# *Time And The Web: Representing And Reasoning About Temporal Properties Of Interaction With Distributed Systems*


## Chris Johnson

Department of Computing Science,
University of Glasgow,
Glasgow, G12 8QQ, UK.

Tel: +44 (041) 339 8855 ext. 6053

Fax: +44 (041) 330 4913

EMail: johnson@dcs.glasgow.ac.uk



**New groups of users are learning to exploit the facilities provided by Internet browsers, such as Mosaic and Netscape. The graphical interaction style supported by these systems eases the traditional burdens of information retrieval using methods such as the file transfer protocol (ftp). In consequence, distributed information sources are being made available to people whose interests do not lie solely in computer science or systems engineering. This rapid growth in the Internet has exposed users to interaction problems that rarely occur with stand-alone computer systems. In particular, people are often faced with unpredictable timing delays over remote networks. These delays lead to frustration and error which can, in turn, prevent users from successfully retrieving necessary information. This paper exploits Clarke and Emerson's Computation Tree Logic (CTL) to identify presentation techniques that can support retrieval tasks. It is argued that CTL provides a convenient means of representing and reasoning about temporal properties of interaction with distributed systems.**



**Keywords:** interface design; temporal logic; World Wide Web; distributed systems.


## 1. Introduction

Since the inception of the Internet under the Pentagon's Advanced Research Project Agency, there has been a rapid growth in world-wide computer networks. In 1980 there were approximately one hundred computers attached to the Internet. In 1990 there were one hundred thousand. In 1994, the number of systems connected to the Internet exceeded one million. A recent estimate placed the number of Internet users at just over twenty-five million (Schofield, 1994). Hundreds of sites in many different domains provide access to a vast range of information sources. The growth of these information sources and the development of applications such as Gopher and Mosaic has encouraged the active participation of new groups of users (Berners-Lee, Cailliau and Groff, 1992). Most of these

participants only possess a minimal knowledge of the communications mechanisms that support computer networks. Remote site failures, network bottlenecks and the problems of data fusion have little relevance to their tasks. Unfortunately, it is not possible to engineer out these problems given current technology. In consequence, users are faced with a range of apparently 'bizarre' temporal behaviours when interacting with distributed computer networks. The resulting usability problems frustrate attempts to exploit Internet resources.

### *1.1. The Problem: Unpredictability*

This paper focuses upon one particular interface problem for distributed systems; unpredictability. This is defined to occur whenever a user cannot determine the consequences of their actions based upon the information presented by the system at the time that they perform those actions (Dix, Harrison, Runciman and Thimbleby, 1987). Such unpredictability has a profound impact upon the quality of interaction with a range of computer applications. For example, it can be difficult for the users of a text-editor to predict the effects of a keypress if they cannot determine its current mode. A consequence of this is that novice users quickly become frustrated by the apparently random effects of entering the same command line in different contexts. Unpredictability also has important consequences for multi-user systems ranging from conferencing systems to steel casting plants (Miles, Johnson, McCarthy and Harrison, 1991). For instance, an operator cannot predict that commands to start a piece of equipment will be successful if one of their colleagues has already closed it down. In the worst case, this can lead to a complete break-down in co-operative working techniques.

This paper focuses upon the problems that unpredictability creates for a single user accessing information over a computer network. Figure 1 characterises the architecture of interaction with distributed information retrieval systems over the Internet.

