Agent Model on Information Kiosk Systems Supporting Mobile Users

Yasuhiko Miyazaki Kenji Fujimoto Kazuhiro Sugiyama

Nippon Telegraph and Telephone Corp. (NTT)
Cyber Communications Laboratory Group
1-1 Hikarinooka
Yokosuka 239-0847 Japan
{miyazaki, fujimoto, sugiyama}@marsh.hil.ntt.co.jp

Abstract

We propose PET (Personal Elegant assistanT) agent model, an agent model suitable for information kiosk systems supporting mobile users, which provides cyberspace services in user-adaptive manner. PET agent model consists of four types of agents; Interface Agent (IA) which supports user interactions on the terminals, User Agent (UA) which manages user's profiles, Service Agent (SA) which generalizes network services, and Application Agent (AA) which controls application procedures. Since interaction functionality between users and the system is separately modeled as IA and UA, their combination efficiently enables user-adaptive information provision for multiple mobile users accessing multiple public kiosk terminals. AA makes it possible for application developers to implement practical applications quickly by controlling a procedural scenario to ask for many cyberspace services, which are generalized as SAs. Based on this idea, we implemented a prototype agent systems using Java to evaluate how our agent model is applicable.

Introduction

Today, cyberspace becomes rich information sources; we can access various kind of information residing at cyberspace while sitting in front of a computer. However, it is still difficult to get information from the network while walking around specific areas, where we want to get various information, such as theme parks, exhibitions, air port terminals and large office buildings. The difficulty comes from the facts;

- There are much more limitations for mobile users to operate devices than for static users.
- The users' circumstances and purposes to get information are dynamically changing while they are moving.

In other words, the followings are the requirements from the viewpoint of the users.

(1) The system should be easily accessible to mobile users without complicated operations.
(2) The system should adapt personal circumstances of mobile users.

From the viewpoint of the providers, they need to implement the systems satisfying (1) and (2). Therefore, we should consider the followings.

(3) The system should have some software models which outline how to implement the platform.
(4) The system should be easy to implement the applications on the platform.

We researched the architecture and the model of information systems for mobile users that is applicable to (1)-(4). We first investigated the system architecture and found that it was a good choice for satisfying (1) to use information kiosk terminals with user detection capability as access devices. We also considered that it was a good candidate solving (2) to integrate generic agent model as a fundamental software architecture. Therefore, we focused on the problem how to model generic agents on kiosk systems to satisfy (3) and (4).

In this paper, we state the system architecture we assumed, point out problems to construct information systems on the architecture, and describe our proposed kiosk agent model to solve the problems. We also mention the prototype systems to evaluate the model.

System Architecture

There were mainly two architectures to provide network information to mobile users. One was using personal digital assistance (PDA) e.g. (Nishibe et al 1998) and the other was using information kiosk. However, we do not consider that either architecture is efficient for information systems supporting mobile users in theme parks etc. In this section, we state the requirements, and, from these viewpoints, evaluate which architecture we think is suitable to build such systems. Then, we mention software architectures.

Requirements for System Architecture

We should consider the following functionality when we build an information system for mobile users comparing to that for static users.

- Operability
- Portability
- Personal Awareness

Operability and Portability correspond to (1) and Personal Awareness corresponds to (2) in the previous section.

