DIVISION OF COMPUTER SCIENCE

Formal Dialogue Specification for Hypertext and Multimedia Systems

Lisa Jacob
Sara Jones

Technical Report No.175

January 1994

Technical Report No. 175 January 1994

Formal Dialogue Specification for Hypertext and Multimedia Systems

Lisa Jacob, Sara Jones
L.Jacob,S.Jones@herts.ac.uk

School of Information Sciences, University of Hertfordshire,
College Lane, Hatfield, Herts, AL10 9AB, UK.

Tel: +44-707-284370
Fax: +44-707-284303

Abstract

We discuss issues relating to the use of CSP, a process-based formal notation, in the specification of human-computer dialogue with hypertext and multimedia systems. We illustrate our discussion with a small example and describe how this demonstrates important features of three systems: a simple bibliographic database; a multimedia information system; and a hypermedia exhibition guide. We consider the way in which CSP dialogue specifications can be used in the design and development of such systems and identify some areas for further work.

Submitted for presentation at EWHCI '94, the 1994 East-West International Conference on Human-Computer Interaction, to be held in St Petersburg, Russia, August 1994.

Formal Dialogue Specification for Hypertext and Multimedia Systems

Lisa Jacob, Sara Jones[*]
L.Jacob,S.Jones@herts.ac.uk

School of Information Sciences, University of Hertfordshire,
College Lane, Hatfield, Herts, AL10 9AB, UK.

Tel: +44-707-284370
Fax: +44-707-284303

Abstract

We discuss issues relating to the use of CSP, a process-based formal notation, in the specification of human-computer dialogue with hypertext and multimedia systems. We illustrate our discussion with a small example and describe how this demonstrates important features of three systems: a simple bibliographic database; a multimedia information system; and a hypermedia exhibition guide. We consider the way in which CSP dialogue specifications can be used in the design and development of such systems and identify some areas for further work.

1. Introduction

The Human Factors Consultancy team at the University of Hertfordshire have developed a number of hypertext and multimedia systems over the past 5 years. As the range of technologies available for supporting input and output of various kinds of information has grown, the task of designing and developing such systems has become increasingly complex. We have therefore begun to investigate the way in which methods and tools currently used in software engineering can be incorporated into the process of developing systems in which the human-computer interface is of primary importance.

This paper focuses on the question of human-computer dialogue design. It considers the way in which the use of a particular notation (CSP) might support the specification, implementation and evaluation of hypertext and multimedia systems by providing a basis for modelling the kind of dialogue such systems support. We begin with an introduction to existing work on dialogue modelling and the use of CSP. We then describe some of the particular issues confronting developers of hypertext and multimedia systems. We present a small example which demonstrates how important features of hypermedia system dialogue can be represented using CSP. We describe how the example could be extended for use in representing more realistic examples, and explain its relation to three systems built at the University of Hertfordshire. We suggest a number of ways in which CSP dialogue models might be used in the development and evaluation of hypermedia systems, and identify directions for further development in this area.

[*] denotes primary contact

2. Modelling Human-Computer Dialogue

In HCI design and research, the term 'dialogue' is used to refer to the sequence of events through which a computer and its human user communicate with each other. The structure of a dialogue - that is, the nature and order of actions through which communication takes place - depends both on the nature of the computer system and the interface it presents, and on characteristics of the user such as familiarity with the system and task of interest.

Models of human-computer dialogue provide system designers with a basis on which to discuss and reason about a variety of design options before even the first prototype is built. In large or complex systems, discussion of such options is only possible if concerns of dialogue design are separated from those relating to other aspects of the system. The existence of a concrete model which can be presented graphically or animated supports communication between different members of a design team, and if the model is written using a formal notation, it can also be used as a basis for rigorous mathematical reasoning about the consequences of particular design decisions.

Various approaches to human-computer dialogue modelling have been reviewed by Green [1] and Palanque et al [2]. These include the use of state transition diagrams, context-free grammars, events and Petri nets. The continuing development of interface technologies has placed increasing demands on notations for dialogue description as developers have needed to model not just command-line interfaces, or even menu driven interaction, but direct manipulation interfaces in which several forms of communication can be used at the same time. Neither grammars nor transition diagrams are well-suited to the representation of concurrency. Petri nets are sufficiently expressive but can become unwieldy if the systems to be represented are large.

