Interacting with the Web

Brian R Gaines and Mildred L G Shaw
Knowledge Science Institute
University of Calgary
Calgary, Alberta, Canada T2N 1N4
{gaines, mildred}@cpsc.ucalgary.ca

ABSTRACT
Widespread access to the Internet has led to a new era of distributed client-server computing. In particular, the introduction of support for *forms* on World-Wide Web in late 1993 has provided an easily programmable, cross-platform graphic user interface that has become widely used in innovative interactive systems. This article analyzes the human factors of WWW client-server applications in terms of both general principles and the constraints of existing technology. It gives examples of innovative applications that illustrate the issues involved. Currently the technology is characterized by the limited functionality of clients, communication delays in interacting with the server, and lack of incremental updatiug at the iuterface. These constraints limit interactivity in a way that dominates usability issues. The article develops system design guidelines to minimize the impact of these limitations, and proposes developments to improve interactivity.

KEYWORDS: Internet, world-wide web, client-server architectures, active documents, human factors, usability.

INTRODUCTION
Advances in computing technology that generate new opportunities for interactive computing generally also introduce new human factors considerations. In the 1980s the major new technology was the personal computer with graphic interaction leading to the *direct manipulation* paradigm [32]. In the 1990s it is the interconnection of such personal computers world-wide through the Internet that is leading to innovation through client-server distributed interaction. New paradigms for *groupware* and *agent-based computing* are some well-known outcomes. However, there are other impacts of the partitioning of interactive functionality between client and server that are less obvious, equally momentous, and highly relevant to the support of the groupware and agent paradigms.

The explosive growth of the World-Wide Web (WWW) [6] in the 1990s is a significant example of new trends in human-computer interaction evolving through grass-roots development world-wide. In particular, WWW *forms* support a graphic user interface (GUI) that can be simply programmed as an interactive document. Anyone who can use a word processor can become a user interface designer. The embedding of user interface elements within a document in itself promotes a new approach to interaction in which the interface emerges from the content and context. The flexibility of a dynamically programmed *emergent interface* may itself constitute a new paradigm for human-computer interaction in the 1990s.

The next section gives an overview of WWW, the html document type definition, the http communication protocol, and the model of client-server computing that results. The following section exemplifies WWW functionality through a range of applications that illustrate the human factors issues involved. Finally, usability guidelines are developed that address the capabilities and limitations of the current technology, followed by proposals for future development of client-server computing on the Internet.

WORLD-WIDE WEB
World-Wide Web was conceived by Berners-Lee in March 1989 [10] as a "hypertext project" to organize documents at CERN in an information retrieval system [5]. The design involved: a simple hypertext markup language that authors could enter through a word processor; distributed servers running on machines anywhere on the network; and access through any terminal, even line mode browsers. WWW today still conforms to this basic model.

A poster and demonstration at HT91 in December 1991 announced WWW to the computing community. However, major usage only began to grow with the February 1993 of Andreessen's Mosaic for X [1]. Whereas the original WWW proposal specifically states it will not aim to "do research into fancy multimedia facilities such as sound and video" [5], the http protocol for document transmission was designed to be content neutral and as well-suited to multimedia material as to text. The availability of the rich X environment on workstations supporting color graphics and sound led naturally to multimedia support, although the initial objective of meaningful access through any terminal was retained. Most web material can still be browsed effectively through a line mode browser.

Figure 1 shows a typical WWW application providing access to a hypermedia document collection. A document has been fetched from a remote server across the Internet. It appears on the screen, and can be printed, with the typography, layout, and embedded colored diagrams expected of a high-quality document processor. It supports embedded hypertext links which can be used to access other documents across the net.

Figure 1 Hypermedia document in World-Wide Web

For example, the cursor at the top of the page is over the underlined word "LINCS" and the status line at the bottom of the screen shows this as a hypertext link to "http://ksi.cpsc.ucalgary.ca/LINCS/ LINCS.html". Clicking on this word fetches the document LINCS.html in the directory /www/LINCS on the machine whose Internet address is ksi.cpsc.ucalgary.ca using the http protocol. The entire string constitutes a *uniform resource locator* (url) for addressing material across the Internet. What is fetched could equally well be a sound such as someone discussing LINCS, or a movie on the project, or a set of data on its finances in a spreadsheet. The http protocol uses MIME encoding to represent arbitrary datasets, and the WWW client passes data it can present to a *helper* application that displays it, plays it, executes it, or otherwise presents or utilizes it. Since a helper application can send url back to the WWW client and request it to fetch a file, it is simple to extend client functionality.

Similar considerations apply to servers which need not be passive retrievers of existing documents but can in addition generate new documents and other datasets in response to a request. This was used initially to allow a server to generate lists of files with hypertext links for retrieval from dynamic directories. It was also used to support search facilities in which a collection of documents could be accessed by content, for example, through the WAIS protocol. Servers dynamically generating documents now constitute one of the major resources on WWW, for example, providing weather maps from satellite imagery.