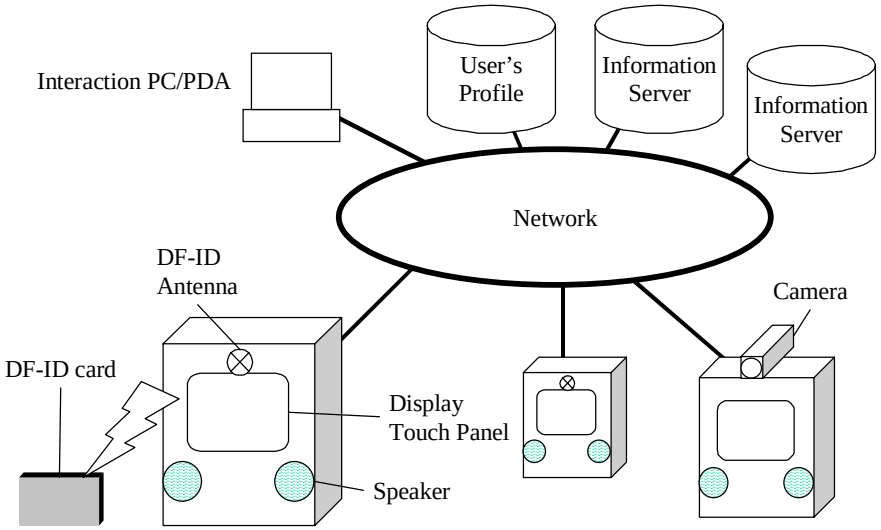

Figure 1 System Architecture

Architecture / Evaluation	**PDA**		**Kiosk terminal**	***Kiosk terminal with user detection function***
	palmtop size	wrist watch size		
Operability	OK	Hard to operate. Hard to see pictures and maps.	OK	*OK*
Portability	Annoying to hold it anytime anywhere.	OK	OK	*OK*
Personal Awareness	OK, Configurable for the user.	OK, Configurable for the user.	Public use.	*OK, Possible to integrate with personal profile.*

Table 1 Comparisons of system architectures

Operability means how easily users can access to the network information without complex user operations. Operations available outside are usually very restricted. They cannot use sensitive pointer devices or keyboards, for example.

Portability means how easily users can carry equipment necessary to access information. Mobile users, especially walking users, do not want to bring disturbing devices only for information systems.

Personal awareness means how much the system can consider each user's needs. The destinations and purposes of mobile users are usually different from those of others. In addition, appropriate interactions are depending on the environments. For example, audio output should be louder in noisy place while it should be avoided in quit library. Furthermore, as the users move, such needs and environments are dynamically changing. The architecture has to be aware of each user's circumstances.

Comparison of Hardware Architectures

Table 1 shows comparisons for information system architectures. PDA cannot solve the trade-off as for operability and portability. Existing kiosk terminals ca nnot support personal awareness, either.

The solution we thought of is using information kiosk terminals with user identification function, which can detect and identify the person in front of them (see the right most column in *Table* 1). It is feasible by using

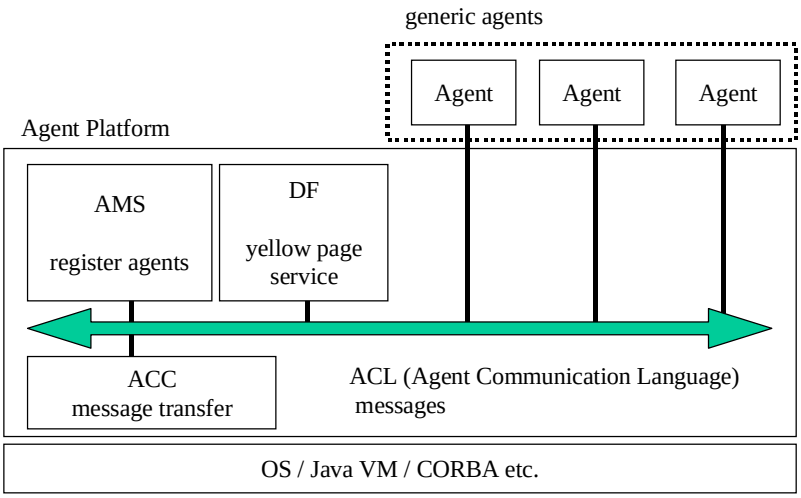

Figure 2 FIPA Agent Reference Model

recent face recognition technology, for example. More practical solution is using RF-ID (Radio Frequency Identification) technology. Users bring a tiny RF-ID tag in their pockets, which terminals detect and identify the user.

Hence, we adopted the system architecture shown in *Figure 1*. The system consists of information servers, user profile servers and interaction terminals which can identify users.

Generic Agent Architecture

We considered an agent model as a software architecture which enables the personal awareness.

Recently, a lot of agent researches have been going on as Bradshaw reviewed in his book (Bradshaw 1997). Gilbert (Gilbert et al 1995) described agents by three dimensions, agency (= degree of autonomy), intelligence, and mobility. Three primitive attributes of agents in (Nwana 1996) are autonomy, cooperation, and learning. Shoham (Shoham 1997) stated that many researchers defined "agent" as an entity that functions continuously and autonomously. We paid attention to this autonomy to build user adaptive information systems for mobile users, for autonomous agents carry out their actions in response to users' environments. Agents would adjust audio volume in response to the environment, for example.