Considerations such as these have lead some researchers (see, for example, [3] and [4]) to investigate the benefits of modelling human-computer dialogue using CSP [5], a process-based formal notation developed for the specification of reactive systems involving communication and concurrency. CSP can model all systems or dialogues which can be modelled using either state transition diagrams or Petri nets. The CSP notation is quite readable for non-mathematicians, even in its textual form. Its utility as a basis for discussions about dialogue design options has, however, been further enhanced by the development of tools such as Alexander's SPI [3] which support animation of CSP specifications. This means that it can be used in discussions not only between different members of a design team, but also with clients negotiating system requirements. Because CSP has a sound mathematical basis and well-defined semantics, it is also possible to reason about specifications in a mathematically rigorous way and to prove that they possess the properties which designers intend or clients require. Automated support is available for carrying out some of these checks as described in the final section.

3. Dialogue in Hypertext and Multimedia Systems

The development of hypertext and multi-media systems presents the HCI designer with unique challenges [7].

Dialogue with hypertext systems can take different forms depending on the platforms on which they are implemented and the purposes for which they are intended. A dialogue with a system implemented in HyperCard may be similar to that with many windows-based systems in that the user simply moves between different screens of information by clicking on buttons in the interface. However, other hypertext systems, such as NoteCards, permit windows displaying the contents of a number of nodes to be open and 'active' (in the sense that they are ready to receive input) at any time.

The defining feature of multimedia systems is their use of a wide range of input and output technologies. Input may be through keyboard, mouse, speech recognition or touch screen devices. Output may take the form of text, graphic stills, animation, video, speech, sound or music. Multimedia system designers must take account of the temporal nature of audio and video sequences in synchronising presentations, and must also consider which inputs and outputs may be used concurrently. A series of graphic stills may, for example, be accompanied by explanatory speech output, or music may be played while a video clip is shown. On the other hand, a designer would probably want to avoid playing two segments of speech output simultaneously, and may also wish to permit only one video to be shown at any one time.

Hypermedia systems combine elements of both hypertext and multimedia and designers of such systems are currently forced to make decisions about the kinds of issues identified above in the absence of any guiding principles or tools for reasoning about the available options. As one text puts it: 'Multimedia extensions to current systems have grown like weeds without well-defined or well-understood principles' [6], and the need for such principles becomes increasingly urgent as growing numbers of such systems are produced. It is intended that the thoughts presented below should form a basis for the development of precise models of hypermedia dialogue which will support discussion and experimentation and ultimately lead to the development of sound principles for hypermedia system design.

4. A CSP Model of Hypermedia System Dialogue

An interactive system can be described in terms of the activities or behaviours in which it appears to engage. CSP is a mathematical notation which allows us to describe systems precisely in these terms. Using CSP, systems can be described in terms of *processes* or behaviour patterns. These processes can in turn be described as sets of *events,* or atomic actions in which some element of the system engages. In the example below, the names of processes are shown beginning with upper case letters, and the names of events are written all in lower case.

CSP provides many different constructs for modelling aspects of the interactions between processes and events. The example below uses only a small subset of the notation. Symbols used include: $\widehat{=}$ which means 'is defined as'; -> which can be read as 'then'; $\sqcap$ and $\Box$ which denote choice between two alternative paths of behaviour, internally and externally to the process in which the choice is made; $||$ which means, roughly speaking, 'in parallel with' or 'at the same time as'; ';' which can be read as 'followed by'; and SKIP which denotes successful termination of a process. For a full description of the precise meaning of these symbols and of their mathematical significance, the reader is referred to [5].

Before building a model of a system or the dialogue which it embodies, we must make decisions about which parts of the system will be included in the model and what level of detail we will use. With CSP, we can model a human-computer system as consisting of two key processes: one representing the user, and the other the system. The interactions between these two processes constitute the human-computer dialogue with which we are concerned here. These top-level processes may be broken down into sub-processes, so that different aspects of the behaviour of either the user or the system may be considered in more detail. For example, we could model the user in terms of separate processes corresponding to auditory and visual attention mechanisms, or to processes dealing with different forms of input device such as the keyboard, mouse or touch screen.