Figure 2 shows the client-server architecture of WWW:

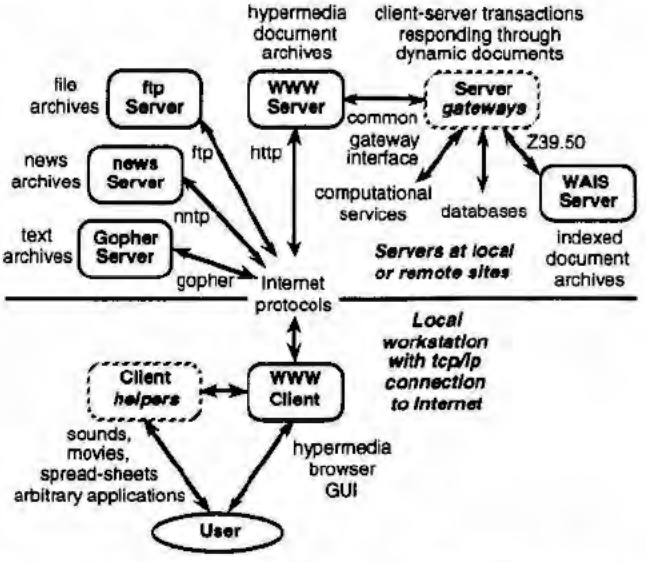

Figure 2 Client-server architecture of World-Wide Web

The partitioning of protocol handling between client and server is somewhat arbitrary. It has become conventional for WWW clients to handle the simple Internet protocols such as ftp [29], nntp [23] and gopher [2] directly, but use gateway access to more complex protocols through the server. The smtp (mail) [28], Z39.50 (WAIS) [34] and Prospero (ARCHIE) protocols may also be handled directly or, in the case of WAIS and ARCHIE, through gateways to a WWW server. In all cases the client attempts to present the interaction to the user through a uniform hypertext document format.

The protocol used between WWW clients and servers is http (hypertext transfer protocol) [3], a generic, stateless, object-oriented protocol that use MIME (multipurpose Internet mail extensions) [9] content encoding to transmit arbitrary data. The content encoding used for WWW documents is html (hypertext markup language) [7] which is a document type definition (DTD) in the ISO 8879 Standard Generalized Markup Language (SGML) [20].

This use of, and promotion of, standards has been a major strength of WWW, leading to consistency between a wide variety of client and server applications across many platforms. In particular, the adoption of an SGML tagged encoding schema has allowed WWW documents to include not only text, typographic and multimedia material but also to carry arbitrary additional data through simple, backwards compatible extensions. The html DTD is currently being standardized at four levels:

level 0 - text with embedded links;

level 1 - adds typographic text with embedded images;

level 2 - adds embedded graphic user interfaces (forms);

level 3 - adds tables, mathematics, and...

Level 0 functionality can be supported on an alphanumeric terminal. Level 2 adds typography and pictures. Level 2 adds embedded GUIs which are the primary topic of this article. Level 3 is still somewhat open-ended and will evolve through prototype implementations [30].

WWW FORMS—PORTABLE, EMBEDDED GUIs

In March 1993 WWW was still being presented [4] as primarily a hypermedia retrieval system, but in November that year a development took place that so changed the nature of WWW as to constitute a revolution in its own right. Andreessen [1] issued NCSA Mosaic version 2 using SGML tags to encode definitions of Motif widgets embedded within a hypermedia document, and allowed the state of those widgets within the client to be transmitted to the server. Suddenly the WWW protocols transcended their original conception to become the basis of general interactive, distributed, client-server information systems.

The following examples show the changes in functionality that html level 2 provides compared with the level 1 hypermedia information retrieval of Figure 1. Figure 3 shows the genesis of using html documents as GUIs through hypertext links attached to pictures. It is a level 1 document fetched through the same client, but now resembling, and having the functionality of, an iconic GUI. To the user it appears as a computer interface where clicking on one of 6 icons gives access to one of 6 services. The document generating Figure 3 is shown in Figure 4.

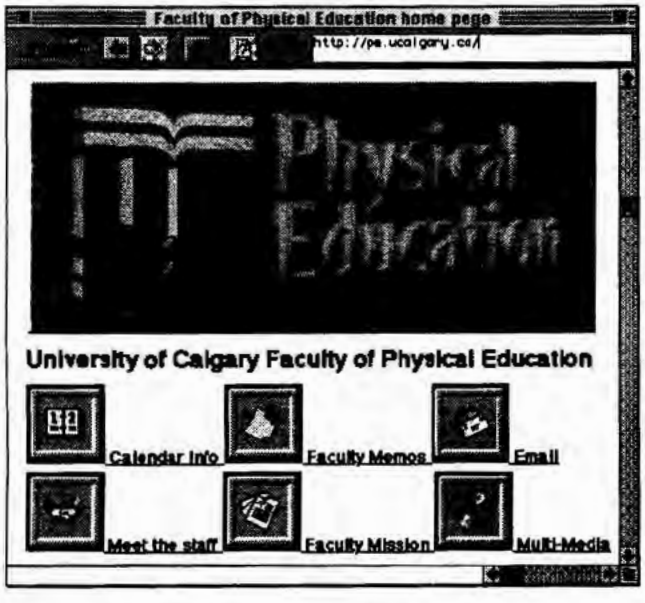

Figure 3 Interface to a university information system

```
<TITLE>Faculty of Physical Education home page</TITLE>
<IMG SRC="pelogo.gif">
<H1>University of Calgary Faculty of Physical Education</H1>
<B><A HREF="calendar.html">
<IMG SRC="cal.gif" ALT=" "> Calendar Info</A></B>
<B><A HREF="memos/readmem.html">
<IMG SRC="memo.gif"> Faculty Memos</A></B>
<B><A HREF = "execute/runmail.jcl">
<IMG SRC="mail.gif"> Email</A></B><BR>
<B><A HREF="meet.html">
<IMG SRC="meet.gif" ALT=" "> Meet the staff</A></B>
<B><A HREF="mission/mission.html">
<IMG SRC="mission.gif" ALT=" "> Faculty Mission</A></B>
<B><A HREF="media.html">
<IMG SRC="media.gif" ALT=" "> Multi-Media</A></B>
```

