

# Modeling and Prototyping of Distributed Multimedia Systems for Flexible Working

Koji Takeda Xiaobo Wang Mitsuyuki Inaba Kazuo Sugihara Isao Miyamoto

Department of Information and Computer Sciences

University of Hawaii at Manoa

Honolulu, Hawaii 96822, U. S. A.

Email: {koji|xiw|inaba|sugihara|miyamoto}@nile.ics.hawaii.edu

Fax: +1 808 956 3548

## Abstract

*Due to recent advances in the areas of multimedia systems and groupware, flexible working is emerging as a new way of working, where workers carry out tasks using computers no matter where, when and how they work. This paper discusses modeling and prototyping of a distributed multimedia system which is used to provide a flexible working environment. First, we propose a model to specify distributed multimedia systems (especially, user interfaces) for flexible working. The model is based on finite state machines and information exchanged between them. The specifications in our model are executable so that the specifications can be used as a prototype of the distributed multimedia system. Second, a prototyping tool is introduced which is an interpreter of executable specifications in the model. The tool is currently operational on the Mac family such as Mac II, Power Macintosh and PowerBook.*

**Keywords and Phrases:** Flexible working, distributed multimedia systems, modeling, prototyping, agent-based technology, multimedia user interfaces, and executable specifications

## 1 Introduction

The concept of *flexible working* [7] has been with us for many years. It has many aliases: Teleworking, home working, outworking, mobile computing, telecommuting, telecomputing, etc. However, flexible working practices have not been widespread. Some of reasons for limiting the uptake of flexible working so far are cost; immaturity of required technologies; the need for a major change in the way that work is valued; data security; etc.

Due to recent advances in the areas of *multimedia systems* [1, 3, 4] and *groupware* [5, 9], however, very few barriers will prevent the upcoming growth of flexible working applications in the next decade. For example, in 1989, the State of Hawaii established the *Telework Center* which is a very first one of its kind in the United States. Its goals are to demonstrate the use of state-of-the-art technology; acquire experience in designing and operating future offices; and evaluate the performance of telework employees, including benefits, problems and the overall effectiveness of technology.

In a flexible working environment, multimedia will be used to enhance communication not only between a user and a computer, but also among users (i.e., workers). A system for supporting flexible working will be a groupware. Hence multimedia data and computing in the system will be distributed. Therefore, a *distributed multimedia system* will play a key role in supporting flexible working.

Another key to flexible working is *agent-based* technology [10, 12]. If applications for performing tasks are fairly autonomous (i.e., each application can carry out its task by itself independently to some extent), then the user interaction required to control the application can be lessened. In contrast to the traditional client-server approach, the amount of communication needed to perform the task can also be reduced.

A flexible working environment is supported by a collection of intelligent agents, each of which performs a particular task for a worker. As an example, a stereotypical organization of agents is shown in Figure 1.

This paper discusses modeling and prototyping of distributed multimedia systems (especially, user interfaces) for flexible working. Prototyping is very important to develop a flexible working environment, since

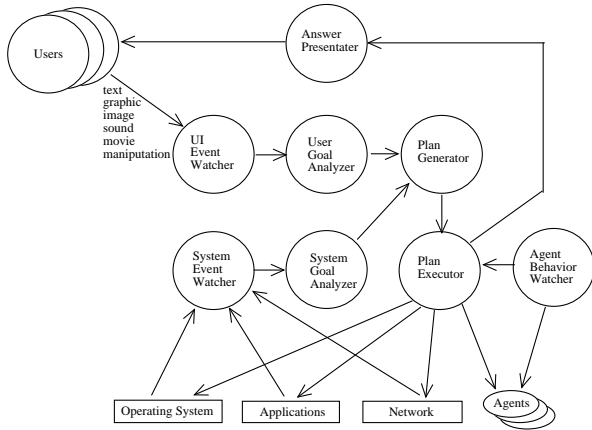


Figure 1: A stereotypical organization of agents for flexible working.

available hardware and software for multimedia data and computing are rapidly changing and they drastically affect the design of a flexible working environment. It is inevitable in the requirements analysis and design of a system, where a customer must be able to understand and confirm what is being developed. It enables us to detect some potential problems in an early stage of the development. The prototyping also allows us to assess advantages of a new multimedia product, which can be used in a distributed multimedia system for flexible working, and examine its potential advantages and disadvantages.

First, we propose a model to specify a distributed multimedia system for flexible working. The model is based on finite state machines and information exchanged between them. The specifications in our model are executable so that the specifications can also be used as a prototype of the distributed multimedia system. Second, a prototyping tool is introduced which is an interpreter of executable specifications in the model. The tool is currently operational on the Mac family such as Mac II, Power Macintosh and PowerBook.

## 2 Flexible Working Environments

There are the following four major aspects which the design of a flexible working environment should take into account.

