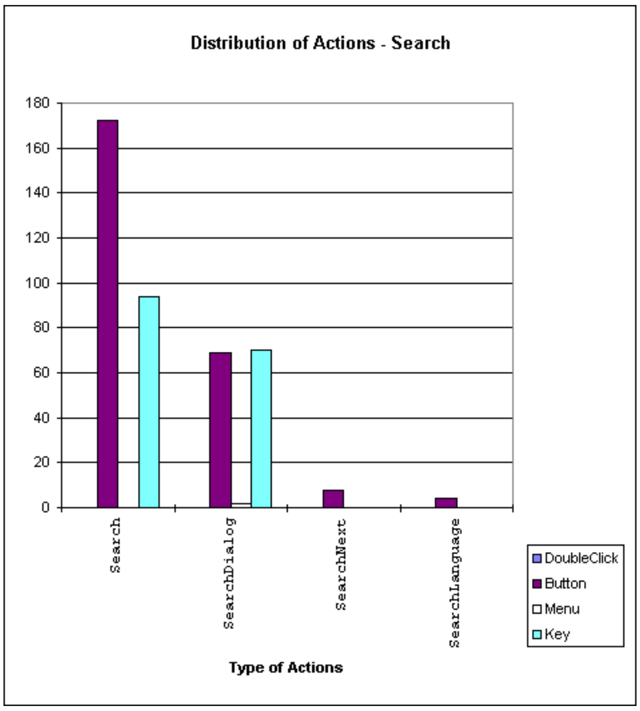


Figure 8.4: Distribution of user actions for the category Search.

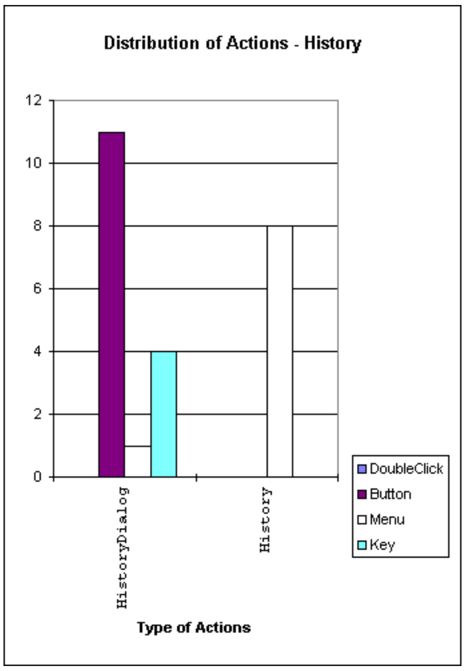


Figure 8.5: Distribution of user actions for the category History.

Type of Action	*DoubleClick*	*Button*	*Menu*	*Key*		*Total*
System_Clear_Cache	7	881	59	12		959
View_Attributes	0	0	0	150		150
File_View_Source	0	0	82	0		82
Display_Show_Anchors	0	35	0	0		35
Options_Language	0	30	0	0		30
Navigate_Attributes	0	0	25	0		25
View_ShowCount	0	0	24	0		24
Navigate_Hold	0	0	21	0		21
Display_Fonts	0	0	16	0		16
Display_AutoFit	0	11	0	0		11
Display_Color_Anchor	0	0	6	0		6
Display_ZoomOut	0	6	0	0		6
File_Mail_SendMessage	0	0	6	0		6
Display_ZoomIn	0	4	0	0		4
View_ShowHost	0	0	0	2		2
Anchors_Shape	0	0	2	0		2
Display_Text_SelectColor	0	0	1	0		1
Options_General_Global_Viewer	0	0	1	0		1
Display_Anchor	0	0	1	0		1

Table 8.17: Details of all actions used to set various options.