Figure 4 Document generating Figure 3

Clicking on an icon in Figure 3 requests another document but this can have the side-effect of taking other actions at either server or client. For example, the "runmail.jcl" document in line 8 of Figure 4 runs a script in a job control application on the client that starts the user's normal email application and brings its window to the front.

The interface of Figure 3 is achieved though level 1 html hypermedia functionality. However, such applications suggested the provision of embedded GUIs as a natural extension to html. The level 2 forms extension enhances the capability of WWW documents to act as GUIs by allowing other widgets such as buttons, check boxes, radio buttons, popup menus, scrolling lists, and text entry boxes to be embedded. Figure 5 shows the memo entry facility selected through the "Faculty Memos" icon in Figure 3. The user can type information into what appears to be a normal GUI dialog box, and submit that information to the server.

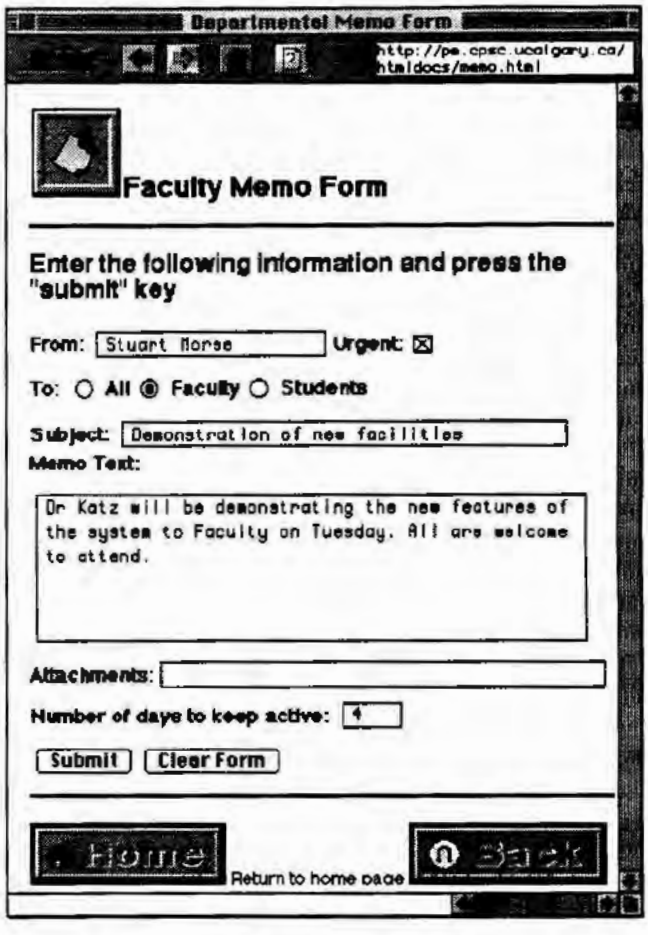

Figure 5 Memo entry in a university information system

Figure 6 shows the html specification for the form of Figure 5. The text boxes are simply specified, and the ACTION field gives the url to which their contents will be sent using the POST facility of the http protocol. When the "Submit" button is clicked the client sends the contents of the form to the server which passes them to the program "post-memo" which files them. Users may retrieve memos through an index document with hypertext links to available memos.

```
<FORM METHOD="POST"
        ACTION="http://pe.cpsc.ucalgary.ca/cgi-bin/post-memo">
<H2>Enter the following information and press the "submit"
        key</H2> <BR>
<STRONG>From: </STRONG>
<INPUT SIZE=20 NAME="username">
<STRONG>Urgent:</STRONG>
<INPUT TYPE=CHECKBOX NAME="urgent" VALUE="yes"> <BR>
<STRONG>To: </STRONG>
<INPUT TYPE=RADIO NAME="to" VALUE="All" CHECKED>
<STRONG>All</STRONG>
<INPUT TYPE=RADIO NAME="to" VALUE="Faculty">
<STRONG>Faculty</STRONG>
<INPUT TYPE=RADIO NAME="to" VALUE="Students">
<STRONG>Students</STRONG>
<BR> <STRONG>Subject: </STRONG>
<INPUT SIZE=40 NAME="subject"> <BR>
<STRONG>Memo Text: </STRONG><BR>
<TEXTAREA NAME="Content" cols=50 rows=6></textarea> <BR>
<STRONG>Attachments:</STRONG>
<INPUT SIZE=40 NAME="attachments"><BR>
<STRONG>Number of days to keep active: </STRONG>
<INPUT SIZE=4 NAME="duration"> <BR>
<INPUT TYPE=SUBMIT VALUE="Submit">
<INPUT TYPE=RESET VALUE="Clear Form">
</FORM>
```

Figure 6 Form generating user interface in Figure 5

From a human-computer interaction perspective, what is significant about Figures 3 and 5 is that they represent graphic user interfaces operating on Unix, Mac, PC and other platforms, giving access to a functionality on a local or remote server operating on any platform supporting tcp/ip connections. Moreover, the user interface is programmed through a simple script that can be written in any text editor. It is this simplicity of development of interactive applications which has generated widespread interest in programming client-server applications through WWW in the short period since the inception of the forms capability.

