# User Interface and Agent Prototyping for Flexible Working

Koji Takeda, Mitsuyuki Inaba, and Kazuo Sugihara
*University of Hawaii at Manoa*

**A collaborative work group that links workers through distributed computers is only as productive as the system itself. Software agents are emerging as a way to boost workers' productivity, performing group tasks independently. The authors use a typical business conference as an example to show how graphical models and agent prototyping empower flexible working environments.**

The flexible working[1] concept—also known as teleworking, telecommuting, home working, outworking, mobile computing, and telecomputing—has been with us for awhile. Flexible working lets employees perform tasks with computers no matter where, when, or how they work. (See the "Teleworking and Mobile Computing" sidebar). For example, Hawaii's Department of Transportation established the Telework Center in 1989, chiefly to reduce Honolulu's traffic congestion. The center is a good example of state-of-the-art technology, office design and operation, and telework employee performance. Last year, in fact, a center was established on another island. Flexible working practices are not yet widespread, however, primarily because of cost, immature technologies, data security issues, and the need for individuals and corporations to change the way they value work. We believe that recent advances in multimedia systems[2,3] and groupware[4] will lower the barriers to flexible working and spur an upswing in new applications over the next decade.

In flexible working, multimedia enhances communications—both the electronic kind (between users and computers) and the personal kind (between workers). Much as groupware does, then, a distributed multimedia system[5] plays a key role in flexible working support.

The focus of our research, current results of which we describe here, is on prototyping of distributed multimedia systems for flexible working. Ultimately, we hope to develop a prototyping tool for an entire flexible-working environment. The design for such an environment integrates three major components: workflow, agents, and user interfaces.

In this article, we present graphical models for specifying a flexible working environment, especially multimedia interfaces and agents. The models are based on Petri nets and finite state machines, and the information exchanged between them. Our models' specifications are executable and so can be used as a prototype of the distributed multimedia system. We also introduce a prototyping tool that interprets the models' executable specifications. To illustrate our process, we use a hypothetical example of a typical flexible-working situation: arranging a business conference.

For our purposes, we define prototyping as system modeling and simulation. Prototyping is essential especially in system requirements analysis and design, where a customer must understand and confirm what is being developed. It also enables us to detect potential problems early in development, evaluate possible solutions, and obtain feedback from prospective users.

Prototyping contributes to multimedia system development in two ways. First, it lets us assess new products so that we can evaluate advantages and disadvantages. This is vital because multimedia hardware and software change rapidly, which drastically affects system design. Second, prototyping helps users make a smoother transition from an older system to a new one through exposure to the new system.

In designing a system for flexible working, prototyping is desirable for other reasons, too. First, it is difficult to completely specify such a complex system in advance, where elements execute and interact asynchronously. Another reason is that user interface evaluation strongly depends on users' subjective viewpoints, which can vary. Finally, prototyping promotes users' early involvement in system development, which is necessary for developers to specify requirements and to acquire application domain knowledge.

## Flexible-working concepts

As mentioned, a flexible-working environment design includes workflow, agents, and user interfaces. In each component, the communication aspects are specified and analyzed. Here we briefly overview research areas related to workflow and agents.



1070-986X/96/$5.00 © 1996 IEEE

## Teleworking and Mobile Computing

The following are on-line information sources for teleworking and mobile computing.

| Topic | URL |
|---|---|
| Index on mobile computing research | http:/www.citi.umich.edu/mobile.html |
| Index on telecommuting | http://www.engr.ucdavis.edu/~its/telecom/related.html |
| Searchable bibliographies | http://liinwww.ira.uka.de/bibliography/Distributed/ |
| Telecommuting advisory council | http://www.telecommute.org/ |
| Telecommuting and teleworking | http://www.gilgordon.com/ |
| Telework center in Hawaii | http://www.htdc.org/~lttc/ |
| Teleworking in Europe | http://www.agora.stm.it/ectf/ectfhome.html |
| Teleworking in Japan | http://www.mpt.go.jp/MPT-News/vol6-4/contents.html |
| Virtual library on mobile and wireless computing | http://snapple.cs.washington.edu/mobile/ |
| Virtual office | http://virtual.office.com/tech/home.html |
| Web resources on mobile computing | http://www.mcl.cs.columbia.edu/ |

Additional information can be found with search engines or by browsing Internet directories such as Yahoo at http://www.yahoo.com/, WWW Virtual Library at http://www.w3.org/hypertext/DataSources/bySubject/Overview.html, EINet's Galaxy at http://galaxy.einet.net/galaxy.html, and Virtual Computer Library at http://www.utexas.edu/computer/vcl/.

## Computer-Supported Collaborative Work

The following are on-line information sources for computer-supported collaborative work (CSCW).

| Topic | URL |
|---|---|
| CSCW annotated bibliography | ftp://ftp.cpsc.ucalgary.ca/pub/projects/CSCWbibliography/ |
| CSCW directory | http://www.demon.co.uk/jrac/cscwdir.html |
| CSCW Yellow Pages | http://www.tft.tele.no/cscw/ |
| CSCW Yellow Pages and bibliography | http://www11.informatik.tu-muenchen.de/cscw/ |
| FAQ | http://www.cis.ohio-state.edu/hypertext/faq/usenet/comp-groupware-faq/top.html |
| Groupware Yellow Pages | http://www.consensus.com/groupware/ |
| Newsgroup on groupware | news:comp.groupware |
| Tutorial on workflow | http://cne.gmu.edu/modules/workflow/ |
| Workflow on the Web | http://master.www.cityscape.co.uk:1081/users/ef48/ |
| WWW collaboration projects | http://union.ncsa.uiuc.edu/HyperNews/get/www/collaboration.html |