1. Workflow
2. Agents
3. User interfaces

## 4. Communications

*Workflow* captures procedures and activities that workers have to perform. *Petri nets* [8] are often used to describe it [2]. We also adopt Petri nets to specify workflow in a flexible working environment. An example of workflow for ordering, invoicing and payment is shown in Figure 2.

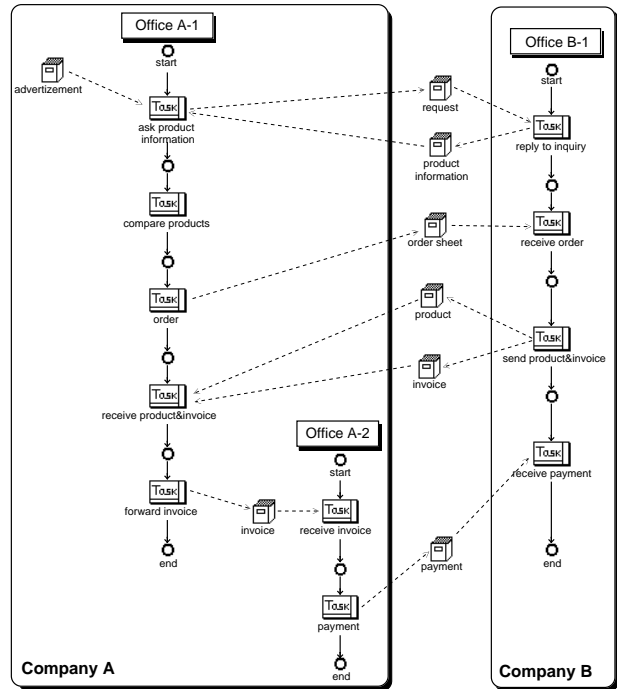


Figure 2: An example of workflow involving multiple parties.

In order to describe the four aspects, we use graphical models whose primitive constructs are entities, relations and attributes. Formally, the graphical model is defined in *MERA* (Meta-Entity-Relation-Attribute) [11] which is a generalization of the well-known ER (Entity-Relationship) model. However, this paper gives intuitive examples of models so that the reader is not required to know about MERA.

As an example of a flexible environment, let us consider the following scenario.

In a company, two executives *A* and *B* are involved in some R&D project. *B* wants to have a discussion on an issue with *A*, a chief *C* of the division in charge of the project, and a project leader *D*. *B* asks a secretary *S* to arrange a way of the discussion. *A* is now on a business trip, but reads emails by using a portable computer while traveling. *C* and *D* are working in a branch of the company

which is far from the company's headquarters where *A*'s office is located.

Typical workflow for conferencing is shown in Figure 3. It describes how the secretary *S* would arrange the conference and the participants *A*, *B*, *C* and *D* would discuss the issue at the conference. A "conference" in a flexible working environment may not be a meeting in an ordinary sense. It can be video-conferencing and/or exchanging opinions via email. This example will be used in Sections 3 and 4 to illustrate our model and prototyping tool.

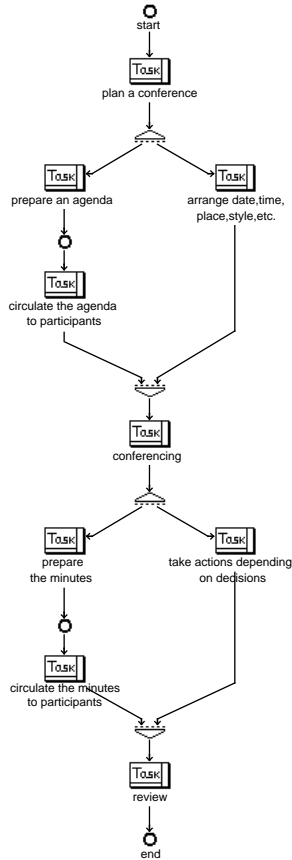


Figure 3: An example of workflow for conferencing.

### 3 A Model for Distributed Multimedia Systems

Although we aim at modeling of an entire flexible working environment, in this paper we focus on multimedia user interfaces for flexible working. A flexible working environment can be regarded as a collection of distributed agents which carry out their tasks by

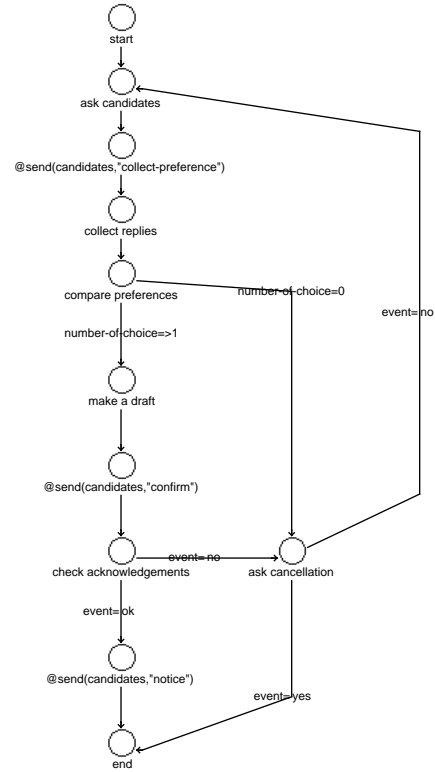


Figure 4: A state transition diagram of an agent for arranging a conference.

communicating to each other. In our model, a user interface is specified for each agent and interactions among agents are explicitly specified as communications among the agents. Thus, a distributed multimedia system as a whole has a user interface which is a collection of the interrelated user interfaces of the agents.

