Journal of Universal Computer Science, vol. 16, no. 15 (2010), 1929-1952
submitted: 31/1/10, accepted: 28/7/10, appeared: 1/8/10

Mobile Agent-based Context-aware Services

Ichiro Satoh
(National Institute of Informatics, Tokyo, Japan
ichiro@nii.ac.jp)

Abstract: This paper presents an agent-based system for building and operating agent-based context-aware services in public spaces, including museums. The system provides users with agents and detects the locations of users and deploys location-aware user-assistant agents at computers near the their current locations by using active RFID-tags. When a visitor moves between exhibits in a museum, this dynamically deploys his/her agent at the computers close to the exhibits by using mobile agent technology. It annotates the exhibits in his/her personalized form and navigate him/her user to the next exhibits along his/her routes. It also introduces user movement as a natural approach to interacting between users and agents. To demonstrate the utility and effectiveness of the system, we constructed location/user-aware visitor-guide services and experimented them for two weeks in a public museum.

Key Words: User Navigation, Mobile Agent, Context-aware Service, RFID

Category: SD C.2.4 Distributed applications; SD D.2.9 Software configuration management; SD F.1.2 Interactive Computation; SD H.5.2 User interface management systems

1 Introduction

The use of user/location-aware user-assistant services in museums has attracted a great deal of attention from researchers over the past few years. This is because most visitors to museums lack sufficient knowledge about the exhibits and they need supplementary annotations on these. However, as their knowledge and experiences are varied, they may become puzzled (or bored) if the annotations provided to them are beyond (or beneath) their knowledge or interest. Nevertheless, most existing museums are unaware of their visitors and unable to assist them personally. In this paper, we explore a user/location-aware system for assisting visitors in museums by using agent technology, because agent technology can provide information for users and interact with users in their personalized forms. There have been several attempts to use agents to assist visitors in museums, e.g., [Kuflik et al., 2006, Kruppa et al., 2005, Rocchi et al., 2004].

However, there is a serious gap between the requirements of user assistant services in museums and agent-based services. Most of existing works have been developed to the prototype stage and tested in small-scale laboratory-based experiments. In fact, they have been designed in an ad-hoc manner or centralized manner to provide specific single services in particular spaces, i.e., research laboratories and buildings. As a result, they are not available for public spaces or for applications that they were not initially designed to support. Most existing projects for context aware visitor guide supported portable terminals, e.g., cellular phones and PDAs [Cheverst et al., 2000, Fleck et al., 2002, Rocchi et al., 2004]. However, such terminals often prevent visitors from fully experiencing the actual exhibits, because they tend to pay much attention to the terminals rather than the exhibits.

To solve these problems, we constructed an agent-based system for providing user/ location-aware services in a real museum with real users for real applications. It provides each user with mobile agent-based software components to deploy application-specific services at stationary computers independently of the underlying infrastructure and other services. It can also spatially bind a user to their agent/s using location-sensing systems. For example, when a user stands in front of an exhibit, his/her agent is deployed at a computer close to his/her position and provides him/her with annotation services about the exhibit in a personalized form that has been adapted to the individual user. We also introduce visitor movements as a user-friendly interaction with agents, because visitor movement is one of the most natural user behaviors in museums.

Our final goal was to offer agent-based context-aware services in large public spaces, e.g., cities, in addition to museums. Nevertheless, we designed, implemented, and operated several context-aware services in a science museum as a case study before we provide context-aware services through the wide public spaces of cities. This paper addresses the design, implementation, and application of the agent-based visitor-guide system, instead of user evaluation. However, we intend to present our experiences, e.g., usability and human aspects, in future papers.

In the remainder of this paper, we discuss the requirements of visitor-guide agents in museums (Section 2). We present the design of our agent-based visitor-guide system (Section 3) and agent architecture (Section 4). We describe the current implementation of the system (Section 5) and some experience from our experiment in a museum (Section 6). We briefly review related work (Section 7), provide a summary, and discuss some future issues (Section 8).

2 Basic Approach

This paper addresses the issue of location/user-aware services from software agents running on stationary computers located at museums instead of on portable computers carried by visitors. Our system was inspired by the real requirements of museums rather than mere academic interest.

2.1 Requirements

One of the goals of public museums, in particular science museums, is to provide experiences to visitors that enhance their knowledge of science from exhibitions. Existing science museums are passive in the sense that they are unaware of their visitors and unable to assist them in a meaningful way. That is, annotations about exhibitions should be provided according to the visitors' context, e.g., who, where, and when they are. Many museums assign explicit or implicit routes to visitors, because the order in which they view exhibits often affects what they learn about the exhibits. Our system should navigate visitors along such routes. It is difficult for visitors, particularly children, elderly, and handicapped people, to interact with agents through the buttons or touch panels of mobile or stationary terminals. To free visitors from the burden of complex operations of interacting with their agents and configuring their annotations, we use their movements between exhibits as implicit operations to select the annotations that they want to view/hear and to evaluate what they have learned from the exhibits, because visitor movement between exhibits is one of the most basic and natural behaviors in museums. Many museums tend to avoid using portable terminals because, they are too expensive to lend to visitors except for temporal experiments and they also require regular maintenance, e.g., replacing or recharging the batteries every day.

2.2 Design principles

Most visitors want to interact with human museum guides rather than computer-based guide services. However, it is impossible for museums to provide each visitor with human guides. Instead, our system enables agents to follow and interact with visitors. It is unique among other existing agent-based museum guide systems, because it detects the presence of visitors standing in front of specified exhibits and it deploys their agents at stationary computers close to their current exhibits. Agents provide annotations about exhibits to assist visitors to learn about or be impressed by them. When a visitor is in front of an exhibit, his/her agent should be deployed at computers close to his/her current location. The agent selects/configures annotations according to his/her current or previous behaviors and provides him/her with the annotations without his/her explicit operations. Our design principles are discussed as follows:

- Agents should accompany their visitors and annotate exhibits in front of the visitors in the real-world on behalf of museum guides or curators. They also should navigate their visitors to exhibits along routes assigned to the visitors by museums.
- Agents are responsible for recording and maintaining user preferences and experiences. They should select and provide visitors with annotation about exhibits in personalized form according to their knowledge and experience, e.g., the exhibits that the visitors previous viewed, in addition to their current locations.
- User-manipulation must be simple and natural, because it is difficult for visitors to explicitly operate devices, including buttons, mice, touch panels, and smart cards. We introduce visitor movements as a user-friendly interaction with agents, because visitor movement is one of the most basic and natural behaviors in museums.
- Annotations about exhibits may often be changed and modified, because real museums frequently replace and relocate displays in their exhibition spaces. Non-professional administrator, e.g., curators, should be able to easily define and customize visitor-guide agents.