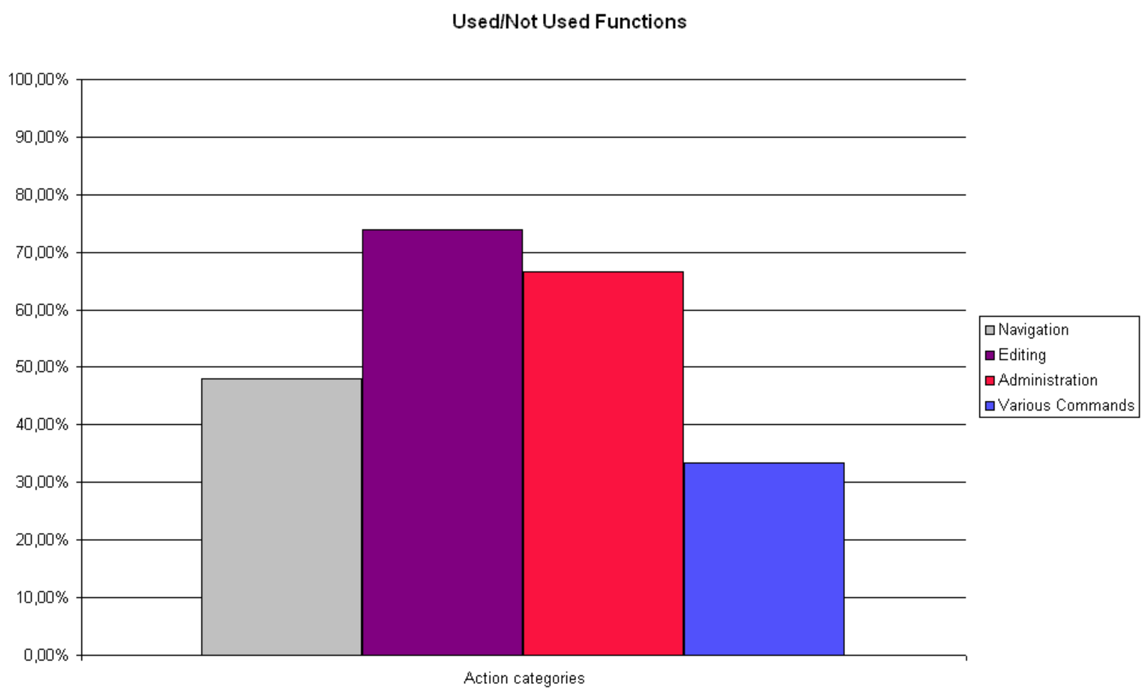


Figure 8.6: Distribution of user actions for all categories.

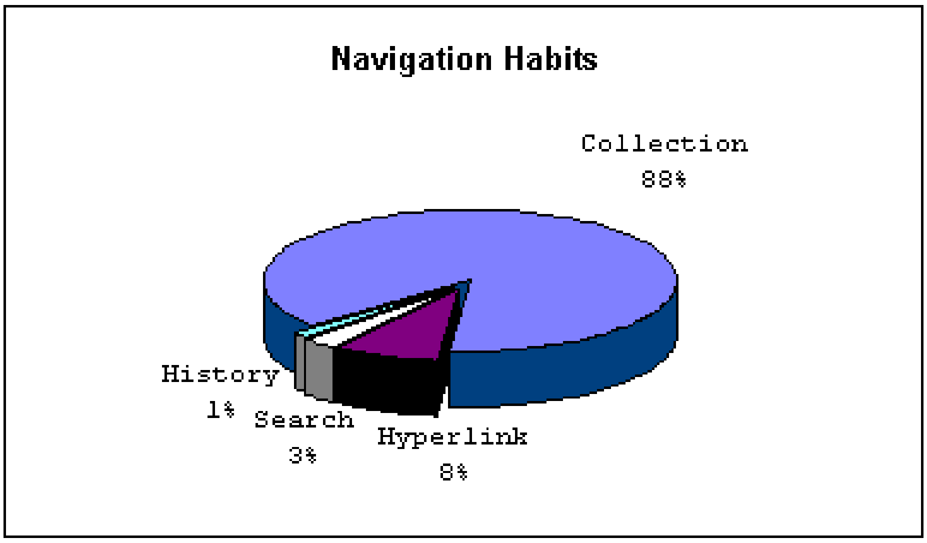


Figure 8.7: Navigation habits of the real test users.

Option	*Usage*	*Incl. Non-terminated sessions*
Whole Database	63	76
Selected Collections	153	156
Active Collections	0	0
Result	9	9
In Title	174	187
In Keyword	149	152
In Name	0	0
In Content	101	104
Extended	35	35

Table 8.18: Search dialog of Harmony including non-terminated sessions.

- German
- English

The actual search language was changed 4 times using the language button in the search dialog of Harmony.