### Workflow

Workflow, a hot topic in computer-supported cooperative work (CSCW), refers to group activity automation by task sequencing and information routing. The workflow concept, however, originated about 15 years ago and was investigated in the context of office automation from the late 1970s through early 1980s.[6] Since the mid-1980s, workflow has been examined in a process programming framework within software engineering.[7]

The aim of CSCW is to develop groupware—that is, software and/or hardware for collaborative work among a group of people. Groupware applications to support workflow include, for example, the document-centric Notes, developed by Lotus; the process-centric FlowMark, developed by IBM's Exotica project at its Almaden Research Center; InConcert, developed by XSoft of Xerox to model, coordinate, and automate work process components; and ProcessIT, developed by AT&T GIS (Global Information Systems).

In academia, groupware development includes efforts by Clarence A. Ellis and colleagues at the University of Colorado at Boulder (goal-based workflow modeling and dynamic change in workflow systems); and efforts by Amit Sheth and colleagues at the University of Georgia (workflow management systems). For further information, see the "Computer-Supported Collaborative Work" sidebar.

## Agents

The following are on-line information sources for agents.

| Topic | URL |
|---|---|
| Agent language KQML | http://www.cs.umbc.edu/kqml/ |
| Agents directory | http://www.cl.cam.ac.uk/users/rwab1/ag-pages.html |
| Agents, infobots, and knowbots | http:/union.ncsa.uiuc.edu/HyperNews/get/computing/agents.html |
| Agents research at MIT | http://agents.www.media.mit.edu/groups/agents/ |
| AI resources | http://ai.iit.nrc.ca/ai_point.html |
| DAI | http://dis.cs.umass.edu/ |
| Robots on the Web | http://info.webcrawler.com/mak/projects/robots/robots.html |
| Security | ftp://www.offshore.com.ai/security/ |
| Security | ftp://ftp.csua.berkeley.edu/pub/cypherpunks/Home.html |
| Software agents | http://www.cs.umbc.edu/agents/ |

### Agents

Agents[8] are entities regarded as weak or strong; essentially, agents classified as strong simply have more capabilities than those classified as weak. Our research focused on weak, or software, agents as we discuss later. A software agent can (1) operate autonomously, interacting with other agents—including humans—through communications; (2) reactively adapt to changes in an environment; and (3) proactively exhibit goal-directed behavior. A strong agent can do all this; moreover, it possesses mentalistic notions—such as belief, desire, obligation—and it is capable of learning.

The two issues in designing an agent-based system are

- how to implement an agent (microlevel), and
- how to coordinate multiple agents (macrolevel).

Macro-level issues are the province of distributed artificial intelligence. One problem with AI agents is the lack of a "definitive" theory.[8] So far, there is no general model to describe different types of agents that form a large, complex system. (See the "Agents" sidebar.)

Agent types include *information agents* such as softbots[9] to access Internet resources, *task agents* such as Telescript agents,[8] and *interface agents* such as Maxims[10] to assist users.

Agents are particularly essential to a flexible-working environment. If computer programs for performing tasks are fairly autonomous (that is, each application can carry out its task independently to some extent), then less user interaction is required to operate the computer programs. A collection of agents provides a flexible-working environment in which each agent performs a particular task for a worker. Furthermore, less communication is needed to perform tasks when agent-based systems are compared with the traditional client-server approach, and reducing the amount of communication is a primary concern in distributed multimedia systems to conserve bandwidth and speed response times.

### Prototyping

Graphical models are at the heart of our prototyping process for a flexible-working environment, along with a prototyping tool we developed called AP (Agent Prototyper). AP includes a graphical editor for the models and an interpreter of executable specifications in the models. Formally, all the graphical models are defined in MERA (Meta-Entity-Relation-Attribute), a specification language whose primitive constructs are entities, relations, and attributes. MERA is a generalization of the well-known ER (Entity-Relationship) model. Our examples are sufficiently self-explanatory that details on MERA are unnecessary here.

The prototyping of a flexible-working environment consists of the following steps, where Steps 2 and 3 can be done concurrently.

- *Step 1.* Identify elements and activities in the working environment and describe the workflow activities.
- *Step 2.* Clarify the workflow activities' tasks and model agents for performing the tasks, together with interactions among the agents and system resources.
- *Step 3.* Clarify interactions between the agents

and the users, create sample multimedia data used in the interactions, and model multimedia user interfaces.

- *Step 4*. Simulate the models and evaluate the system.

In Step 1, we use Petri nets[11] to describe workflow in a flexible-working environment. Petri nets are frequently used to model office procedures and process programming because of the nature of concurrency and synchronization in office work and software development. In addition, Petri nets are convenient and powerful when modeling entity communications.

As the basis for the rest of our discussions, the following scenario for a flexible-working environment will serve.

> In a company, two executives (A and B) are involved in an R&D project. Executive A wants to discuss an issue with three people: Executive B, a division chief (C) in charge of the project, and a project leader (D). A asks a secretary (S) to arrange the discussion. B is traveling on business but can receive e-mail with a portable computer. C and D are working at a branch location that is far from the company's headquarters, which is where B's office is located.