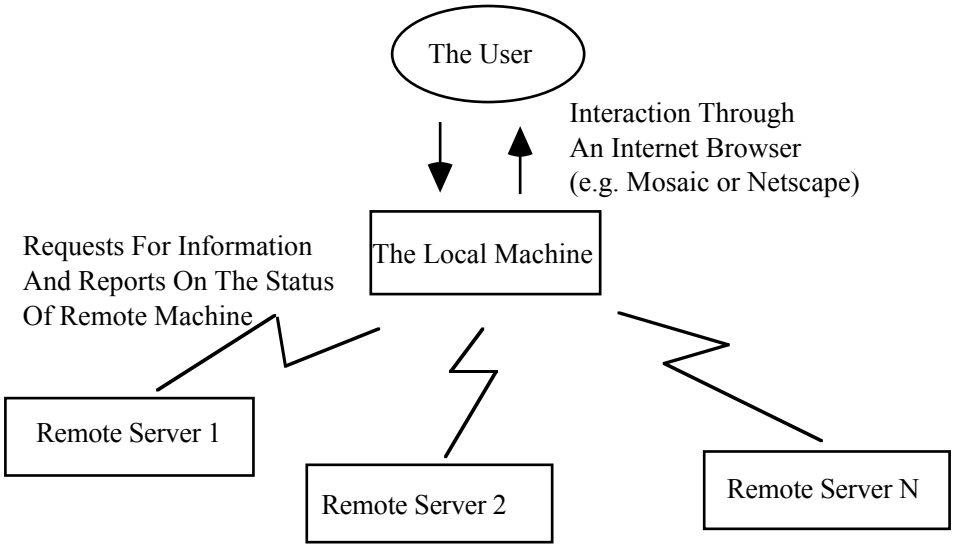


Figure 1: Information Retrieval Over A Distributed Network

Application programs, called browsers, translate user request for information into the communications primitives that are necessary to transfer relevant data from remote servers. Internet browsers hide the details of the underlying networks from the user. This leads to unpredictability because without this information it is difficult to determine whether a

retrieval command will be successful. Remote site failure and communications bottlenecks can delay the transmission of data between the browser and the remote servers.

### *1.2. Outline Of The Paper*

This section has briefly introduced the problem of unpredictability in the context of user interfaces to distributed information sources. Section 2 describes how this problem affects the Mosaic and Netscape World Wide Web browsers. These applications typify many of the usability problems that frustrate interaction with distributed information systems. Section 3 argues that formal notations, such as Clarke and Emerson's Computation Tree Logic (CTL), can be used to represent the complex temporal behaviours that create unpredictability. Section 4 applies CTL to analyse the underlying causes of unpredictability in distributed systems: retrieval failure and presentation delay. Section 5 uses the logic to represent techniques, such as failure dialogues, that reduce the problems created by retrieval failure. Section 6 presents solutions to the problem of presentation delay: incremental updates and request termination. Section 7 identifies areas for further research and presents the conclusions that can be drawn from this work.

## 2. The World Wide Web

Figure 2 shows the user interface to the National Center for Computer Applications' Mosaic browser. This system and its commercial successor, Netscape, are used to illustrate this paper. They provide concrete examples of the architecture shown in Figure 1. Users, typically, request information by selecting a labelled button, such as How To Brew Your First Beer. The browser then assembles the requested information by sending messages to local or remote servers.

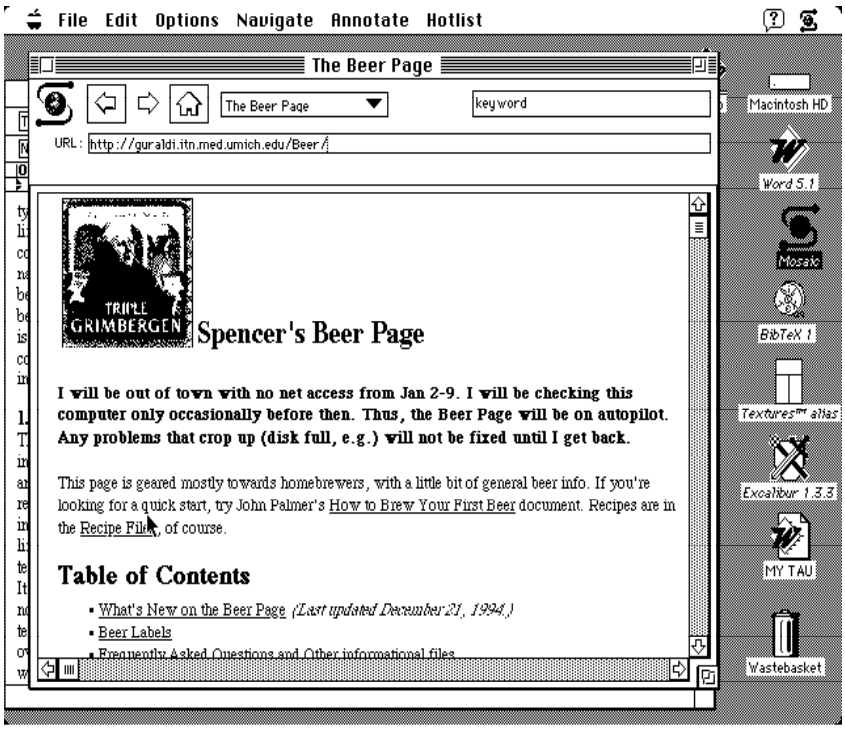


Figure 2: The User Interface To The Mosaic Browser.