Some work forces to standardize agent technologies have been emerged, such as FIPA and OMG. *Figure 2* shows the FIPA agent reference model described in (FIPA 1997), which was widely discussed and accepted. It defined that an agent is the fundamental actor that has service capabilities and interacts with each other by communicating with conceptual message written in Agent Communication Language (ACL) through Agent Communication Channel router (ACC). An agent resides in an Agent Platform (AP) by registering into Agent Management System (AMS). Directory Facilitator (DF) provides yellow page service; it tells which agent has what services. We found that this model was applicable to our target systems.

Problem Description

In order to build real kiosk information systems, we have to implement personal-aware software systems on an agent model. This implies that we should define a particular agent model based on the generic agent architecture like FIPA model. In this section, we discuss what is necessary for the agent model on kiosk information systems.

Our model must satisfy (3) and (4) described before. As for (3), the specific problem of kiosk information systems is the user interactions (i.e. functionality for communication between the user and the system such as input and output) for mobile users in application-independent. Therefore, we focused on the following two problems to define kiosk agent model, which have not been well addressed before.

- [Problem 1] How to support user-adaptive interactions on multiple and shared terminals, which is derived from (3).
- [Problem 2] How easily to define practical applications, which corresponds to (4).

We discuss the detailed issues of the problems below.

Personalization of Public Kiosk Terminals

[Problem 1]

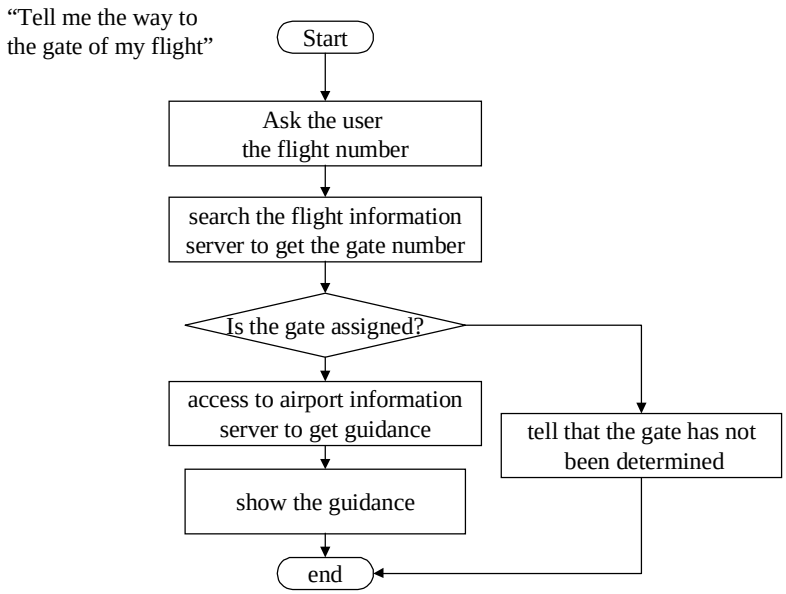

Figure 3 Procedural User Request

As is stated in the previous section, we concluded that in a viewpoint of user operations, kiosk terminals are better than personal devices for walking users. However, we should consider how to realize personal awareness by combination of user identification functionality of the kiosk terminals and user preference servers on the network. It is not trivial because the relationship between mobile users and public kiosk terminals is *m*-to-*n*; a user accesses many terminals and a terminal is used by many users. We have to take the following issues into consideration.

- [P1-1] Since a kiosk terminal is in public use, the appearances and operations should be configured dynamically according to the user's preferences.
- [P1-2] As the devices of the terminals are not exactly same each other, the system should encapsulate the deference of their capability.
- [P1-3] A terminal may simultaneously find several users in front of it but can interact with only one of them due to device limitations. A mechanism would be useful to let the other users know who has a message from the system.
- [P1-4] Several information sources may try to access the users via public terminals at the same time. On the other hand, a terminal needs to serialize them if it presents them by synthesized voice, for example. Therefore, it is important to control conversation threads to the users. Otherwise, the users are confused to whom and in which context the system tells the information.

Application Development of Agent-based Systems

[Problem 2]

We found that many user requests for information systems were procedural. In the airport, for example, the request "Tell me the way to the gate of my flight." can be defined as a flow chart shown in *Figure 3*. Therefore, applications of the kiosk information systems are often described as scenarios. On the other hand, most multi-agent architectures discussed in AI (Artificial Intelligence) area have each agent to make its own planning to achieve a given goal or sub-goal. This approach makes it difficult to foresee the total scenario done by many involved agents. We believe that directly defining application scenarios is as important as multi-agent planning. Hence, the following points are necessary for practical agent applications to emerge.