In the model below, we have given a highly simplified view of the user in which we are interested only in his or her ability to perform one of two actions. Our aim has been to focus on elements of the system which characterise it as a hypermedia system, so that we can demonstrate how CSP helps us to tackle some of the unique challenges of hypermedia design described above. While the system we consider here is necessarily small (it contains only two hypermedia nodes), we hope our treatment of it illustrates the way in which more realistic systems could be modelled.

In our example, the two operations of which a user is capable involve asking the system to display the contents of either the first node in the hypermedia network, or the second:

```
User ≙ (open_node_1 -> User) ⊓ (open_node_2 -> User)
```

The system is represented as a collection of processes, some managing the use of the various output media, and some monitoring the user's requests for displaying nodes:

```
System ≙ Text_Displayer || Graphic_Displayer ||

         Video_Player || Sound_Player || Voice_Player

         ||

         (open_node_1 -> Node_1_Display

         □

          open_node_2 -> Node_2_Display)
```

This division of output media managers allows us to reason about the fact that different output media must be handled in different ways. For example, while text and graphics can be presented instantaneously, and may simply remain on the screen until they are cleared, video, sound and voice must be output over a period of time. For this reason, static and dynamic media must be treated differently in composing hypermedia presentations. 'Sound' output has been distinguished from 'Voice' or 'Speech'. This allows us, for example, to model the fact that two fragments of speech should not be played simultaneously, whereas a single speech fragment may be accompanied by other sound such as music. Finally, we have assumed here that computer-generated animation can be viewed as a kind of video output, and that music can be managed by a general-purpose 'Sound_Player' as we have so far not encountered any situations in which these kinds of output need to be distinguished.

Now, to model the fact that node 1 contains text and graphics and that a user's request to open that node should result in the relevant text and graphics being displayed on the screen, we use the following notation:

```
Node_1_Display ≙ Node_1_Text_Display || Node_1_Graphic_Display
```

This states that the process for displaying node 1 consists of two processes running in parallel, one of which displays the relevant text, and the other, the graphic. These two processes may be described in more detail as follows:

```
Node_1_Text_Display ≙ begin_node_1 -> display_node_1_text -> SKIP

Node_1_Graphic_Display ≙ begin_node_1 -> display_node_1_graphic
                                    -> SKIP
```

Notice that both begin with an event called 'begin_node_1'. This event is used to synchronise the two processes, so that we have some assurance that the text and graphics will be displayed at the same time.

The system processes which manage the display of text and graphics are described as:

```
Text_Displayer ≙ display_node_1_text -> Text_Displayer

Graphic_Displayer ≙ display_node_1_graphic -> Graphic_Displayer
```

These descriptions seem rather unwieldy in the context of such a small specification, but in a larger example, they would be expanded and parameterised to describe the fact that text or graphics from any node in the hypermedia network could be displayed.

Moving on to the second of the nodes in our toy network, we wish to describe the fact that activating node 2 starts a video. The beginning of the video is accompanied by music, but part way through the video, the music stops, and a fragment of speech is played in its place. We model this arrangement as follows:

```
Node_2_Display ≙ Node_2_Video_Play ||

                 (Node_2_Sound_Play ; Node_2_Voice_Play)
```

Thus the video for node 2 is specified as being shown in parallel with a sound sequence consisting of a short piece of music followed by some speech.

The description of playing a video is slightly more complicated than that of displaying a static graphic, owing to the fact that it is extended in time. We need to model both the beginning and end points of the video so that we can synchronise the presentation of other media with those points. In the description below, we have also modelled a mid-point in the video which is the time when the second part of the soundtrack (containing the speech fragment) should be begun.

```
Node_2_Video_Play ≙ begin_node_2 -> begin_node_2_video_play ->
                    mid_node_2_video_play -> end_node_2_video_play
                    -> SKIP
```

