The Challenge Of Time *

Chris Johnson

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

Abstract. In the past, temporal problems have not played a central role in the development of human-machine interfaces. The timing of individual commands seldom affects single users interacting with standalone applications, such as text-editors and spreadsheets. This situation is changing. The increasing use of mutlimedia applications, distributed systems, computer supported cooperative work tools and process control environments is forcing designers to consider temporal aspects of usability. It is a non-trivial task to synchronise the activities of multiple operators working through several different modalities on many different machines. Unfortunately, traditional techniques that rely upon rapid prototyping or iterative development cannot easily be used to address the many different timing issues that arise during the design of this new generation of interactive systems. In contrast, formal specification techniques provide concise and precise means of representing and reasoning about such interactive behaviour. This paper argues that existing notations are, however, poorly equipped to face the challenges posed by temporal aspects of usability.

1 Introduction

Designers are facing an increasing range of challenges when they attempt to synchronise human-machine interaction. For instance, users of the World Wide Web will recognise the frustration and error that can arise when information is delayed or cannot be retrieved from remote sites [9]. Similarly, users of multimedia and virtual reality systems will recognise that time plays a critical role in determining the quality of interaction [4]. Video images and audio output may be difficult to follow if they are delivered at an unacceptably slow rate. In computer supported cooperative work systems, delays in speech and text based communication can break established protocols and modes of working [7]. Automated control systems are increasingly providing information at a rate that exceeds the operators' ability to sample them [14]. Traditional techniques that rely upon rapid prototyping or iterative development cannot easily be used to address the range of temporal behaviours exhibited by such systems. The challenge of time is, therefore, to develop formalisms that are well equipped to represent and reason about the temporal properties of interaction.

* This document summarises the findings of a working group on temporal aspects of usability. The members of the group are listed at the end of the paper.

2 What Are The Problems?

Many of the problems that arise during the design of human-computer interfaces have temporal components. For instance, at a high level of abstraction developers must consider the time necessary to complete particular tasks. Frequently performed operations should not, wherever possible, take longer than infrequent tasks. Temporal problems also affect interaction at lower levels of granularity. The developers of window managers must determine the period that distinguishes double-click selections from two separate mouse-down events. The apparently ubiquitous nature of temporal problems is, however, an illusion. In many stand-alone applications, such as text-editors and spreadsheets, these issues are not as critical as the development of appropriate display formats or input sequences. In contrast, however, temporal aspects of usability are playing an increasingly important role in determining the quality of interaction with distributed systems and multimedia applications.

2.1 Distribution And The Pace Of Interaction

Figure 1 illustrates the sources of temporal usability problems in distributed systems. The physical properties of communications networks impose delays upon the transmission of information. This leads to frustration and error because it is difficult for operators to predict how long it will take before their commands are effective. For instance, attempts to retrieve remote information may be abandoned if an individual thinks that their request has failed. The new groups of users who are exploiting Internet resources cannot be expected to understand the mechanisms that support their communications with remote colleagues and information servers. This point is illustrated by the problems that arise when using browsers, such as Mosaic and Netscape. The temporal characteristics of these systems appear to be both bizarre and arbitrary for many of their users [6].

Designers can adopt a number of techniques to support interaction with distributed systems. Unfortunately, many of these approaches are extremely complex. They frequently involve the integration of advanced communications facilities and novel interface design techniques. The complexity of such an integration makes this an important area for the application of formal methods. For example, the development of interlaced graphics and mirror sites enables information providers to alter the pace of interaction between users and their systems. Images may be gradually presented as they are retrieved rather than being displayed at the end of some unpredictable delay. Information may be cached at local sites so that users are not forced to repeat retrieval requests on distant machines. A problem with the current use of these techniques is that they introduce entirely new temporal usability problems. For instance, mirror sites can prevent users from predicting how long their requests will take. Some information can be retrieved from local sources whilst the rest must be requested from the original, remote, machine. These problems might be overcome by developing displays to provide more detailed feedback about the location of information and its rate

