Electronic Gridlock, Information Saturation and the Unpredictability Of Information Retrieval Over the World Wide Web

Chris Johnson

Glasgow Accident Analysis Group,
Department of Computing Science, University of Glasgow,
Glasgow, G12 8QQ, Scotland.

Tel: +44 041 330 6053, Fax: +44 041 330 4913
EMail: johnson@dcs.glasgow.ac.uk
http://www.dcs.gla.ac.uk/~johnson

Abstract. New groups of users are learning to exploit the facilities provided by Mosaic, Netscape and the Internet Explorer. This rapid growth in the Internet has exposed users to interaction problems that rarely occur with stand-alone computer systems. For example, electronic gridlock occurs when remote systems cannot satisfy the number of demands that users make upon their services. This leads to frustration and error as retrieval delays increase. In contrast, information saturation occurs when remote sites provide so much data that users cannot extract important documents from a mass of background 'noise'. Compounding both of these problems is the unpredictability that characterises web-based systems. Users simply cannot determine whether a query will result in a long delay or an instantaneous response. These are non-trivial problems because it can be difficult for designers to represent and reason abut the complex temporal nature of interaction with distributed systems. This paper argues that temporal logics can be used to address the problems caused by electronic gridlock, information saturation and unpredictability with web-based information systems. These notations are appropriate because they describe **what** an interface must do in order to support web-based interaction. Designers are not forced to consider **how** such requirements might be satisfied using particular protocols or browsing mechanisms. Low level implementation details can be gradually introduced as development progresses. Unfortunately, formal methods do not describe **why** such requirements are appropriate. Later sections of the paper, therefore, describe how literate specifications techniques can be used to provide links to the informal argumentation and empirical evidence that supports the formal design of interactive systems.

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

1. Introduction

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 mass-market browsers has encouraged the active participation of new groups of users (Berners-Lee, Cailliau and Groff, 1992).

The very success of distributed computing has, however, exacerbated a range of existing usability problems. For example, retrieval delays have reached a stage where many sites suffer from a form of 'electronic gridlock' (Johnson, 1995). At peak times it can take thirty or forty minutes to download relatively small files. Other usability problems arise from the sheer volume of information that can be accessed from remote sites (Schofield, 1994). When users select a link, they have little idea if the information that they have selected will be relevant to their task.

These problems are exacerbated by the unpredictable nature of information retrieval over the web. Users, typically, cannot predict the time taken to retrieve a file. Server performance varies according to the time of day, local network capacity, the availability of mirror sites etc. Unpredictability arises because many users have little understanding of the underlying technology that supports interaction with the web. 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.

1.2. The Problems: Electronic Gridlock

The term 'electronic gridlock' is used to refer to situations in which networks and servers cannot cope with the amount of information that is being requested from them. Recent innovations in caching, including the widespread use of mirror sites, have done much to increase the efficiency of network communications. However, the majority of Internet users still suffer from the 'bottlenecks' created by modem and ethernet connections. As more and more people access a greater and greater volume of data over these low bandwidth systems, there has been an increase in the amount of time that is required to download information (Johnson, 1995, Johnson and Kavanagh, 1996). Kuhmann (1989) argues that these retrieval delays have a profound impact upon the quality of interaction with distributed systems. Prolonged periods waiting to retrieve a file frequently leads to frustration and error.

1.2. The Problems: Information Saturation

The term 'information saturation' refers to situations in which users simply cannot cope with the volume of information that can be accessed from remote sites. Hiltz and Turoff (1985) argue that this problem stems from 'information entropy'; electronic documents are insufficiently well structured for users to select the items that they need. This problem affects a wide range of applications, not just web based systems. For example, it can often take ten or twenty minutes to extract critical items from the morning's electronic mail messages. The use of in-page indexing and graphical navigation techniques through image maps helps to address these problems for web users. However, the low marginal cost associated with creating web documents and the large number of potential authors helps to offset any long-term benefits that these techniques might offer. Search engines also help to address the problems of information saturation but even these systems are being swamped by the difficulty of effectively indexing the millions of documents that are created every day.

1.3. The Problems: Unpredictability

The problems of electronic gridlock and information saturation are made much worse for many users because of the unpredictability of interaction with the world wide web. Unpredictability occurs 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). Figure 1 illustrates the way in which the architecture of interaction with distributed information retrieval systems leads to unpredictability.

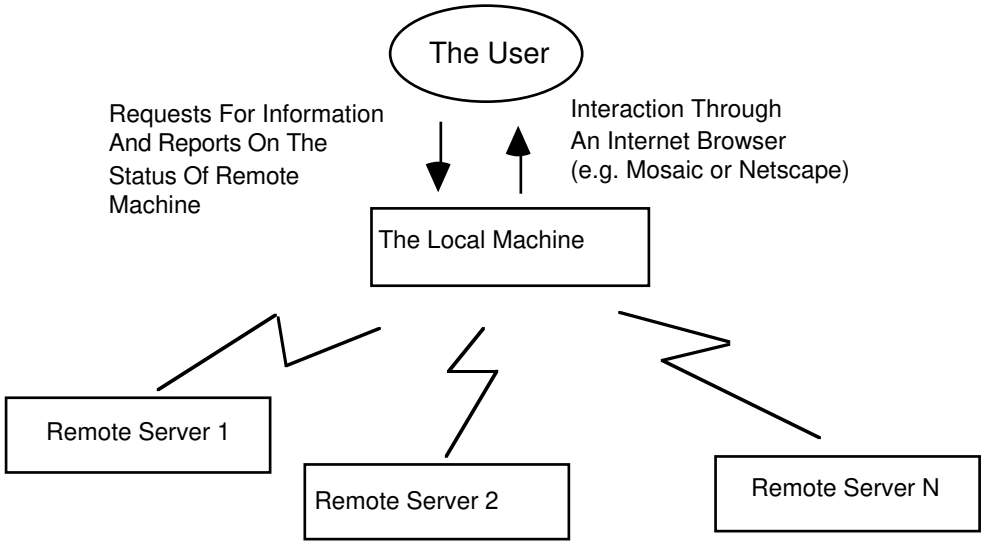


Figure 1: Information Retrieval Over A Distributed Network

Browsers translate user request for information into the communications primitives that are necessary to transfer relevant data from remote servers. These 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. 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 that 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. However, these problems are also linked to the nature of the requested resource. For instance, users may not be able to predict the time necessary to retrieve a particular file given a simple hypertext label. This paper argues that designers can exploit formal notations to reason about the presentation techniques that reduce these problems.

1.2. Outline Of The Paper

This section has briefly introduced the problems of electronic gridlock, information saturation and unpredictability for the world wide web. Section 2 uses a temporal extension to first order logic as a means of assessing the causes of electronic gridlock. In particular, it is argued that this notation provides suitable abstractions for reasoning about the geographical and temporal factors that lead to delays during information retrieval over the web. Section 3 builds on the analysis and argues that the simple temporal logic of Section 2 cannot easily be applied to reason about the unpredictable nature of interaction with distributed systems. In particular, a branching time logic is required if designers are to analyse the many different failures that can prevent users from accessing remote sites. This section also identifies a number of techniques that can be used to combat the unpredictability of web-based interaction. Section 4 argues that the techniques which are advocated as a means of reducing unpredictability do not address the problem of information saturation. In particular, they do not help users to assess the value of remote resources. The branching time temporal logic is, therefore, used to describe a number of presentation techniques that avoid this limitation. Unfortunately, none of the formal techniques used in Sections 2, 3 and 4 provide any evidence that a particular implementation will actually improve the usability of an interface. Section 4, therefore, goes on to describe how empirical evidence can be used to support the formal analysis of web-based systems Section 5 presents the conclusions that can be drawn from this work and suggests areas for further research.