In describing specification workflow, it's helpful to first write down a description of typical interactions among people as shown in Figure 1, where the secretary is denoted by S, the executive who requested a conference is denoted by A, and participants are denoted by A, B, C, and D. Then write the workflow based on the text description.

Now, we can develop the workflow shown in Figure 2 (next page). In this Petri-net-based graphical model, rectangular icons depict activities, diamond icons depict branches, and document-like icons depict information. Solid lines denote precedence relationships among the activities, and dashed lines denote information flow among the activities (that is, input/output information of each activity).

Once the workflow has been described, the system designer starts to model agents. At Step 2, the designer first associates agents with tasks performed by the activities. Then, the agents are modeled based on interactions among agents, users, and system resources. Because workflow represents activities to be done, it's natural that the resulting agent descriptions resemble the workflow. In an extreme case where all activities can be automated by agents, the workflow could be regarded as a model of the agents.

Next, we explain the models for describing agents and user interfaces at Steps 2 and 3, respectively. For continuity, we will refer to the conference arrangement scenario in Figure 1.

| From —> To | Message |
|---|---|
| A —> S | "Please schedule a conference." |
| S —> A | "Please give me the following information:"<br>- Purpose of the conference<br>- Names of conference participants<br>- Expected length of the conference<br>- Meeting place<br>- By when the conference must be held |
| A —> S | "Here is the information."<br>- Purpose of the conference: To discuss a prototype of our new product<br>- Names of conference participants: A, B, C, D<br>- Expected length of the conference: Short<br>- Meeting place: Somewhere in Hawaii<br>- By when the conference must be held: As soon as possible |
| S —> A,B,C,D | "Please provide me your availability for the following conference. Also let me know the island where you want to have the conference."<br>- Purpose of the conference: To discuss a prototype of our new product<br>- Names of conference participants: A, B, C, D - Expected length of the conference: One hour<br>- Meeting place: Somewhere in Hawaii<br>- By when the conference must be held: Within a week |
| A —> S | "Any time, except Monday and Friday. I prefer Maui." |
| B —> S | "Any time after June 11 because I am on a trip. Any island is OK with me." |
| C —> S | "Monday afternoon or Wednesday afternoon. Maui is fine." |
| D —> S | "Any time. Anywhere." |
| S —> A,B,C,D | "Please give me your preference over the following choices."<br>(1) Early afternoon of Wednesday<br>(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."<br>- On June 12, Wednesday, 1:00 pm<br>- At Maui branch<br>- To discuss a prototype of our new product<br>- By A, B, C, D |
| A —> S | "Confirmed." |
| B —> S | "Confirmed." |
| C —> S | "Confirmed." |
| D —> S | "Confirmed." |

*Figure 1. Description of typical interactions among people in setting up a conference.*