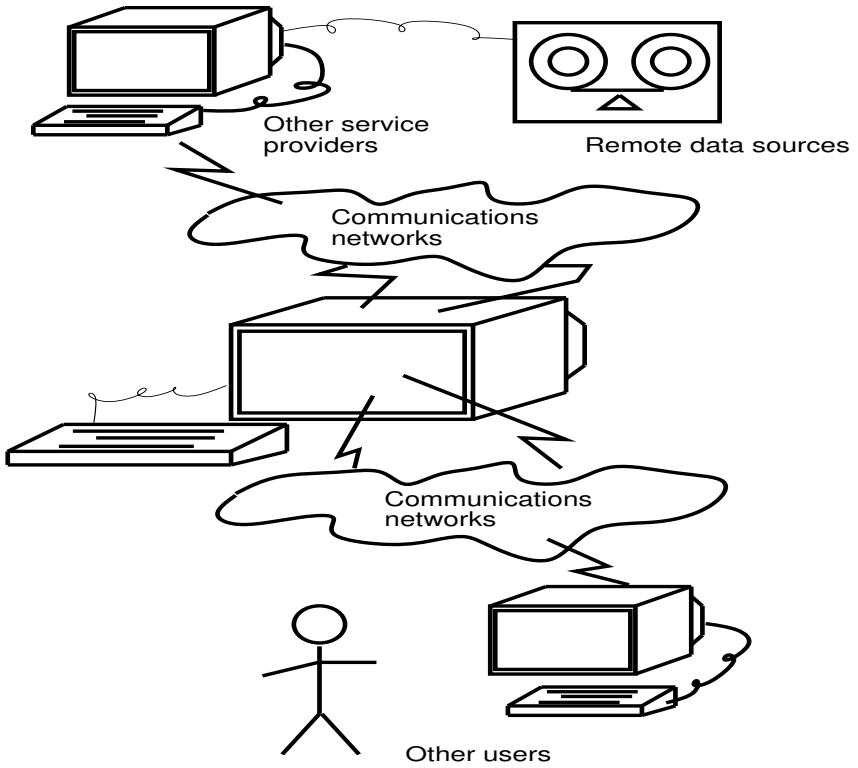

Fig. 1. The Problems Of Synchronising Distributed Interaction.

of transfer. Such presentation techniques create further temporal problems; designers must monitor and display the progress of information between various sites. The limited success of previous ad hoc solutions to these problems suggests that the precision and rigour of specification techniques might support interface development for distributed systems. However, if formal methods are to be of real benefit in this area then there are two temporal challenges that must be addressed:

- how can users be provided with an adequate representation of the flow of information from remote sites when the rate of transfer may have more to do with technological considerations rather than the geographical distances involved?
- how can users be provided with sufficient information about other users and systems so that they can make accurate predictions about the changing state of interaction over time.

2.2 Multimedia Bottlenecks

Time plays a critical role in determining the quality of interaction with multimedia applications. For instance, delays between frame updates can have a profound effect upon the use of video signals. If rates fall below 25Hz then users may perceive flicker, objects will appear to be blurred and motion will not be continuous. In such circumstances, users will frequently abandon video output in favour of more 'conventional' media. The causes of such temporal problems are illustrated by Figure 2. In one sense, the problems are similar to those cre-