- [P2-1] Agent architecture should allow application developers to define total scenarios.
- [P2-2] The architecture should still have flexibility to varieties of environments with agents' autonomy.

Proposal of Kiosk Agent Model

Our first approach toward Problem 1 was modeling the terminal and the users' profile as separate types of agents called IA and UA. Then, we can realize the user-adaptive information systems by the combination of a public kiosk terminal (IA) and a profile (UA) of the user who is identified by the terminal. We thought that sub-problems [P1-1]-[P1-4] could be solved by defining mandatory functions that each type of agent should provide.

Similarly, our approach toward Problem 2 was modeling total application scenario controller as an agent called AA. Also we modeled various service providing agents called

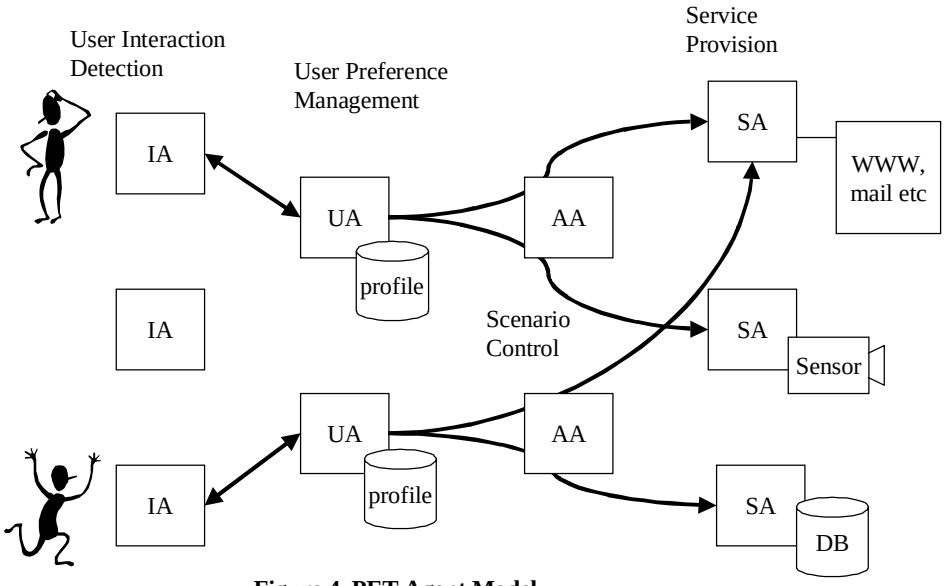

Figure 4 PET Agent Model

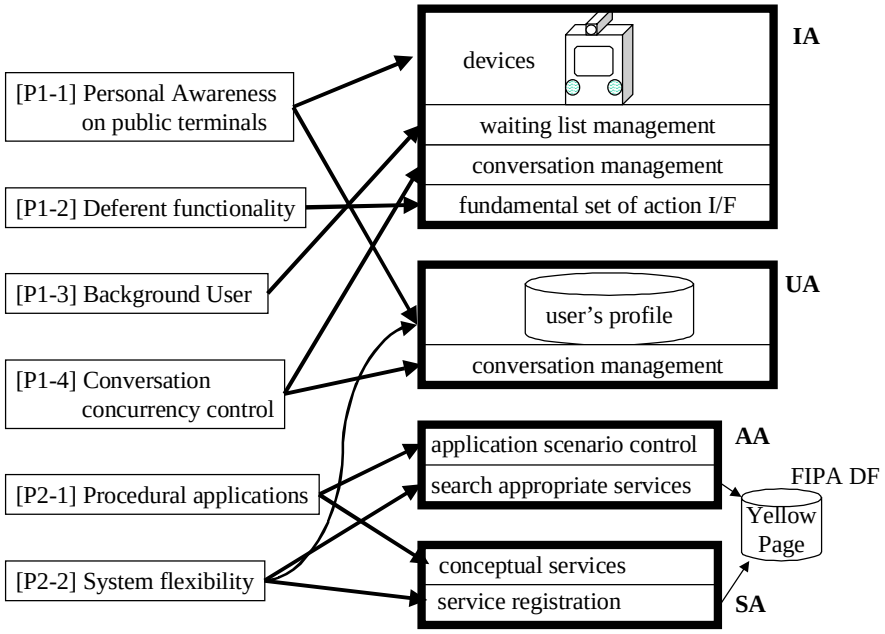

Figure 5 Approaches based on PET Model

SA. We thought that defining AA was a solution to [P2-1] while SA kept flexibility pointed out in [P2-2].