The examples above were chosen because they show the capabilities of html forms in the rapid development of conventional GUIs. However, the embedding of the user interface in high-quality documents offers the possibility of innovative applications in which the interface functionality emerges naturally from the application context. Figure 7 shows a screen from GroupWriteNet (GWN), a system for supporting collaboration in document production. GWN is an extension of our group writing [13] and active document [15] tools to operate across the Internet, and part of the development of a suite of tools supporting distributed scientific communities [17, 18].

GWN documents are stored in html format and can be accessed normally through WWW. A member of a registered group can request that a document belonging to that group be retrieved in annotatable form as shown in Figure 7. Each paragraph in the annotatable document is followed by a form allowing annotation to be entered and submitted to the server. Members of the group can then retrieve the document with each paragraph followed by hypertext links to the related annotation.

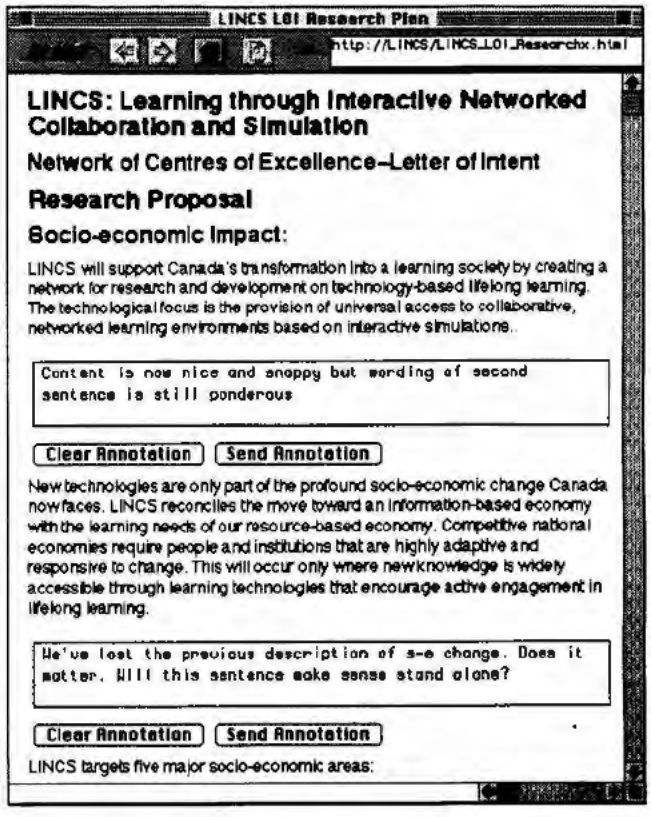

Figure 7 Annotation in collaborative writing

Figure 8 shows the html specification for the forms in Figure 7. Note the HIDDEN field in line 2 which contains an identification code for the annotator. The http protocol is designed to be stateless so that the server does not in itself keep track of a sequence of related transactions with the same client. However, the required state information can be embedded in documents sent to the client so that when the server receives a request the associated service can determine the state of a transaction sequence. Since a client can save an html document and reopen it at a later time, this also allows transaction sequences to be suspended with the state information retained by the client. The fact that this is achieved through the normal local document storage mechanism also makes it a simple and natural activity for the user.

```
<FORM ACTION="ISC4.script" METHOD=POST>
<INPUT TYPE=HIDDEN NAME="Annotator"
        VALUE="BRGpGF9y">
<TEXTAREA NAME="ISC4.1.1-1" ROWS=3 COLS=60>
</TEXTAREA> <BR>
<INPUT TYPE=RESET VALUE="Clear Annotation">
<INPUT TYPE=SUBMIT VALUE="Send Annotation">
</FORM>
```

Figure 8 Annotation forms in Figure 7

The level 2 html capabilities make WWW an excellent environment for developing client-server applications, and have triggered a wide range of innovative interactive systems. Figure 9 shows interaction with Ibrahim's [22] system for teaching Pascal programming through the web.

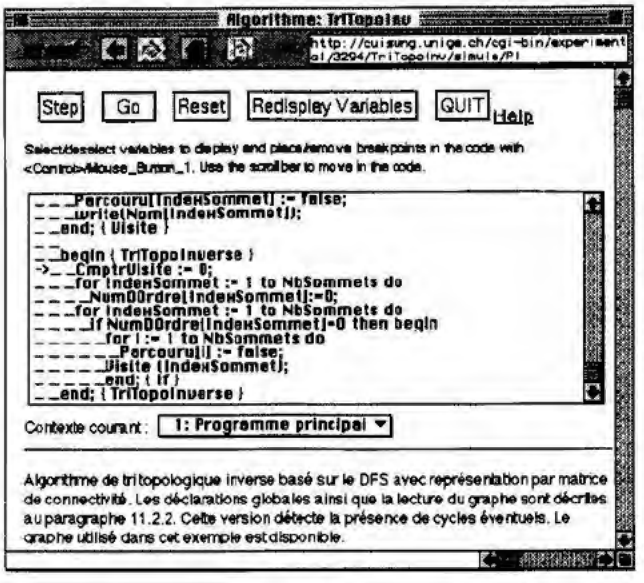

Figure 9 Teaching Pascal over the web

The student sees a Pascal program in a scrolling text area and can request the server to run the program, reporting the results, step through it, run to a breakpoint, and show the values of selected variables, allowing them to be changed. Breakpoints and variable to be displayed are defined by clicking on the appropriate program line.