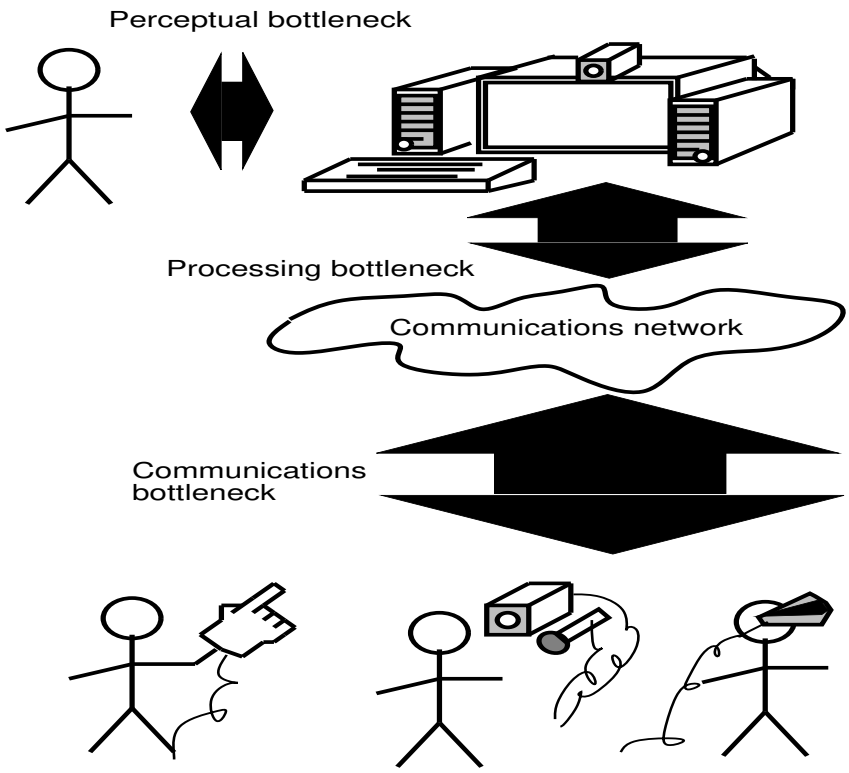

Fig. 2. The Problems Of Synchronising Multimedia Interaction.

ated by distributed systems. Limited communications facilities prevent designers from polling devices or providing updates as frequently as they would like. The delays caused by this bottleneck increase as the available bandwidth has to be shared amongst more and more devices. There are further problems. A second bottleneck arises from the limited processing resources that are currently provided by multimedia workstations. These limitations introduce delays into

the continuous presentation of real-time media such as audio and video output. Finally, there is the perceptual bottleneck. The user's ability to sample and provide information would still create temporal usability problems even if the other two bottlenecks were solved. Designers must match the presentation of diverse information sources to the operators' finite perceptual resources.

A number of techniques can be used to reduce the problems caused by the various bottlenecks in multimedia systems. For instance, interlacing can be used to reduce the communications loading without seriously increasing screen flicker. Similarly, communications standards, such as the Asynchronous Transfer Mode (ATM), can be defined to improve the distribution of multi-media information [3]. None of the techniques, however, provides a panacea for the development of multimedia interfaces. For instance, ATM does not address the low level synchronisation issues that must be considered in order to guarantee a particular quality of service at the user interface. A more fundamental point is that quality does not simply relate to technological factors. It is closely tied to the user's tasks. This reflects a recent preoccupation with the communications and processing bottlenecks rather than the perceptual bottleneck in multimedia systems. For instance, if the loading on particular networks reduced the transfer rate of video images then users might be prepared to degrade that service in order to maintain good response times over the audio channel. This would maximise the utility obtained from one particular perceptual modality. Conversely, an audio link might be degraded in order to improve the transfer of graphical images or video information. At present, there are no convenient tools that designers can use to represent and reason about such strategies. The complexity of such synchronisation techniques makes this another important area for the exploitation of formal notations. Multimedia applications, therefore, provide further challenges for the specification and verification of interactive systems:

- how can designers represent the changing priorities that might be assigned to different modalities during the course of interaction?
- how can these priorities be adequately related to the changing demands of particular user tasks?

3 What's The Time?

In order to select an appropriate means of representing and reasoning about temporal properties of interaction, designers must first make certain decision about the nature of time. In particular, they must decide whether their model is based around real-time or intervals and whether their view of the future is linear or branching.

3.1 Models Of Time

Representing temporal properties of interaction is not as easy as it might first appear. The most obvious approach is to specify interface properties in terms of

minutes, hours and seconds. For example, designers might decide to present a warning if a retrieval request on a remote machine has not been answered within ten seconds. The first diagram in Figure 3 illustrates this approach. The timeline