A *state transition diagram* is used as a *script* in which an agent (especially, its user interface) is specified. If the agent communicates with another agent to perform its own task, the state transition diagram also specifies information to be exchanged between them. For example, Figures 4, 5 and 6 show state transition diagrams of the agents which perform the task to arrange a date, time, place, style, etc. as shown in Figure 3.

The agent (say, *agent-1*) shown in Figure 4 serves for the person who is arranging a conference (i.e., the secretary *S* in the previously mentioned scenario). A state with a label *@send* is to send a message to sites where candidates of participants reside. The first argument of the command *@send* is a list of sites to which the message is sent. The second argument indicates the message to be sent. The content of a message can

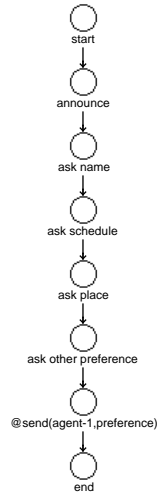


Figure 5: A state transition diagram of agent *collect-preference*.

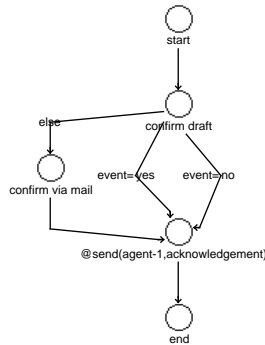


Figure 6: A state transition diagram of agent *confirm*.

be more than just a text. For example, the command `@send(candidates, "collect-preference")` is to send another agent (named `collect-preference`) to sites in the list `candidates`. The agent *collect-preference* is executed at each of those sites when it is received. Similarly, Figure 4 includes states in which two more agents *confirm* (see Figure 6) and *notice* are sent to the sites in `candidates`. As we will mentioned in Section 4, sending an agent is implemented in our prototyping tool by sending a state transition diagram of the agent.

The agent *collect-preference* shown in Figure 5 asks questions to a prospective participant; collects his/her preferences on the conference; and then returns the preferences to *agent-1*. The agent *confirm* shown in Figure 6 informs a participant a draft of the conference plan arranged by *agent-1* and has him/her confirm the draft.

A list of actions is associated with each state in

Op-Code	Function	Argument 1	Argument 2	Evaluation
Assign	Assign value to field	Field Name	Symbol or Field Name	Field Value
Beep	Sound beep	Number (Option)		
Clear	Clear a field value	Field Name		NULL
Clr	Clear a screen			
Display	Display a field	Field Name	Display Type	
Flash	Flash a field	Field Name	Number (Option)	
Hide	Hide a field	Field Name		
Inkey	Input one character			Input Character
Input	Input a value	Field Name	Number (Option)	Input String
Device	Select a field			Field Name
Timeout	Set input waiting time	Number (Option)		
Reset	Initialize a field	Field Name		Initial Value
Test	Check a field value	Field Name		Field Value
wait	Delay	Number (Option)		
Menu	Display a pull down Menu	Field Name		Item Name
PopUp	Display a popup menu	Field Name		Item Name
Enable	Enable a menu	Field Name	Menu Name	
Disable	Disable a menu	Field Name	Menu Name (Option)	
Color	Set color	Symbol		
RdField	Read a file into a field	Field Name	Symbol or File Name	
WrField	Write a field into a file	Field Name	Symbol or File Name	
Wopen	Open a window			
Wclose	Close a window			
Invoke	Invoke a command	Symbol or Field Name	Symbol or Field Name	
Play	Play a movie	Field Name		
Speak	Speak a text	Field Name	Voice type	

Figure 7: A list of available actions.

a state transition diagram. The action list describes what actions are executed at the state and in which order. Available actions are shown in Figure 7. The last two actions are typical ones relevant to multimedia data.

An action list of the state “ask place” in Figure 5 is shown in Figure 8. Each action consists of an icon and arguments (typically one or two). For example, actions in Figure 5 are the following.