Option	*Usage*
Whole Database	63
Selected Collections	153
Active Collections	0
Result	9
In Title	174
In Keyword	149
In Name	0
In Content	101
Extended	35

Table 8.19: Usage of the various options of the Search dialog of Harmony.

Table 8.19 shows that the real test users performed 35 detailed searching operations with the *Extended* option. Another interesting aspect is that the real test users used the possibility to start another search based on the result of a previous one only 9 times.

Table 8.20 shows that the most popular combination of parameters in search operations was *Selected Collection + Title + Keyword*. Table 8.20 shows also that searches including *Title* and *Fulltext* were used very often. It is noticeable that the real test users used several different combinations of parameters for search operations.

Option	*Frequency*
Real test users	
Sel. Coll. + Title + Keyword	37
Sel. Coll. + Title + Keyword + Content	20
Whole Database + Result + Title	3
Sel. Coll. + Title + Keyword + Content + Extended	3
Whole Database + Result + Title + Extended	2
Sel. Coll. + Title	2
Whole Database + Sel. Coll. + Result + Title	2
Sel. Coll. + Title + Content	1
Whole Database + Title + Result + Content + Extended	1
Sel. Coll. + Title + Content + Extended	1
Whole Database + Sel. Coll. + Result + Title + Content	1

Table 8.20: Search operations with parameters performed by the real test users.

8.2.6 Error Messages

The real test users encountered 18 different error messages during the test. In total 35 error messages were encountered from the test participants in the real test. However, the contents of the list depends on the data which is lying on the server.

- ImageViewer: dgif(1997/05/01 - 1997/05/01) connection closed
- WARNING: This object is only member of one collection! Object will be deleted! Continue anyway?
- Identify not successful!
- Collection not found!
- Prior request not finished
- Warning: object has no title! Proceed anyway?
- Warning: generic object has no subtype! Proceed anyway?
- Can't lock Object access denied
- Insert Document access denied
- Insert Document entry in name-field not unique
- Could not connect to host
- Pipe Document:Document Cache error: Cannot read from local store
- Text Viewer: About This Server connection closed

- Message from: hgsystem(system) DBServer stopped
- Server not responding
- No Previous Version
- Access denied
- File is a Directory!
- Collection does not exist, or is not accessible

8.2.7 Sequences Including Error Messages

Tables 8.21, 8.22,and 8.23 represent sequences of user actions, which include one or more error messages. The tables contain information like

1. The action which was done before the error occurred
2. The error message
3. The action which was done after the error message was received

During the analysis process it was found out that more than one such sequence occurred during a session with Harmony. Examples for such situations are shown in Table 8.21, sequence 1, Table 8.22, sequence 3–6, and Table 8.21, sequence 7. In these cases a greater part of the logfile was displayed in the table to show the whole situation where the errors occurred. The analysis showed that there were three main ways how the individual test users dealt with error messages. It is noticeable, that the individual test users, after they have received an error message,

- simply finished their Harmony session as shown in Table 8.21, or
- tried to identify themselves, as shown in Table 8.22, or
- continued working with the `Navigate_Open_collection` command, see Table 8.23

Other users, who encountered the following errors

- Collection not found!
- Identify not successful!
- Can't lock object access denied
- Collection does not exist, or is not accessible

tried to identify themselves using the `System_Identify` command. Examples for this behavior are shown in Table 8.22.
However, the reason for finishing the session was not due lack of understanding the error message.

No	*Actions Before*	*Error Sequence*	*Actions After*
s1	**startviewer** StartSession_-Harmony	**errormessage** Text Viewer: About This Server connection closed **dclick** Navigate_Open_document **dclick** Navigate_Open_collection **errormessage** Message from: hgsystem(system) DBServer stopped **dclick** Navigate_Open_collection	**endviewer** EndSession_-Harmony
s2	**dclick** Navigate_-Open_-document	**errormessage** Pipe Document:Document Cache error: Cannot read from local store	**endviewer** EndSession_-Harmony

Table 8.21: User reaction after an error message - finish Harmony session.