From these considerations, we proposed an agent model called PET (Personal Elegant assistanT) model, which consists of four types of agents. (*Figure 4*)

- Interface Agent (IA)
- User Agent (UA)
- Service Agent (SA)
- Application Agent (AA)

Figure 5 illustrates our approaches toward the above problems with the PET model. We consider that this approach is the most appropriate because it is simple and enough to give solutions to the problems.

In the following subsections, we describe detailed functionality which each agent type should support in order to solve them.

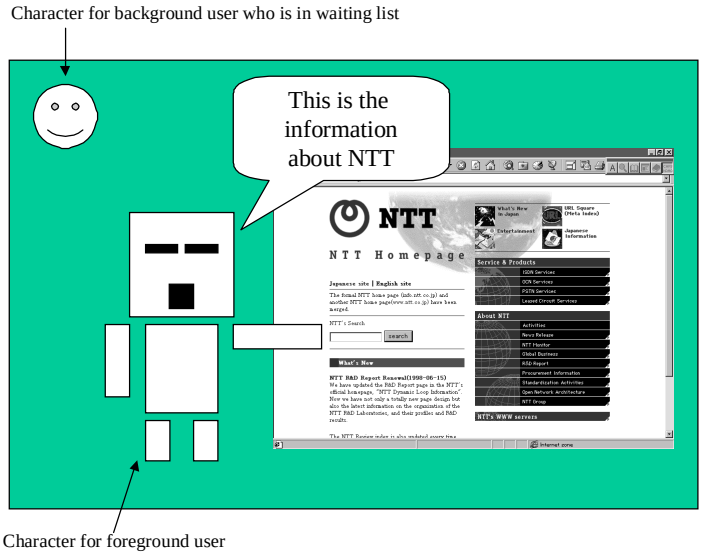

Figure 6 Foreground and Background Characters

Action	Description
conversation-setup	begin the conversation.
conversation-terminate	complete the conversation.
conversation-control	put the conversation in waiting queue since the IA cannot directly interact the user currently.
present	output the information to the user.
listen	input the information from the user.
select	ask the user a selective questions and get the answer.

Table 2 Actions of IA

Interface Agent (IA)

IA is an agent of a terminal which deals with user interactions directly.

IA should have interaction and detection capabilities corresponding to kiosk terminals we assumed. The interaction can be done according to devices installed in the terminal, for example, by GUI, speech synthesis and recognition, animation display, and/or gesture recognition. The detection, by which IA can detect and identify the user, can be realized by RF-ID, face recognition, voice recognition, and/or user's login operation. This functionality may allow users to access information by only approaching the terminal.

Followings are detailed actions which IA should support to solve [P1-1]-[P1-4].

[P1-1] IA gets preferences of the interacting user from his/her UA and executes configuration. The configuration is done autonomously according to the modality the IA supports. For example, IA which interacts by animation gets the character the user likes; IA which is able to present multilingual information gets the user's native language. The user's favorite configuration is useful especially for animation display. Since the same character appears wherever the user approaches, it seems that the personal secretary accompanies with him/her.

[P1-2] To encapsulate IA's functionality, we defined fundamental actions shown in *Table 2* which every IA should support. Each IA performs these actions

autonomously depending on its capability. Similar actions were also approved in [FIPA 1998] Part 12, Human-Agent Interaction specification.

[P1-3] Although IA finds several users to whom the system wants to tell information, the IA may not interact with all of them simultaneously because its audio device is exclusive or display area is not wide enough to open multiple windows. Even if the IA does not fully interact with multiple users, however, it may inform those who are not on the interaction (called background users) that they have messages from the system by showing their small characters (*Figure 6*), for example. To model this functionality, we defined *conversation-control* action to control user interactions in the waiting list. When the foreground user leaves, one of the background users in the list is chosen and promoted to the foreground user, then the IA starts full interactions with him/her.

[P1-4] IA needs to control interactions for multiple users by switching the conversations in case it is requested by multiple UAs at the same time. *Conversation-setup* and *conversation-terminate* actions, with *conversation-id* parameter in each action message, are used for concurrency control of user interactions. Although control policy is determined autonomously by each IA, we believe that IA should switch the conversation as little as possible to avoid users' confusions, which is different from the DBMS policy where high concurrency is desirable.