We also have the following system requirements.

– User-assistant services, including visitor-guide services in museums, are likely to be accessed often by users. Such services should be executed at nearby computers to minimize communication delays because the services, including user-interfaces, are provided by the computers without any communication between computers.[1]

– Computers in public spaces may only have limited resources, such as restricted levels of CPU power and amount of memory. They cannot support all the services that may be needed. We therefore have to deploy services, i.e., content and software for playing the content, at computers while those services are needed.

– Our final aim is widespread building-wide and city-wide deployment of ubiquitous computing systems. It is almost impossible to deploy and administer a system in a scalable way when all of the control and management functions are centralized.

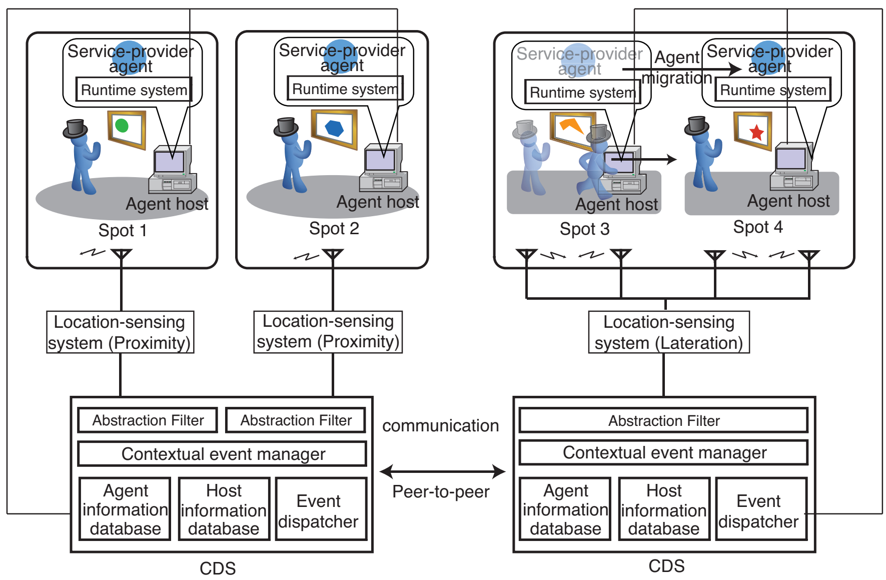


Figure 1: Architecture of Context-aware Visitor-guide Agent System.

[1] If such services are provided through other approaches, e.g., client-server, communication delays between client-side computers and server-side computers may seriously affect interaction between users and agents.

3 Agent Platform

Our user/location-aware visitor-guide system consists of three subsystems: 1) context-aware directory servers, called CDSs, 2) agent runtime systems, and 3) context-aware mobile agents (Fig. 1). The first is responsible for reflecting changes in the real world and the location of users when services are deployed at appropriate computers. The second is running on stationary computers, which are located at specified spots close to exhibits in a museum and are equipped with user-interface devices, e.g., display screens and loudspeakers. It is responsible for executing and migrating user-assistant agents, like existing runtime systems for mobile agents [Satoh, 2010]. The third is an autonomous entity that defines application-specific services for visitors. It is implemented as one or more mobile agents. Each mobile agent is a self-contained autonomous programming entity. User/location-aware visitor-guide services are encapsulated within the third subsystem so that the first and second subsystems are independent of any application specific services or other agents, which are simultaneously running to provide different services. Our system can consist of multiple CDSs, which are individually connected to other servers in a peer-to-peer manner. Each CDS only maintains up-to-date information on partial contextual information instead of on tags in the whole space.

Each agent can maintain per-user preferences on a user and record the user's behavior, e.g., exhibits that they have looked at. The agent can also define user-personalized services adapted to the user and access location-dependent services provided at its current computer. Each agent is spatially bound to, at most, one user. When a user gets closer to an exhibit, our system detects the migration of the user by using location-sensing systems and then instructs the user's agents to migrate to a computer close to the exhibit. Mobile agents help to conserve limited resources, because each agent only needs to be present at the computer while the computer needs the services provided by that agent. Agents can be managed in a non-centralized manner. When an agent migrates to another computer, it does not have to interact with the source computer.

3.1 Agent runtime system

Each agent runtime system is responsible for executing and migrating agents to other agent runtime systems running on different computers through a TCP channel using mobile-agent technology. It is built on the Java virtual machine (Java VM) version 1.5 or later versions, which conceals differences between the platform architectures of the source and destination computers.

It provides each agent with one or more active threads but it governs all the agents inside it and maintains the life-cycle state of each agent. When the life-cycle state of an agent changes, e.g., when it is created, terminates, or migrates to another runtime system, its current runtime system issues specific events to the agent to execute specified callback methods defined in the agent.

Agent runtime systems can exchange agents with other runtime systems through TCP/IP. When an agent runtime system migrates an agent to another runtime system over the network, not only the agent's code but also its state is transformed into a bit-stream through Java's object serialization package and the bit stream is then transferred to the destination.

Since the package does not support the capturing of stack frames of threads, when an agent migrates to another computer, the agent needs to stop its active threads. Its runtime system propagates certain events to invoke specified callback methods defined in agents so that they can stop their active threads and release their previously acquired resources. The agent runtime system on the receiving side receives and unmarshals the bit stream. Since arriving agents may explicitly have to acquire various resources, e.g., video and sound, or release previously acquired resources, the runtime systems propagate certain events to agents to invoke callback methods defined in the agents. Agent runtime systems can explicitly store agents on secondary storage, so that the agents can continue to assist their users when they want the services provided by the agents.

To offer protection against malicious agents being passed between computers, each runtime system supports a Kerberos-based authentication mechanism for agent migration. It authenticates users without exposing their passwords on the network and generates secret encryption keys that can selectively be shared between parties that are mutually suspicious parties. Since it can inherit the security mechanisms provided in the Java language environment, the Java VM explicitly restricts agents so that they can only access specified resources to protect computers from malicious agents. Agent runtime systems can explicitly permit their visiting agents to access resources, e.g., multimedia content, inside them and resources provided via specified URLs over an intranet or external network, e.g., the Internet, instead of the resources provided at their current computers.

3.2 Context-aware deployment of agents

Each CDS is responsible for monitoring location-sensing systems and spatially binding more than one user-assistant agent to each user. It maintains two databases. The first stores information about each of the agent runtime systems and the second stores each of the agents attached to users. It can exchange this information with other CDSs in a peer-to-peer manner.