Figure 10 shows Gruber and Gauthier's Device Modeling Environment (DME) [21] being used in the simulation of a leak in the Space Shuttle's reaction control system.

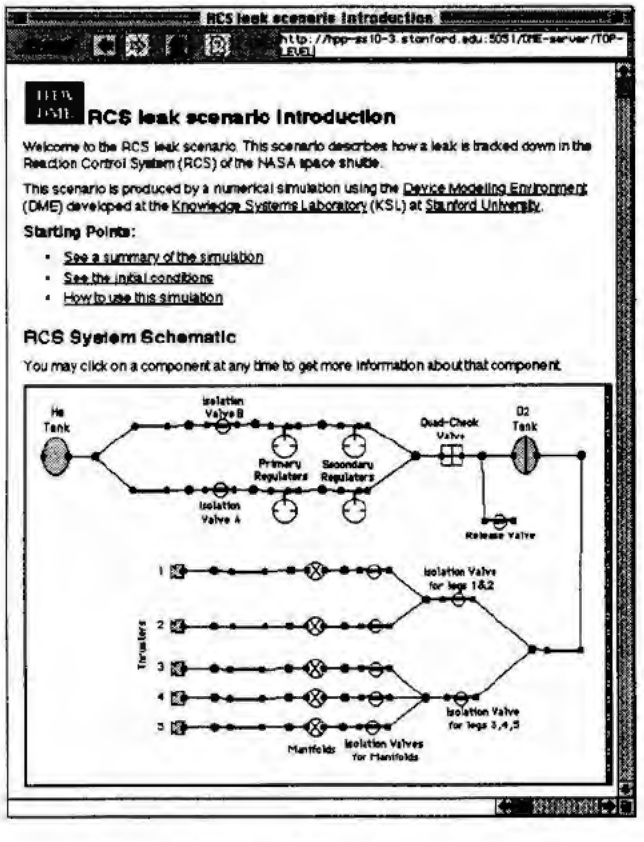

Figure 10 Fault diagnosis over the web

When the user clicks on a component in the schematic at the bottom of Figure 10 the server returns a document giving information about its role in the device model linked to other information, such as its current state. This uses another html level 1 interface feature, the ISMAP attribute that designates an image as a 'map' which when clicked returns the position of the cursor in the image to the server. The DME system is also interesting because it uses a distributed server based on agents communicating through the kqml protocol [12] to make inferences about the model.

Maps allow the WWW user interface to be extended with additional widgets provided a single click interaction is adequate. For example, Figure 11 shows our KMapServer using KMap [16], a concept mapping tool, to provide access to a multimedia database. The server runs KMap locally to determine in which concept the user has clicked, treating the click as if it has occurred in a server window holding the map. This allows any concept map developed in KMap to be delivered as an interactive widget on WWW.

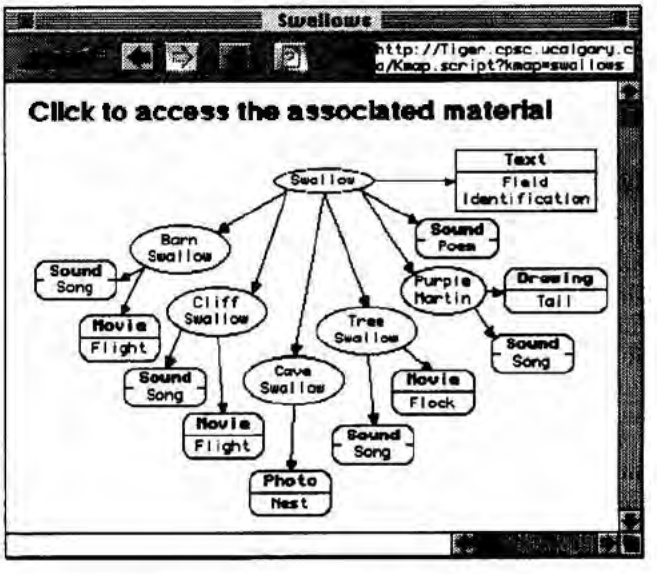

Figure 11 Concept map server

KMap can also operate as a helper at the client end, and this has been used to implement Mediator [14], a system for managing a geographically dispersed virtual factory management system. Attaching the concept mapping tool as a helper to the client allows local use of its full interactive functionality. Attaching it through a gateway to the server allows significant parts of its functionality to be offered through any client. Both are useful system architectures, and determining where to place such functionality is one of the interesting decisions to be made in designing distributed client-server applications.