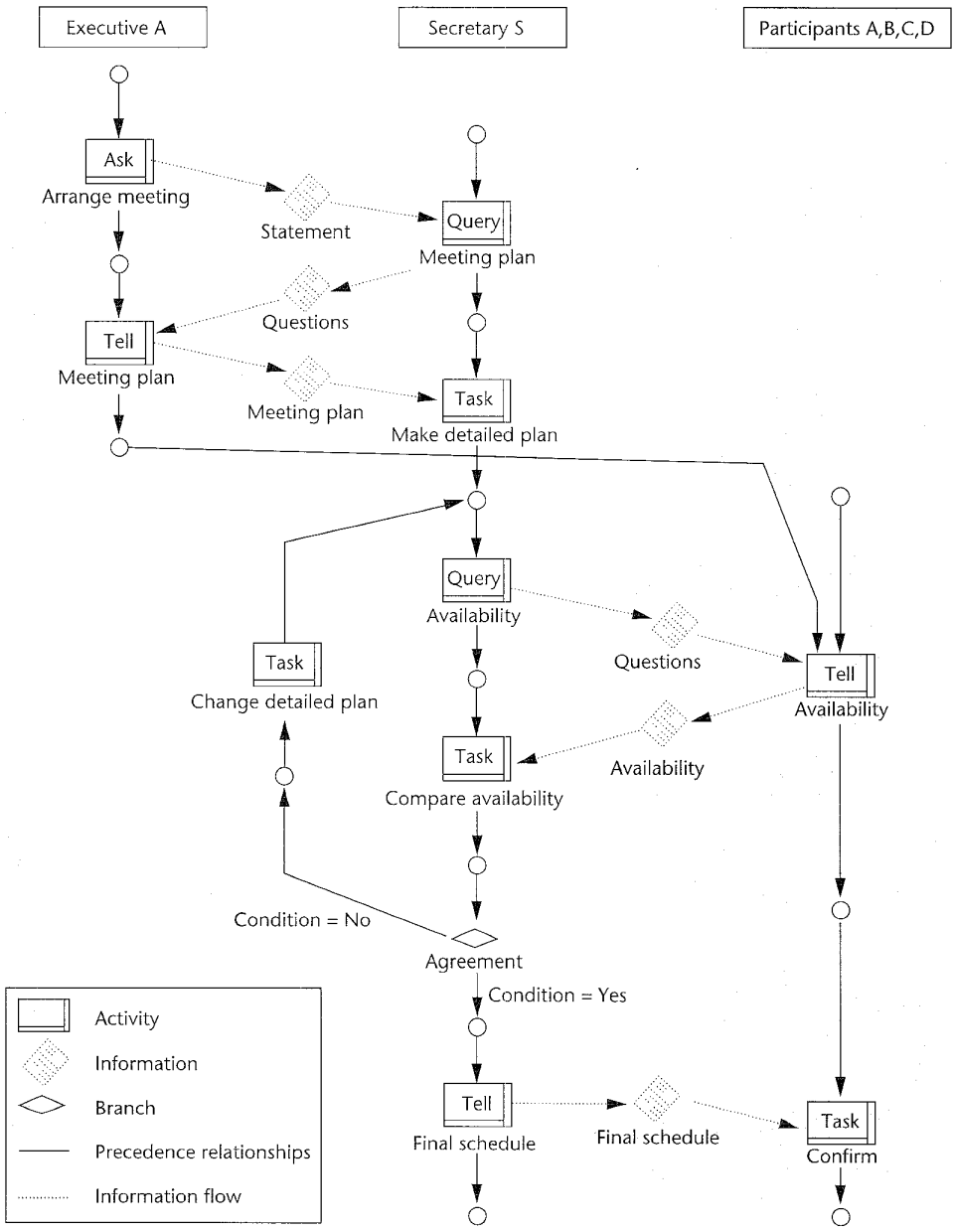



*Figure 2. An example of workflow to arrange a conference.*

## Modeling

Our *agent model* specifies agent behavior and its interactions with other agents and system resources. The top half of Table 1 summarizes the entity types that represent actions to be done by an agent. The precedence relationships among them (that is, control flow) are denoted by solid arrows in the workflow figure. The bottom half summarizes the entity types that represent objects with which an agent interacts. The interaction relationships are denoted by dashed arrows in the workflow figure. An agent can have two types of interactions: *direct* or *indirect* (via another agent). The system designer chooses between the two based on criteria regarding communication, reliability, and security, for example.

Our conference arrangement agent is modeled as shown in Figure 3 (p. 46). This model represents the viewpoint of the secretary's agent, which performs tasks on the secretary's behalf. There are

three User icons: Secretary, Executive, and Participants. The designer specified indirect interactions between the agent and Executive and between the agent and Participants, which is reasonable because synchronous communications among our flexible-working participants is impractical.

The model in Figure 3 represents the following task of an agent for the Secretary.

1. The Secretary's agent first creates another agent, Agent001, to get a meeting plan from Executive A.

2. The created agent (Agent001) is sent to and executed on Executive A's computer.

3. The meeting plan is then sent from Agent001 to the Secretary's agent.

4. Because the plan may not be detailed or definite enough, the task of making a detailed plan is left to the Secretary.

5. The Secretary's agent creates another agent, Agent002, to communicate with, and obtain availability from, each Participant.

6. The created agent (Agent002) is sent to and executed on the Participant's computer.

7. The availability information is then sent from Agent002 to the Secretary's agent.

8. When the Secretary's agent obtains all the Participants' availability, it uses system resources such as application programs to select the best choice of meeting time and location. If the best choice exists, it's sent to the Secretary for approval and then distributed to all the Participants for confirmation. Otherwise, the agent notifies the Secretary that the task failed and asks for help.

Although the agent model in Figure 3 closely resembles the workflow in Figure 2, there are some differences. If the workflow includes an activity that cannot be automated (such as the Secretary's task to make a detail plan), the activity is replaced by a query in the model, which asks the Secretary to give the agent the required information. If the agent independently makes an important decision (for instance, the agent's task to compare availability and pick the best time), a new query is added in the model to have the Secretary double-check and approve the decision.

*Table 1. Entity types of the agent model.*

| Icon | Type | Description |
|---|---|---|
| | Place | Presence of token(s) in the place indicates the end of the previous action or ready for the next action |
| Query | Query | Send a question to receiver(s) and wait for a reply |
| Tell | Tell | Send information to receiver(s) |
| Ask | Ask | Ask receiver(s) to do something |
| Task | Task | Perform a task |
| | Branch | Branch depending on the result of preivous activities |
| | Par-begin | Beginning of parallel executions |
| | Par-end | Synchronization of parallel executions |
| | User | Human user |
| | Multiuser | Multiple users |
| | Agent | Software agent on a computer |
| | Multiagent | Multiple agents |
| | Model | Executable model (such as specifications of an agent in the .agent model) |
| | Information | Information to be exchanged |
| | Resource | System resource |

Next, we explain our *user interface model* for specifying user interfaces. A system's user interface is actually an interrelated collection of separate agent user interfaces. We specify an agent user interface with an outline, screen layout, and action graph, as follows.

