

# Design and Implementation of Context-Aware Museum Guide Agents

Ichiro SATOH<sup>†a)</sup>, *Member*

**SUMMARY** This paper presents an agent-based system for building and operating 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 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 navigates 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*

## 1. Introduction

The use of user/location-aware agents for assisting users in public spaces, in particular museums, has attracted much attention from and explored by several researchers over the past few years. This is because agents are useful to provide their own services. There have been several agent-based user-assistant 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. Agent technology can provide information for users and interact with users in their personalized forms. However, there is a serious gap between the requirements of user assistant services in museums and agent-based services. Most of existing attempts have been developed to the prototype stage and tested in small-scale laboratory-based experiments. 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. Also, we need a methodology for easily making user-assistant agents, because such agents are often required to be configured according to the requirements of individual visitors.

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 application. It pro-

vides each user with mobile agent-based software components to deploy application-specific services at 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. Nevertheless, we designed, implemented, and operated several context-aware services in a science museum as a case study before we provide ambient computing 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

The requirements of public museums, in particular science museums, are different to the requirements of agents, in particular interactive agents, in experimental spaces, e.g., laboratories. One of the goals of museums is to provide experiences to visitors that enhance their knowledge of science from exhibitions, but not with characters or avatars. Inter-

Manuscript received July 4, 2009.

Manuscript revised October 19, 2009.

<sup>†</sup>The author is with National Institute of Informatics, Tokyo, 101-8403 Japan.

a) E-mail: ichiro@nii.ac.jp

DOI: 10.1587/transinf.E93.D.789

actions between visitors and agents may lower the extent of the visitors' learning experiences in museums, because they tend to be preoccupied with the interactions. It may also be difficult for visitors, particularly children, elderly, and handicapped people, to interact with agents through the buttons or touch panels of portable computers, cellular phones, and stationary terminals.

On the other hand, our agents is designed to play supporting or extra roles, because exhibits should play the leading roles in museums. The system detects the presence of visitors standing in front of specified exhibits and it deploys their agents at stationary computers close to their current exhibits. They are responsible for providing visitors with the personalized annotations about the exhibits that they are standing in front of and for recording the behaviors of visitors, e.g., a history of the exhibits that the visitors had previously viewed. Also, 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.

## 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. 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 and provides him/her with the annotations without his/her explicit operations. That is, our user assistant agents should accompany their visitors and annotate exhibits in front of the visitors in the real-world on behalf of museum guides or curators. Our design principles are discussed as follows:

- 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. Agents as visitor guides should navigate their visitors to exhibits along routes assigned to the visitors.
- 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 in-

teraction 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.

Our agents can provide users with enrich content, e.g., interactions and animation, but 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.

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.<sup>†</sup>
- Computers in public spaces may have only 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 software for agents at computers only 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.

## 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 located at specified spots close to exhibits in a museum. It can execute user-assistant agents, where we assume that the computers are located at specified spots in public spaces and are

<sup>†</sup>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.

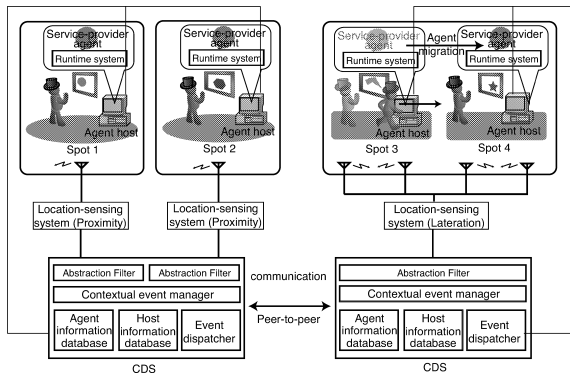


Fig. 1 Architecture of context-aware visitor-guide agent system.

equipped with user-interface devices, e.g., display screens and loudspeakers. 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 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 sys-

tem, its current runtime system issues specific events to the agent. Some navigation or annotation content, e.g., audio-annotation, should be played without any interruptions.

When an agent is transferred over the network, not only the code of the agent but also its state is transformed into a bitstream by using Java's object serialization package and then the bit stream is transferred to the destination. Since the package does not support the capturing of stack frames of threads, when an agent is deployed at another computer, its runtime system propagates certain events to instruct it to stop its active threads. The agent runtime system on the receiving side receives and unmarshals the bit stream. Arriving agents may explicitly have to acquire various resources, e.g., video and sound, or release previously acquired resources.

To protect 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 be shared selectively 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.

### 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.

#### Location-sensing management:

Tracking systems can be classified into two types: proximity and lateration. The first approach detects the presence of objects within known spots or close to known points, and the second estimates the positions of objects from multiple measurements of the distance between known points. The current implementation assumes that museums have provided visitors with 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. To abstract away differences between the underlying location-sensing systems, the CDSs map geometric information measured by the sensing systems to specified areas, called *spots*, where the exhibits and the computers that play the annotations are located.