The processes for playing the music and speech clips are modelled in an analogous way, with the event 'begin_node_2' being used to synchronise the beginning of the video with the start of the music, and the event 'mid_node_2_video_play' being used to time the introduction of the speech as intended:

```
Node_2_Sound_Play ≙ begin_node_2 -> begin_node_2_sound_play ->
                     end_node_2_sound_play -> SKIP

Node_2_Voice_Play ≙ mid_node_2_video_play ->
                     begin_node_2_voice_play ->
                     end_node_2_voice_play -> SKIP
```

Finally, we model the process managing the display of the video in a manner analogous to that used in 'Text_Displayer' and 'Graphic_Displayer' above. The only difference is that the ongoing playing of successive sections of the video is described recursively as a process, rather than being modelled as a single event (as in the case of 'display_node_1_text' or 'display_node_1_graphic'):

```
Video_Player ≙ begin_node_2_video_play -> Play_Video_2

Play_Video_2 ≙ Play_First_Section_Video_2 ; Play_Rest_Video_2

Play_First_Section_Video_2 ≙ first_section_video_2 -> SKIP

Play_Rest_Video_2 ≙ next_section_video_2 -> Play_Rest_Video_2
                    □
                    mid_node_2_video_play -> Play_Rest_Video_2
                    □
                    last_section_video_2 ->
                    end_node_2_video_play -> SKIP
```

The 'Sound_Player' and 'Voice_Player' would also be modelled recursively, but will not be described here due to constraints of space.

Considering the hypermedia systems built at the University of Hertfordshire, we can see that the simple CSP constructs used in the example above can be used to model most of their distinctive features. Our simple bibliographic database system was built using HyperCard [7], a hypertext development environment for the Apple Macintosh. However, a brief analysis of the dialogue it supports revealed that interaction with this system was much like interaction with a wide range of other windows-based systems in which commands can be issued to the system through the use of buttons and menus. Dialogue of this kind can be described using CSP in the way demonstrated by Alexander in [3] and [8].

A second system, MODEMA [9], was built using KnowledgePro [10], a knowledge-based system and hypertext development tool which runs under Microsoft Windows. MODEMA is an information system for employers of people with disabilities. Its main use of distinctive hypermedia features is in the concurrent display of text and graphic information about products and case studies of interest. This method of displaying information can be described with the CSP constructs used to model Node 1 in the example above.

Finally, we may consider Libtech [11], a hypermedia exhibition guide, again developed for the Macintosh. Libtech also displays information using a combination of text and graphic stills. Additionally, it makes use of video sequences which were intended to introduce exhibition attendees to features of the location of the exhibition for which it was built. As we have seen, the use of video can be described with simple CSP constructs such as those used in Node 2 above. If we wanted, in future, to consider the addition of speech or music fragments to accompany the videos, we would also be able to do this using the techniques described above.

Some features of our systems are not incorporated in the simple example presented here. For example, we have not modelled the fact that the user may interrupt a video at any time and switch to viewing a different node in the system. Nor have we modelled the co-ordination of low-level events which are intended to happen at the same time in displays of different forms. For example, in Libtech, a mouse click on an element in a piece of text can result in highlighting, not only of that piece of text, but also of a corresponding element in an accompanying graphic. Perhaps the most important omission, however, is that of error conditions. We aim in future to extend models like that shown above, to investigate how effectively they allow us to reason about aspects of hypermedia systems such as these.

5. Conclusions

A number of the distinctive features of dialogue with hypermedia systems can be modelled quite naturally using the CSP notation. The example presented in section 4 shows how it provides us with a basis for reasoning about key concerns of integrating presentations using a number of different output media, some of which are temporally extended and therefore pose particular problems of synchronisation. It thus provides a formal basis for modelling at least some of the features identified as being of interest in the recent Amsterdam model of hypermedia systems [12], and assists the hypermedia designer in avoiding 'media ghettos' [13] by supporting better integration of information represented using different media.