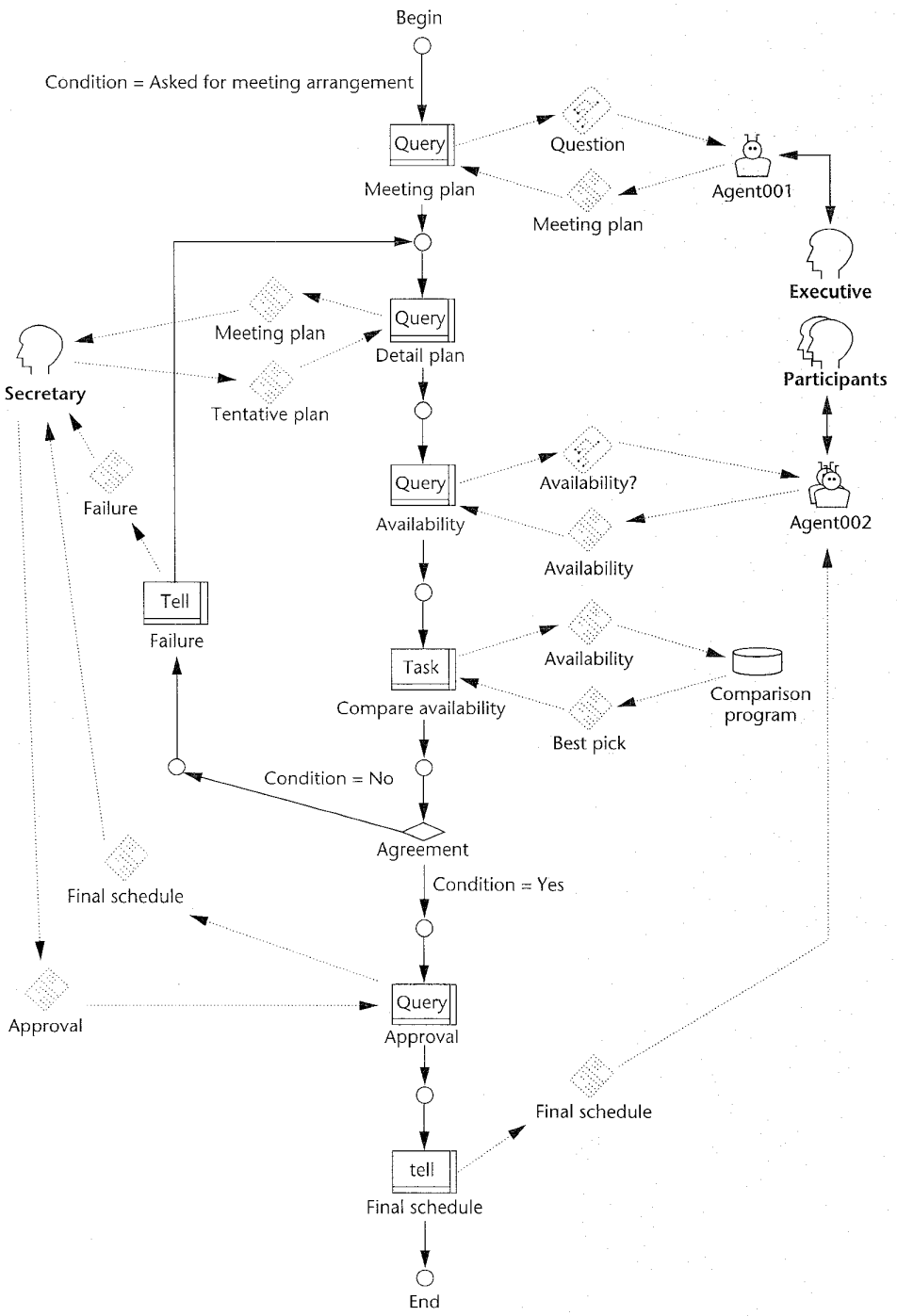
- *Outline.* A state transition diagram outlines user interactions, in effect. Our agent prototyper tool (AP) provides a diagram editor.

- *Screen layout.* For each state in the state transition diagram, the designer creates a screen layout with AP's graphical editor and generates sample multimedia data for exchange between the user and the agent. The screen layout consists of rectangular fields on the screen in their relative positions. Sample field data can be specified by either putting the data in the field or listing the data's file name.

- *Action graphs.* For each state in the state transition diagram, the designer uses AP's diagram editor to graph the actions applied to the screen fields and, through Petri nets, orders the actions for execution.

To illustrate this part of the process, let's consider Agent002 interacting with Participants as was shown in Figure 3. A sequence of four states—Start,

*Figure 3. An agent model for conference arrangement.*




Get attention, Get info, and End—represents Agent002's state transition diagram. *Start* initializes the user interface. *Get attention* gets the user's attention. *Get info* gets information about the user's availability. *End* terminates the user interaction.

Figure 4 shows an action graph only for the Get info state, and Table 2 lists the action types used. Because the action graph is a Petri net, parallel execution and action synchronization can be represented with entities of types Par-begin and Par-end. This is very useful when a multimodal user interface is modeled.

*Table 2. Types of actions used in Agent002.*

| Icon | Type | Description |
|---|---|---|
| | Assign | Assign a given value to a field |
| | Clear | Clear a screen |
| | Device | Select one of the fields on the screen |
| | Display | Display the current value of a field at the location designated to the field |
| | Invoke | Invoke a command or an application program |
| | Play | Play animation |
| | Speak | Read text data aloud |