Servers can interact with the real-world, for example to support tele-presence at a remote site. Figure 12 shows the Mercury Project [19] developed by the Departments of Anthropology and Computer Science at USC in which users tele-operate a robot arm moving over a terrain filled with buried artifacts. A CCD camera and pneumatic nozzle mounted on the robot allow users to select viewpoints and to direct short bursts of compressed air into the terrain.

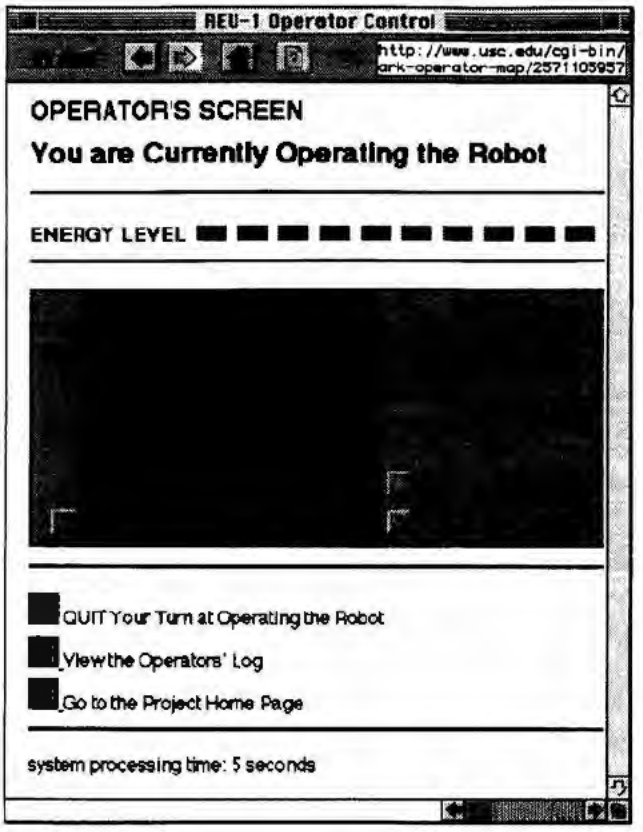

Figure 12 Tele-operation of a robot arm

HUMAN FACTORS OF WWW APPLICATIONS

WWW technology is evolving rapidly. Impediments to effective interaction are being removed as they are noted. In analyzing existing limitations one must distinguish between current constraints, those that will change in the near future, and those that are comparatively long-term. For example, lack of incremental updating of the screen will disappear in a short time as clients become more intelligent, lack of down-loadable interactive widgets will disappear as a standard is developed, and delays in communication between client and server will decrease as the 'information highway' becomes a reality. However, there may always be users with minimally-featured clients and lower speed connections, so taking such limitations into account will remain significant for many years.

The technology involves four main areas of human-computer interaction:

- **Graphic user interfaces** subject to the guidelines that have been developed over the years. The usual issues of uniformity in style of layout, presentation, vocabulary, use of color, and so on apply, and human factors design may be modeled on existing practice [31-33].
- **Hypermedia navigation** where issues such as cognitive overload in hyperspace navigation, and the associated guidelines, are also becoming well-defined in the hypertext community [11, 26].
- **Typography, layout and visualization** which have human factors guidelines arising in graphic design and information visualization [8, 35].
- **Client-server computing** where communication delays and partitioning of functionality create problems that generally fall under the normal user interface guidelines. However, because they are not problems in local interaction with a personal computer, they need to be treated as peculiar to the client-server environment. It is these that form the focus of the remainder of this article.

The client-server computing issues most strongly affecting the usability of WWW are:

- **Communication delays** which can be long and variable, and are crucially dependent on features of the data structures and client architecture. System designers need to be aware of far more detail about WWW operation than is apparent at first sight.
- **Limited client functionality** which interacts with delays to reduce interactivity. For example, it is difficult to give help information rapidly and in context.
- **Stateless transactions** modeled on an information retrieval paradigm. As already noted, facilities exist for the designer to track state information very effectively through hidden fields. However, the retrieval paradigm has led to the neglect of incremental updating in current clients and this again reduces interactivity.

Communication delays

Variable and long communication delays are probably the most critical factor that the system designer must take into account currently. If client helper applications are not used then the application can be factored into a human interface at the client communicating with the major functionality at the server. Most interactions then require client-server communication, and expediting this is crucial to interactivity, particularly given users' experience of high-speed local interactive applications.

The main data structures transmitted are tagged text, forms (which are also tagged text), and images. Images are currently encoded in the CompuServe gif format which uses Lempel-Ziv run-length encoding making compression critically dependent on the image structure [25].

Figure 13 shows the transaction times for typical data over different speed lines chosen to represent: a modem over a dial-up line; a typical Internet data rate between continents; and a typical one in North America. V32bis compression gives a data rate of about 3.4Kbytes for text and 1.5Kbytes for gifs. The 50 Kbytes figure also typifies local ethernet communications. The main components of the transaction time are time to: perform domain name system (dns) lookup; transmit the request; process the request; transmit text; transmit graphics; decode graphics; and layout text and graphics. The times shown do not take into account processing at the server because this is application-dependent, or dns lookup since this is variable and usually cached both at the local site and within the client.

The document types have been chosen to typify a wide range of applications, and to illustrate some of the features of gif encoding and caching. The times in the first column represents a typical remote user over a modem, and it can be seen that the transmission of images is the primary factor in determining the response time.