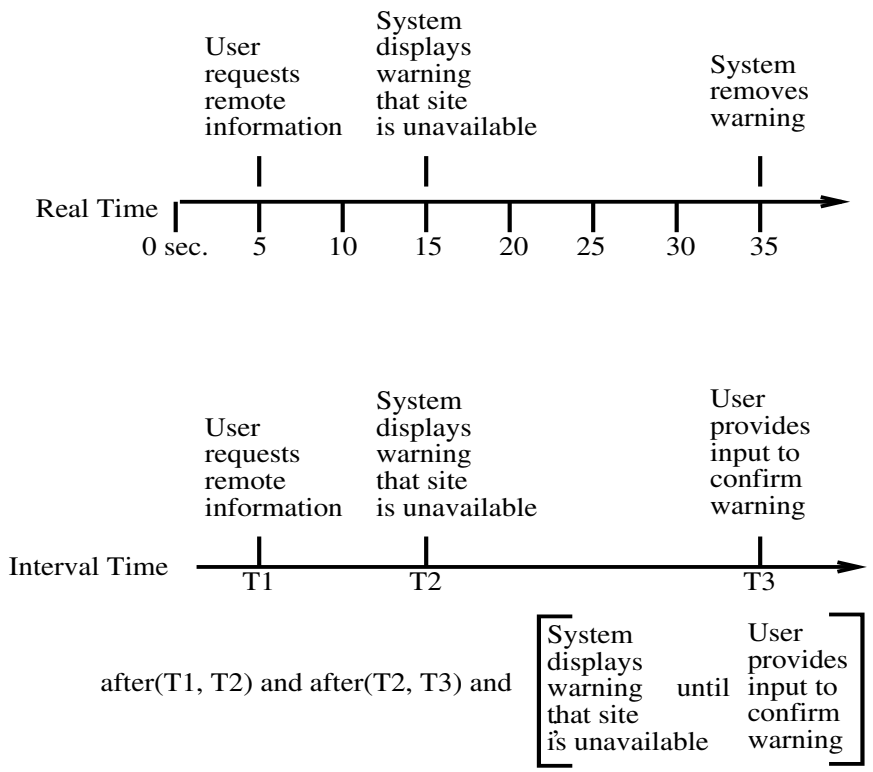

Fig. 3. Real And Interval Time.

shown in this picture can be formalised using various real-time extensions to first order logic:

$$retrieval_failure(12:00:35) \Leftarrow user_request(12:00:05) \wedge$$
$$failure_warning(12:00:15) \wedge warning_removed(12:00:35) \qquad (1)$$

There are a number of problems with real-time as a basis upon which to design distributed, multimedia applications. In particular, clauses such as (1) are extremely deterministic. The clause would not hold if the user requested the information at 12:00:06, 12:00:07, 12:00:08 and so on. This determinism has important consequences for the use of real-time notations as a means of satisfying the challenges mentioned in previous sections. For instance, a standard warning

that is always presented after a ten second timeout hardly provides appropriate feedback about many different retrieval requests. Such a delay would be expected for retrieval tasks that involve low performance networks. Users would continually be presented with a failure warning even though their retrieval were taking the expected amount of time. Similarly, a ten second delay might be perfectly acceptable on heavily loaded machines, such as the National Centre for Supercomputing Applications' servers. In such circumstance, designers must either tailor the real-time requirement to particular retrieval tasks or seek alternative means of representing temporal properties that avoid such determinism.

The limitations associated with real-time notations can be avoided through the use of interval based formalisms. This approach is illustrated by the second diagram shown in Figure 3. A timeline is used to model the flow of events, interval requirements can then be constructed in terms of this underlying model. The example specifies that the user is informed about the failure of their command until they issue input to confirm that they have observed the warning. This is a temporal requirement in that it describes how interaction develops over time. It does not, however, specify the exact number of hours, minutes and seconds during which the display is presented. This approach avoids the temporal determinism of real-time notations. Interval formalisms can also be captured using textual notations, such as Manna and Pnueli's temporal logic [13]. In the following $\bigcirc$ can be read as 'in the next interval', $\mathcal{U}$ can be read as 'until':

$$\begin{aligned} &\mathit{retrieval_failure} \Leftarrow \bigcirc(\mathit{user_request} \wedge \\ &\quad \bigcirc(\mathit{failure_warning} \mathcal{U}\, \mathit{user_confirmation})) \end{aligned} \tag{2}$$

Interval notations are appropriate for the early stages of development when the exact timings of interactive behaviour may not be known. Unfortunely, they lack the precision that is required during more detailed development. A more particular problem for the previous challenges is that interval based notations cannot easily be used to relate users' tasks in multimedia applications to the real-time properties of their underlying systems.