1. Display data given in “message field” (where the number “8” indicates to display the data in a rectangular area).
2. Output the same data “message field” with a fe-

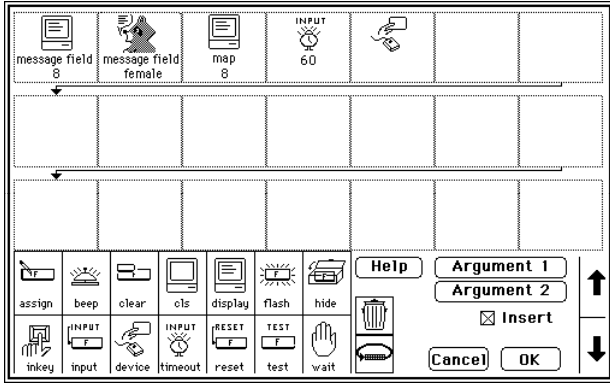


Figure 8: An action list for the state “ask place”.

male voice.

3. Display the data “map” which is an image of a map.
4. Set a timeout for the next action to 60 seconds.
5. Wait for a user’s selection (choice of a location on the map).

An action list of the state “confirm draft” in Figure 6 is shown in Figure 9.

1. Display data given in “draft-explain”.
2. Output the same data “draft-explain” with a female voice.
3. Display a button specified by “yes”.
4. Display a button specified by “no”.
5. Set a timeout for the next action to be 60 seconds.
6. Wait for a user’s selection.

State transition diagrams of the agents for conferencing (see Figure 3) are shown in Figures 10 and 11. The agent shown in Figure 10 serves the person who called for the conference (i.e., the executive *B* in the previously mentioned scenario). It sends a state transition diagram of another agent *start-video-conference* to sites where participants are. Each copy of *start-video-conference* serves for a participant, sets up a video conference and starts it. A command `@wait` in Figure 11 is to wait for a message from another agent. Its argument is a list of agents from which a message is expected.

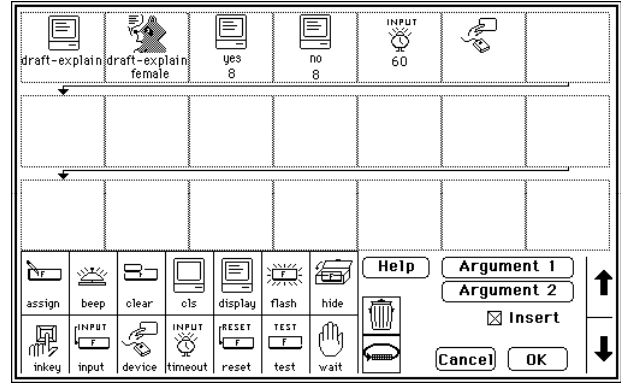


Figure 9: An action list for the state “confirm draft”.

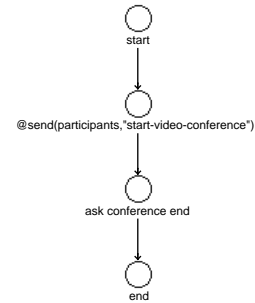


Figure 10: A state transition diagram of an agent for conferencing.

## 4 A Prototyping Tool

We introduce a prototyping tool called *AP* (Agent Prototyper). It is an interpreter of specifications in the model of a distributed multimedia system for flexible working. A copy of *AP* resides on each machine in a network and executes the specifications written in the model.

*AP* is currently implemented in C and runs on the Mac family such as Mac II, Power Macintosh and PowerBook. It deals with the following requirements for prototyping of distributed multimedia systems.

1. Multimedia data: *AP* uses QuickTime and Mac-inTalk to accommodate multimedia data including image, movie and sound.
2. Communications: *AP* supports distributed applications through a network. Apple event-based protocol is used for communication between applications. A lower level communication protocol available in the network can also be used for more flexible communication.
3. Agents: Two or more copies of *AP* communicate with each other whenever they encounter

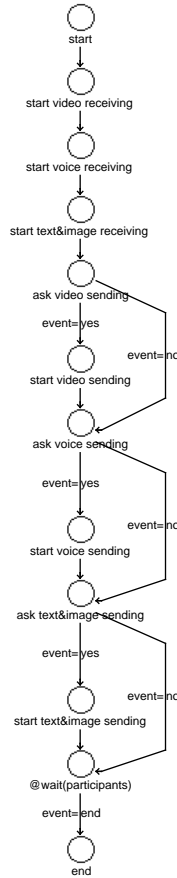


Figure 11: A state transition diagram of agent *start-video-conference*.

a command `@send` or `@wait` in an interpreted model. This enables us to prototype multiple agents which communicate with each other.

Next, we explain how we prototype a detailed user interface of an agent by using AP. Here the agent to be prototyped will support the task of arranging a conference which is defined as the second task in the workflow model for conferencing discussed in Section 2 (see Figure 3).