User Agent (UA)

UA behaves on behalf of the user, by being always on the network to interact with other agents. UA plays an important role for [P1-1] and [P1-4].

[P1-1] UA manages user's preference data for IA to personalize the interactions like animation character data and his/her native language. In addition, UA has preference data such as his/her name, address, email, hobby, interest etc. to personalize information provision in two ways. First, as UA gives the user's preference data to provider agents on his/her behalf[1], they can provide more appropriate information. Second, as messages to the user are sent via UA, it can filter or prioritize them according to the user's preference. In case of transportation guidance application, for example, UA sorts several candidates by fees if the user cares the cost, or it filters out the ways to walk more than the given threshold if the user is unwilling to walk long distance. Therefore, the system can provide user-adaptive information; the user can access what he/she wants in the way he/she likes.

[P1-4] On behalf of a user, UA receives messages from several agents having information for him/her while the user is out of terminals as well as he/she is interacting. Therefore, UA has to manage the messages to keep the conversation threads. Like the messages to IA, *Conversation-id* to UA is also used to indicate the thread. Although UA could merge several conversation threads concurrently, we believe that, when a certain IA detects the user and notifies the UA, it should send the interaction messages to the IA sequentially to avoid his/her confusion.

[1] The user's authorization mechanism should be involved for real systems.

Service Agent (SA)

SA provides services to other agents in the system by the form of agent messages. Typical examples are the Internet services (e.g. WWW and email), personal schedule service, databases (e.g. flight reservation, employee and map), and environmental sensors (e.g. to see traffic congestion, length of queues or door lock status).

In the FIPA agent reference model, SA can be defined as a generic agent providing service. In order to solve [P2-1] and [P2-2], the followings are points.

- [P2-1] SA provides services in the conceptual level as Agent Communication Language (ACL) messages.
- [P2-2] SA registers its service contents to Directory Facilitator (DF) so that other agents can dynamically find appropriate SA by asking DF.

Application Agent (AA)

AA controls an service scenario by combining several SAs providing simple services to satisfy user's relatively complex request.

[P2-1] Upon user's request or another event trigger, AA is invoked. It controls the application scenario by requesting appropriate SAs to get services or information and by interacting with the user through his/her UA. AA is a quite procedural agent; the framework of its behavior is predetermined. We believe that the procedural approach to applications of information systems is necessary because users expect the system to perform the procedure they expect. From practical point of view, it is easier to implement a procedural application as a single agent, AA than to make each agent have a scenario of the procedure separately.

[P2-2] Although AA has a predetermined scenario, it can still keep autonomy and flexibility as an agent; AA dynamically chooses appropriate instances of SA by searching DF so that it can avoid congested or stopped SA. Also, AA can provide user-adaptive information by interacting with the user through the mechanism of UA and IA described above.

Prototype Systems

We implemented an agent platform for PET model using Java. Then, we realized several applications to evaluate the model.

PET Agent Platform

We defined most agents as Java *Object* which has its own *Thread*. Basic *Agent* class has common methods such as receiving and sending messages, registering to the platform,

IA / Action	Kiosk terminal based on browser	Kiosk terminal based on animation	Telephone
conversation-setup	open a conversation page	animation character (avatar) shows up	make a phone call
present	display on the browser	avatar speaks and displays in a balloon	speak
select	display radio buttons using HTML forms and accept inputs	avatar speaks prompts and shows buttons on the touch panel screen and/or set up voice recognition engine with the candidates	speak prompts and receive touch tone
listen	display input field using HTML forms and accept input	record voice and/or accept keyboard input	record voice and/or receive facsimile
conversation-terminate	back to home page	avatar disappears	hang up

Table 3 Sample implementations of IA actions

and monitoring. IA, UA, SA and AA classes are defined by extending the *Agent* class. Actual services and application scenarios are hierarchically defined in the subclasses of SA and AA. So are the functions of IA and UA.

Some other agents are implemented by C++ since they need to link C++ libraries such as voice recognition engine and CTI (Computer Telephony Integration) engine.

Messages are based on XML (W3C)and sent via TCP/IP socket to agents on the remote hosts and directly via message queue to those on the local host.

Implementation of Agents

Here, we state some ideas to realize functions of each type of PET agent described in previous section.