Designers must decide how they are to represent the future course of interaction with distributed or multimedia systems. For instance, interaction might be described in terms of a linear sequence of user inputs and system responses. This approach is illustrated in the first diagram of Figure 4. These linear models are captured in the textual forms of the interval (2) and real-time (1) clauses, shown above. Unfortunately, these approaches do not capture the uncertainty that frequently frustrates the operation of distributed and multimedia applications. Branching-time models avoid this problem by representing the future as a series of alternative paths. The second diagram in Figure 4 shows one future in which the user is presented with confirmation that their command is being dealt. In another 'parallel' future they are presented with a warning that the remote site has failed. As before, a range of textual notations have been developed to exploit these temporal models. For instance, Clarke and Emerson's Computation Tree Logic [2] might be used to require that users must confirm warnings that a retrieval request has failed. The AX quantifier is used to specify that the input

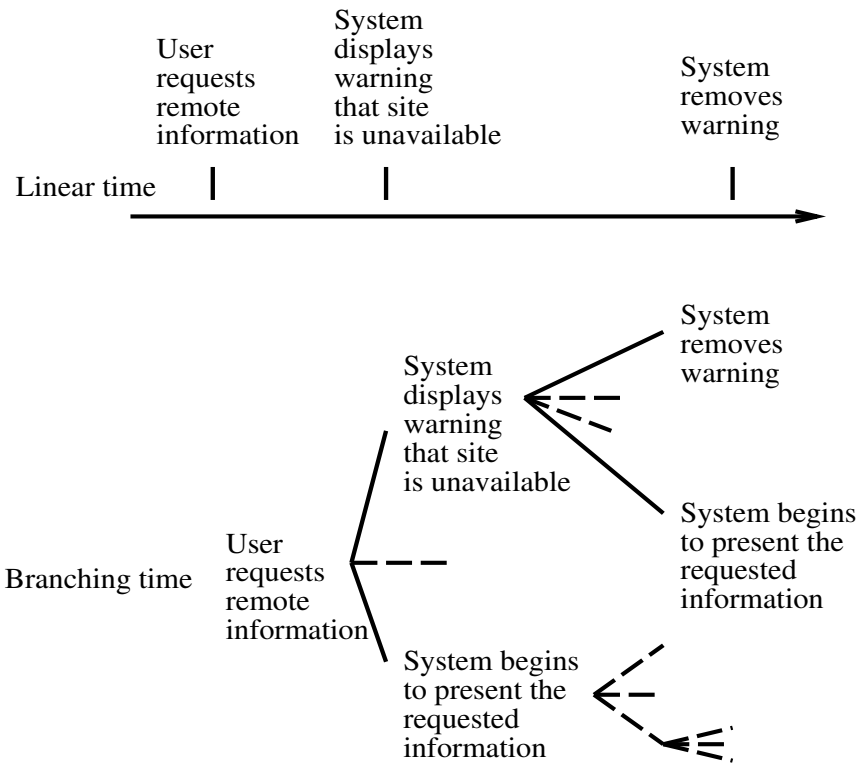

Fig. 4. Branching And Linear Time.

must be provided in all possible next intervals:

$$\begin{aligned} &\mathit{retrieval_confirmation} \Leftarrow \\ &\quad \mathit{failure_warning} \wedge AX \quad \mathit{user_confirmation} \end{aligned} \tag{3}$$

Clearly, the choice of real or interval time, linear or branching time will have an important impact upon the successful application of formal notations. This poses further challenges:

- what criteria can designers use to select an appropriate temporal model for their development tasks?
- is it possible to move between notation that support different temporal models as development progresses, possibly from imprecise temporal intervals to more precise real-time requirements?

3.2 Temporal Notations

Time can be represented by a number of different notationals. The major distinction here is between textual and graphical approaches. Petri Nets are, perhaps, the most widely applied graphical notation [1, 8]. The basic form provides means of capturing intervals and sequences of interaction. There also exist real-time and stochastic extensions that can be used to capture alternative future traces. Figure 5 shows how Petri Nets can be used to represent complex sequences of interaction in multi-user and distributed systems. This is the approach followed in [11]. They can also support user modelling [15]. A powerful attraction for the use of such graphical notations is that they have an intuitive appeal. The flow of event over time can be represented by the movement of tokens through the network. Token are shown as filled dots in Figure 5. Unfortunately, Petri Nets and similar graphical formalisms suffer from a number of limitations. They can quickly become intractable for large systems. Certain features, such as undo, are extremely difficult to capture using this approach to interface specification.