One important feature of formal models such as that presented above is that they provide a basis for stating and proving properties which the system modelled is intended to have. A property of hypermedia systems in general is that they should not require the user to attend to more than one fragment of speech at any one time. This property can be expressed in terms of CSP event traces, and can be shown to hold (or not) for a particular system by a process of mathematical reasoning for which automated support can be provided. Individual clients may also wish to state particular requirements for hypermedia systems in terms of, for example, their resource usage. If a client does not have the hardware necessary for producing sound output, we can model the fact that their system should not make use of sound or voice output, and show, for any proposed system design, that this is the case. We can also, as system designers, compare different models of a system to discuss the effectiveness with which they support particular activities. For example, the fact that one potential implementation might require a user to engage in more low-level activities (such as keypresses and mouse clicks) than another would be evident from CSP traces of system events.

We aim to develop the work described in this paper in a number of directions. As described earlier, our first priority will be to extend models such as those presented in section 4 to investigate the extent to which they can be scaled up to model complete systems such as those we have already built, and the way in which they can then be used to support design and development activities as suggested above. We also intend to investigate the use of extensions to CSP such as Timed CSP (see, for example, [14]) in describing timing and synchronisation constraints between presentations using different media more precisely. Finally, once we have developed a fuller understanding of the potential applications of CSP in the design of hypermedia systems, we would like to evaluate its utility with respect to other notations for modelling concurrent systems.

Acknowledgements

The authors would like to thank Ben Potter, Jean Baillie, Martin Loomes, Jill Hewitt, Mike Bearne and Ian French for the advice and discussions which contributed to the writing of this paper.

References

[1] M. Green. A survey of three dialogue models. *ACM Transactions on Graphics,* 5(3), July 1986.

[2] P.A. Palanque, R. Bastide, L. Dourte and C. Sibertin-Blanc. Design of user-driven interfaces using Petri nets and objects. In *Advanced Information Systems Engineering,* C. Rolland, F. Bodart, C. Cauvet (eds), Proceedings of the Fifth International

Conference, CAiSE '93, Paris, France, June 1993. Lecture Notes in Computer Science 685, Springer-Verlag.

[3] H. Alexander. *Formally-Based Tools and Techniques for Human-Computer Dialogues.* Ellis Horwood, 1987.

[4] G.D. Abowd. Agents: Communicating interactive processes. In *Human-Computer Interaction - INTERACT '90,* D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds), Elsevier Science Publishers B.V. (North-Holland), 990.

[5] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[6] R. Dannenberg and M. Blattner. The trend toward multimedia interfaces. In *Multimedia Interface Design,* M.M. Blattner and R.B. Dannenberg (eds), Addison-Wesley, 1992.

[7] HyperCard Reference, Claris, 1989.

[8] H. Alexander. Structuring dialogues using CSP. In *Formal Methods in Human-Computer Interaction,* M. Harrison and H. Thimbleby (eds), chapter 9, Cambridge University Press, 1990.

[9] J. Hewitt, J. Sapsford-Francis, G. Bolstad, O. Eftedal, P. Halford, D. Vervenne and M. Verheyen. MODEMA - A knowledge based browsing system to facilitate the employment of people with disabilities. In *Rehabilitation Technology,* E. Ballabio, I. Placencia-Porrero and R. Puig de la Bellacasa (eds), IOS Press, 1993.

[10] KnowledgePro Windows User Manual, Knowledge Garden Inc, 1991.

[11] J. Hewitt, J. Sapsford-Francis and P. Halford. Use of multi-media in a public information system. In *Proceedings of the 1993 International Symposium on Multi-Media Technologies and Future Applications,* BCS/IEEE/RTS, Southampton, April 1993.

[12] L. Hardyman, D.C.A. Bulterman and G. Van Rossum. The Amsterdam hypermedia model: Extending hypertext to support real multimedia. *Hypermedia* 5(1), 1993.

[13] B. Laurel, T. Oren and A. Don. Issues in multimedia interface design: Media integration and interface agents. In *Multimedia Interface Design,* M.M. Blattner and R.B. Dannenberg (eds), chapter 3, Addison-Wesley, 1992.

[14] G.M. Reed, A.W. Roscoe and S.A. Schneider. CSP and timewise refinement. In *4th Refinement Workshop,* J.M. Morris and R.C. Shaw (eds), Proceedings of the 4th refinement workshop, organised by BCS-FACS, January 1991, Cambridge. Springer-Verlag, 1991.