3.2.1 Location-sensing management

Tracking systems can be classified into two types, i.e., proximity and lateration. The first approach detects the presence of objects within known areas or close to known points, and the second estimates the positions of objects from multiple measurements of the distance between known points. The CDSs support the two types, but it maps geometric information measured by the latter sensing systems to specified areas, called *spots*, where the exhibits and the computers that play the annotations are located. This is because most context-aware services in public spaces should be provided within specified spaces rather than at specified geometric points. Each CDS has its own local database to maintain the locations of visitors and their agents.

Although our system itself is independent of the underlying sensing systems, the experiment presented in this paper supported active RFID tag systems, where museums have provided one or group visitors with RFID tags. These tags are small RF transmitters that periodically broadcast beacons, including the identifiers of the tags, to receivers located in exhibition spaces. The receivers locate the presence or position of the tags. CDSs generate sensing-system-independent identifiers from the identifiers of tags so that agent runtime systems and agents should be independent of the underlying location sensing systems.

3.2.2 Agent discovery mechanism

When the underlying sensing system detects the presence (or absence) of a visitor (or the RFID tag tied to the visitor) in a spot, that is, the visitor in the spot, the CDS attempts to instruct the agent attached to the visitor to the agent runtime system close to his/her current location.

- The CDS multicasts a query message that contains the identity of the visitor (or the identity of the tag) to the CDS that manages the system attempts to query the locations of the agent tied to the visitor from its local database.
- If the database does not contain any information about the identifier of the visitor (or the tag tied to him/his), it multicasts a query message that contains the identity of the new visitor (or the tag) to other CDSs through UDP multicasting.
- It then waits for reply messages from other CDSs. Next, if the CDS knows the location of the agent tied to the visitor (or the tag), it sends a control message to the agent runtime system that runs the agent so that it instructs the agent to migrate to computers close to the visitor (or the tag).

For example, when a tag bound to a user who has his/her agents is close to a computer, the system identifies the user and deploys his/her agents at computers that are close to him/her. The system relies on UDP multicasting, but CDSs can communicate other CDSs, which are not in their same domain, through TCP/IP communication. When CDSs sends control messages to agent runtime systems, they map the identifier of each tag into the corresponding sensing-system-independent identifier. When each agent runtime system migrates or receives agents, it sends a message about the identifier of the agents that leaves from or arrives at, to nearby CDSs through UDP multicasting.

3.2.3 Queuing mechanism

There are many visitors in museums so that we cannot cope with conflicts caused by multiple users. For example, more than two visitors may simultaneously view and hear at most one annotation provided from a stationary computer at an exhibition space under the impression that the annotation is for them. Some navigation or annotation content, e.g., audio-annotation, should also be played without any interruptions.

To solve this problem, we support two approaches. The first is to use the visual representation of agents, e.g. characters, as a method of assisting visitors to know who the annotation is for. The second is to provide each agent runtime system with a queuing mechanism for exclusively executing agents for multiple simultaneous users. When two users enter the same spot, the CDS sends two notification messages to the agent runtime system in the spot in the order in which they entered it. The runtime system can send events to the two agents bound to the two users in that order, or it can explicitly send an event to one of the agents. After the first agent has handled the event, it sends the same event to the second agent. It supports several typical queuing policies, e.g., LIFO and synchronization among more than two users.

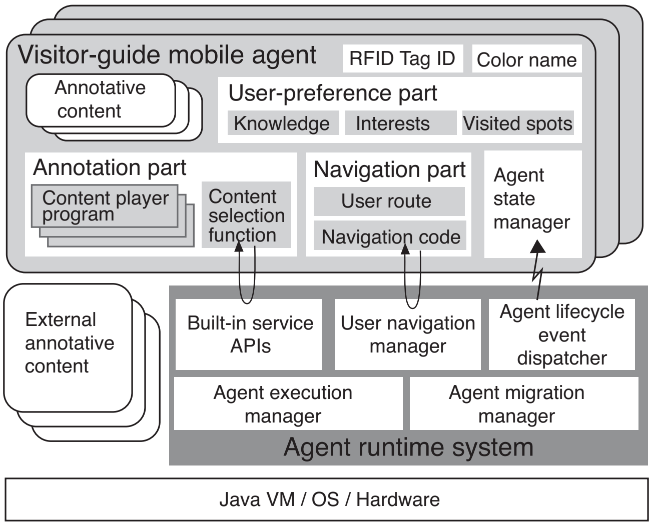


Figure 2: Architecture of agent runtime system.

4 Visitor-guide Mobile Agent

Each agent is attached to at most one visitor and maintains his/her preference information and programs that provide annotation and navigation to him/her (Fig. 2). Each agent keeps the sensing-system-independent identifier of its visitor. It is defined as a collection of Java objects. Nevertheless, to easily define/customize user/location-dependent services, it can be dynamically assembled from the following three parts:

- **The user-preference part** maintains and records information about visitors, e.g., their knowledge, interests, routes, name, and the durations and times they spend at exhibits they visit.
- **The annotation part** defines a task for playing annotations about exhibits or interacting with visitors.
- **The navigation part** defines a task for navigating visitors to their destinations.

These parts are loosely connected with one another through data attributes by using Java's introspection mechanism so that they can be replaced without the need for any compilations or linkages for their programs. We use the standard JAR file format for archiving these parts because this format can support digital signatures, which enables authentication.

4.1 User-preference part

This is responsible for profiling and maintaining information about a visitor, because user information tends to depend on application-specific services rather than the system itself. Each agent has a built-in database for maintaining the identifiers of, the number of visits to, and the length of stays at spots by visitors. Its current runtime system updates the database according to changes in the contextual information about of the visitor maintained by CDSs. For example, when a visitor leaves from a spot, the length of stays at the spot is recorded in the user profile maintained in his/her agent. The annotation and navigation parts can access entries with key names so that these parts can be combined loosely and replaced with compatible parts. Therefore, agents can infer what exhibit their target visitors are interested in and adapt their annotation to their interests, because the visitors tend to stay longer in front of exhibits in which they are interested.

The database is implemented as a hashtable to maintain the collection of data entries. Each entry is a pair of a name and a value, where the former is string data and the latter has an arbitrary data structure represented as Java objects. Since its format is compatible with Binary Large OBject (BLOB), used as one of the most basic data structure in existing cloud computing infrastructures, e.g., Google's App Engine and Microsoft's Windows Azure, these entries can be naturally stored on cloud computing to enable the entries to be analyzed later. The database can also write information about such user behaviors on a spread-sheet format when users complete their routes. It is possible to make an educated guess with some degree of probability as to what a visitor may be interested in, if we know which spots he/she has visited, how many he/she has visited, and how long he/she spend in visiting them.