No	*Actions Before*	*Error Sequence*	*Actions After*
s1	**menu** Edit_Copy	**errormessage** Collection does not exist, or is not accessible	**menu** System_Identify
s2	**key** System_Identify	**errormessage** Identify not successful!	**key** System_Identify
s3	**key** Navigate_Home	**errormessage** Collection not found! **menu** System_Identify **errormessage** Identify not successful!	**key** Navigate_-Open_-collection
s4	**menu** System_Identify	**errormessage** Identify not successful! **menu** System_Identify	**menu** Navigate_-CloseAll
s5	**key** Navigate_Home	**errormessage** Collection not found! **key** System_Identify **key** Navigate_Home **dclick** Navigate_Open_collection **errormessage** Prior request not finished	**dclick** Navigate_-Open_-collection
s6	**key** File_Insert	**errormessage** Insert Document entry in name-field not unique **key** Edit_Delete **key** System_Identify **errormessage** Identify not successful! **errormessage** Could not connect to host **key** System_Identify **errormessage** Identify not successful! **dclick** Navigate_Open_collection **errormessage** Prior request not finished	

Table 8.22: User reaction after an error message - try to identify.

No	*Actions Before*	*Error Sequence*	*Actions After*
s1	**dclick** Navigate_-Open_-document	**errormessage** ImageViewer: dgif(1997/05/01 - 1997/05/01) connection closed	**dclick** Navigate_-Open_-collection
s2	**dclick** Navigate_-Open_-collection	**errormessage** WARNING:This object is only member of one collection! Object will be deleted! Continue anyway?	**dclick** Navigate_-Open_-collection
s3	**dclick** Navigate_-Open_-collection	**errormessage** No Previous Version	**dclick** Navigate_-Open_-collection
s4	**key** File_Insert	**errormessage** Insert Document access denied	**dclick** Navigate_-Open_-collection
s5	**dclick** Navigate_-Open_-collection	**errormessage** Access denied	**dclick** Navigate_-Open_-collection
s6	**dclick** Navigate_-Open_-collection	**errormessage** File is a Directory!	**menu** Anchors_-Define_As_-Destination
s7	**key** File_Insert	**errormessage** Warning:object has no title! Proceed anyway? **errormessage** Warning:generic object has no subtype! Proceed anyway?	**dclick** Navigate_-Open_-collection

Table 8.23: User reaction after an error message - continue working.

Chapter 9

Concluding Remarks

This thesis presented a usability field study of Harmony, the Unix-based client for the Hyperwave web server. A technique called *software instrumentation* was used. For this technique, a specially augmented version of the software was created. The goal of this method is to create logfiles of actual use of the software over a longer period without interfering with the test users. As a first step of the study such an instrumented version of Harmony was created. This version was tested and improved with internal test users. Internal test users were members and students of the IICM. After this pre-testing and development phase was concluded a short analysis process was done. The purpose of these pre-tests was to estimate the amount of data of each user. The analysis was performed with a Perl script and the purpose was to see if the behavior of the test users was the same as expected. It was expected that the navigational methods of the users might be that about 80% of them use the collection hierarchy, 10% use hyperlink navigation and 10% use the search facilities.

The next step in the study was to perform pilot tests to assure the functionality of the instrumented version of Harmony and to test the data transmission process. As a result of these pilot tests the data transmission process had to be changed. Finally, a user database was established and the real test was started. A period of 30 days was scheduled for the test. A survey was done during establishing of the user database. Many users of Hyperwave could not participate in the study because they used Netscape rather than Harmony.

The concluding step of the study was the analysis of the received logfiles. The Perl script, which was used before for the short analysis, was refined for this purpose and the results of the script were analyzed manually for a detailed analysis. A remarkable result was that less than 50% of all instrumented functions were used. About 75% of the used actions dealt with navigational actions and 14% were used for editing purposes. These facts characterized the way how the users used Harmony. It is also noticeable that 88% of the navigational actions dealt with the collection hierarchy. These navigational habits of the test users suggest that the concept of a graphical display of the hierarchical structure of information on the server is very intuitive for the users. The fact that this structure is always visible and therefore easily and quickly available, contributes as well to this behavior of the users. Hyperlink navigation (7%) and searching (4%) were used much less frequently.

Most of the test users used the mouse for navigation (doubleclicks), but they used other interface techniques like menus, buttons and shortcuts, if these techniques were available as an alternative, as well. One remarkable result was also that 50,68% of the users who used the search dialog did not change the predefined options in the search dialog at all. The pilot test users showed the same behavior. It is also interesting, that the possibility to change the language of the search as of the Graphical User Interface (GUI) was hardly used.