Tabular notations, such as the XUAN [5] shown in Figure 6, avoid some of the problems associated with the large-scale application of Petri Nets. The structuring mechanisms of the columns and rows provide some impression of the flow of time. XUAN also possesses a process algebra that can be used to describe additional temporal constraints. These provide a convenient means of composing large scale specifications in a manner that is, arguably, more tractable than the composition facilities provided by graphical notations. Unfortunately, tabular notations are less well developed than many of their textual and graphical counterparts. Whilst Petri Nets have 'established' real-time variants, the corresponding facilities in XUAN do not possess a clear semantics. There are no facilities for representing 'alternative' or branching futures in this tabular notation.

There are many different textual notations that designers might exploit to represent and reason about temporal properties of interaction. These range from higher order logics [2] to more established approaches such as Z, CSP and VDM [12]. For example, the following schema shows how Z can be extended to include temporal operators such as $\mathcal{U}$ (read as 'until') [10]:

$$
\begin{array}{|l}
\hline
\mathit{Select_Steam_Circuit_High_Shut_Down} \\
\mathit{Present_Steam_Circuit_Display} \\
\mathit{Multi_User_Events_Display_Attributes} \\
\hline
\forall\, user_1, user_2 : \mathit{USERS} \bullet \\
\quad user_input_event(user_1, pump_a_icon) = on_select \wedge \\
\quad\quad \neg\, (value(pump_a_press) = min(pump_a_press))\ \mathcal{U} \\
\quad\quad\quad user_input_event(user_2, pump_a_icon) = on_select \\
\hline
\end{array}
$$

The advantage of such textual approaches is that they have well-developed semantics and proof techniques. Tool support is available and many of these notations have been used in large-scale development tasks. It can, however, be