Graphical images, sounds, video clips and pages of text are referenced by what are known as Uniform Resource Locators (URLs). The URL of the page in Figure 2 is:

**http://guraldi.itn.med.umich.edu/Beer/**

The **http** refers to the hypertext transfer protocol that is used to convey the information from remote sites. The machine that the information is retrieved from is specified after the protocol. In our example, it is the Internet site at **guraldi.itn.med.umich.edu**. The location of the information on that machine is given after the address of the site; **Beer/**. The important point here is that vast resources of information can be accessed by the selection of labelled buttons. Users are not required to understand the underlying architecture that supports the various protocols and URL retrieval mechanisms.

Unfortunately, it is not always possible to isolate users from the underlying communications mechanisms that support Web browsers. Delays can occur when bottlenecks form in the passage of information back to the browser. These problems come and go as the network loading changes over time. From the point of view of the user, it is impossible to predict the effects that such problems will have upon the duration of their retrieval task. The information that they request may be provided relatively quickly if none of these factors intervene. Alternatively, their retrieval task may fail completely if a remote site goes down. Several studies have shown that users find it difficult to complete retrieval tasks under such circumstances, especially if they have a fixed amount of time at their disposal (Pejtersen, 1989). Other studies have shown that variable delays in system response times lead to frustration and error (Kuhmann, 1989). It is important to emphasise these problems are tied both to the temporal behaviour of the network and to the temporal characteristics of the browser. If the interface to the application provides feedback on the rate of information transfer then users may be able to make better predictions about the likely costs of a retrieval as it progresses. This paper argues that designers can exploit formal notations to reason about such temporal presentation techniques.

## 3. Logic And Time

Formal specifications are increasingly being recruited to support the software engineering of distributed, real-time systems (Johnson, McCarthy and Wright 1995). Mathematical techniques can also support interface development (Harrison and Thimbleby, 1990). For example, the following clause illustrates how a Horn clause notation can be used to represent requirements for a Web browser. This formalism is appropriate because it eases the transition between abstract logic requirements and executable prototypes using variants of the Prolog programming environment (Johnson , 1993, 1994) . These systems can be used to validate the claims that are embodied within formal specifications of user interfaces. The following formula states that a resource is retrieved if the user selects an area of text which is linked with that resource. The logic provides a means of abstracting away from the low level input and output device details that obscure important properties of an interface. This information can be added gradually as development progresses:

$$display_url(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge fetch(Url) \wedge present(Url) \quad (1)$$

Unfortunately, there is no notion of time in classical logic; formula (1) would be true if the resource were retrieved and presented before it had been requested by the user. This is clearly not what the designer intended. The previous clause also fails to capture the potential delay that can occur between the user selecting the label and the resource being returned from a remote machine. These limitations can be avoided by using Clarke and Emerson's (1982) Computation Tree Logic . CTL is based upon a branching model of time; it can be used to

describe alternative future traces of interaction. This provides great flexibility in the range of temporal properties that can be specified. Figure 3 illustrates CTL's model of time.

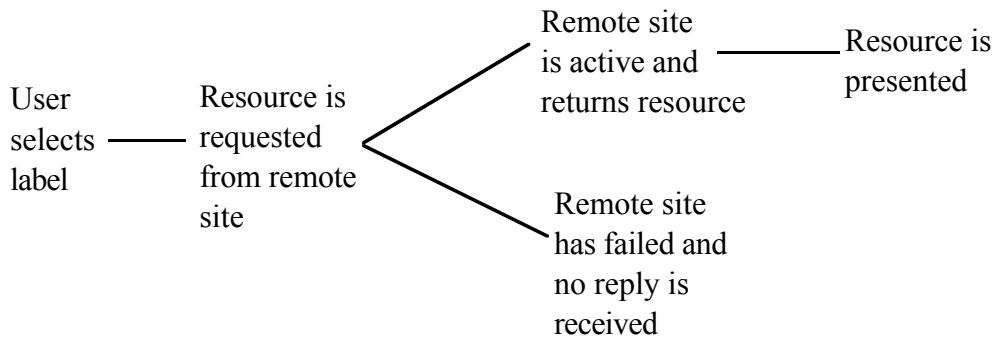


Figure 3: CTL's Branching View Of Possible Futures

In this diagram two possible futures lead from the user selecting the label. In one path the resource is successfully retrieved from a remote site. In another path it cannot be retrieved because the remote site is unavailable. This captures the uncertainty that frustrates attempts to predict the outcome of particular commands. Brevity prevents a complete presentation of the syntax and semantics for CTL. Johnson (1993a) discusses the formal underpinnings of the notation. In contrast, the following pages apply the notation to analyse the problem of unpredictability in distributed systems. Informal descriptions of the logic clauses will also be provided throughout the paper. The intention is to focus upon the application of the formalism to support interface development.