Test users, after receiving an error message, either finished their Harmony session, tried to identify themselves or simply continued working and ignored the error. The error messages were obviously understandable, because it was not observed that users received the same error message more than once during a session.

Chapter 10

Outlook

Software instrumentation is a very powerful technique for performing usability field studies of software. The concept of recording users actions and analyzing the logfiles can be used in every kind of software which sports a graphical user interface. The creation of instrumented software is time-consuming, but once there is such a instrumented version available, the testing process is very simple. For analysis purposes it is necessary to write tools which support the analysis process. However, it is necessary to manually analyze the results of such tools to get a more detailed analysis of the behavior of the test users. The results of such a field study with instrumented software can be used for further improvements of the software or for further usability testing. For instance, a list of most used commands can be established, tasks can be designed for these commands and used for usability testing.

A very important aspect is to get in touch with the development team and to establish a connection. This is especially useful, because of changes in the implementation process. One of the most difficult tasks after the creation of the instrumented version is to estimate the amount of data produced by each test user. A proper approach to this problem might be to create short data files, eg. one data file per session. This approach was taken in this thesis. The problem of transferring the data files can be solved with the use of FTP or email. The idea to implement a data transfer utility based on TCP/IP to transfer the dat files over the Internet is not so practicable, because of firewalls and network bandwidth problems. If a user database is not already available it can be very hard to find test users. Therefore it is a good method to look for test users in the first stages of the development process, eg. shortly after the first test version is implemented.

One potential problem of software instrumentation is the “big brother” effect[Nie93], since it is possible to misuse the concept of software instrumentation. About one year ago a company released a software tool which run under Windows95 and WindowsNT platforms and recorded the caption of the active window, the number of keystrokes, total distance of mouse travel, number of mouse clicks and the URL if a browser is used. The danger of this program was that it could be set up so that it would not pop up or even show up on the task bar. Therefore a regular Windows95 or WindowsNT user would not know that this tool is running in the background and collecting data. Therefore a company, which uses this tool can observe its employees and their productivity. This is a severe abuse of privacy and employee rights.

Appendix A

Perl Scripts used for Logfile Transmission

A.1 Prepare.pl

```
eval '(exit $?0)' && eval '[ -f /usr/local/bin/perl ] && exec
/usr/local/bin/perl -S $0 ${1+"$@"}; exec perl -S $0 ${1+"$@"};'
& eval 'if ( -f /usr/local/bin/perl ) exec /usr/local/bin/perl -S $0
$argv:q ; exec perl -S $0 $argv:q'
     if 0;


($d_path) = @ARGV;
$isok = 0;

($isok = 1) if -x("/usr/local/bin/gzip");

sub prepare_to_send_logs {
  $username = $ENV{'USER'};
  $userdir = $ENV{'HOME'};
  chdir($d_path);

  $bef1 = "tar cvf $username.logfiles.tar * \n";
  system $bef1;
  if ($isok == 1) {
     $bef1 = "gzip $username.logfiles.tar \n";
  }
  else {
     $bef1 = "compress $username.logfiles.tar \n";
  }
  system $bef1;
  if ($isok == 1) {
    $dummy = "$username.logfiles.tar.gz.uue";
    $name = "$username.logfiles.tar.gz";
  }
  else {
    $dummy = "$username.logfiles.tar.Z.uue";
    $name = "$username.logfiles.tar.Z";
  }
  $bef = "uuencode $name <$name >$dummy\n";
  system $bef;

  $bef9 = "mv $dummy $userdir/$dummy \n";
  system $bef9;

  $beff = "rm  -r *\n";
  system $beff;

  $beff1 = "rm .start-time\n";
  system $beff1;
}

&prepare_to_send_logs();
```