2. Analysing Electronic Gridlock

Formal methods provide designers with means of abstracting away from the mass of detailed information that is available on the world wide web. Logic can be used to reason about underlying problems, such as electronic gridlock and information saturation, rather than focusing in upon surface characteristics of particular browsers. For example, a Horn clause notation might be used to represent the interaction between a user's request and a server's response. 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) . The following formula states that a resource is retrieved if the user selects an area of text which is linked with that resource. The term *url* stands for Uniform Resource Locator. The *http* addresses that are so familiar to web users provide examples of these addresses:

$$display_url(url) \Leftarrow user_selects(label) \wedge linked(label, url) \wedge fetch(url) \wedge present(url) \quad (1)$$

Unfortunately, there is no notion of time in classical logic. The previous clause fails to capture the potential delay that can occur between the user selecting the label and the resource being returned from a remote machine. This is a significant limitation because retrieval delays have a considerable impact upon the design of web-based systems. For example, many users will abandon their requests rather than tie up their machine while large files are downloaded. Figure 2 illustrates this point. The graph shows the times that are required to access different lengths of video clips using the Netscape browser on a stable network. As can be seen, the bigger the file, the more time the user needs to complete their task.

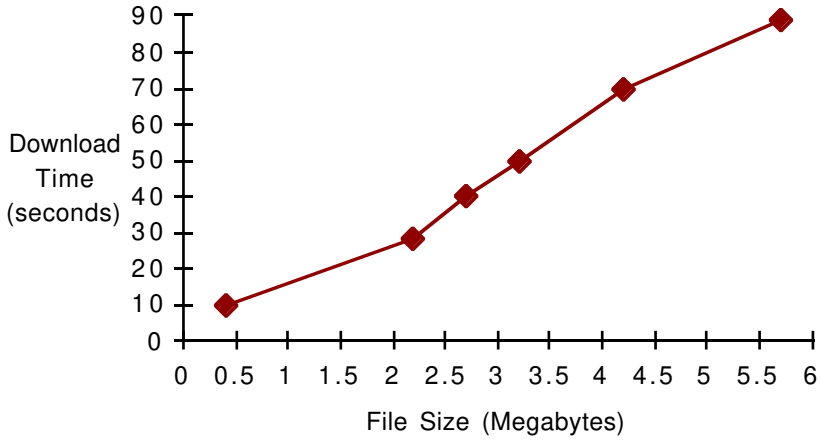


Figure 2: Variable Download Latency With File Size.

Several notations can be used to represent the impact of time upon information retrieval. For example, designers might exploit temporal extensions to graphical notations, such as timed Petri Nets. Alternatively, real-time constraints can be introduced into first order logic. For example, the following clause represents the ninety second delay that might be expected for the largest of the files shown in Figure 2. The user requests the resource at 12:00:00hrs but it is not presented until 12:01:30hrs:

$$\begin{aligned} &\textit{acceptable_delay(url, 12_00_00, 12_01_30)} \Leftarrow \\ &\quad \textit{user_selects(label, 12_00_00)} \wedge \textit{linked(label, url, 12_00_00)} \wedge \\ &\quad \textit{fetch(url, 12_00_00)} \wedge \textit{present(url, 12_01_30)} \end{aligned} \tag{2}$$

Such clauses quantify the real-time delays that occur during information retrieval tasks. In this case the delay was only ninety minutes. This might be deemed 'acceptable' for some applications. This might not be the case if delays extended beyond ten minutes. Under such circumstances, designers might consider splitting resources, such as video files, into smaller segments. The following clause states that delays of ten minutes are considered to be 'unacceptable':

$$\begin{aligned} &\textit{unacceptable_delay(url, 12_00_00, 12_10_00)} \Leftarrow \\ &\quad \textit{user_selects(label, 12_00_00)} \wedge \textit{linked(label, url, 12_00_00)} \wedge \\ &\quad \textit{fetch(url, 12_00_00)} \wedge \textit{present(url, 12_10_00)} \end{aligned} \tag{3}$$

Previous clauses such as (2) and (3) help to address the problems created by electronic gridlock because they explicitly represent the boundaries for acceptable retrieval delays. If 'average' delays increased beyond these limits then designers must re-design their resources, for instance, by reducing the amount of graphics or by editing audio and video clips. These clauses also raise a number of further issues. For instance, they do not state the precise conditions under which timings may be taken for 'average' delays. This is a significant limitation because download times vary between different parts of the globe. It would be of little benefit if some users had an *acceptable_delay* but had no interest in a resource while others had to suffer an *unacceptable_delay* for information that they really needed. For example, I can rapidly access meteorological information about Seattle but frequently have great difficulty in accessing information about the West of Scotland. Figure 3 presents the results of a trial that we conducted to illustrate the importance of geographical location for retrieval delays. A number of users were asked to download the same file at approximately the same time of day.

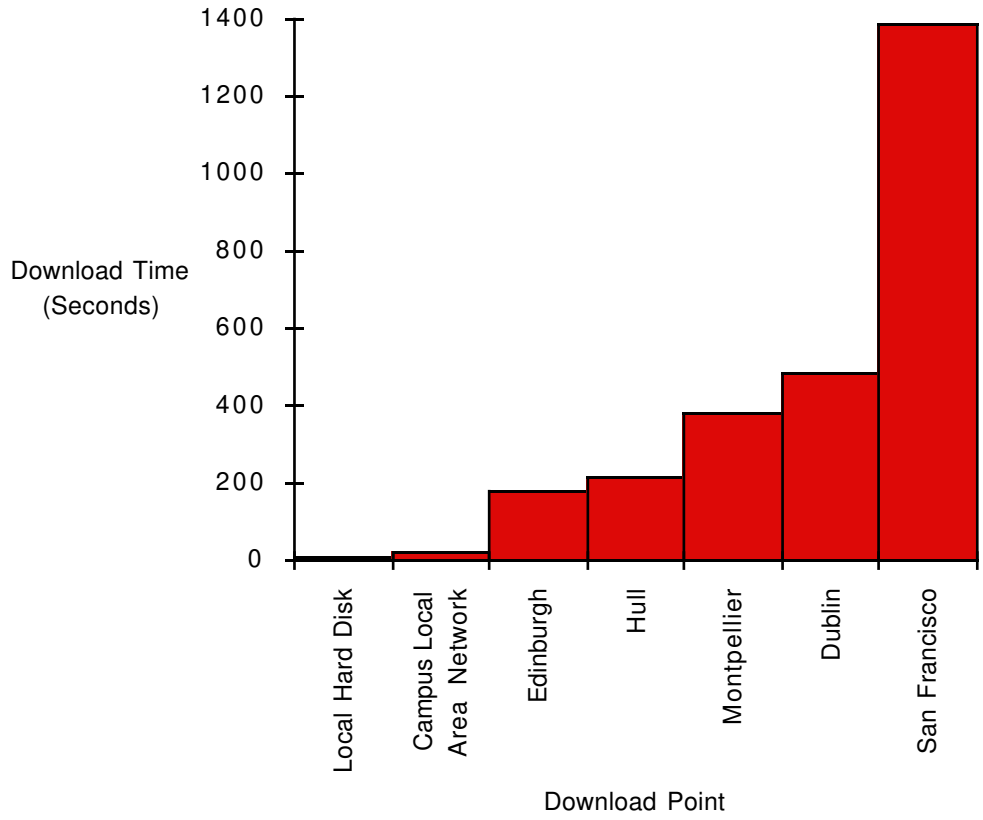


Figure 3: Variable Download Latency With Users' Location.