#### Agent discovery mechanism:

When the underlying sensing system detects the presence (or absence) of a tag in a spot, the CDS that manages the system attempts to query the locations of the agent tied to

the tag from its database. If the database does not contain any information about the identifier of the tag, it multicasts a query message that contains the identity of the new 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 newly visiting tag, it instructs the agent to migrate to computers close to 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 close to him/her.

#### 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. 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. The runtime system can send events to the agents bound to the two users in that order, or can explicitly send an event to one of the agents. After the first has handled the event, it sends the same event to the second one. The current implementation supports several queuing policies, e.g., LIFO and synchronization among more than two users.

## 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 identifier of the tag attached to 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 were loosely connected with one another through data attributes by using Java's introspection mechanism so that they can be replaced without any compilations or linkages for their programs. The current implementation uses the standard JAR file format for archiving these parts because this format can support digital signatures, enabling

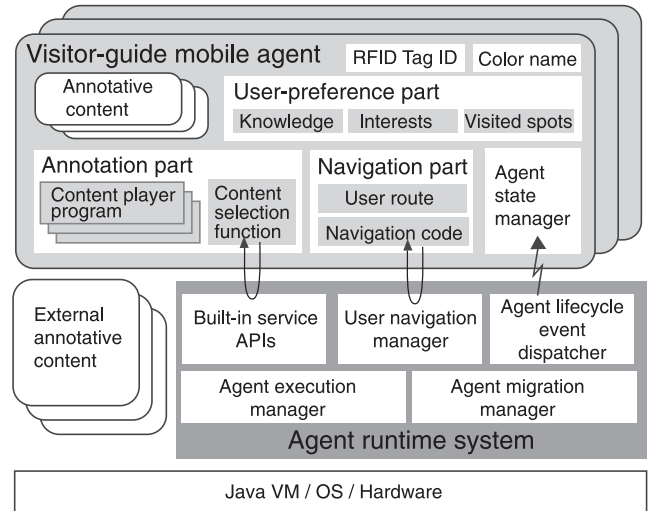


Fig. 2 Architecture of agent runtime system.

authentication.

### 4.1 User-Preference Part

This is responsible for profiling and maintaining information about a visitor. Each agent has a mechanism to automatically record the identifiers, the number of visits to, and the length of stays at spots by visitors. This part 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 App Engine and Microsoft Windows Azure, these entries can be naturally stored in cloud computing. The annotation and navigation parts can access entries with key names so that these parts can be combined loosely and replaced by compatible parts. The current implementation enables administrators to explicitly ask visitors about their knowledge and interests and manually input the information into this part. This is because it is difficult to accurately infer what a visitor knows or is interested in from data that are measured by existing sensing systems. The current implementation can also write information about such user behaviors in a spread-sheet format when users finished their courses. It is still possible to make an educated guess with some 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 was visited.

### 4.2 Annotation Part

This part is responsible for selecting and playing annotations according to the current spot and route in addition to the information stored in the user-preference part and play the content in the personalized form of its user. It is defined

as a content-selection function and programs for playing the selected content.

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 subparts: opening, annotation, and closing, which are played in turn.

Content player program:

Annotation content is various, e.g., text, image, video, and sound. The annotation part defines programs for playing this content. The current implementation supports (rich) text data, html, 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 interactions between visitors and agents.

#### 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.<sup>†</sup>
- *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 user's route is described as a sequences of primitives corresponding to the above free patterns with our language

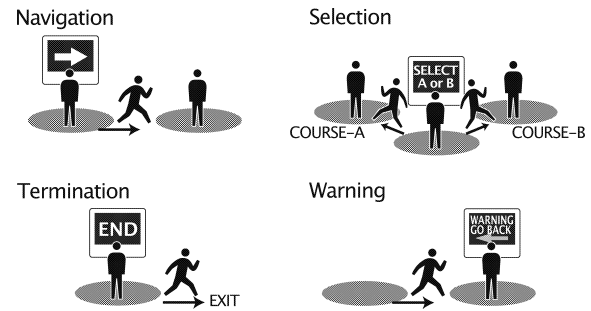


Fig. 3 User-navigation patterns.

for specifying the itineraries of mobile agents for network management [9] 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. The system can explicitly suspend and store agents on secondary storage after the agents terminate so that the agents can continue to assist their users when the users visit museums again.

#### 5. Current Implementation

This section describes the current implementation of our system.

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 for 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 timings 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.

System management:

There may be no space for any management systems on ex-

<sup>†</sup>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.



Portable administration terminal

Fig. 4 Portable management terminal.

hibition 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.

#### Agent configuration:

We also provide another GUI-based management system for configuring agents on PCs by using a compound document framework, called *MobiDoc* [10] 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 the next section could continue the experiment without redefining any classes.