The first step of prototyping an agent for conference arrangement is to write a more detail workflow model of that particular task. By doing so, we can understand the conventional procedure for conference arrangement and make the behavior of a computer agent familiar to the user. If the detail workflow model is difficult to write, an alternative way is to write down in text a typical interaction among people in such a setting as shown below, where participants are secretary (S), executive on a trip (A), another executive (B), division chief (C), and project leader (D).

From — To Message

B — S "Please schedule a conference."

S — B "Please give me the following information:"  
 - Purpose of the conference  
 - Names of conference participants  
 - Expected length of conference  
 - Meeting place  
 - By when conference must be held

B — S "Here is the information."  
 - Purpose of the conference: To discuss impact of this war  
 - Names of conference participants: A, B, C, D  
 - Expected length of conference: One hour  
 - Meeting place: Somewhere in Hawaii  
 - By when conference must be held: Within one week

S — A, B, C, D "Please provide me your availability for the following conference. Also let me know which island you want to have the conference."  
 - Purpose of the conference: To discuss impact of this war  
 - Names of conference participants: A, B, C, D  
 - Expected length of conference: One hour  
 - Meeting place: Somewhere in Hawaii  
 - By when conference must be held: Within one week

A — S "Anytime after 10/25 because I am on a trip. Any island is OK with me."

B — S "Anytime, except for Monday and Friday. I prefer Maui."

C — S "Monday afternoon or Wednesday afternoon. Maui is fine."

D — S "Anytime. Anywhere."

S — A "Please give me your preference over the following choices."  
 (1) Early afternoon of Wednesday  
 (2) Late afternoon of Wednesday

A — S "I prefer (1)."

B — S "I prefer (1)."

C — S "I prefer (2)."

D — S "I prefer (1)."

S — A, B, C, D "Please confirm the following conference schedule."  
 - On November 2, Wednesday, 1:00 pm  
 - At Maui branch  
 - To discuss impact of this war  
 - By A, B, C, D

A — S "Confirmed."

B — S "Confirmed."

C — S "Confirmed."

D — S "Confirmed."

The second step is to write a script of the agent, which arranges the conference, by using a state transition diagram (see Figure 12). In this script, the main agent performs the task by generating and sending sub-agents which assist communication with candidates of conference participants at dispersed locations. We will proceed with prototyping of "Get-availability" sub-agent below.

The third step, which may be done concurrently with the fourth step, is to design the layout of the user interface for the sub-agent (see Figure 13) and list up data to be given to the user such as texts and

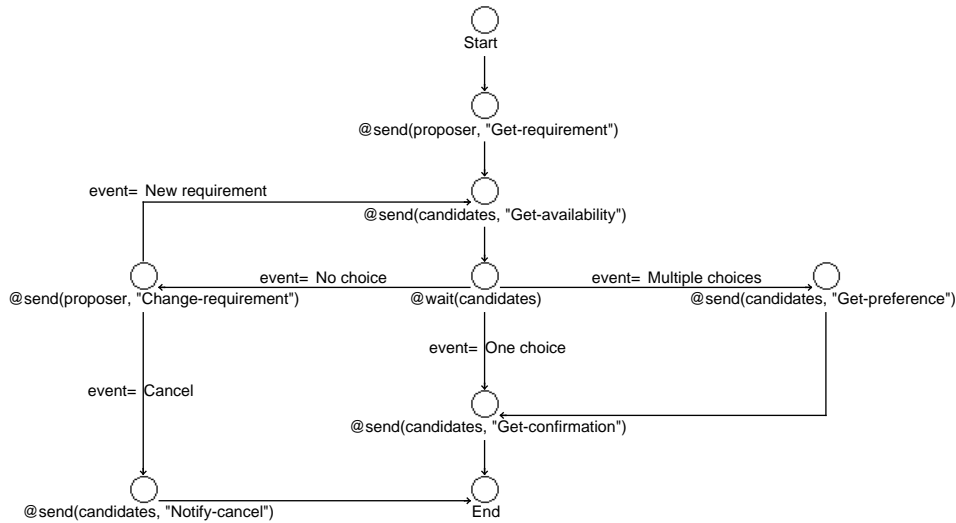


Figure 12: A script for an agent arranging a conference.

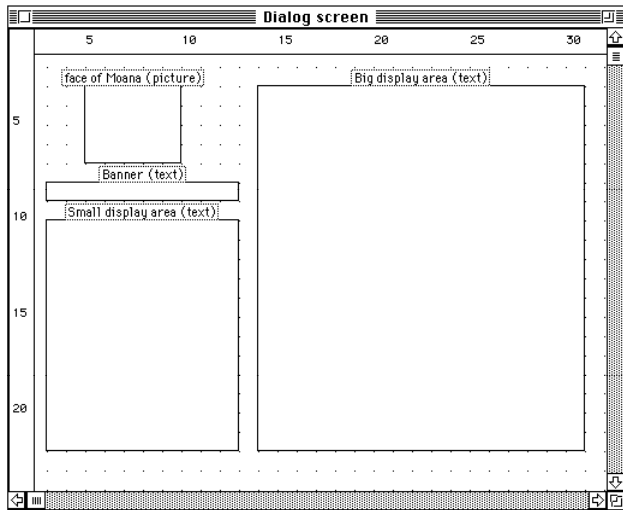


Figure 13: Display layout.