As can be seen from the graph, retrieval delays are determined by the user's location. It is important to note that this does **not** mean that delays are not determined by the user's geographical distance from a resource. Dublin is 200 miles away from Glasgow and yet it takes longer to retrieve the file than Montpellier which is 900 miles away. This is due to a 256kbit line between Ireland and the United Kingdom. If designers were providing a resource for use by Dubliners then the web page would have to be designed as if Dublin were geographically further away than Montpellier. This implies that any attempt by designers to define acceptable and unacceptable delays must be indexed by the relative position of the user to the resource. Logic provides a means of representing these objectives or criteria for the development of a web-based system:

$$\begin{aligned} &unacceptable_delay(url, 12_00_00, 12_10_00, hull) \Leftarrow \\ &\quad user_selects(label, 12_00_00) \wedge linked(label, url, 12_00_00) \wedge \\ &\quad user_position(hull, 12_00_00) \wedge resource_position(glasgow, 12_00_00) \wedge \\ &\quad fetch(url, 12_00_00) \wedge present(url, 12_10_00) \end{aligned} \quad (4)$$

$$\begin{aligned} &acceptable_delay(url, 12_00_00, 12_10_00, san_francisco) \Leftarrow \\ &\quad user_selects(label, 12_00_00) \wedge linked(label, url, 12_00_00) \wedge \\ &\quad user_position(san_fransisco, 12_00_00) \wedge resource_position(glasgow, 12_00_00) \wedge \\ &\quad fetch(url, 12_00_00) \wedge present(url, 12_10_00) \end{aligned} \quad (5)$$

Too many designers have, perhaps, become preoccupied with the quantity of information that can be accessed over the world wide web. Pages have become crammed with complex graphical images, trivial Java Applets and unnecessary frames. As a result, many users feel exasperated at the amount of time that is wasted while huge files are retrieved from distant sites (Johnson, 1995). Clauses (4) and (5) show how designers can use a formal notation to explicitly represent the temporal and geographical properties that determine 'quality of service'. The introduction of additional features is only justified if users can access particular resources within an acceptable period of time. If these features have to be introduced and response time fall outside *acceptible* limits then designers must consider the use of mirror sites to reduce the impact of physical, or rather network, distance upon retrieval delays. Alternatively, improvements can be made to the supporting infrastructure. For example, the delays between Glasgow and Dublin, shown in Figure 3, are being addressed by the provision of a 512kbit line across the Atlantic.

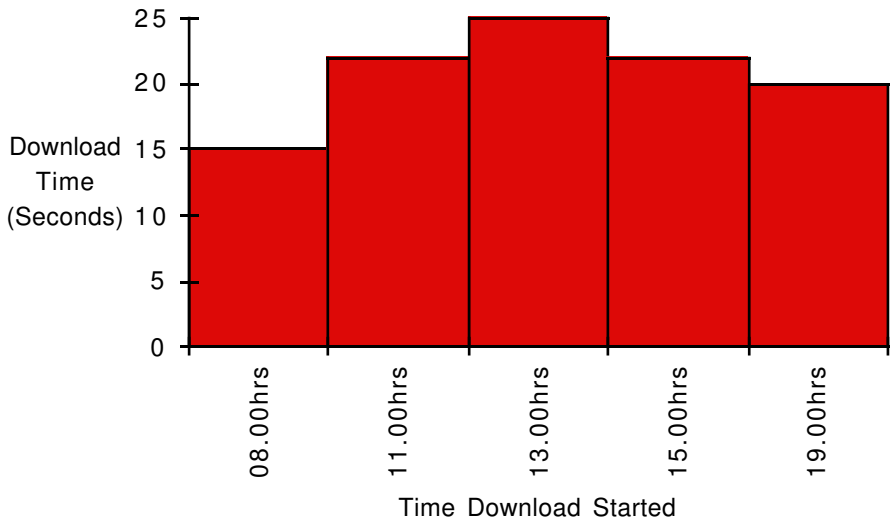


Figure 4: Variable Download Latency With Time Of Day.

There are, however, further issues that must be considered if designers are to address the problems created by electronic gridlock. Figure 4 presents the delays that occur when accessing a video file over the Glasgow University network at different times during the day. Variations stem from both network and server loading. From this it follows that acceptable and unacceptable delays are not only determined by file size and the relative position of the user, they are also determined by the time of day when a request is issued. For instance, a ten minute delay might be quite acceptable at midday when machines and networks are heavily loaded. The same delay might be less acceptable at nine in the morning when less demands are placed on supporting infrastructures:

$$\begin{aligned} &\textit{unacceptable_delay_1(url, 09_00_00, 09_10_00, hull)} \Leftarrow \\ &\quad \textit{user_selects(label, 09_00_00)} \wedge \textit{linked(label, url, 09_00_00)} \wedge \\ &\quad \textit{user_position(hull, 09_00_00)} \wedge \textit{resource_position(glasgow, 09_00_00)} \wedge \\ &\quad \textit{fetch(url, 09_00_00)} \wedge \textit{present(url, 09_10_00)} \end{aligned} \tag{6}$$

$$\begin{aligned} &\textit{acceptable_delay(url, 12_00_00, 12_10_00, hull)} \Leftarrow \\ &\quad \textit{user_selects(label, 12_00_00)} \wedge \textit{linked(label, url, 12_00_00)} \wedge \\ &\quad \textit{user_position(hull, 12_00_00)} \wedge \textit{resource_position(glasgow, 12_00_00)} \wedge \\ &\quad \textit{fetch(url, 12_00_00)} \wedge \textit{present(url, 12_10_00)} \end{aligned} \tag{7}$$