## 4. Analysing Unpredictability In Distributed Systems

CTL can be used to specify properties over traces of interaction between a user and a distributed system. For instance, the *AX* quantifier specifies that a particular property is true in the next interval of all possible traces. This provides a means of introducing the sequential information that was missing from clause (1). The following formula requires that the user selects the label before the corresponding resource is retrieved:

$$\textit{strong_predictability(Url)} \Leftarrow \textit{user_selects(Label)} \wedge \textit{linked(Label, Url)} \wedge \textit{AX (fetch(Url)} \wedge \textit{AX present(Url))} \quad (2)$$

This formula represents an ideal state of affairs; the resource would be retrieved immediately after it had been requested in every trace of interaction. Unfortunately, network delays and site failures make it impossible for designers to guarantee that a particular resource will be retrieved immediately after it has been requested. It is only possible to ensure that there are some traces in which a resource is returned immediately after it has been requested. This weaker objective can be represented using the *EX* operator (read as 'there exists some next state'):

$$\textit{weak_predictability(Url)} \Leftarrow \textit{user_selects(Label)} \wedge$$

$$linked(Label, Url) \wedge EX\,(fetch(Url) \wedge AX\,present(Url)) \quad (3)$$

Unpredictability occurs because users cannot determine whether a request will be one of those for which the resource is immediately retrieved. In other words, a browser may behave unpredictably if the user selects a label and it is possible that the resource cannot be retrieved and presented in the next interval:

$$unpredictable(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EX\ not(fetch(Url) \wedge AX\ present(Url)) \quad (4)$$

Formal reasoning techniques provide means of re-writing temporal logic clauses in order to clarify the properties that they describe. For example, a temporal form of De Morgan's law can be applied to the previous formula in order to identify the causes of unpredictability in distributed systems. The technical details of this re-writing are described in Johnson (1992). Informally, the following clause specifies that unpredictability occurs if the browser fails to fetch the resource referenced by the URL in the next interval or if there exists a subsequent interval in which the browser does not present that resource:

$$unpredictable(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EX\ (not(fetch(Url)) \vee EX\ not(present(Url))) \quad (5)$$

The following sections analyse interface design techniques that can be used to reduce the impact of this unpredictability.

## 5. Unpredictability And Retrieval Failure

Logic formulae, such as (5), provide a focus for the development of user interfaces to distributed systems. For example, one of the causes of unpredictability was identified as retrieval failure. This occurs when information is not returned to the browser immediately after the initial request; *EX not(fetch(Url))*. Typically, this might be due to high network loading or to the failure of a machine at a remote site. This delay is propagated into the user interface of the browser and can create considerable frustration when using the system (Kuhmann, 1989). A number of techniques can reduce the impact of such retrieval failures. For instance, browsers frequently warn users as soon as a transmission failure has been detected. In CTL, the *EF* operator (read as 'there exists some future') can be used in combination with the *AX* operator to specify that the user is alerted to a problem if eventually a resource is not retrieved and a warning is presented:

$$failure_warning(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EF\ (not(fetch(Url)) \wedge AX\ present(request_failure)) \quad (6)$$

The Mosaic and Netscape browsers both exploit this approach. In Mosaic, the number of bytes read from a remote server is displayed. The Netscape browser presents the percentage of data already received for a particular resource. If failures eventually occur then these numbers remain unchanged. The problem with both of these techniques is that the failure may only be detected well into a transfer. This interval is represented by the *EF* quantifier in