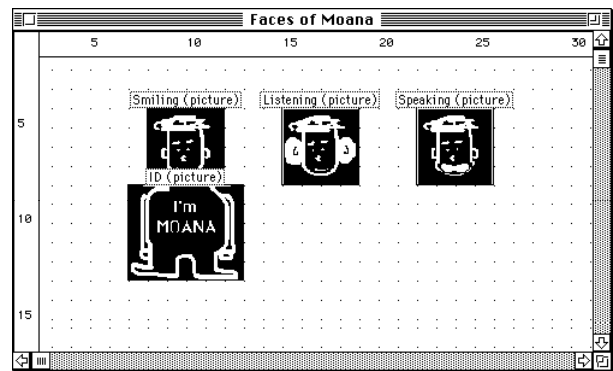


Figure 14: Images to be display.

images to be displayed on the screen (see Figures 14 and 15).

The fourth step is to write a script of “Get-availability” agent (see Figure 16).

Finally, a list of actions is attached to each state in the script (see Figures 17, 18 and 19). Note that in this example, an “invoke” action is used to invoke “Get time & place preference” agent in a specific area of the screen. In this way, it is possible to build a user interface consisting of multiple agents.

The prototype specification given above is executable. For example, Figure 20 is a snapshot of the screen while “Get answer” state is executed.

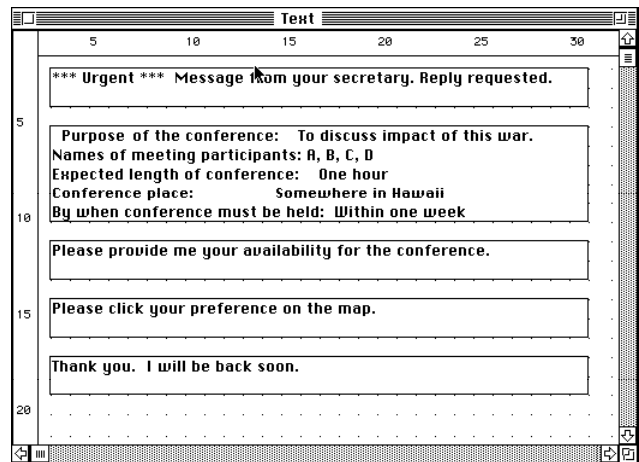


Figure 15: Texts to be display.

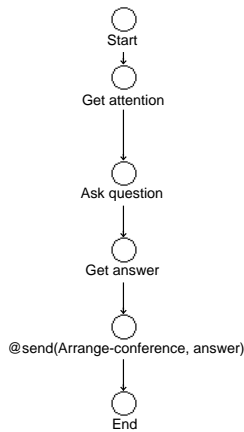


Figure 16: A script for “Get-availability” agent.

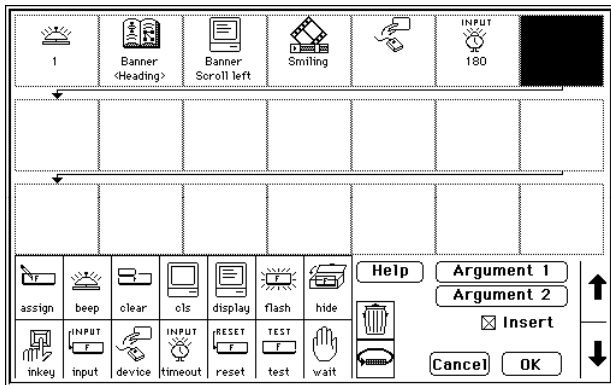


Figure 17: A list of actions for “Get attention”.

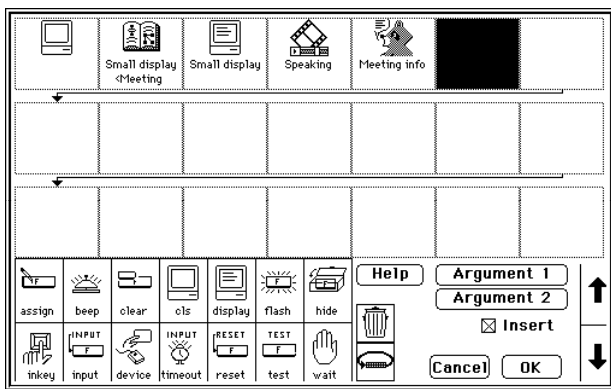


Figure 18: A list of actions for “Ask question”.