Clauses (6) and (7) help to specify the boundaries of acceptable and unacceptable delays for particular locations at particular times of the day. If these limits were regularly broken then, as before, designers might consider dividing resources into smaller files with faster access times. The number of graphical images in a page might be reduced. Mirror sites might be provided. However, a number of limitations restrict the utility of clauses, such as (6) and (7), for reasoning about information retrieval systems. In particular, they only describe fixed intervals of time. Designers would be forced to introduce additional clauses to specify that delays of eleven minutes, twelve minutes, thirty minutes, one hour... were also unacceptable. Such problems can be avoided by introducing variables to stand for precise moments of time. The following clause states that for all times, *t,* there is an unacceptable delay in presenting a *url* if the resource is selected at *12:00:00hrs* and presented at *t* and *t* occurs more than ten minutes after *12:00:00* :

$$\begin{aligned} &\forall\, t: \textit{unacceptable_delay(url, 12_00_00, t, hull)} \Leftarrow \\ &\quad \textit{user_selects(label, 12_00_00)} \wedge \textit{linked(label, 12_00_00)} \wedge \\ &\quad \textit{user_position(hull, 12_00_00)} \wedge \textit{resource_position(glasgow, 12_00_00)} \wedge \\ &\quad \textit{fetch(url, t)} \wedge \textit{present(url, t)} \wedge \textit{after(12_10_00, t)} \end{aligned} \tag{8}$$

The same abstraction techniques can be used to represent a range of geographical locations. This avoids the needless duplication of clauses such as (4) and (5) for many different points in the globe. The following clause states that delays of more that two hours are unacceptable at any time, anywhere in the world:

$$\forall t,t',c: unacceptable_delay(url,\ t,\ t',\ c) \Leftarrow$$
$$user_selects(label, t) \wedge linked(label, t) \wedge$$
$$user_position(c, t) \wedge resource_position(glasgow, t) \wedge$$
$$fetch(url, t') \wedge present(url, t') \wedge after(t+2_00_00, t') \quad (9)$$

The important point about clauses (8) and (9) is that they force designers to focus upon quality of service issues for web based interaction. It is of little use providing rich multimedia resources over the Internet if no user can afford to wait while they download. In this sense, (8) and (9) illustrate how formal notations can be used to establish high level objectives for Internet providers. If such standards cannot be achieved then designers must take specific steps to combat electronic gridlock by splitting resources, reducing multimedia information and constructing mirror sites.

3. Analysing Unpredictability

Our previous formalisations have not captured the tremendous unpredictability that frustrates everyday retrieval tasks. For instance, a request might block owing to site failure, it might be stalled by heavy server loading, it might fail because a resource is missing. In order to capture these various eventualities, we must recruit a notation that can describe a range of alternative futures. For example, Clarke and Emerson's (1982) Computation Tree Logic (CTL) uses a branching model of time to represent a number of alternative traces of possible interaction. This approach is illustrated in figure 5. 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.

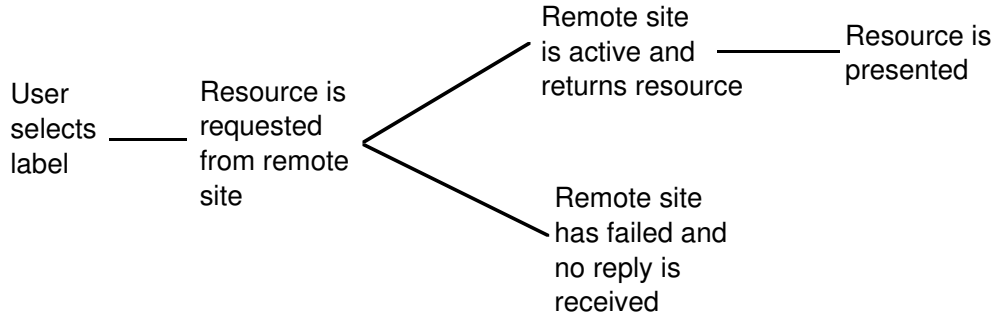


Figure 5: CTL's Branching View Of Possible Futures

Readers who are more concerned about the application rather than the formal foundations of the notation should move to the next section. CTL represents future time in terms of a number of possible sequences of states . This contrasts with the logic of (8) and (9) where the notation is built around a single sequence of future states. Following Clarke and Emerson, our description of CTL uses p to denote an atomic proposition. p is a formula, $f_\{i\}$ denotes a formula or sub-formula. $f_\{1\} \wedge f_\{2\}$ is a formula denoting conjunction, $\neg f$ is a formula denoting negation. $EX\ F_\{j\}f$ is a formula; f is true in the next state of possible future $F_\{j\}$. $A[f_\{1\}\ U\ f_\{2\}]$ is a formula; there exists a prefix for all futures such that $f_\{2\}$ is true in the last state of the prefix and $f_\{1\}$ holds at all other states in that prefix. $E[f_\{1\}Uf_\{2\}]$ is a formula; there exists a prefix for a future such that $f_\{2\}$ is true in the last state of the prefix and $f_\{1\}$ holds at all other states in that prefix. The semantics of CTL are represented by a model $M = (S, F_\{1\},...,F_\{k\}, L)$ where: S is a countable set of states; $F_\{i\} \subset S \times S$ is a relation on S giving the transitions over a possible future i ; L is an assignment of true atomic propositions in each state. Let $F = F_\{1\} \cup F_\{2\} \ldots \cup F_\{k\}$ and $\forall s \in S,\ \exists s'$ in $S\ (s, s') \in F$. A trace is an infinite sequence of states $(s_\{0\},\ s_\{1\},\ s_\{2\},\ \ldots)$ in $S^{\wedge}w$ where $\forall i(s_\{i\},\ s_\{i+1\}) \in F$. To any structure M and state $s \in S$ of M there corresponds a tree of possible future states with root labelled $s_\{0\}$ such that $s \rightarrow s_\{i\}$ is a branch in the tree iff $(s, s_\{i\}) \in F$. $M,\ s_\{0\} \models f$ denotes that formula f is true in state $s_\{0\}$ of future tree M . The M is omitted if it is understood by the preceding context. The $\models$ relation can be defined inductively:

- $s_\{0\} \models p$ iff $p \in L(s_\{0\})$;
- $s_\{0\} \models \neg f$ iff not $(s_\{0\} \models f)$;

- $s_\{0\} \models f_\{1\} \wedge f_\{2\}$ iff $s_\{0\} \models f_\{1\}$ and $s_\{0\} \models f_\{2\}$;
- $s_\{0\} \models EX\ F_\{k\}f$ iff $\exists\ s_\{1\}$ *in* $S(s_\{0\}, s_\{1\})$ *in* $F_\{k\} \wedge s_\{1\} \models f$;
- $s_\{0\} \models A[f_\{1\} U f_\{2\}]$ iff for all traces
  $\exists\ i\ [i \geq 0 \wedge s_\{i\} \models f_\{2\} \wedge \forall j(0 \leq j \wedge j < i \wedge s_\{j\} \models f_\{1\})]$;
- $s_\{0\} \models E[f_\{1\} U f_\{2\}]$ iff for some trace
  $\exists\ i\ [i \geq 0 \wedge s_\{i\} \models f_\{2\} \wedge \forall j(0 \leq j \wedge j < i \wedge s_\{j\} \models f_\{1\})]$;
- $f_\{1\} \vee f_\{2\} \equiv \neg(\neg f_\{1\} \wedge \neg f_\{2\})$;
- $f_\{1\} \Rightarrow f_\{2\} \equiv \neg f_\{1\} \vee f_\{2\}$;
- $EFf_\{1\} \equiv E[true\ U f_\{1\}]$, in some trace there exists a state where $f_\{1\}$ is true;
- $AGf_\{1\} \equiv \neg\ EF\ \neg f_\{1\}$ states that $f_\{1\}$ is true for all states in all traces;
- $AXf \equiv \neg\ EX \neg f_\{1\}$ states that $f_\{1\}$ holds in all next states.