IA
We implemented three types of IAs. Browser-based IA is for low performance terminals, Animation-based IA is for high performance terminals, and Telephone IA is for calling mobile phones. The executions for the requested actions are shown in *Table 3*.

UA
UA needs to have a conversation queue to keep messages from multiple AAs and send them to an IA sequentially in order to avoid too much conversation switching for the user. After receiving the reply from IA and forwarding it to the requesting AA, UA should not process another AA's request immediately. Therefore, we have the UA wait for a few seconds to see if the first AA sends the next request continuously.

AA
For rapid application development, scripting is very useful. We implemented Interpreter AA which interprets XML-based scripts. Since the scripts can directly embed XML-based agent messages, the Interpreter AA enables us to program the contents of the agent messages and control their sequences easily.

Museum Guide Application System

We implemented a prototype application of PET agent system for museum guide.

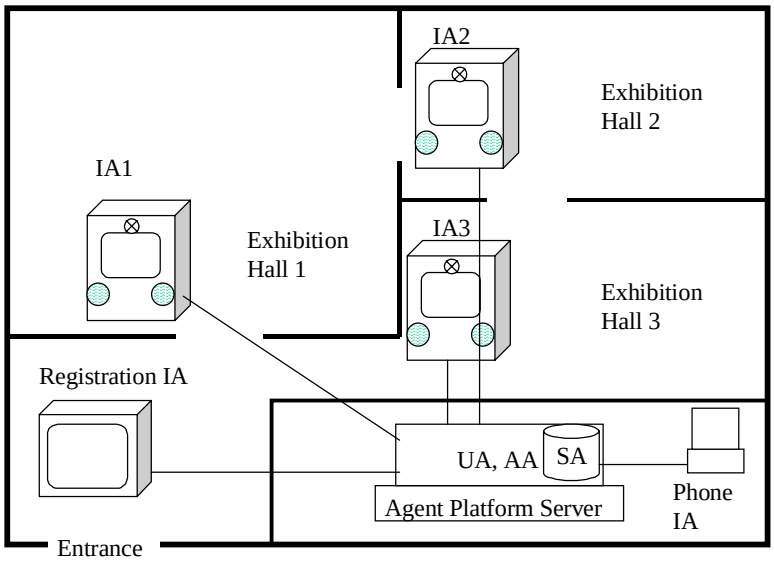

Figure 7 Museum Guide Application Architecture

We assume that guests of the art museum are provided RF-ID tag at the entrance, then, if it is the first time, they register their names, mobile phone number, favorite animation characters (avatars) and their interested events (e.g. special movie, demonstration, tour, etc.). There are kiosk terminals with RF-ID antenna at the corner of exhibition rooms. Each kiosk terminal has IA which is connected to the PET agent platform server where UAs, AAs and SAs are running. We install Telephone IA which can call the user's mobile phone to inform him/her of emergency messages. (See *Figure 7*)

Application Scenario 1 - Guidance

We implemented the art guidance application:
"When the user, say Mary, enters Exhibition Hall 1, her avatar automatically shows up and explains pictures and/or sculptures in the hall according to her interests."
This scenario is executed by the following agents' interactions.

1. IA 1 detects Mary by RF-ID antenna.
2. IA 1 sends *user-login* to her UA.
3. UA starts Interpreter AA with the script for Hall 1 as a default.
4. AA requests UA to start conversation with her.
5. Since UA knows that the user is coming in front of IA 1, UA sends *conversation-setup* to IA 1.
6. IA 1 has her avatar show up.
7. AA gets her interests from UA
8. According to the interests, AA gathers necessary information from SAs and makes a guidance scenario.
9. AA requests UA to play it
10. UA sends *present* (and *select* if necessary) to IA 1.
11. The avatar on IA 1 explains the work of art.
12. UA sends *select* to ask if she wants other presentations.
13. If she chooses another presentation, UA starts the corresponding Interpreter AA, then repeat 7-12.
14. If she wants no more, UA sends *conversation-terminate*.
15. IA 1 has the avatar disappear.

Application Scenario 2 - Messaging

In addition to the guidance application, we implemented the messaging application:
"When the other user, say Tom, needs to contact Mary in the museum, he sends a message to her via PET agent system, which tells her by the appropriate way."
Here, "appropriate way" means that

- if Mary is at a terminal, it interrupts the guidance and tells the message. (case 1)
- if Mary is not at a terminal, the system holds the message and tells it when any IA finds her. (case 2)
- if Mary has a mobile phone and the message is important, the system calls her and tells it. (case 3)

The agents' interactions in this scenario is as follows.