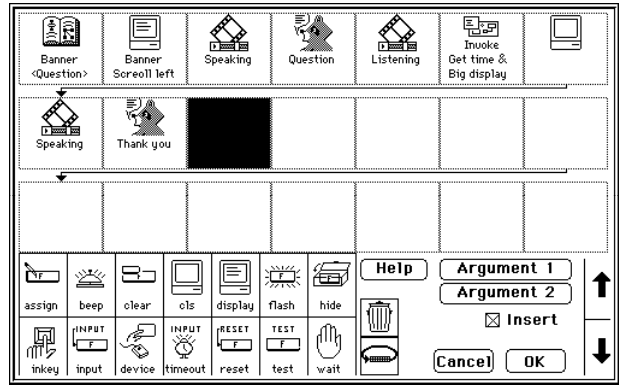


Figure 19: A list of actions for “Get answer”.

Figure 21 shows another snapshot of the screen where an agent for the task “conferencing” in Figure 3 is being prototyped.

## 5 Conclusion

This paper presented a model for distributed multimedia systems (especially, multimedia user interfaces) which are used to support flexible working. A prototyping tool was also introduced which executes specifications of a distributed multimedia system. The current version of the tool is operational on the Mac family such as Mac II, Power Macintosh and PowerBook.

We plan to improve and extend the model and the tool in the following directions.

1. *Hypermedia* [6]: A certain trend in flexible working is to support multimedia data that are structured for information sharing among workers in a sophisticated way. Hypermedia is a promising approach to this demand.
2. *Platform Independence*: For practical use, the prototyping tool should be able to run on a variety of platforms, rather than only the Mac family.
3. *Collaboration, Negotiation and Migration*: Agents are often required to perform their tasks as a group through collaboration, negotiation and migration. A model for a flexible working environment should be able to describe the behavior of each agent in the group and the process of collaboration, negotiation and migration. The model will enable us to prototype “intelligent” agents.

## Acknowledgements

This research is supported in part by PFU, Inc. in Japan. The authors would like to thank anonymous

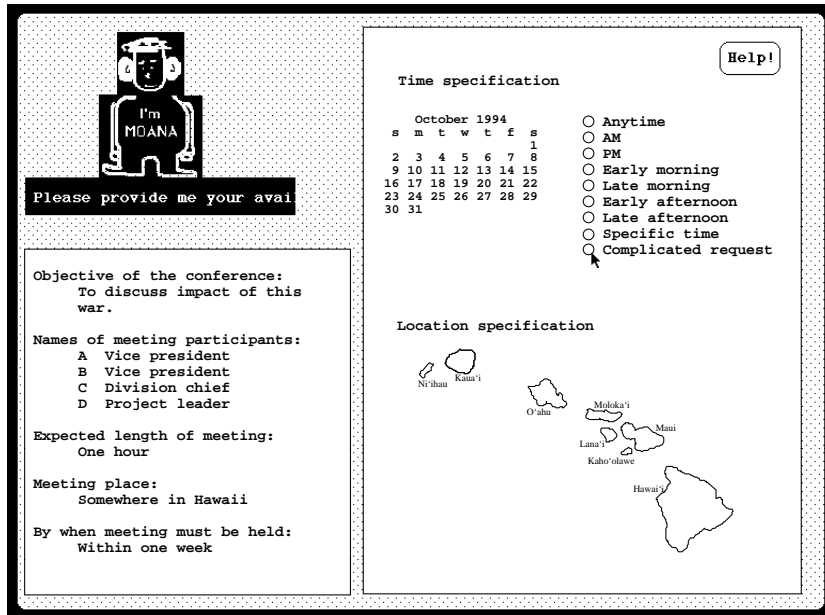


Figure 20: A snapshot of executing “Get answer” state.