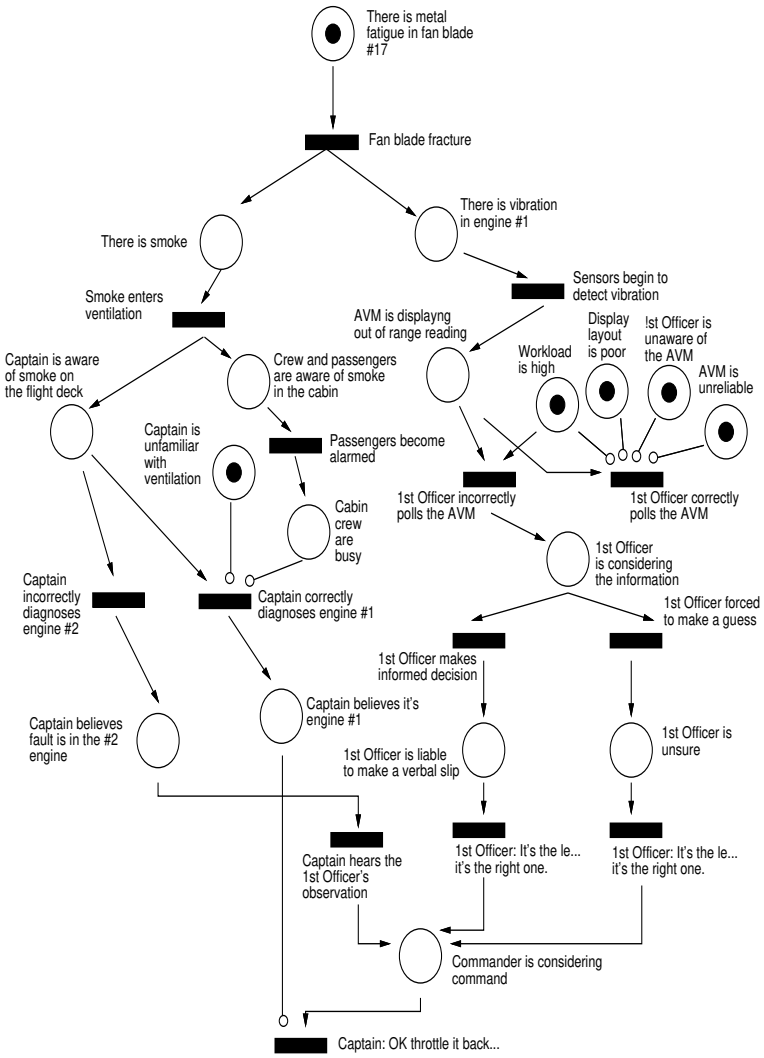

Fig. 5. A Petri Net Describing The Kegworth Air Accident

extremely difficult to visualise the interactive behaviour of an application from its textual description. This is significant because abstract notations should ease communication between and within design teams. If it is difficult for other design groups to identify the temporal properties within textual descriptions then this benefit may be lost.

It is not a simple matter to determine which of these different notational forms provides the greatest support for interface development. There are a num-

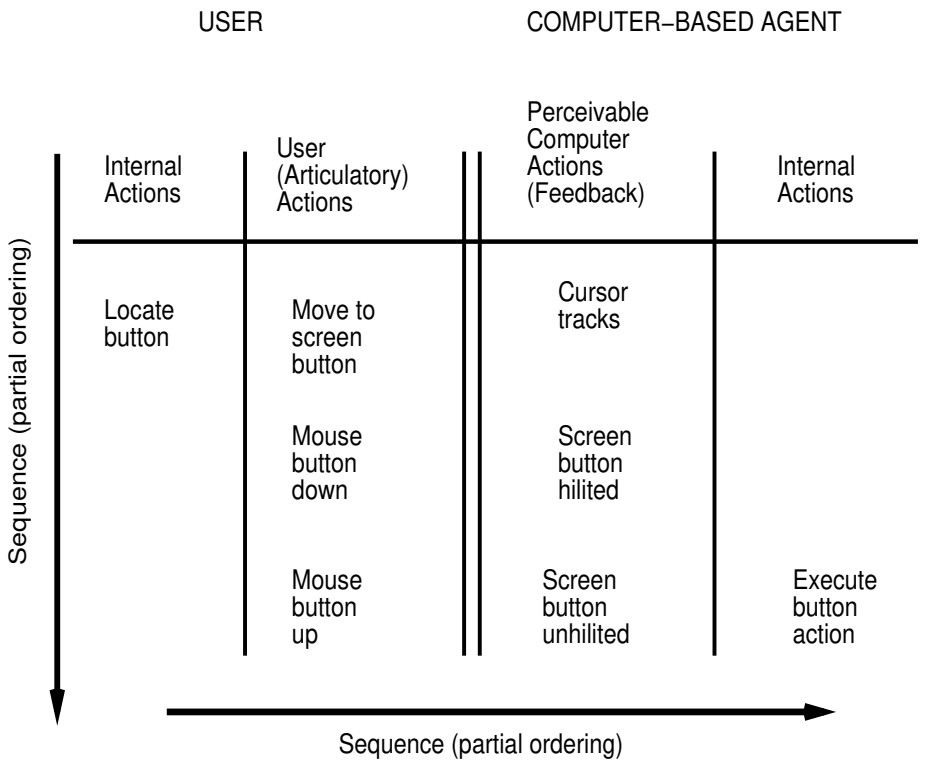

Fig. 6. An XUAN Task Description

ber of notational challenges that must be addressed before formal methods can be used to address temporal usability problems:

- how can a designer determine whether graphical, tabular or textual notations are more appropriate for their development task?
- how can the chosen notation be used most effectively to communicate temporal properties to the other designers, clients and users who will be affected by their observations?

4 Conclusions

In order to meet all of the challenges being posed by the temporal properties of modern interactive systems, we need an interval notation with real-time operators, a branching time approach that can also be used to express constraints over all possible futures, a graphical representation that is intuitive and easy to follow and yet has the power of textual notations for formal reasoning and proof.