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) without representing a particular moment in time which was a problem in clauses (6) and (7). The following formula requires that the user selects the label before the corresponding resource is retrieved:

$$strong_predictability(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge AX\ (fetch(Url) \wedge AX\ present(Url)) \quad (10)$$

This formula represents an ideal state of affairs; the resource would be retrieved immediately after it had been requested in every trace of interaction. As we have seen, however, 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'):

$$weak_predictability(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EX\ (fetch(Url) \wedge AX\ present(Url)) \quad (11)$$

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:

$$unpredictable(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EX\ not(fetch(Url) \wedge AX\ present(Url)) \quad (12)$$

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. 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:

$$unpredictable(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EX\ (not(fetch(Url)) \vee EX\ not(present(Url))) \quad (13)$$

Logic formulae, such as (13), 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:

$$failure_warning(Url) \Leftarrow user_selects(Label) \wedge linked(Label, Url) \wedge EF\ (not(fetch(Url)) \wedge AX\ present(request_failure)) \quad (14)$$

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 (14). 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, designers might specify that users must select a button to confirm warnings about remote site failure. Such techniques reduce the likelihood that users will miss the warning and continue to request unavailable information:

$$\begin{aligned} &\mathit{failure_confirmation(Url)} \Leftarrow \mathit{user_selects(Label)} \wedge \\ &\quad \mathit{linked(Label,\ Url)} \wedge \mathit{EF\ (not(fetch(Url))} \wedge \\ &\quad \mathit{AX\ \ present(request_failure)} \wedge \mathit{user_selects(request_failure))} \end{aligned} \qquad (15)$$

The previous clause represents a high-level approach that might be implemented in a number of different ways. From this it follows that the use of a logic notation does not guarantee the 'usability' of a particular interface. Some implementations of the formal requirement will be better than others. For example, Netscape v3.0 exploits a variation of the approach advocated in (15). Users are required to click a button labelled OK after being presented with the following warning:

> Netscape is unable to locate the server: www.giraffe.gla.ac.uk. The server does not have a DNS entry. Check the server name in the location (URL) and try again. the domain name server cannot locate a requested machine.

Most users have little or no idea about the importance of DNS entries. This warning, therefore, causes considerable confusion when learning to use the browser. Other problems have arisen when browsers fail to satisfy the temporal properties described by (15). For example, early versions of the Mosaic browser would only provide warnings of transmission failures if the problem was immediately detected, $EX(not(fetch(Url))$. The browser would 'hang' if the failure occurred at some mid-point during transmission, $EF(not(fetch(Url))$. In multi-processing environments, there were few technical reasons for this behaviour and subsequent versions of the browser allow the termination of a request at any point in a transfer. The use of the logic, and in particular the distinction between *EX* and *EF*, captures this problem in a precise and concise way.

The second cause of unpredictability, identified in clause (13), 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 example, interface designers can provide some indication about the size of remote resources. This does not imply that designers must include the file size of every item that can be accessed from a web page. For instance, the interface shown in Figure 6 provides schoolchildren with access to videos about the Romans in Scotland. Each clip is accompanied by an indication of the running time which has more meaning for the intended users than a fie size measured in bytes.

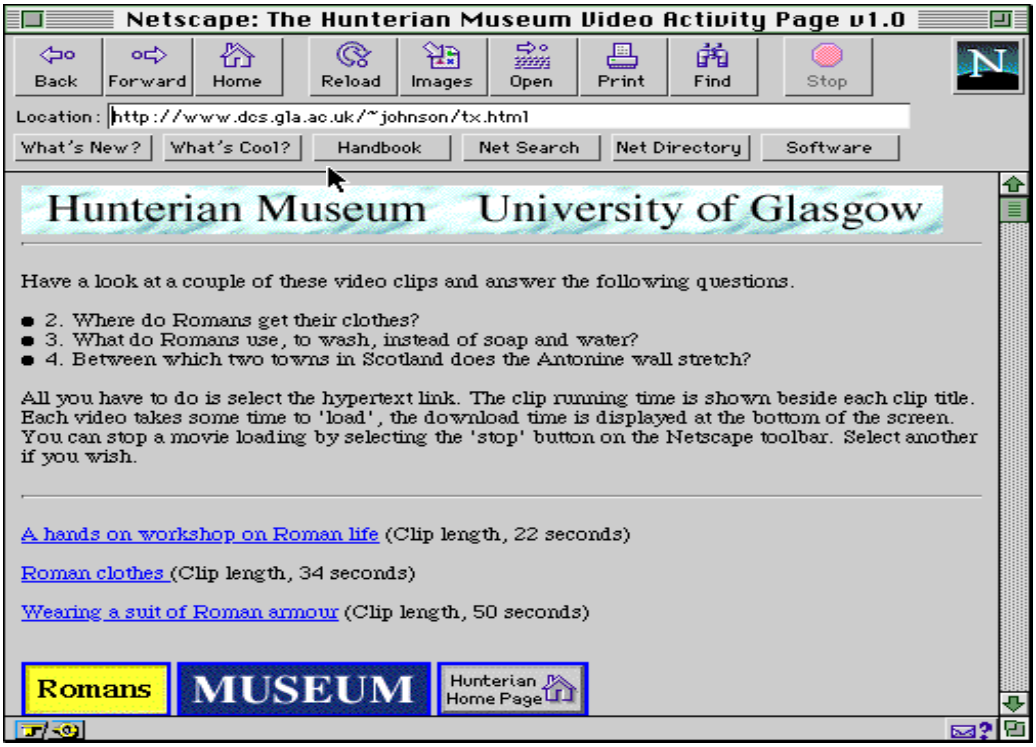


Figure 6: Reducing Unpredictability By Indicating Resource Size

As before, logic abstractions can be used to represent generic requirements for web pages. This is important because, as Figure 6 indicates, the precise means of presenting the size of a resource is determined by the particular characteristics of the user population. File lengths were inappropriate for the school-children accessing this web page. Users may predict retrieval delays if the size of a resource is presented until the user makes a selection:

$$predict_delay(Url) \Leftarrow [present_size(Url)\ U\ user_selects(Label)] \qquad (16)$$

Other techniques are exploited by the designers of browsers so that users gain a better impression of the likely delays while pages of text and graphics are assembled from remote resources. Mosaic and Netscape process and present the text on a page before displaying any associated images. A temporary icon is presented while the images are processed. This supports predictability. The number of temporary icons gives the user some idea of the amount of time required to retrieve and present all of the information on the page:

$$update(Url) \Leftarrow fetch(Url) \wedge AX((present_text(Url) \wedge present_temporary_icon(Url))\ U\ present_images(Url)) \qquad (17)$$

One of the benefits of the distributed architecture behind the Web is that the text and graphics in any page may be located on different machines. If the graphics machine goes down then, under the approach described in (17), 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:

$$fault_tolerant_update(Url) \Leftarrow fetch_text(Url) \wedge fetch_images(Url) \wedge AX(present_text(Url) \wedge present_temporary_icon(Url)\ U\ (present_images(Url) \vee present(request_failure))) \qquad (18)$$