clause (6). Given the distractions of many working environments it can be difficult for users to monitor the precise amount of data that has been transferred from a remote site. Internet browsers, typically, provide additional displays to attract users' attention when connections cannot be made to remote sites. For instance, Mosaic uses the warning 'Unable to connect to remote host'. The results of an initial 'think aloud' evaluation show that the warning is easily overlooked by many users and that this can lead to further unpredictability. For instance, novice users frequently respond to transmission delays by re-selecting a label. Rather than repeating the retrieval request, subsequent mouse clicks select the text under the cursor. Given such problems, it is important that designers can exploit logic to identify alternative solutions. For example, logic can be used to describe alternative warning displays without considering the details of particular presentation devices (Johnson, 1991). The following clauses state that the request failure icon has dimensions that occupy half of the width and one twentieth of height of the browser's window. The icon is positioned at Cartesian co-ordinates (0.5, 0.3) and contains the text 'Unable to connect to remote host':

$$dimension(request_failure, 0.5, 0.05). \quad (7)$$
$$position(request_failure, 0.5, 0.3). \quad (8)$$
$$text(request_failure, \text{'Unable to connect to remote host'}). \quad (9)$$

These graphical descriptions can be integrated into dialogue requirements:

$$larger_failure_warning(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EF\,(not(fetch(Url)) \wedge AX\ present(request_failure) \wedge dimension(request_failure, 0.5, 0.3)). \quad (10)$$

An important point here is that empirical analyses are required in order to determine whether operators are liable to observe warnings placed in particular positions with particular dimensions on their display. Johnson (1993a) describes prototyping tools that can be used to directly derive partial implementations from clauses such as (10). In contrast, this paper goes on to explore further techniques that can be used to reduce the impact of unpredictability in distributed systems. A limitation with the previous requirement is that users are often distracted by some secondary task while operating systems such as Mosaic. Under such circumstances, additional prompts are required to ensure that users recognise the failure of their request. For instance, Netscape users must select a button to confirm the 'Unable to locate host' warning:

$$failure_confirmation(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EF\,(not(fetch(Url)) \wedge AX\ present(request_failure) \wedge user_selects(request_failure)) \quad (11)$$

Unfortunately, a number of criticisms can be made about the temporal properties of these presentation techniques. The warnings provided for transmission failure only seem to occur if the problem is immediately detected, $EX(not(fetch(Url)))$. Initial trials with the Mosaic browser indicate that it will 'hang' if the failures occur at some mid-point during transmission, $EF(not(fetch(Url)))$. The use of the logic captures this problem in a precise and concise way. In multi-processing environments, there seem to be few technical reasons for this behaviour.

## 6. Unpredictability And Presentation Delay

The second cause of unpredictability, identified in clause (5), can be termed presentation delay. This occurs when information is successfully retrieved from a remote site but its presentation is delayed by the demands of processing text, graphics etc. This leads to unpredictability because people cannot determine how long it will take to assemble a page before they select it. Users can be discouraged from exploring Internet resources if they are forced to commit large amounts of time whenever information is requested. A number of techniques can be used to reduce the unpredictability caused by such processing delays. For instance, the page shown in figure 2 is composed of some text and an image of a monk. When requesting this page using Mosaic, the text is immediately processed. The image takes longer; a temporary icon is presented until the relevant data is processed. This supports predictability. The number of temporary icons gives the user some idea of the amount of time required to retrieve and present the information on the page:

$$update(Url) \Leftarrow fetch(Url) \wedge AX((present_text(Url) \wedge present_temporary_icon(Url))\ U\ present_images(Url)) \quad (12)$$

This approach can be improved by separating the retrieval of text and graphics This is a relatively simple fix in terms of systems engineering but it can have a significant effect upon 'usability'. One of the benefits of the distributed architecture behind the Web is that the text and graphics may be located on different machines. If the graphics machine goes down then, under the approach described in (12), the browser would 'hang'. It would not be possible to present_images(Url) until the machine came back up. By splitting the retrieval and presentation processes, the page can still be viewed even if the temporary icons are all that is available for some of the images. Mosaic and Netscape, therefore, not only support predictability through the use of temporary icons but also provide a high degree of fault tolerance:

$$\begin{aligned} fault_tolerant_update(Url) \Leftarrow\ & fetch_text(Url) \wedge fetch_images(Url) \wedge \\ & AX(present_text(Url) \wedge present_temporary_icon(Url)\ U \\ & (present_images(Url) \vee present(request_failure))) \end{aligned} \quad (13)$$