A.2 Client.pl

```
eval '(exit $?0)' && eval '[ -f /usr/local/bin/perl ] && exec
/usr/local/bin/perl -S $0 ${1+"$@"}; exec perl -S $0 ${1+"$@"};'
& eval 'if ( -f /usr/local/bin/perl ) exec /usr/local/bin/perl -S $0
$argv:q ; exec perl -S $0 $argv:q'
     if 0;

$username = $ENV{'USER'};
$userdir = $ENV{'HOME'};
chdir($userdir);

$dummy = "$username.logfiles.tar.gz.uue" if -e("$username.logfiles.tar.gz.uue");
$dummy = "$username.logfiles.tar.Z.uue" if -e("$username.logfiles.tar.Z.uue");
$size = -s($dummy);


sub transfer_logs {
  open(TAR,"$dummy") || die "Sorry - tar-archive could not be opened -
        Try : >prepare.pl ~/.harmony-log and then start manually sendlogs.pl \n";
  $trans = "mail iegger\@iicm.edu < $dummy \n";
  system $trans;
  printf "File $dummy was send to iegger\@iicm.edu \n";
  $tempname = "$userdir/.harmony-log/.transmission-info";
  open(TRANS,">$tempname") || die "Sorry - Could not create
                                   file ~/.harmony-log/.transmission-info\n";
  printf TRANS "999";
  close(TRANS);
}
```

```
&transfer_logs();
```

A.3 Sendlogs.pl

```
eval '(exit $?0)' && eval '[ -f /usr/local/bin/perl ] && exec
/usr/local/bin/perl -S $0 ${1+"$@"}; exec perl -S $0 ${1+"$@"};'
& eval 'if ( -f /usr/local/bin/perl ) exec /usr/local/bin/perl -S $0
$argv:q ; exec perl -S $0 $argv:q'
     if 0;

($u_path) = @ARGV;
$port = 2345 unless $port;
$them = 'iicm.tu-graz.ac.at' unless $them;

use Socket;

$username = $ENV{'USER'};
$userdir = $ENV{'HOME'};
chdir($userdir);

$l_up = length($u_path);
if ($l_up > 2) { chdir($u_path); }

($dummy = "$username.logfiles.tar.gz.uue") if -e("$username.logfiles.tar.gz.uue");
($dummy = "$username.logfiles.tar.Z.uue") if -e("$username.logfiles.tar.Z.uue");

$trans = "mail iegger\@iicm.edu < $dummy \n";
system $trans;
printf "File $dummy was send to iegger\@iicm.edu \n";
```

A.4 Server.pl

```
eval '(exit $?0)' && eval '[ -f /usr/local/bin/perl ] && exec
/usr/local/bin/perl -S $0 ${1+"$@"}; exec perl -S $0 ${1+"$@"};'
& eval 'if ( -f /usr/local/bin/perl ) exec /usr/local/bin/perl -S $0
$argv:q ; exec perl -S $0 $argv:q'
     if 0;

($destination_path,$them,$port) = @ARGV;
$port = 2345 unless $port;
$them = 'localhost' unless $them;

$lang = length("FinishOfDataTransmissionProtocolTag\n");
```

```
use Socket;

$l_up = length($destination_path);
if ($l_up > 2) { chdir($destination_path); }

$sockaddr = ´S n a4 x8´;

($name,$aliases,$proto) = getprotobyname(´tcp´);
if ($port !~ /^\d+$/) {
  ($name,$aliases,$port) = getservbyport($port,´tcp´);
}

$this = pack($sockaddr, AF_INET, $port, "\0\0\0\0");

select(NS); $| = 1; select(STDOUT);

socket(S, PF_INET, SOCK_STREAM, $proto) || die "socket: $!";
bind(S, $this) || die "bind: $!";
listen(S, 5) || die "connect: $!";

select(S); $| = 1; select(STDOUT);

for($con = 1; ; $con++) {
##    printf("Listening for connection %d .....\n",$con);
    ($addr = accept(NS,S)) || die $!;

    if (($child = fork()) == 0) {

        $actual = 0;
        $name = <NS>;
        $bytes_to_receive = <NS>;
        chop($bytes_to_receive);
        open(TAR,">$name") || die "Sorry \n";
        while (<NS>) {
          last if (/FinishOfDataTransmissionProtocolTag$/);
          print TAR $_;
          $actual = $actual + length($_);
        }
        close(TAR);
##        $actual = $actual - $lang;
        print NS "$actual\n";
        close(NS);
        exit;
    }
    close(NS);
}
```

Bibliography

[And96] Keith Andrews. Information systems and the internet. In Hermann Maurer, editor, *HyperWave: The Next Generation Web Solution*, chapter 2, pages 9–18. Addison-Wesley, May 1996. http://www.iicm.edu/hgbook.