Unfortunately, this approach only addresses some of the problems that are caused by unpredictability. Contextual information about the costs of retrieval 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:

$$unpredictable_request(Url2) \Leftarrow user_selects(Label) \wedge$$
$$linked(Label, Url) \wedge fetch_text(Url) \wedge fetch_images(Url) \wedge$$
$$EF(user_selects(Label2) \wedge linked(Label, Url2)) \wedge$$
$$(present_text(Url) \wedge present_temporary_icon(Url)\ U$$
$$(present_images(Url) \vee present(request_failure)))) \quad (19)$$

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:

$$ordered_request_processing(Url, Url2) \Leftarrow user_selects(Label) \wedge$$
$$linked(Label, Url) \wedge fetch_text(Url) \wedge fetch_images(Url) \wedge$$
$$AX(user_selects(Label2) \wedge linked(Label2, Url2) \wedge$$
$$(present_text(Url) \wedge present_temporary_icon(Url) \wedge$$
$$not(fetch_text(Url2) \wedge fetch_images(Url2))\ U$$
$$(present_images(Url) \vee present(request_failure)))) \quad (20)$$

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:

$$support_for_halt(Url) \Leftarrow$$
$$fetch_text(Url) \wedge fetch_images(Url) \wedge$$
$$AX(present_text(Url) \wedge present_temporary_icon(Url)\ U$$
$$(present_images(Url) \vee present(request_failure) \vee user_selects(halt))) \quad (21)$$

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:

$$ordered_request_processing(Url, Url2) \Leftarrow$$
$$user_selects(Label) \wedge linked(Label, Url) \wedge send_request(Url) \wedge$$
$$AX(user_selects(Label2) \wedge linked(Label2, Url2) \wedge$$
$$EF(fetch_text(Url) \wedge fetch_images(Url))) \quad (22)$$

This section has used CTL to represent techniques that reduce the unpredictability of web-based systems. Page designers can provide indications of the size of a remote resource. Browser designers can provide termination mechanisms and support the parallel presentation of text and graphics. It is important to emphasise, however, that these techniques also help to reduce the problems created by electronic gridlock. If users fail to accurately predict the delays that affect a particular request, from a particular geographical location at a particular time of day then termination mechanisms provide an escape route. Similarly, by indicating the size of a resource, designers can help users to judge whether it is worth requesting a file from a particular machine.

4. Analysing Information Saturation

Previous sections have shown how formal methods can be used to represent and reason about the problems of electronic gridlock and unpredictability. In contrast, this section uses the problem of information saturation to identify two critical weaknesses in the application of formal methods. Firstly, there is no means of assessing **why** an interface is appropriate given its formal specification. Secondly, there is no guarantee that the application of formal methods will actually result in more 'usable' interfaces.

4.1 Information Saturation and the Creativity of Design

The most irritating feature of the world wide web is not, perhaps, that users cannot predict the time that it will take to retrieve a resource but rather that they cannot predict the quality of the resource from its link. In other words, delays would be far less irritating if users could be sure that the final page would be worth the wait. Much of this frustration stems from information saturation. We have reached a point where the sheer volume of information makes it difficult for users to distinguish valuable resources from worthless trivia. For example, many web pages now provide access to video clips. Most of these clips suffer from extremely poor production quality and inadequate direction. In consequence, many users are reluctant to tie up their machines while large

files of poor quality footage are retrieved from remote sites. This problem affected the interface shown in Figure 6. Initial trials with the page showed that 30-40% of requests were abandoned before they were completed (Johnson and Kavanagh, 1996). The actual percentage varied with the length of the clip. These results were extremely depressing because a great deal of time, effort and money had gone into the development of the web site. The videos were produced by a professional crew working for a national broadcaster. They were processed using 'state of the art' compression software. In spite of these precautions, many users were still not prepared to wait for the films to download. Users were, however, more likely to complete a retrieval request if they had already downloaded one of our videos and hence were aware of the production standards. It might, therefore, be argued that the design technique described in clause (16) ought to be strengthened. Not only must designers indicate the size of a resource but they must also provide users with some indication of its quality or value to their task:

$$display_value_size(Url) \Leftarrow [(present_size(Url) \wedge present_value(Url))\ U\ user_selects(Label)] \quad (23)$$

For example, Figure 7 illustrates a revised page design for the video interface shown in Figure 6. Thumb-nail icons and descriptive paragraphs are used to indicate the production and compression quality of the associated material. Even if users cannot predict the delays imposed by network loading, our experience with this application shows that designers can affect attitudes to retrieval delays by providing a clear indication of the quality of the resources that users request.

Figure 7: Indicating the Quality of Remote Information

Additional clauses might be introduced to represent the techniques, such as those used in Figure 7, that satisfy the high-level requirements of clause (23). For example, the following formula states that the value of the movie located at a particular URL is represented if a *Gif* is used to present a screen shot from that movie:

$$present_value('http://www.dcs.gla.ac.uk/\sim johnson/mov1') \Leftarrow display_screen_shot(Gif, http://www.dcs.gla.ac.uk/\sim johnson/mov1') \quad (24)$$

This clause illustrates both the strength and the weakness of formal approaches to interface development. On the one hand, mathematical notations provide designers with means of specifying high-level design objectives. On the other hand, they provide little indication of how this requirement might be satisfied within a particular interface. Further problems stem from the fact that formal specifications, typically, provide an indication of **what** an interface must do. They do not say **why** it must do it. This is significant because it can be difficult for non-formalists to understand why the design of Figure 7 is an improvement over the page layout shown in Figure 6. In previous papers, we have addressed these problems by developing literate specification techniques

(Johnson, 1996a, 1996b). This approach uses the semi-formal argumentation of design rationale to support the use of formal methods during systems development. Figure 8 illustrates this approach. Rank Xerox's Questions, Options and Criteria (QOC) notation is used to document the reasons why the previous clause might be adopted within the design of a particular system. QOC diagrams are built by identifying the key questions that must be addressed during the development of an interactive system (Buckingham Shum, 1995). In this case, the question is which style of interface best addresses the problems created by information saturation. The options that answer a particular question are then linked to it using the lines shown in Figure 8. The textual interface that satisfies clause 16 is one alternative that might reduce information saturation. The 'cinematic' interface that satisfies clauses 23 and 24 is a further alternative. Finally, these options are linked to the criteria that support them, using solid lines, or weaken them, using broken lines. In Figure 8, the cinematic interface is justified by the criteria that it represents both the size and the quality of the remote resources. It is not supported by the argument that the initial screen, shown in Figure 8, will take a relatively short time to download. Additional delays will be introduced by the introduction of the thumb-nail sketches. In contrast, the textual interface of Figure 6 is supported by a short down-load time, it also indicates the size of the remote resource but unlike the cinematic design it does not indicate the production quality of the video. This example, therefore, helps to illustrate the trade-off that is being made when using clauses such as (23) and (24) to address the problem of information saturation. Users are given some indication of the quality of remote information. This is achieved at the cost of increased retrieval delays from the additional information that provides an indication of this quality on the indexing screen shown in Figure 7.