Unfortunately, the approach described above only addresses some of the problems caused by unpredictability. Contextual information about the costs of retrieval and presentation are provided after the user has made a commitment to accessing the entire page. This increases the frustration associated with distributed retrieval; temporary icons enable the user to predict that a large amount of time will be wasted while graphical images are needlessly processed by the browser. Further problems occur if the user issues requests to access other resources before the original retrieval is completed. It is difficult to predict when the second request, *user_selects(Label2)*, will be processed:

$$\begin{aligned} unpredictable_request(Url2) \Leftarrow\ & user_selects(Label) \wedge linked(Label, Url) \wedge \\ & fetch_text(Url) \wedge fetch_images(Url) \wedge \\ & EF(user_selects(Label2) \wedge linked(Label, Url2)) \wedge \\ & (present_text(Url) \wedge present_temporary_icon(Url)\ U \\ & (present_images(Url) \vee present(request_failure)))) \end{aligned} \quad (14)$$

This problem arises because browsers will, typically, hand over the processing of graphics, sound and video to helper applications. Once this has been done, any subsequent interaction will be ignored until the processing has been completed. There is no way for the user to detect this 'hand-over' from the information presented by browsers such as Mosaic and Netscape. This helps to create the 'bizarre' temporal behaviour mentioned in the introduction. Users are unable to predict when their input will take effect because they cannot identify the critical intervals before the helper applications are called. This can be resolved in a number of ways. For instance, all subsequent input might be ignored until the initial request has been completed:

$$\begin{aligned} &ordered_request_processing(Url, Url2) \Leftarrow user_selects(Label) \wedge \\ &\quad linked(Label, Url) \wedge fetch_text(Url) \wedge fetch_images(Url) \wedge \\ &\quad AX(user_selects(Label2) \wedge linked(Label2, Url2) \wedge (present_text(Url) \wedge \\ &\quad present_temporary_icon(Url) \wedge not(fetch_text(Url2) \wedge fetch_images(Url2)) \; U \\ &\quad (present_images(Url) \vee \; present(request_failure)))) \end{aligned} \quad (15)$$

The designers of Mosaic and Netscape have adopted an alternative solution. They provide a mechanism for halting a request. This enables users to interrupt the processing of a resource in order to revise their initial selection:

$$\begin{aligned} &support_for_halt(Url) \Leftarrow fetch_text(Url) \wedge fetch_images(Url) \wedge \\ &\quad AX(present_text(Url) \wedge present_temporary_icon(Url) \; U \\ &\quad (present_images(Url) \vee \; present(request_failure) \vee user_selects(halt))) \end{aligned} \quad (16)$$

This is a partial solution. For instance, the following clause describes a trace in which the halt request is issued before all of the information is retrieved from the remote servers. Under such circumstances, there will be an inevitable delay while the browser handles the information that has already been requested from other sites:

$$\begin{aligned} &ordered_request_processing(Url, Url2) \Leftarrow \\ &\qquad user_selects(Label) \wedge \; linked(Label, Url) \wedge send_request(Url) \wedge \\ &\qquad AX(user_selects(Label2) \wedge linked(Label2, Url2) \wedge \\ &\qquad EF(fetch_text(Url) \wedge \; fetch_images(Url))) \end{aligned} \quad (17)$$

Attempts to abort NCSA Mac Mosaic (Version 1.0.3) caused the system to crash. Later versions of the browser have resolved this problem. They are, however, still subject to the variable termination delays that lead to unpredictability. Until such usability problems are addressed, there will continue to be moments of intense frustration and irritation when using the Web.

## 7. Conclusion

This paper has argued that communications failures, network loading and processing overheads make it difficult for users to predict the effects of their commands. This leads to frustration and error when delays occur during retrieval operations. These problems are exacerbated by the fact that most users cannot be expected to have a detailed understanding

of network engineering. As a consequence, the temporal properties exhibited by distributed systems often appear to be 'bizarre' and 'unreasonable'. In order for designers to address these concerns, it is important that they are provided with some means of reasoning about the complex temporal properties of user interfaces. This paper has exploited CTL to examine a range of interaction techniques, such as incremental presentation and request termination, that can be used to reduce the impact of unpredictability in distributed systems.

Temporal logic does not provide a panacea for the development of distributed, interactive systems. In particular, it is possible for designers to specify presentation techniques that exacerbate rather than resolve the problems of unpredictability. Fortunately, tool support can be recruited to address this concern (Johnson, 1993a). Executable temporal logics can directly derive prototype systems from abstract clauses such as (15). Further work intends to develop these tools so that they can be used to assess alternative designs for a range of Web browsers.