[CP95] L.D. Catledge and J.E. Pitkow. Characterizing browsing strategies in the world-wide web. In *Computer Networks and ISDN Systems*, volume 27, pages 1065–1073, 1995.

[CR91] J.M. Carroll and M.B. Rosson. Deliberated evolution: Stalking the view matcher in design space. In *Human-Computer Interaction*, volume 6, 3 and 4, pages 281–318, 1991.

[CRM91] S.K. Card, G.G. Robertson, and J.D. Mackinlay. The information visualizer: An information workspace. In *Proc. CHI'91 Conf.*, pages 181–188, New Orleans, April 1991. ACM.

[CW88] J.F. Cove and B.C. Walsh. Online text retrieval via browsing. *Information Processing and Management*, 24(1):31–37, 1988.

[DR93] J.S. Dumas and J.C. Redish. *A Practical Guide to Usability Testing.* Ablex, 1993.

[ERG+89] D.E. Egan, J.R. Remde, L.M. Gomez, T.K. Landauer, J. Eberhardt, and C.C. Lochbaum. Formative design-evaluation of superbook. In *Transactions on Information Systems*, volume 7, pages 30–57. ACM, January 1989.

[Gru90] J. Grudin. Obstacles to user involvement in interface design in large product development organizations. In *Proc. INTERACT'90 Third Intl. Conf. Human-Computer Interaction*, pages 219–224, Cambridge, August 1990. IFIP.

[Gru91a] J. Grudin. Interactive systems: Bridging the gaps between developers and systems. In *IEEE Computer*, volume 24, pages 59–69, April 1991.

[Gru91b] J. Grudin. Systematic sources of suboptimal interface design in large product development organizations. In *Human-Computer Interaction*, volume 6, 2, pages 147–196, 1991.

[Mau96] Hermann Maurer. Hyper-g: How does it differ from www and gopher? In Hermann Maurer, editor, *HyperWave: The Next Generation Web Solution*, chapter 1, pages 2–8. Addison-Wesley, May 1996. http://www.iicm.edu/hgbook.

[Mil68] R.B. Miller. Response time in man-computer conversational transactions. In *Proc. Spring Joint Computer Conference*, volume 33, pages 267–277. AFIPS, 1968.

[MS88] J.E. McDonald and R.W. Schvaneveldt. The application of user knowledge to interface design. In Lawrence Erlbaum Associates, editor, *Cognitive Science and its Applications for Human-Computer Interaction*, pages 289–338, Hillsdale, 1988.

[Nie93] Jakob Nielsen. *Usability Engineering.* Academic Press, London, 1993.

[Rei88] P. Reitman. Streamlining your documentation using quick references. In *IEEE Trans. Professional Communication*, volume 31, pages 75–83, June 1988.

[Ric91] J.F. Rice. Display color coding: 10 rules of thumb. In *IEEE Software*, volume 8, pages 86–88, January 1991.

[Sch88] K. Schmidt. Functional analysis instrument. In G. Schaefer, R. Hirschheim, M. Harper, R. Hansjee, M. Domke, and N. Bjoern-Andersen, editors, *Functional Analysis of Office Requirements - A Multiperspective Approach*, pages 261–289, Chichester, 1988.

[Shn82] Ben Shneiderman. Designing computer system messages. In *Communication of the ACM*, volume 25, pages 610–611. ACM, September 1982.

[Tra91] D. Travis. *Effective Color Displays: Theory and Practice.* Academic Press, London, 1991.

[WBH88] J. Whiteside, J. Bennett, and K. Holtzblatt. Usability engineering: Our experience and evolution. In M. Helander, editor, *Handbook of Human-Computer Interaction*, pages 791–817, Amsterdam, 1988.

[Wik94] M.E. Wiklund. *Usability in Practice.* Academic Press, Boston, 1994.

[Wri83] P. Wright. Manual dexterity: A user-oriented approach to creating computer documentation. In *Proc. CHI'83 Conf.*, pages 11–18, Boston, December 1983. ACM.

[Wri91] P. Wright. Designing and evaluating documentation for i.t. users. In B. Shackel and S. Richardson, editors, *Human Factors for Informatics Usability*, pages 343–358, Cambridge, 1991. Cambridge University Press.