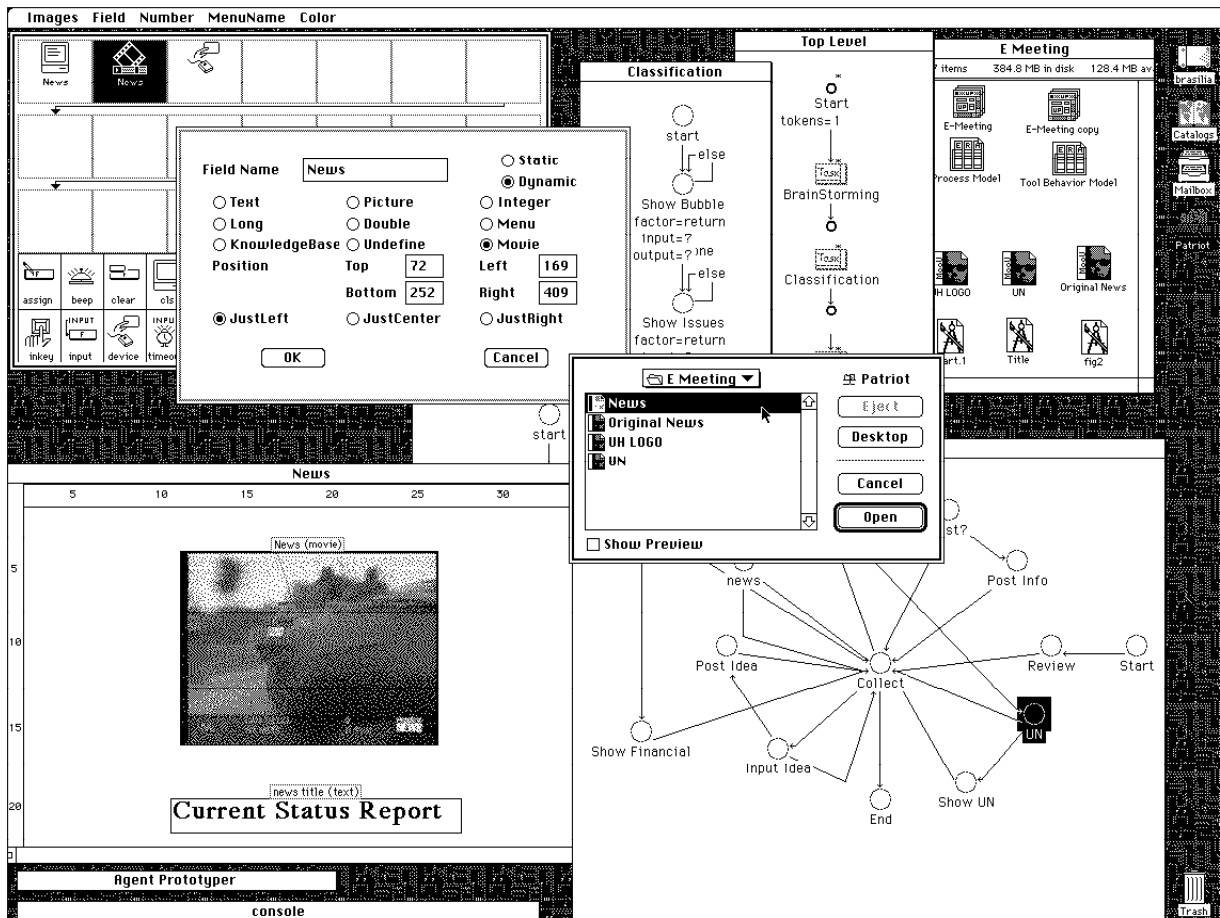


Figure 21: A snapshot of prototyping an agent for “conferencing”.

reviewers for their comments on an earlier version of this paper.

## References

- [1] John A. Adam, "Interactive multimedia: Applications, implications," *IEEE Spectrum*, Vol. 30, No. 3, pp. 22-31, March 1993.
- [2] S.-K. Chang, *Tutorial on Distributed Multimedia Information System*, 1994 Pacific Workshop on Distributed Multimedia Systems, Feb. 23, 1994, Taipei, Taiwan, Knowledge Systems Institute, 1994.
- [3] Bernard Cole, "Interactive multimedia: The technology framework," *IEEE Spectrum*, Vol. 30, No. 3, pp. 22-31, March 1993.
- [4] Borko Furht, "Multimedia systems: An overview," *IEEE Multimedia*, Vol. 1, No. 1, pp. 47-59, Spring 1994.
- [5] Jonathan Grudin, "Computer supported cooperative work: History and focus," *IEEE Computer*, Vol. 27, No. 5, pp. 19-26, May 1994.
- [6] L. Hardman, D. Bulterman and G. van Rossum, "The Amsterdam hypermedia model: Adding time and context to the Dexter model," *Comm. of ACM*, Vol. 37, No. 2, pp. 50-62, Feb. 1994.
- [7] Stephen McClelland, "Problems, Perceptions, Prescriptions and Policy Options in Flexible Working," *Pacific Telecommunications Review*, pp. 4-9, March 1994.
- [8] Tadao Murata, "Petri nets: Properties, analysis and applications," *Proc. of IEEE*, Vol. 77, No. 4, pp. 541-580, April 1989.
- [9] James D. Palmer and N. Ann Fields, "Computer-supported cooperative work," *IEEE Computer*, Vol. 27, No. 5, pp. 15-17, May 1994.
- [10] Andy Reinhardt, "The network with smarts," *BYTE*, Vol. 19, No. 10, pp. 51-64, Oct. 1994.
- [11] Koji Takeda, David N. Chin and Isao Miyamoto, "MERA: Meta language for software engineering," *Proc. 4th Int'l Conf. on Software Engineering and Knowledge Engineering*, Capri, Italy, June 15-20, 1992, pp. 495-502.
- [12] Peter Wayner, "Agents away," *BYTE*, Vol. 19, No. 5, pp. 113-118, May 1994.