Administrators in the experiment a explicitly asked visitors about their knowledge and interests and manually input the information into this part. This was because it was difficult to accurately infer what visitors knew or were interested in only from data that were measured with existing sensing systems due to measurement errors the data contained. For example, if sensing systems miss detecting the presence of a visitor at a spot due to the absence of radio signals emitted from his/her tag, his/her agent will record that he/she spent a shorter time than the actual duration he/she remained at the spot. Instead, our system allows users to explicitly select the routes they are interested in according to their movements will be explained later.

4.2 Annotation part

This part is responsible for selecting and playing annotations according to the current spot and route in addition to information stored in the user-preference part and it plays content in the personalized form of its user. It is defined as a content-selection function and offers programs for playing the selected content.

4.2.1 Content-selection function

The function is state-less and maps more than one argument, e.g., the current spot, the user's selected route, and the number of times he/she has visited the spot into a URL referring to the annotative content. The content can be stored in the agent, the current agent runtime system, or external http servers. That is, each agent can carry a set of its content, play the selected content at its destinations, directly play the content stored at its destinations, or download and play the content stored in Web servers on the Internet. The current implementation can divide this part into three sub-parts: opening, annotation, and closing, which are played in turn.

4.2.2 Content player program

Annotation content is varied, e.g., text, image, video, and sound. The annotation part defines programs for playing this content. The current implementation supports (rich) text data, html, and image data, e.g, JPEG and GIF, video data, e.g., animation GIF and MPEG, and sound data, e.g., WAV and MP3. The format for content is specified in an MIME-based attribute description. Since the annotation part is defined as Java-based general-purpose programs, we can easily define programs to play other formats and to interact with visitors.

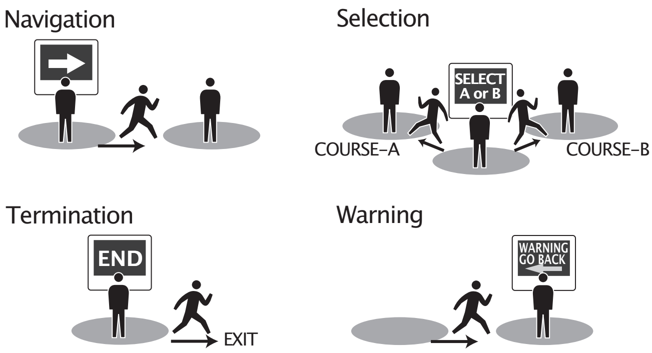


Figure 3: User-navigation patterns.

4.3 Navigation part

Our agents are required to navigate visitors to their destinations along routes recommended by museums or visitors. After executing their annotation part, the navigation part is invoked by the runtime system to provide visual (or audio) information on the screens of the displays (or from loudspeakers) of the current agent runtime system. For example, the agents display directions to exhibits that their visitors should next see. We also introduced visitor movements between exhibits as an implicit operation for selecting the routes that they wanted and evaluating what they had learned from the exhibits, because visitor movement is one of the most basic and natural behaviors in museums. As shown in Fig. 3, the current implementation supports the four navigation patterns as follows:

- *Navigation* instructs users to move to at least one specified destination spot.
- *Selection* enables users to explicitly or implicitly select one spot or route from one or more spots or routes close to their current spots by moving to the selected spot or one spot along the selected route.[2]
- *Termination* informs users that they have arrived at the final destination spot.
- *Warning* informs users that they had missed their destination exhibit or their routes.

The second pattern is also useful to capture users' interesting and knowledge. For example, an agent enables its user to select one of the spots corresponding to his/her interesting or his/her answer for quizzes.

[2] The current implementation does not limit the number of the spots that users can select, but in fact it depends on the layout of spots to avoid users to pass through other spots.

The user's route is described as a sequences of primitives corresponding to the above free patterns with our language for specifying the itineraries of mobile agents for network management [Satoh, 2006] through a GUI-based configuration system by administrators. The user-preference part maintains the route inside it. No agent knows the spatial directions to the destinations because the directions themselves depend on the spatial relationships between the locations of the current agent runtime system and the locations of the destinations, as well as the direction to the current computer's screen. Agent runtime systems provide several built-in APIs to their visiting agents. For example, if an agent has at least one destination, it invokes a specified API corresponding to the first pattern with the name of the destination; its current runtime system returns the direction to the destination to it or displays the direction on the screen on its behalf.

5 Current Implementation

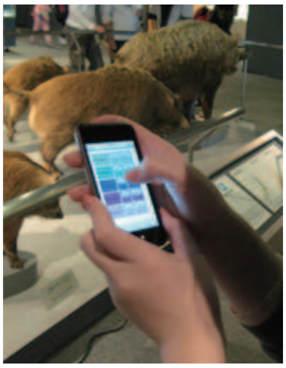
This section describes the current implementation of our system.

5.1 Support for location-sensing systems

The current implementation supports two commercial tracking systems. The first is the Spider active RFID tag system, which is a typical example of proximity-based tracking. It provides active RF-tags to users. Each tag has a unique identifier that periodically emits an RF-beacon (every second) that conveys an identifier within a range of 1-20 meters. The second system is the Aeroscout positioning system, which consists of four or more readers located in a room. These readers can measure differences in the arrival times of WiFi-based RF-pulses emitted from tags and estimate the positions of the tags from multiple measurements of the distance between the readers and tags; these measurement units correspond to about two meters. The experiments presented in the following section used the first system.

5.2 System management

There may be no space for any management systems on exhibition rooms in museums. Our system needs to enable administrators to operate and monitor the system through portable terminals instead of any stationary terminals. In the current implementation, the CDS provides Web-based APIs with AJAX technology to create, control, and terminate agents. Therefore, an operator can create and customize agents through a Web browser running on his/her (portable) computer. Figure 4 shows our GUI interface for binding agents to users, running on a portable terminal equipped with WiFi interface (Apple iPod Touch). Each agent runtime system provides an HTTP server to be monitored from the external systems for the reason of administration.



Portable administration terminal

Figure 4: Portable management terminal

5.3 Agent configuration

We also provide another GUI-based management system for configuring agents on PCs by using a compound document framework, called *MobiDoc* [Satoh, 2004] so that curators, who has no knowledge of context-aware systems, can easily change annotation content by doing drag-and-drop manipulations on the maps of their target exhibition rooms displayed on the screen. Such a configuration system is useful because museums need to continuously provide and configure services for visitors. The current implementation also enabled curators to manually input the directions of possible destinations by doing drag-and-drop manipulations. It permitted developers to define agents by overwriting built-in classes for these parts, but the experiment presented in this paper could continue the experiment without redefining any classes.