The CTL notation cannot be used to represent real-time properties. We are addressing this limitation as part of the Joint Council Initiative's Temporal Aspects of Usability project. Formal notations, including temporal logics, are being used to represent the temporal behaviours that are predicted to affect the usability of interactive systems. Simulation tools are then used to provide experimental evidence as to whether these behaviours do indeed have an impact upon the successful operation of human-computer interfaces.

Finally, it is important to emphasise some of the wider issues that arise from this research. Many of the usability problems that occur during interaction with distributed systems are directly connected to the physical characteristics of the underlying networks. We have argued that users cannot be expected to possess a detailed knowledge of these characteristics. Interfaces must be developed to hide the underlying applications. In order to do this, human factors specialists will themselves have to develop a clear understanding of increasingly complex systems. It, therefore, follows that we must not only 'know our user' but also 'know our system' if we are to improve the usability of an increasing number of distributed applications.

## Acknowledgements

NCSA Mosaic is a trademark of the National Center for Computer Applications. Netscape is a trademark of the Netscape Communications Corporation. This work was conducted as part of the Temporal Aspects of Usability project SPG9201233, funded by the Joint Council Initiative in Cognitive Science and Human Computer Interaction.

## References

Berners-Lee, T.J., Cailliau, R. and Groff, J.-F. (1992) *The World Wide Web*, Computer Networks And ISDN Systems, (25):454--459.

Clarke, E.M. and Emerson, E.A. (1982) The Design And Synthesis Of Synchronisation Skeletons Using Branching Time Temporal Logic. In D. Kozen (ed.) *Logic Of Programs 1981*, 52--71, LNCS 131, Springer Verlag, Berlin, Germany.

Dix, A., Harrison, M., Runciman, C. and Thimbleby, H. (1987) Interaction Models And The Principled Design Of Interactive Programs. In H.K. Nichols and D. Simpson, editors,

*Proceedings Of The First European Software Engineering Conference*, LNCS 289, 118--126, Springer Verlag, Berlin, Germany.

Harrison, M. and Thimbleby, H. (1989), editors, *Formal Methods In Human Computer Interaction*, Cambridge University Press, Cambridge, United Kingdom.

Johnson, C.W., McCarthy, J. and Wright, P.C. (1995) *Using Petri Nets To Support Natural Language In Accident Reports*. Ergonomics, (38)6:1265--1283.

Johnson, C.W. (1991) Declarative Graphics And Dynamic Interaction. In F.H. Post and W. Barth, editors, *Eurographics '91*, 195--207. Elsevier, North Holland.

Johnson, C.W. (1992) A Formal Approach To The Integration Of Human Factors And Systems Engineering, DPhil thesis, University Of York, York, United Kingdom. Available as technical report YCST 95/05.

Johnson, C.W. (1993) A Formal Approach To The Presentation of CSCW Systems. In J.L. Alty, D. Diaper and S. Guest, editors, *People And Computers VIII*, 335--352, Cambridge University Press, Cambridge, United Kingdom.

Johnson, C.W. (1993a) *A Probabilistic Logic For The Development Of Safety-Critical Interactive Systems*, International Journal of Man-Machine Studies, (38)2:333--351.

Johnson, C.W. (1994) The Formal Analysis Of Human Computer Interaction During Accident Investigations. In G. Cockton, S.W. Draper and G.R.S. Weir, editors, *People And Computers IX*, 285--300, Cambridge University Press, Cambridge, United Kingdom.

Kuhmann, W. (1989) *The Stress Inducing Properties Of System Response Times*, Ergonomics, (32)3:271--280.

Miles, V., Johnson, C.W., McCarthy, J. and Harrison, M. (1991) Supporting Prediction In Complex Dynamic Systems. In D. Diaper and N. Hammond (editors), *People And Computers VI*, 133--144, Cambridge University Press, Cambridge, United Kingdom.

Norman, D. (1990) *The 'Problem' With Automation: Inappropriate Feedback And Interaction Not 'Over-Automation'*. In D.E. Broadbent , J. Reason and A. Baddeley (editors) Human Factors In Hazardous Situations, 137--145, Clarendon Press, Oxford, United Kingdom.

Pejtersen, A.M., (1989) *A Library System For Information Retrieval Based On A Cognitive Task Analysis And Supported By An Icon Based Interface*. In ACM SIGIR Proceedings, Boston. ACM Press, New York, United States of America.

Schofield, S. (1994) *The UK Internet Book*. Addison Wesley, Wokingham, United Kingdom.