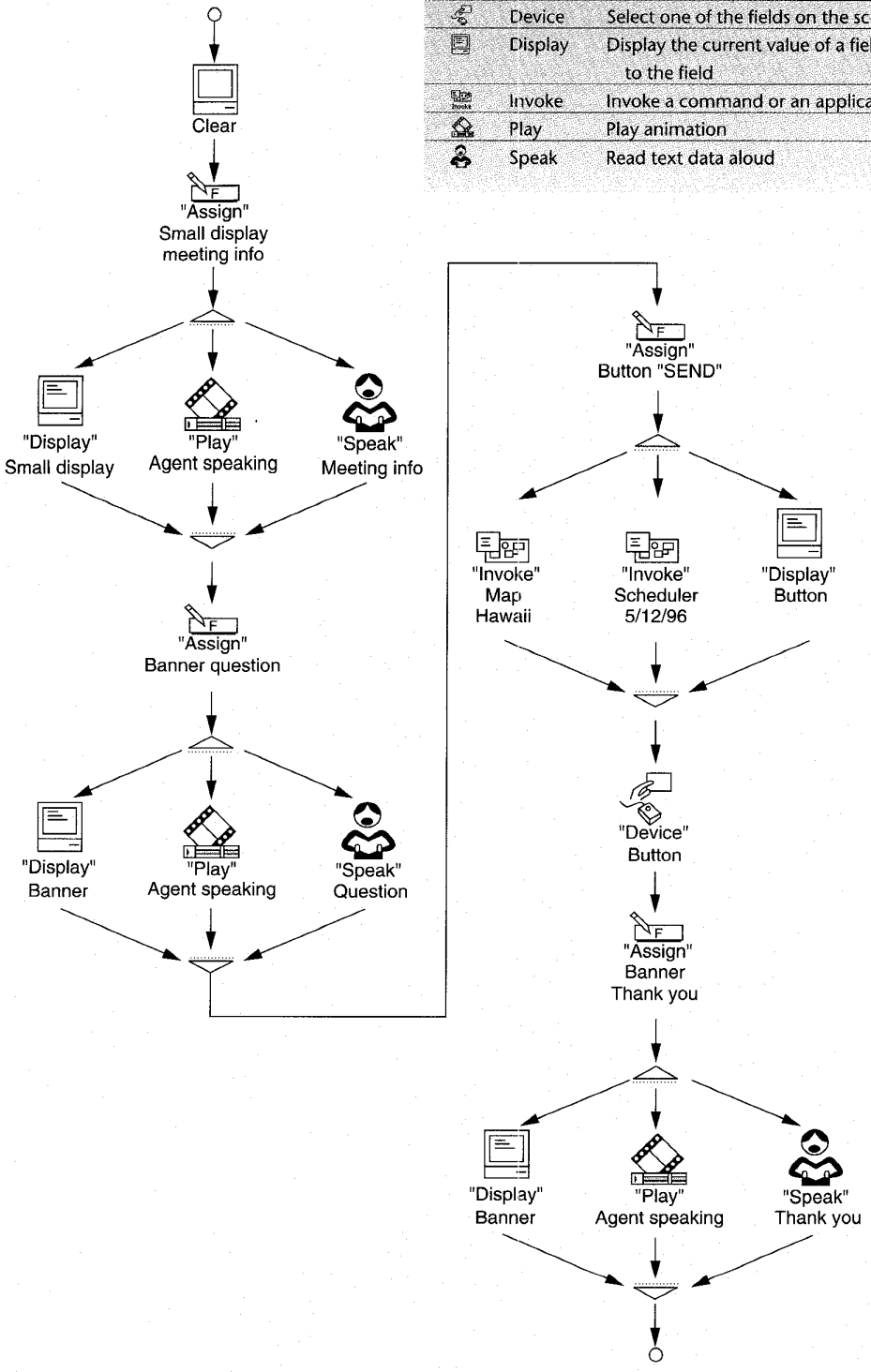



*Figure 4. User interface model for Agent002, showing an action graph for the Get info state of a four-step state diagram: Start, Get attention, Get info, and End.*