5.4 Supports to complex visitor-behaviors

We introduce as a language for defining visitors' recommended routes and annotation contents by extending the author's process calculus for specifying the routes of logistics truck [Satoh, 2008b]. Therefore, the current implementation itself can select and play special contents, only when multiple specified visitors are in the same spot by using the language or when visitors enter the spots where other visitors were being. Nevertheless, the language will be left to our future paper, because the experiment presented in this paper could be operated without the language.

5.5 Performance evaluation

Although the current implementation was not built for performance, we measured the cost of migrating a null agent (a 5-KB agent, zip-compressed) and an annotation agent (1.2-MB agent, zip-compressed) from a source computer to a destination computer that was recommended by the CDSs. The latency of discovering and instructing an agent attached to a tag after the CDS had detected the presence of the tag was 420 ms and the respective cost of migrating the null and annotation agent between two runtime systems running on different computers over a TCP connection was 41 ms and 490 ms. This evaluation was operated with three computers (Intel Core 2 Duo 2 GHz with MacOS 10.5 and Java ver. 5) connected via a Fast Ethernet. This cost is reasonable for migrating agents between computers to follow visitors who are walking moving between exhibits.

5.6 Support for location-sensing systems

The current implementation supports two commercial tracking systems, i.e., the two types of location sensing systems discussed in this paper. The first is the Spider active RFID tag system, which is a typical example of proximity-based tracking. It provides active RF-tags to users. Each tag has a unique identifier that periodically emits an RF-beacon (every second) that conveys an identifier within a range of 1-20 meters. The second system is the Aeroscout positioning system, which consists of four or more readers located in a room. These readers can measure differences in the arrival times of WiFi-based RF pulses emitted from tags and they can estimate the positions of the tags from multiple measurements of the distance between the readers and tags; these measurement units correspond to about two meters.

6 Experience

We constructed and carried out an experiment at the Museum of Nature and Human Activities in Hyogo, Japan, using the proposed system. Figure 5 has a sketch that maps the spots located in the museum.

6.1 Experiment

The experiment utilized four spots in front of exhibits, which were specimens of stuffed animals, i.e., a bear, deer, racoon dog, and wild boar in the exhibition room of the museum.[3] Each spot could provide five different pieces of animation-based annotative content about the animals, e.g., their ethology, footprints, feeding, habitats, and features, and each had a display and Spider's active RFID reader with a coverage range that almost corresponded to the space, as shown in Fig. 6.

[3] The number of spots and their locations depend on the target environments, e.g., museums.

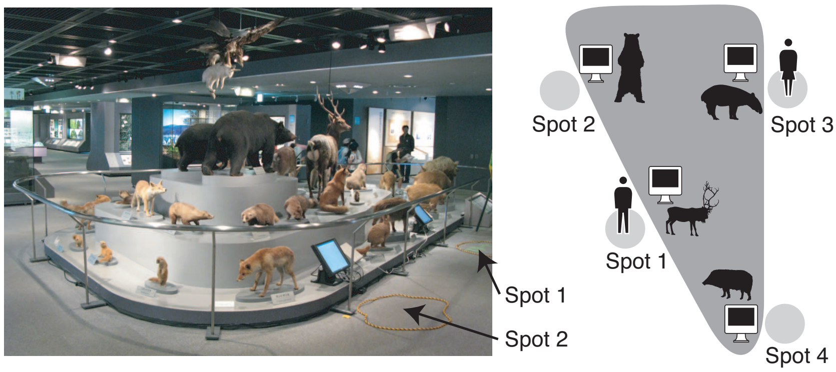


Figure 5: Experiment at Museum of Nature and Human Activities in Hyogo.

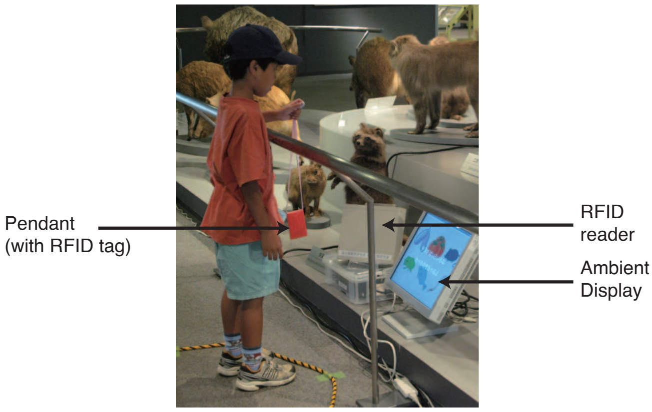


Figure 6: Spot at Museum of Nature and Human Activities in Hyogo.

Each visitor was provided the experiment with a colored pendant including RFID tags, because it was designed to enable visitors to imagine that their agents, which were virtual owls, were within their pendant (Fig. 7). When a visitor with his/her pendant first participated in the experiment, an operator input the point of interest and the route for the new visitor and created his/her agent by using a Web browser running on a portable terminal. In fact, most curators in museums have a positive impression of such portable management terminals rather than stationary management terminals, because stationary terminals require museums to provide spaces to house them.

These pendants are green, orange, blue, yellow, or red. A newly created agent is provided with its own color corresponding to the color of the pendant attached to the agent, because visitors could distinguish between their agents and others' agents through their

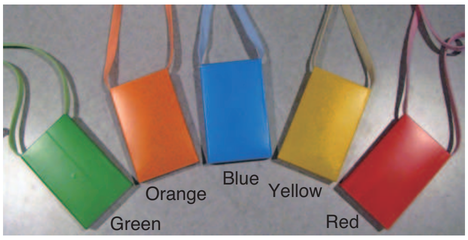


Figure 7: Five colorful pendants

pendants' colors. This may seem to be naive but it effectively enables visitors to readily identify their agents. For example, suppose that a visitor enters the spot with the specimen of the racoon dog, his/her agent migrated from his/her pendant to the display located in the spot. As shown in Fig. 8, an agent tied to the orange pendant plays the opening animation to inform that its target is a visitor with an orange pendant, where the animation shows the agent's character appearing as an orange pendant. It next plays the annotation and then the closing animation. The durations of the opening animation, annotation, and closing are 7 sec, shorter than 40 sec, and 5 sec.

We simultaneously provided two kinds of routes for visitors to evaluate the utility of our user-navigation support. Both routes navigated visitors to destination spots along the way (Fig. 9). They enabled all visitors to walk around an exhibition both consisting of four spots two or three times, as shown on the right of Fig. 5. That is, a visitor might visit the same spots two or three times depending on the navigation providing by his/her agent (Fig. 10). In addition, the first route enabled visitors to explicitly select subjects they preferred by moving to one of the neighboring spots corresponding to the subjects selected in specified spots at specified times. The second route provided visitors with several quizzes to review what they had learnt about the animals by selecting neighboring spots corresponding to their answers in specified spots at specified times. Both the routes were defined in the language discussed in the previous section and offered visitors animation-based annotative content about the animal in front of them so they could learn about it while observing the corresponding specimen.