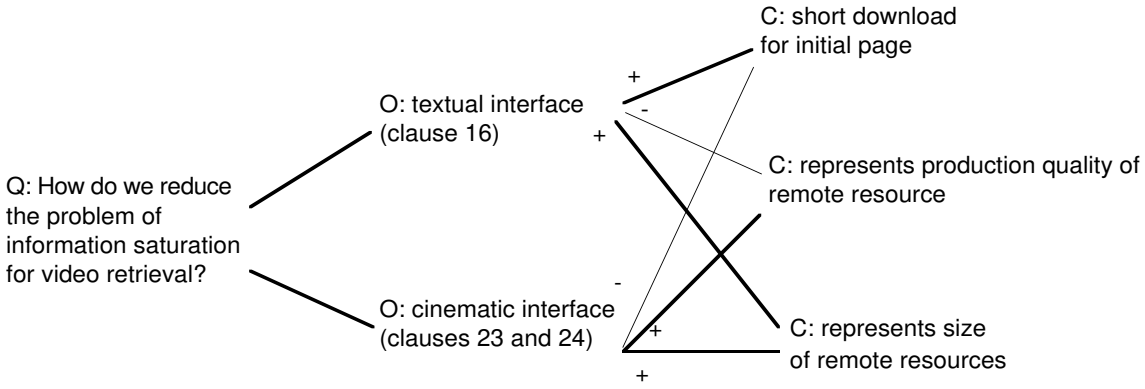
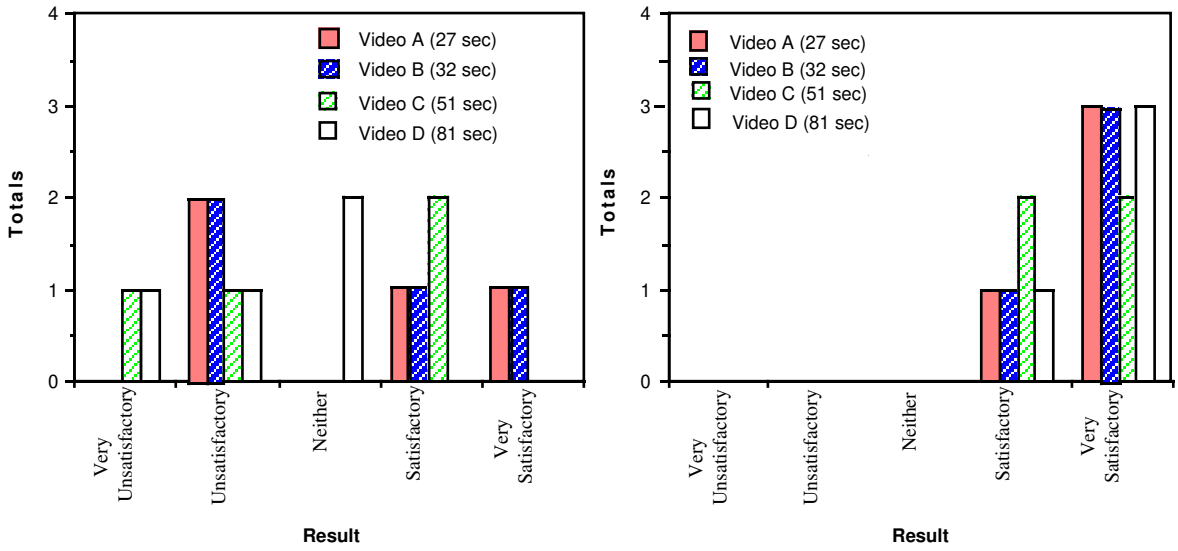


Figure 8: Literate Specification for Information Saturation.

Natural language annotations of the Questions and Criteria provide non-formalists with an entry-point into the clauses that represent particular Options. In literate specification, these annotations provide the justifications for and against formal design requirements. The interested reader is directed to Johnson (1996b) for more detail on the application of this approach. It is important to note, however, that both formal and literate specification techniques suffer from a major limitation. Designers have no proof that a particular approach, such as that described by clauses (23) and (24), will actually improve the 'usability' of an interface.

4.2 Information Saturation and the Evaluation of Design

The interface shown in Figure 7 is intended to reduce the problem of information saturation by using thumb-nail sketches to provide an indication of the quality of a remote resource, *present_value(Url)*. Formal methods provide no means of determining whether or not the page design actually achieves this claimed benefit. In order to provide this evidence, designers must recruit techniques from experimental psychology. For example, Figure 9 present the results from a series of tests that were used to gauge subjective attitudes to the video pages (Johnson and Kavanagh, 1996).



(a) Textual Interface (b) Cinematic Interface

Figure 9: Results of an Empirical Evaluation of Web-Based Video Pages

The chart on the left presents subjective attitudes towards the textual interface shown in Figure 6 for video clips of several different lengths. This page design satisfied clause (16) because it represented the size of the video resources that it indexed. It did not satisfy clauses (23) and (24) because it did not provide users with an indication of the production quality. In contrast, the chart on the right shows subjective attitudes to the delays suffered when using the revised page design of Figure 7. As predicted it is possible to see higher satisfaction ratings from the cinematic interface.

Figure 9 illustrates the complementary nature of formal specifications and empirical evaluations. Without empirical evaluation, designers cannot be sure that particular interface techniques will actually support the intended users of an interface. Without formal specification techniques, it is difficult for designers to precisely describe the differences between the various interfaces being tested. In our example, clauses (16), (23) and (24) characterise the two interfaces that are being evaluated. Unfortunately, there has been no research into the integration of formal and experimental design techniques. In particular, designers have no means of documenting the relationship the relationship that exists between the results of an evaluation and the interfaces that are described in the clauses of a formal specification. We are currently extending literate specification techniques to address this problem. For example, the Questions, Options and Criteria notation can be translated into a form that directly represents the empirical evidence that supports a particular option. Instead of using questions to represent critical design issues, diagrams can represent the usability claims that are made about a particular option. These can then be linked to more detailed descriptions of the empirical techniques that support the claim. Figure 10 presents a Claims, Analysis and Evidence (CAE) diagram for the interfaces shown in Figures 6 and 7.