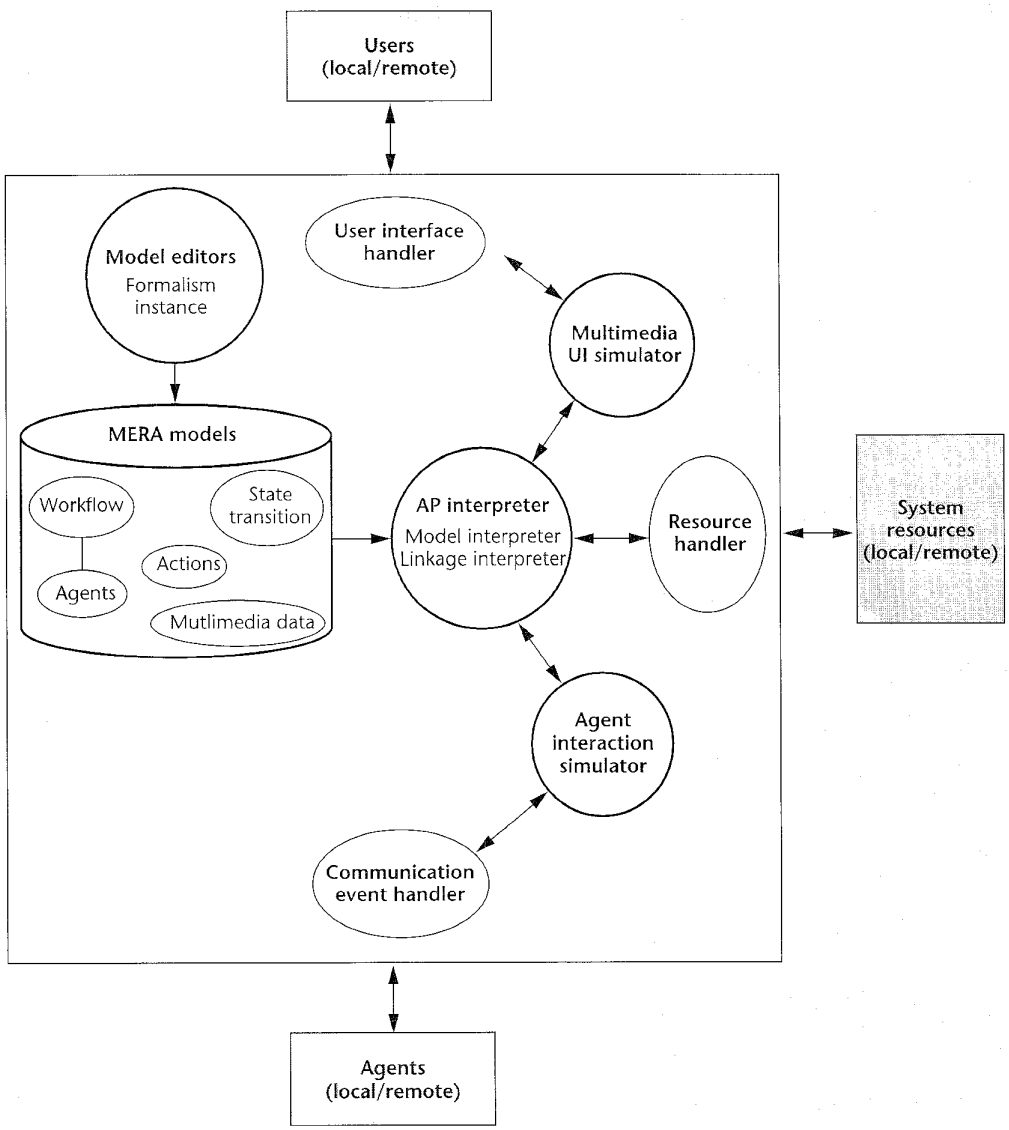



*Figure 5. Configuration of AP.*

### Simulation on AP

When the models are complete, the designer then simulates the system with AP, which can be assumed to reside and run on each machine in a network where simulation is performed. The prototyping tool interprets the system's executable specifications that the designer wrote in the graphical models.

The AP configuration, decomposed into two parts, is shown in Figure 5. One part supports system modeling, and it provides editors for the prototyping models. Another part simulates the modeled system. This part has three handlers that handle low-level events in interactions with users, system resources, and other agents, respectively. The Multimedia User Interface (UI) Simulator and the Agent Interaction Simulator command the UI Event Handler and Communication Event Handler, respectively. The AP Interpreter interprets specifications of the system and commands the other components to control simulation at a high level.

### User interface example

Let's look now at how the agent prototyper would simulate the user interface model for Agent002 in Figure 4. First, AP clears the screen. Second, AP displays Meeting Info data in the Small Display field, then plays animation to visualize an agent speaking while reading the Meeting Info text

data aloud. Third, AP displays Question data in the Banner field and plays the animation of the speaking agent while reading the Question data aloud. Fourth, AP displays the SEND string in the Button field and invokes two application programs named Map and Scheduler with parameters "Hawaii" and "6/12/96," respectively. Then, AP waits for a user to click SEND on the screen. See Figure 6 for a snapshot of the simulation at this moment. Finally, after the user clicks SEND, AP displays Thank-you data in the Banner field and plays the animation for the "speaking" agent while reading aloud the text data. The simulation then ends.

### AP system requirements

The agent prototyper tool is currently implemented in C and runs on the Macintosh platform, including Mac II, Power Macintosh, and PowerBook computers. The following requirements for prototyping distributed multimedia systems apply.

*Multimedia data.* AP handles seven types of multimedia data: image, animation, map, menu, audio, menu, written NL (Natural Language), and spoken NL. It uses QuickTime and MacinTalk to accommodate image, animation, and audio.

*Communications.* AP supports distributed network applications. Apple event-based protocol handles communications between applications. A lower level communication protocol in the network can also be used for communications with a larger variety of applications.

*Distributed agents.* Two or more APs on remote machines communicate with each other when they encounter Query, Tell, and Ask in their interpreted models. This lets us prototype distributed multiagents that communicate with each other.

## Conclusions

Our prototyping results to date, with respect to distributed systems for flexible working, do have some limitations. At the same time, it is clear where we should direct future research.

*Domain knowledge acquisition for agents.* The focus of our AP tool is multimedia user interface and multiagent interaction prototyping. These are the system components for which prototyping is most effective in system design. In developing an agent system, however, acquiring domain knowledge is one of the most time-consuming tasks. Without describing detailed domain knowledge, modeling of both agent behavior and user interface is somewhat limited. Thus, AP should support domain knowledge acquisition.

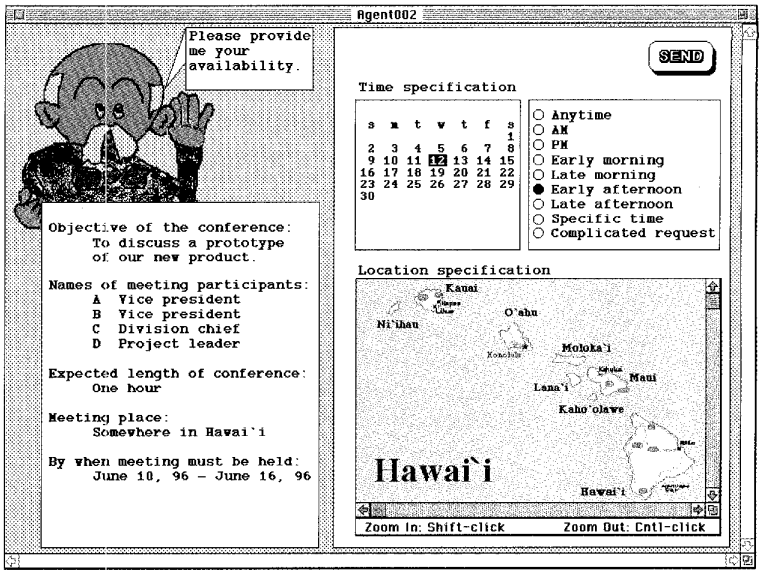



*Figure 6. Simulation of a user interface for Agent002.*

*Agent specifications reuse.* AP doesn't currently support methods for reuse. For example, it should support the stereotype concept and object-oriented methodology.