The experimental system consisted of one CDS and runtime systems running on four computers. It provided GUI-based monitoring and configuration for agents. When the CDS detected the presence of a tag bound to a visitor at a spot, it instructed the agent bound to the user to migrate a computer contained in the spot. After arriving at the computer, the runtime system invoked a specified callback method defined in the annotative part of the agent. The method first played the opening animation for the color of the agent and then called a content-selection function with his/her route, the name of the current spot, and the number of times that he/she had visited the spot. The latency of migrating an agent and starting its opening animation at the destination

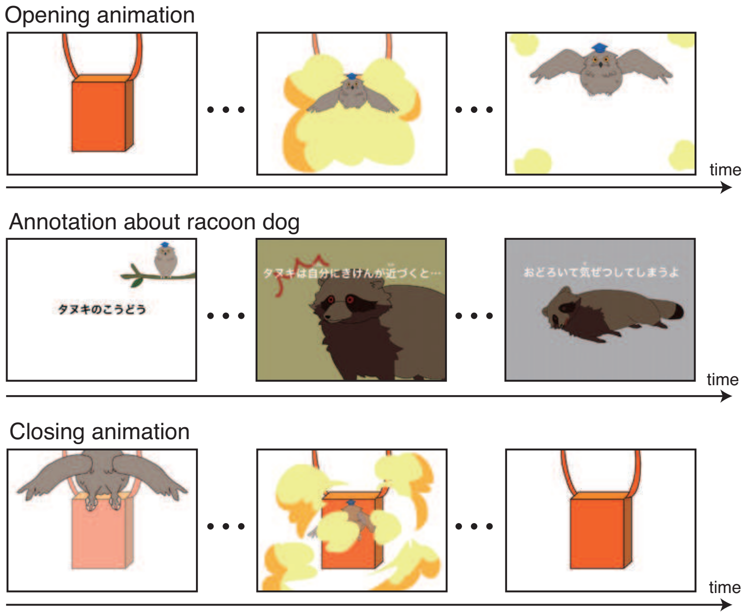
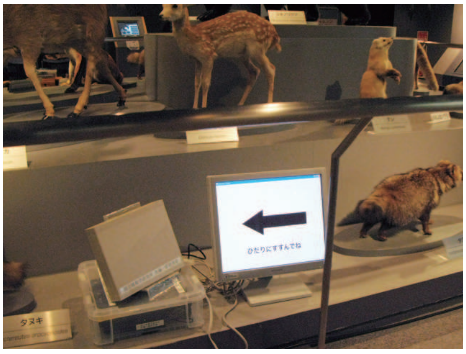


Figure 8: Opening animation, annotation animation, and closing animation for orange pendant

after visitors had arrived at a spot was within 2 seconds, so that visitors could view the opening animation soon after they began standing in front of the exhibits. The method next played the selected content and then played the closing animation. After that, the runtime system invoked a specified callback method defined in the navigation part. An agent bound to a user could recommend two or more destination spots by using the *selection* pattern provided on its current computer. When a visitor moved to one of the spots, his/her agent could record their selection. If the selection corresponded to a quiz choice, when a user moved to a spot corresponding to a correct or incorrect answer, their agent modified the visitor's profile that was maintained within it. Furthermore, if a user left out his/her route, the navigation part invoked a method to play warning content for him/her to return to his/her previous spot.

We did the experiment over two weeks. More than 60 individuals or groups took part in it each day. Most of the participants were groups of families or friends aged from 7 to 16. Most visitors answered questionnaires about their answers to the quizzes and their feedback on the system in addition to their gender and age. Although the aim of this paper is not user evaluation, we will present some basic user evaluation. Almost all the participants (more than 95 percent) provided positive feedback on the system. Typical feedback was "We were very interested in or enjoyed the system", "We could easily



Navigation to one destination

Selection from two destinations

Figure 9: Navigation patterns for user navigation.

answer the quizzes by moving between the spots", and "We gained detail knowledge about the animals by watching them in front of where we were standing positions." Most visitors appeared to be interested in watching the specimens of animals instead of the visual effects, e.g., the animations of agents.

6.2 Discussion

We learned the four main things from the above experiment.

– In the experiment, the author could deploy, install, and configure the entire system, including location-sensing systems and servers, by him within one day, i.e, the day the museum was closed, without conducting any preparatory experiments in the exhibition room.

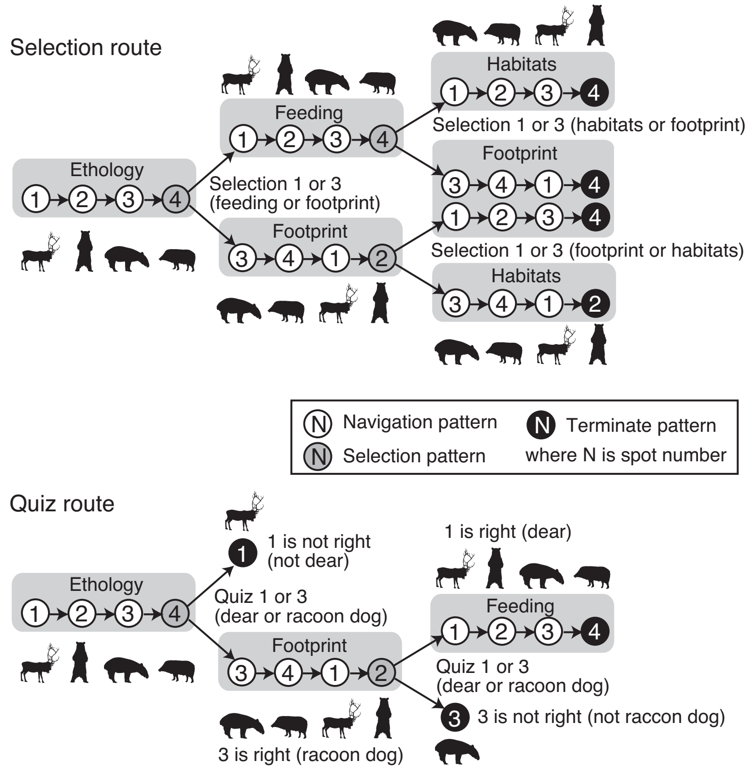


Figure 10: Two routes (selection and quiz)

- The experiment provided visitors with two different visitor-guide services. It could provide more kinds of services, because application-specific services could be defined and encapsulated within the agents and provided independently of other agents by the underlying system. In fact, this is the main advantage of our system in comparison with other existing agent-based context-aware systems, because they explicitly and implicitly assume their agents are predefined at a system-level so that they do not allow end-users to customize agents.