		Transaction Time (sec)		
Document Type \Data Rate text size + gif size \		V32bis modem	5 Kbytes	50 Kbytes
1	1 page paper - 2.5K+0K	1.5	1.3	0.8
2	1 page form - 1K+0K	1.3	1.2	1.0
3	6 pages paper - 15K+0K	5.0	3.8	1.0
4	1 page inc. 1 1/2 page fig. 1.25K+10K	8.6	4.5	2.5
5	6 page inc. 2 1/2 page figs. 12.5K+20K	20.6	10.3	4.4
6	6 page inc. 6 1/2 page figs. 7.5K+60K	51.8	21.3	11.0
7	6 icon gifs (Fig. 3) 0K+6x1K	13.2	10.2	9.0
8	1 page, 6 repeated bullet gifs 2.5K+1x0.4K	3.6	3.2	2.4
9	Complex image - 0K+100K	68.1	21.5	3.5

Figure 13 Transaction times for different data rates

There are two factors that affect image transmission apart from communication speed. First, the run-length encoding of gifs is effective for images that have minimal variations in color along a horizontal scan line. Typically, gif encodes a half page presentation graphic in 10K, but a photographic image in 100K. Designers can improve interactivity by using icons and images that encode well. There are many examples on the web of 3-D rendered icons that look beautiful but are very detrimental to interactivity. Second, clients typically cache all material fetched indexed by its url, and hence a gif that is reused comes from local storage and incurs no communication delay. The reuse of the same icon within a document and across related documents is conducive to both uniformity of style and to interactivity.

However, caching also creates interactivity problems with dynamic documents where the client is unaware that a cache has become out of date. The current protocols need to be extended to support better coordination of caching.

The second and third columns show the merits of a direct connection to the Internet, and the capabilities of WWW on a local network. They also indicate that it is important to specify whether a system is intended to be used from remote sites and through modem connections.

Limited client functionality

While it is good design practice to factor out the human interface from the application functionality, communication delays on the web make it impossible to implement what has become standard practice in local application design. We have become used to programming the state of the interface to change as the user interacts with it so that its affordances accurately track meaningful user choices. We have also become used to providing rapid context-sensitive help that does not disturb the interaction. Neither of these is possible within the html level 2 specification, although the SGML tagging could allow an indefinite amount of user interface programming to be communicated to the client. Raggett [30] is addressing this in the html level 3 specification through the provision of a script language for controlling the dynamic features of the interface.

In future developments it would be simple for clients to support script languages of the power of tcl and tk [27], which would allow interactivity to be programmed equivalent to that of local operation. In particular this would allow widgets to be programmed offering the full variety of click and drag interfaces. There are security problems in clients supporting scripts with full access to file and operating system facilities, but these can be overcome through the use of trusted interpreters with well-defined restrictions such as those proposed for General Magic's Telescript [24].

Stateless transactions

The stateless transaction protocol adds an interesting design factor to WWW systems in that state information is normally associated with the computations taking place at the server rather than with the user interaction taking place at the client. However, one soon becomes used to passing state information back to a client, and making provision to store it there through operations that are natural to the user. This would be aided if clients made better provision for local action to be taken as a result of user interaction with a document. Essentially, one needs to be able to post forms to a local mini-server that can interact directly with the client. This is becoming easier as clients come to support scripting through such mechanisms as Apple Events.

Another addition to client functionality that would greatly improve interactivity is the support of incremental updating of the screen. Much could be achieved by the normal flicker reduction technique of writing new data into an offscreen bitmap and refreshing the complete screen. WWW was designed to retrieve a succession of different items of information, but it is being used increasingly in applications such as those illustrated in Figures 9 and 12 where the user envisions a screen of changing information not a succession of different documents.

CONCLUSIONS

The combination of low-cost, high-performance personal computers, and of low-cost, high-speed communications through the Internet, has provided the enabling technology for a new generation of interactive client-server computing. World-Wide Web, by providing protocols for the rapid development of graphic user interfaces and their embedding in high-quality documents, has made this resource widely available. Computing has changed in the 1990s and there are possibilities for innovation in human-computer interaction that are reminiscent of the early growth of interactive systems in the late 1960s and early 1970s.

The recency, and rate of innovation, in WWW facilities should lead one to expect that what has been achieved to date is by no means optimal in design. It lacks many features which would have been present had the applications of 1994 been available to the WWW designers in 1991 through 1993. There have been calls for a complete zero-base approach to the design of distributed client-server systems on the Internet. They are unrealistic because the will and energy are not available, and any normal specification and standardization process would guarantee obsolescence in all its products.

However, it is important to reverse engineer what has been achieved to date, and to begin to impose on it a rational model that logically completes and extends the system architecture that has evolved. Human factors considerations are paramount to WWW yet they have not been explicit in its design. The web will be an exciting challenge to the CHI community for many years to come.

ACKNOWLEDGMENTS

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada. The system shown in Figures 3 through 6 was specified by Larry Katz, Director of the Computerized Sports Systems Group, and developed by Stuart Morse, as part of an MSc project supervized by Mildred Shaw.