1. Like 1-3 in scenario 1, Tom connects his UA through IA and starts the messaging AA.
2. AA get UA's list which it can contact with.
3. AA asks Tom's UA to whom (which UA) to send the message.
4. His UA sends *select* to IA to choose to whom he wants to tell.
5. AA gets the reply, Mary.
6. AA asks his UA the message to Mary.
7. His UA sends *listen* to IA to input the message.
8. AA sends the message to Mary's UA.

9. Since her UA knows the situation (case 1-3[1]), it sends *present* to
   (in case 1) currently interacting IA
   (in case 2) some IA which sends *user-login* to her UA when it finds her.
   (in case 3) telephone IA
10. Similarly, AA can get a reply from Mary by sending *listen* via Mary's UA and tell it to Tom by sending *present* via Tom's UA.

Discussions

By implementing not only applications to retrieve information but also those to assist user communications, we confirmed that our PET model is well applicable to the museum guide system, which is our typical target, from the following considerations.

For the application developers, they do not need to program how to interact with mobile users using various terminals since UA autonomously does. In Scenario 2, the messaging AA does not care the cases 1-3. They have only to program what to interact in the form of XML. This feature is very useful in the real museum guidance because they have to make explanations for a lot of exhibitions.

For the users, the animation and detection functions of high performance IA are impressive since they can access not only guidance but also messages from other persons with a few voice operations. However, other types of IAs are also important because they are not to restricted in moving around.

Conclusion

We researched the user-adaptive information systems for mobile users in theme parks etc. where they want to access cyberspace services. Mobile users expect the system to be aware of each user's circumstances and easy to operate without carrying any devices that disturb to walk around. Considering these user requirements, we adopted the system architecture which consists of information kiosk terminals with user detection function and generic agent model.

We had to construct an agent model for kiosk information systems to support mobile users' interactions and to enable procedural application development easily. Our proposed model, PET model consists of four types of agents;

- IA, which supports efficient operability for the mobile users by encapsulating functionality of kiosk terminals.
- UA, which manages the user's profile to personalize the system.
- SA, which provides services by conceptual agent messages.
- AA, which controls the scenario to enable practical applications.

The combination of IA and UA enables efficient mobile user's interactions on publicly shared kiosk terminals, which are in *m*-to-*n* relationship. AA allows application developers to define procedural application scenario as a single agent which utilizes generalized cyberspace services, SAs.

By implementing a Java based prototype system, we found that the model was efficiently applicable to the user-adaptive information systems for mobile users.

As future woks, we plan to implement more IA's using various interaction methods, integrate with more intelligent technologies, realize privacy management, and evaluate more aspects of agent systems.

[1] In our current implementation, UA determines the importance of the message only by <priority> tag in XML. If we make more intelligent UA, it can determine totally by the sender, the contents, the user's location, etc.

References

Bradshaw, J. 1997. "An Introduction to Software Agents", *Software Agents*, AAAI Press

FIPA (Foundation for Intelligent Physical Agents) 1997. "FIPA 97 Specification" Part 1-3, http://www.cselt.it/fipa/spec/fipa97/fipa97.htm

FIPA (Foundation for Intelligent Physical Agents) 1998. "FIPA 98 Specification", http://www.fipa.org/spec/fipa98.html,

Gilbert, D.; Aparicio, M.; Atlinson, B.; Brady, S.; Ciccarino, J.; Grosof, B.; O'Connor, P.; Osisek, D.; Pritko, S.; Spagna, R.; and Wilson, L. 1995. "IBM Intelligent Agent Strategy", IBM Corp

Nishibe, Y.; Morihira, I.; Hattori, F.; Nishimura, T.; Yamaki, H.; Ishida, T.; Maeda, H.; and Nishida, T. 1998. "Mobile Digital Assistants for International Conferences", *Community Computing*, pp. 245-284, ed. by Toru Ishida, Wiley

Nwana, H. S. 1996. "Software Agent: An Overview", *Knowledge Engineering Review*, 11(3) pp.205-244

OMG (Object Management Group) http://www.omg.org

Shoham, Y. 1997. "An Overview of Agent-Oriened Programming", *Software Agents*, pp.271 ed. by Jeffrey Bradshaw AAAI Press

W3C (World Wide Web Consortium) "Extensible Markup Language ", http://www.w3.org/XML