- The system used in the experiment allowed curators to monitor user behaviors, e.g., the durations and times they spent at exhibits they visited in real time. The experiment itself could select content according to the spots that users had previously visited. Unfortunately the durations that they spent at spots were not used in the experiments, because the sensing system occasionally could not detect users were present. Nevertheless, curators used information about such user behaviors to evaluate the focus of users in exhibits, because the information was recorded in a spread-sheet format.

- The experiment provided pre-defined parts. When a user registered to take part in our experiment, his/her agent was composed of three pre-defined parts according to his/her initial preference by using a GUI-based manipulation based on a compound document-based framework for building pervasive computing presented in our previous paper [Satoh, 2007b]. Although it enabled developers to define agents by overwriting built-in classes for these parts, the experiment could continue the experiment without having to redefine any classes.

- Our agents should be designed to play extra roles, because exhibits should play the leading roles in museums. That is, although they can provide users with enrich content, e.g., interactions and animation, they must not offer any entertainment, even though entertainment may be useful for children who want to experience annotation services. This is because, children under school age are liable to be more impressed by the entertainment features of annotation services in exhibits rather than the exhibits themselves and their annotative content. In fact, most exhibits in museum are visual objects. Therefore, if agents have overly attractive visual features, e.g., animation and 3D modeling, these actually prevent visitors from fully viewing the exhibits.

- Existing location-sensing systems are not optimal in the sense that errors in their measurements are unavoidable. The ranges and rates of measurement errors tend to depend on location-sensing systems. There are two typical errors. The first error occurs when tags are missed that are attached to visitors. Services expected by users are occasionally not offered because visitors are not detected by sensors. The second error occurs when the positions of tags are estimated far from their real positions. Users in incorrect locations may lead to many users being detected, who are not participating in the experiment, which leads to serious confusion. In a museum setting, the second is more serious than the first, because personalized services to assist particular users need to be provided in correct locations. If other users around the locations receive their services, serious confusion may result.

Our experiments were small scale but it they were a case study in our development of context-aware services in large-scale public spaces. We could gain positive impression about the scalability and availability of the system for large-scale public services for six main reasons:

- The experiment system could be designed and operated without any centralized management mechanisms.

- No runtime system or CDS had to be aware of any global information. In fact, no agent runtime systems had to know about other agent runtime systems, because the CDSs informed runtime systems about the destination of the agents running on the runtime systems.

- CDSs could be managed in a peer-to-peer manner but their management cost was small. This is because they could exchange information about agents with only nearby CDSs, because walking users in a spot could move to only nearby spots but not to any distant spots.
- Since the speed of agent migration between runtime systems was much faster than the speed of user movement between spots, agents could follow their users' movement.
- The latency of detecting the presence of a user and informing an agent attached to the user about the destination was shorter than 160 ms. The interval for active RFID tag system to detect RFID tags used in the experiment was one second. They were reasonably short for migrating agents between computers to enable them to follow walking users.
- The number of agents running or waiting on a single computer was bound to the number of users in the spot that the computer was in. Agents were also only deployed and executed where they were wanted.

7 Related Work

There have been several agent-based or non-agent-based attempts to develop context-aware services for museums with the aim of enabling visitors to view or listen to information about exhibits at the right time and in the right place and to help them navigate between exhibits along recommended routes [Cheverst et al., 2000, Fleck et al., 2002, Oppermann et al., 1999]. However, most of existing attempts have been developed to the prototype stage and tested in laboratory-based or short-time experiments with professional administrators. They also have been designed in an ad-hoc manner to provide specific single services in particular spaces, i.e., research laboratories and buildings. For example, real systems are required to be robust and provide users with the services that they are designed to provide, whereas prototype systems for demonstrations are often allowed to be unreliable and provide insufficient services. There is a serious gap between laboratory-level or prototype-level systems and practical systems. Therefore, this section discusses only several related work that provided real applications for real users in public spaces, in particular museums.

One of the most typical approaches in public museums has been to provide visitors with audio annotations from portable audio players. These have required end-users to carry players and explicitly input numbers attached to the exhibits in front of them if they have wanted to listen to audio annotations about the exhibits. Many academic projects have provided portable multimedia terminals or PDAs to visitors. These have enabled visitors to interactively view and operate annotated information displayed on the screens of their terminals, e.g., the Electronic Guidebook, [Fleck et al., 2002], the Museum Project [Ciavarella et al., 2004], the Hippie system [Oppermann et al., 1999], ImogI [Luyten et al., 2004], and Rememberer [Fleck et al., 2002]. They have assumed that visitors are carrying portable terminals, e.g., PDAs and smart phones and they have explicitly input the identifiers of their positions or nearby exhibits by using user interface devices, e.g., buttons, mice, or the touch panels of terminals. However, such operations are difficult for visitors, particularly children, the elderly, and handicapped people, and tend to prevent them from viewing the exhibits to their maximum extent. These approaches have several serious problems in real museums. In addition, one of the most serious problems associated with portable smart terminals and multimedia systems is that they prevent visitors from focusing on the exhibits because they tend to become interested in the device rather than the exhibitions themselves.

A few researchers have attempted approaches to support users from stationary sensors, actuators, and computers. However, most of these systems have stayed at the prototype- or laboratory-level and have not been operated and evaluated in real museums. Therefore, the results obtained may not be able to be applied to practical applications. Of these, the PEACH project [Rocchi et al., 2004] has been developed as a visitor-guide system for use in museums and it has been evaluated in a museum. The system supported PDAs in addition to ambient displays and estimated the location of visitors by using infrared and computer-vision approaches. The project proposed a system that enabled agents to migrate between computers [Kruppa et al., 2005] by displaying an image of an avatar or character corresponding to the agent on remote computers, including public terminals, but it could not migrate agents themselves to computers. Like the PEACH project, several existing systems have introduced the notion of agent migration, but they have supported only the images of avatars or codes with a piece of specific information, instead of the agents themselves. Therefore, their services could not be defined within their agents independent of their infrastructures, so that they could not customize multiple services while the infrastructures were running, unlike our system. The PEACH project used a RFID tag system to identify users [Kuflik et al., 2006], but it assumes to use portable terminals. The Virtual Anatomy Assistant Ritchie [Wiend et al., 2007] was an attempt to seamlessly integrate characters into user actions by using 3D-optical tracking systems but it focused on conversation between individual users and agents and did not support user navigation.