Clearly, such a notation does not yet exist. This raises a number of important questions. For instance, should current research focus upon the development and extension of existing notations? In this way the particular weaknesses of Z, CSP etc might gradually be addressed so that they can more easily be applied to interface development. Alternatively, should research focus upon the development of novel methodologies that exploit more than one existing formalism? This could build upon the correspondence between Petri Nets and first order logics. Until these issues are addressed, interface designers are poorly equipped to face the challenges that time poses for the new generations of interactive systems.

Acknowledgements

This report is the product of discussions within the group that addressed temporal properties of interaction during the Bonas workshop. The members of the group were Stéphane Chatty, Alan Dix, David Duce, Phil Gray, Mark Green, Munos Jaime, Francis Jambon, Chris Johnson, Panos Markopoulos, Philippe Palanque, Graham Reynolds, Roger Took, Dick Van Schenk Brill and Kees Verhagel.

References

1. R. Bastide and P. Palanque. Petri Net objects for the design, validation and prototyping of user-driven interfaces. In D. Diaper, D. Gilmore, G. Cockton, and B. Shackel, editors, *Human-Computer Interaction—INTERACT'90*, pages 625–631. Elsevier Science Publications, North Holland, Netherlands, 1990.
2. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In D. Kozen, editor, *Logic of Programs 1981 - Proceedings*, LNCS 131, pages 52–71. Springer-Verlag, Berlin, FDR, 1982.
3. G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts And Design*. Addison Wesley, Wokingham, United Kingdom, 1994.
4. S.J. Gibbs and D.C. Tsichritzis. *Multimedia Programming: Objects, Environments And Frameworks*. Addison Wesley, Reading, United Kingdom, 1994.
5. P. Gray, D. England, and S. McGowan. XUAN: Enhancing UAN to capture temporal relationships among actions. In G. Cockton, S.W. Draper, and G.R.S. Weir, editors, *People And Computers IX*, pages 301–312. Cambridge University Press, Cambridge, United Kingdom, 1994.
6. P. Gray and C.W. Johnson. Requirements for the next generation of user interface specification languages. In R. Bastide and P. Palanque, editors, *Proceedings Of The 2nd Eurographics Workshop On The Design, Specification And Verification Of Interactive Systems*. 1995.
7. C.W. Johnson. Applying temporal logic to support the specification and prototyping of concurrent multi-user interfaces. In D. Diaper and N. Hammond, editors, *People And Computers VI: Usability Now*, pages 145–156. Cambridge University Press, Cambridge, United Kingdom, 1991.
8. C.W. Johnson. The application of Petri Nets to represent and reason about human factors problems during accident analyses. In R. Bastide and P. Palanque, editors,

Proceedings Of The 2nd Eurographics Workshop On The Design, Specification And Verification Of Interactive Systems. 1995.

9. C.W. Johnson. Time and the web: Representing temporal properties of interaction with distributed systems. In *People And Computers X*. Cambridge University Press, Cambridge, United Kingdom, 1995.
10. C.W. Johnson. Using Z to support the design of interactive, safety-critical systems. *Software Engineering Journal*, 10(2):49–60, 1995.
11. C.W. Johnson, J.C. McCarthy, and P.C. Wright. Using a formal language to support natural language in accident reports. *Ergonomics*, 38(6):1265–1283, 1995.
12. M. Kooij. Interface specification with temporal logic. In S.J. Greenspan, editor, *The 5th International Workshop On Software Specification And Design*, pages 104–110. IEEE Computer Society Press, Washington, United States of America, 1989.
13. Z. Manna and A. Pnueli. Verification of concurrent programs: The temporal framework. In R.S. Boyer and J. Strother Moore, editors, *The Correctness Problem In Computer Science*, pages 215–273. Academic Press, London, United Kingdom, 1981.
14. V.C. Miles, C.W. Johnson, J.C. McCarthy, and M.D. Harrison. Supporting prediction in complex dynamic systems. In D. Diaper and N. Hammond, editors, *People And Computers VI: Usability Now*, pages 133–144. Cambridge University Press, Cambridge, United Kingdom, 1991.
15. T. Moher and V. Dirda. Revising mental models to accomodate expectation failures in human-computer dialogues. In R. Bastide and P. Palanque, editors, *Proceedings Of The 2nd Eurographics Workshop On The Design, Specification And Verification Of Interactive Systems*. 1995.

This article was processed using the LaTeX macro package with LLNCS style