REFERENCES

1. Andreessen, M., NCSA Mosaic Technical Summary, NCSA, University of Illinois, ftp://ncsa.uiuc.edu/Web/Mosaic/Papers/mosaic.ps.Z, 1993.
2. Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D., and Alberti, B., The Internet Gopher Protocol (a distributed document search and retrieval protocol), Internet, RFC 1436, 1993.
3. Berners-Lee, T., Hypertext Transfer Protocol, CERN, ftp://info.cern.ch/pub/www/doc/http-spec.ps.Z, 1993.
4. Berners-Lee, T., World-Wide Web, http://info.cern.ch/hypertext/WWW/Talks/OnlinePublishing93/Overview.htm.
5. Berners-Lee, T. and Cailliau, R., WorldWideWeb: Proposal for a Hypertext Project, CERN, Geneva, http://info.cern.ch/hypertext/WWW/Proposal.html, 1990.
6. Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H.F., and Secret, A., The World-Wide Web, Communications ACM 37, 8 (1994), 76-82.
7. Berners-Lee, T., Connolly, D., and Muldrow, K., Hypertext Markup Language (HTML) Version 2.0, CERN, Geneva, http://www.hal.com/products/software/olias/Build-html/html-spec.ps, 1994.
8. Bertin, J., Semiology of Graphics, Madison, Wisconsin, University of Wisconsin Press, 1983.
9. Borenstein, N. and Freed, N., MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for specifying and describing the format of internet message bodies, Internet, RFC 1521, 1993.
10. History to date, CERN, Geneva, http://info.cern.ch/hypertext/WWW/History.html, 1994.
11. Conklin, J., Hypertext: An introduction and survey, IEEE Computer 20, 9 (1987), 17-41.
12. Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McKay, D., McGuire, J., Shapiro, S., and Beck, C., Specification of KQML Agent-Communication Language, The DARPA Knowledge Sharing Initiative External Interfaces Working Group, 1992.
13. Gaines, B.R. and Malcolm, N., Supporting collaboration in digital journal production, Journal of Organizational Computing 3, 2 (1993), 195-213.
14. Gaines, B.R. and Norrie, D.H., Mediator: information and knowledge management for the virtual factory, in SIGMAN AAAI-94 Workshop: Reasoning about the Shop Floor (1994), AAAI, Menlo Park, California.
15. Gaines, B.R. and Shaw, M.L.G., Open architecture multimedia documents, in Proceedings of ACM Multimedia 93 (1993), pp. 137-146.
16. Gaines, B.R. and Shaw, M.L.G., Concept maps indexing multimedia knowledge bases, in AAAI-94 Workshop: Indexing and Reuse in Multimedia Systems (1994), AAAI, Menlo Park, California.
17. Gaines, B.R. and Shaw, M.L.G., Knowledge acquisition and representation techniques in scholarly communication, in Proceedings SIGDOC'94 (1994), ACM, New York.
18. Gaines, B.R. and Shaw, M.L.G., Using knowledge acquisition and representation tools to support scientific communities, in AAAI'94: Proceedings of the Twelfth National Conference on Artificial Intelligence (1994), AAAI Press/MIT Press, Menlo Park, California. pp. 707-714.
19. Goldberg, K. and Mascha, M., Mercury Project, University of Southern California, http://www.usc.edu/dept/raiders/story/fmi.html, 1994.
20. Goldfarb, C.F., The SGML Handbook, Oxford, Clarendon Press, 1990.
21. Gruber, T. and Gautier, P., Machine-generated explanations of engineering models: a compositional modeling approach, in IJCAI'93: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (1993), Morgan Kaufmann, San Mateo, California. pp. 1502-1508.
22. Ibrahim, B., World-wide algorithm animation, in Proceedings of WWW94 (1994), http://www1.cern.ch/PapersWWW94/betrand.ps.
23. Kantor, B. and Lapsley, P., Network News Transfer Protocol, Internet, RFC 977, 1986.
24. Knaster, B., Presenting Magic Cap, Reading, Massachusetts, Addison-Wesley, 1994.
25. Luse, M., Bitmapped Graphics Programming in C++, Reading, Massachusetts, Addison-Wesley, 1993.
26. McKnight, C., Dillon, A., and Richardson, J., Hypertext in Context, Cambridge, UK, Cambridge University Press, 1991.
27. Ousterhout, J.K., Tcl and the Tk Toolkit, Reading, Massachusetts, Adison-Wesley, 1994.
28. Postel, J., Simple Mail Transfer Protocol, Internet, RFC 821, 1982.
29. Postel, J. and Reynolds, J., File Transfer Protocol, Internet, RFC 959, 1985.
30. Raggett, D., A review of the HTML+ document format, in Proceedings of WWW94 (1994), http://www1.cern.ch/PapersWWW94/dsr.ps.
31. Shackel, B. and Richardson, S., ed. Human Factors for Informatics Usability, Cambridge University Press, Cambridge, UK, 1991.
32. Shneiderman, B., Direct manipulation: a step beyond programming languages, Computer 16, 8 (1983), 57-69.
33. Smith, S.L. and Mosier, J.N., Guidelines for Designing User Interface Software, Mitre Corporation, ESD-TR-86-278, 1986.
34. St.Pierre, M., J, F., Gamiel, K., Goldman, J., Kahle, B., Kunze, J., Morris, H., and Schiettecatte, F., WAIS over Z39.50-1988, Internet, RFC 1625, 1994.
35. Tutte, E.R., Envisioning Information, Cheshire, Connecticut, Graphics Press, 1990.