We will now discuss differences between the framework presented in this paper and our previous frameworks. We previously presented an approach for deploying mobile agents spatially bound to physical places and objects at computers that moved in the places or were close to the objects [Satoh, 2003]. However, it was not designed for user-navigation, unlike the framework proposed in this paper. We also constructed a location model for ubiquitous computing environments [Satoh, 2005, Satoh, 2007a]. It represented spatial relationships between physical entities (and places) as containment relationships between their programmable counterpart objects and deployed counterpart objects at computers according to the positions of their target objects or places. This was a general-purpose location-model for context-aware services, but was not an infrastructure for deploying and operating such services. We presented an outline of mobile agent-based services in public museums in our early version of this paper [Satoh, 2008], whereas we addressed an implementation of agent-based visitor guide systems and some early experiences with the system in this paper.

8 Conclusion

We designed and implemented an agent-based system for building and operating context-aware visitor-guide services in public spaces, including museums. When a visitor moves from exhibit to exhibit, his/her agent can be dynamically deployed at a computer that is close to the current exhibit to accompany him/her and play annotations about the exhibit according to his/her knowledge, interest, and the exhibits that he/she is watching. Therefore, he/she does not need to carry any portable computers. His/her agent consists of three parts and navigates him/her to exhibits along his/her route. It also introduces visitor-movement between exhibits as a natural interaction between users and agents. It can be managed in a non-centralized manner unlike other existing similar systems to support large-scale context-aware systems.

Finally, we would like to identify further issues that need to be resolved. Some may consider that the non-centralized management architecture we used needed to operate the context-aware visitor-guide services in museums, as described in this paper. However, our final goal is to provide these large-scale context-aware services in extensive spaces, e.g., cities. We therefore need to demonstrate the scalability of the framework for large-scale context-aware services. This paper only described the design and implementation of agents and the effectiveness and utility of the model; however, the experiments provided us with other interesting experiences, e.g., usability, human aspects, and user modelling. We intend to present our findings from these experiences in our future papers. The current implementation supports only four typical navigation patterns, although there are a variety of potential patterns. We plan to support other patterns.

Acknowledgements

We are grateful to Prof. Shigenori Inagaki (Kobe University), Prof. Hiroshi Mizoguchi (Tokyo University of Science), and Prof. Fusako Kusunoki (Tama Art University) who are co-researchers of the experiment presented in this experiment. We thank students of Tama Art University because they created animations and helped the experiment. We thank Prof. Tohru Sakiyama, for his giving a chance to the experiment at We constructed and operated an experiment at the Museum of Nature and Human Activities in Hyogo.

References

[Cheverst et al., 2000] Cheverst, K., Davis, N., Mitchell, K., Friday, A.: "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project"; Proc. Conference on Mobile Computing and Networking (MOBICOM'2000), ACM Press (2000), 20-31.

[Ciavarella et al., 2004] Ciavarella, C., Paterno, F.: "The Design of a Handheld, Location-aware Guide for Indoor Environments"; Personal and Ubiquitous Computing, 8, 2 (2004), 82-91.

[Fleck et al., 2002] Fleck, M., Frid, M., Kindberg, T., Rajani, R., O'BrienStrain, E., Spasojevic, M.: "From Informing to Remembering: Deploying a Ubiquitous System in an Interactive Science Museum"; IEEE Pervasive Computing, 1, 2 (2002), 13-21.

[Kuflik et al., 2006] Kuflik, T., Albertini, A., Busetta, P., Rocchi, C., Stock, O., and Zancanaro, M.: "An Agent-Based Architecture for Museum Visitors' Guide Systems"; Proc. Information and Communication Technologies in Tourism, Springer (2006), 57-60.

[Kruppa et al., 2005] Kruppa M., Kruger, A.: "Performing Physical Object References with Migrating Virtual Characters"; Proc Intelligent Technologies for Interactive Entertainment, LNCS, Vol.3814, Springer (2005), 64-73.

[Luyten et al., 2004] Luyten, K., Coninx, K.: "ImogI: Take Control over a Context-Aware Electronic Mobile Guide for Museums"; Proc. Workshop on HCI in Mobile Guides, (2004).

[Oppermann et al., 1999] Oppermann, R., Specht, M.: "A Context-Sensitive Nomadic Exhibition Guide"; Proc. HUC'2000, LNCS vol.1927, Springer (2000), 127-142.

[Rocchi et al., 2004] Rocchi, C., Stock, O., Zancanaro, M., Kruppa, M., Kruger, A.: "The Museum Visit: Generating Seamless Personalized Presentations on Multiple Devices"; Proc. 9th international conference on Intelligent User Interface, ACM Press (2004), 316-318.

[Satoh, 2003] Satoh, I.: "SpatialAgents: Integrating User Mobility and Program Mobility in Ubiquitous Computing Environments"; Wireless Communications and Mobile Computing, 3, 4 (2003), 411-423.

"Itinerant Agents for Network Management"; IEICE Transactions on Communication, E86-B, 10 (2003), 2865-2873.

[Satoh, 2004] Satoh, I.: "A Component Framework for Document-centric Networking"; IEICE Transactions on Communication, E87-B, 7, (2004), 1826-1833.

[Satoh, 2005] Satoh, I.: "A Location Model for Pervasive Computing Environments"; Proc of IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05), IEEE Computer Society (2005), 215-224.

[Satoh, 2006] Satoh, I.: "Building and Selecting Mobile Agents for Network Management", Journal of Network and Systems Management, 14, 1, Springer (2006), 147-169.

[Satoh, 2007a] Satoh, I.: "A Location Model for Smart Environment"; Pervasive and Mobile Computing, 3, 2, Elsevier (2007), 158-179.

[Satoh, 2007b] Satoh, I.: "Visual Components for Pervasive Computing Management"; Proc. IEEE International Conference on Pervasive Services (ICPS'07), IEEE Computer Society (2007), 19-28.

[Satoh, 2008] Satoh, I.: "Context-aware Agents to Guide Visitors in Museums"; Proc. 8th International Conference on Intelligent Virtual Agents (IVA'08), LNAI/LNCS, Vol. 5208, (2008), 441-455.

[Satoh, 2008b] Satoh, I.: "A Formal Approach for Milk-run Transport Logistics"; IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E91-A, 11 (2008), 3261-3268.

[Satoh, 2010] Satoh, I: "Mobile Agents"; Handbook of Ambient Intelligence and Smart Environments, Springer (2010), 771-791.

[Wiend et al., 2007] Wiendl, V., Dorfmuller-Ulhaas, K., Schulz, N., Andre, E.: "Integrating a Virtual Agent into the Real World: The Virtual Anatomy Assistant Ritchie"; Proc. Intelligent Virtual Agents (IVA'2007), LNCS, Vol.4722, Springer (2007), 211-224.