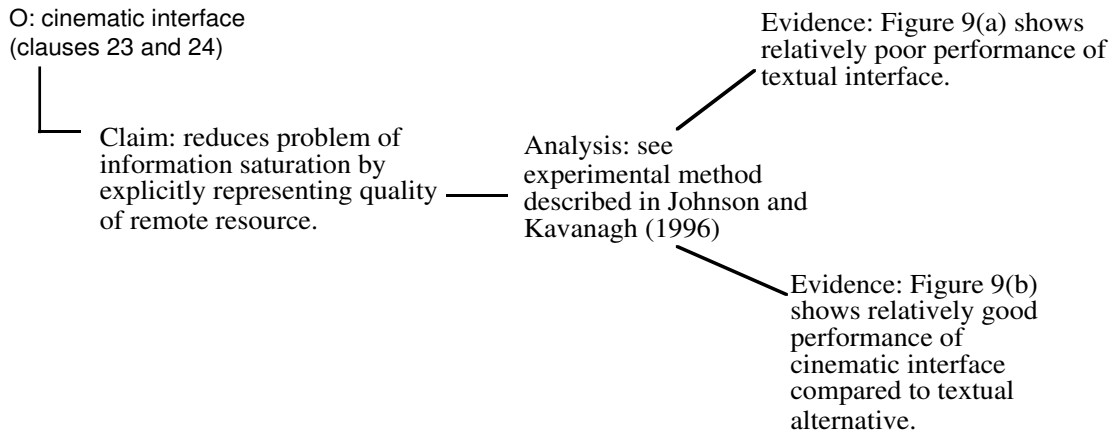


Figure 10: Claim, Analysis, Evidence (CAE) Diagram for the Video Retrieval System

As mentioned, the claim, analysis and evidence links shown in Figure 10 are simple extensions of the literate specification techniques shown in Figure 8. Literate specifications provide a bridge between formal analysis and design rationale. CAE diagrams provide a bridge between formal specifications and empirical evaluations. In short, these complementary techniques are intended to provide a web of supporting evidence for particular interface design decisions. Figure 11 shows how literate specifications and CAE diagrams can be integrated to provide an overview of our formal and empirical analysis of the web-based video system.

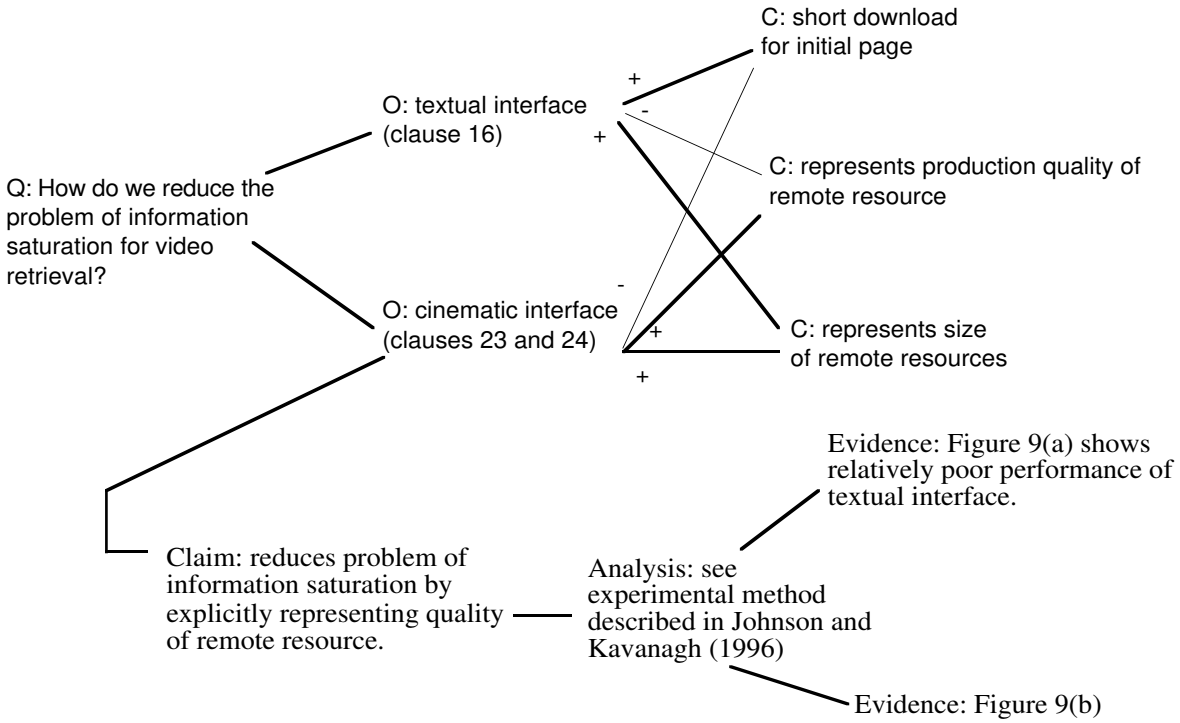


Figure 11: Integrating CAE and Literate Specification

Future work intends to evaluate the usability of these linking structures for different groups of interface designers. Much of this research will focus upon tool support for the resulting diagrams. It is important to stress, however, that we are less concerned about the particular form of the links than with the pressing need to integrate formal and experimental approaches to interface design. Without abstract, mathematical notations it is difficult to reason about the complexity of interaction with distributed information systems. The problems of electronic gridlock, information saturation and unpredictability cannot easily be understood without some precise and concise means of representing temporal properties. Equally, formal techniques cannot be used to validate particular interfaces once the problems of electronic gridlock, information saturation and unpredictability have been better understood.

5. Conclusion

This paper has argued that the increasing demands upon the world's computing resources has created a problem of 'electronic gridlock'. This occurs when users are forced to wait for long periods of time in order to access remote resources. In contrast, the huge increase in the supply or remote information is creating a problem of 'information saturation'. This occurs when users cannot distinguish high quality resources from a background mass of mediocre material. Communications failures, network loading and processing overheads all help to compound these problems because people cannot predict the delays that occur during retrieval operations. 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 that can be used to reduce the impact of electronic gridlock, information saturation and unpredictability in distributed systems. For example, the incremental presentation of text and graphics can help users to predict potential delays. Halting mechanisms free users from systems that would otherwise have been tied up by the problems of electronic gridlock. Representations of the size and quality of remote resources can be used to reduce the frustration that arises from information saturation.

Unfortunately, formal methods do 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. The later sections of this chapter have described a range of techniques that address this problem. Literate specification techniques use formal requirements to document **what** an interface must do. They also provide a rationale that explains **why** such techniques are appropriate. Unfortunately, this approach provides no evidence that the claimed benefits of a particular interface will be satisfied in a particular implementation. We have, therefore, extended our literate specification techniques to include claims, analysis and evidence diagrams. These represent the methods and results of the empirical evaluations that must be recruited to support the formal design of interfaces to distributed information systems.

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.

Buckingham Shum, B., *Analysing The Usability Of A Design Rationale Notation.* In T.P. Moran and J.M. Carroll (eds.) Design Rationale Concepts, Techniques And Use, Lawrence Erlbaum, Hillsdale, New Jersey, United States of America, 1995.

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.

Hiltz, S.R. and M. Turoff, M. (1985). Structuring Computer-Mediated Communication Systems to Avoid Information Overload. *Communications of the ACM*, 28: 680-689.

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. (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.

Johnson, C.W. (1995) Time and the Web: Representing and Reasoning about Temporal Properties of Interaction with Distributed Systems. In M. Kirby, A. Dix and J. Finlay (eds.), *People and Computers X,* 39-50, Cambridge University Press, Cambridge, United Kingdom.

Johnson, C.W. (1996) *Literate Specification: Using Design Rationale to Support Formal Methods in the Development of Human Computer Interfaces,* Human Computer Interaction Journal, (11)4.

Johnson, C.W., (1996a) *Literate Specification.* Software Engineering Journal, (11)4:225-237.

Johnson, C.W. and Kavanagh, J. (1996) Electronic Gridlock. In H. Thimbleby and A. Blandford (eds.) *Adjunct Proceedings of HCI'96*. 78-89. British Computer Society, London, United Kingdom.

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

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.