*Platform.* AP runs on the Mac family, as we mentioned earlier. Its subset also runs on Newton. Although we implemented AP variations that run on different platforms such as Unix/X Window and Smalltalk, they are not compatible. AP should be extended or the variations should be integrated to run on multiple platforms. In addition, we should consider various network configurations. For example, wireless communications such as pagers and portable computers like pen-based Personal Digital Assistants are particularly important for flexible working.

*Various multimedia data for user interfaces.* Currently, AP can handle only text, image, or animation in QuickTime, and audio in MacinTalk. AP should be able to handle a wider variety of I/O data in various formats.

*Addressing the issues.* To address the first two issues, the Agent Languages and AgentWare for Agent Invention (ALAWAI) project has recently started at the Software Engineering Research Laboratory in the University of Hawaii at Manoa.

The project seeks to free the designer from tedious tasks such as describing agent details. While the AP tool supports agents' external specification development (behavior), ALAWAI supports internal specification development (knowledge). The project goes beyond prototyping—it will compile the specifications into stand-alone applications. It has agentware that includes stereotype agents, reusable domain knowledge, and user models, for example.

To develop an agent in ALAWAI, a developer first selects a stereotype agent for a problem domain and adds missing knowledge to it. Inter-agent communications can be specified in a standard agent-language. Then, the specifications can be tested by simulation and finally compiled into executable applications. AP and ALAWAI together will drastically improve the efficiency of three essential tasks of flexible working environment design: workflow modeling, multimedia user interface design, and coding of domain-specific knowledge for agents.

For more information about our research projects and related work, visit our Web site at http://samurai.ics.hawaii.edu/. **MM**

## Acknowledgments

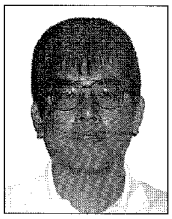
This research has been supported by PFU Inc. in Japan. The authors thank Xiaobo Wang for his help on AP implementation, and anonymous reviewers of this article for their valuable comments and suggestions.

## References

1. S. McClelland, "Problems, Perceptions, Prescriptions and Policy Options in Flexible Working," *Pacific Telecomm. Review*, Vol. 16, Mar. 1994, pp. 4-9.
2. J.A. Adam, "Interactive multimedia: Applications, implications," *IEEE Spectrum*, Vol. 30, No. 3, Mar. 1993, pp. 22-31.
3. B. Furht, "Multimedia systems: An overview," *IEEE MultiMedia*, Vol. 1, No. 1, Spring 1994, pp. 47-59.
4. J. Grudin, "Computer supported cooperative work: History and focus," *Computer*, Vol. 27, No. 5, May 1994, pp. 19-26.
5. S.-K. Chang, Tutorial on Distributed Multimedia Information Systems, *Pacific Workshop on Distributed Multimedia Systems*, Knowledge Systems Inst., Skokie, Ill., 1994.
6. C.A. Ellis and G.J. Nutt, "Office information systems and computer science," *ACM Computing Surveys*, Vol. 12, No. 1, Mar. 1980, pp. 27-60.
7. B. Curtis, M.I. Kellner, and J. Over, "Process modeling," *Comm. ACM*, Vol. 35, No. 9, Sept. 1992, pp. 75-90.
8. M.J. Wooldridge and N.J. Jennings, "Agent theories, architectures, and languages: A survey," *Lecture Notes in Artificial Intelligence*, No. 890, M.J. Wooldridge and N.R. Jennings, eds., Springer-Verlag, Berlin, 1995, pp. 1-39.
9. S.R. Hedberg, "Intelligent agents: The first harvest of softbots looks promising," *IEEE Expert*, Vol. 10, No. 4, Aug. 1995, pp. 6-9.
10. P. Maes, "Agents that reduce work and information overload," *Comm. ACM*, Vol. 37, No. 7, July 1994, pp. 31-40.
11. T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, Vol. 77, No. 4, Apr. 1989, pp. 541-580.

**Koji Takeda** is an assistant director of the Software Engineering Research Laboratory in the Department of Information and Computer Sciences at the University of Hawaii at Manoa. His research interests include visual languages, intelligent agents, and software engineering. Takeda has an MS degree in computer science and is pursuing a PhD degree in communication and information sciences at the University of Hawaii.

**Mitsuyuki Inaba** is a research assistant in the Department of Information and Computer Sciences at the University of Hawaii at Manoa. His research interests include cognitive sciences, natural language processing, and intelligent agents. Inaba received a BA degree in philosophy from Ritsumeikan University in Kyoto, Japan, in 1988. He is a member of the IEEE, ACM, JSAI, and JCSS.



**Kazuo Sugihara** is an associate professor in the Department of Information and Computer Sciences at the University of Hawaii at Manoa. He received a BS degree in electrical engineering (1980), an MS degree (1982) and a PhD degree (1985) in systems engineering from Hiroshima University in Hiroshima, Japan. Sugihara is a member of the IEEE and the IEICE in Japan.

Contact Sugihara at the Department of Information and Computer Sciences, University of Hawaii at Manoa, Honolulu, HI 96822, e-mail sugihara@nile.ics.hawaii.edu.