#### 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 [15]. 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

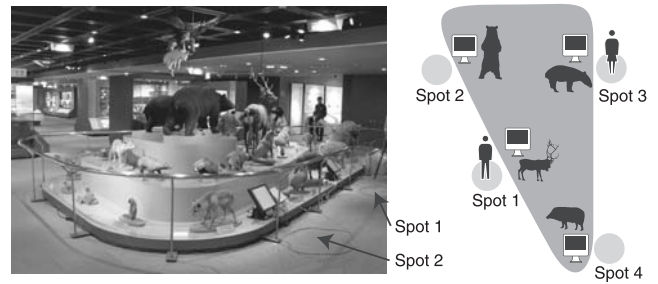


Fig. 5 Experiment at Museum of Nature and Human Activities in Hyogo.

be operated without the language.

#### 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 that visitors moving between exhibits.

## 6. Experience

We constructed and operated 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.

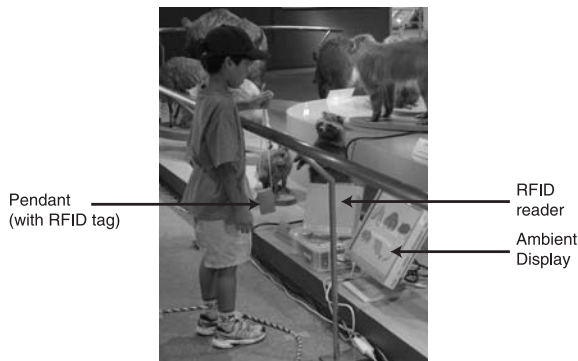
### 6.1 Experiment

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

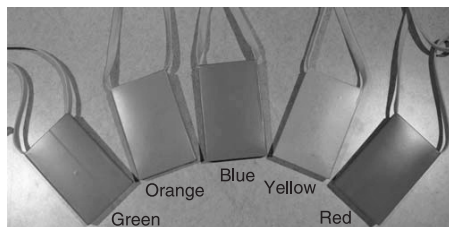
The experiment provides each visitor with a colored pendant including RFID tags, because it was designed to enable visitors to imagine that their agents, which were virtual owls, were within their pendant (Fig. 7). When a visitor with his/her pendant first participated in the experiment, an

<sup>†</sup>The number of spots and their locations depend on the target environments, e.g., museums.





**Fig. 6** Spot at Museum of Nature and Human Activities in Hyogo.

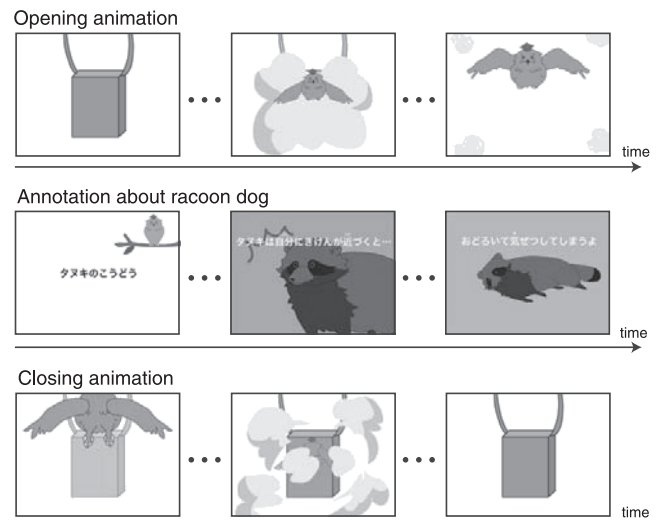


**Fig. 7** Five colorful pendants.

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 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 a spot with the specimen of a 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



**Fig. 8** Opening animation, annotation animation, and closing animation for orange pendant.



Navigation to one destination



Selection from two destinations

**Fig. 9** Navigation patterns for user navigation.

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 sec-

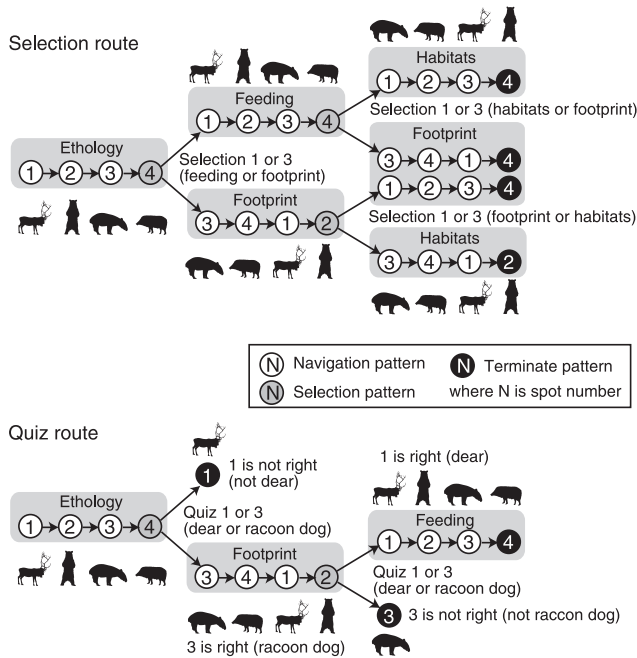


Fig. 10 Two routes. (selection and quiz)

tion 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 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. Each day, more

than 60 individuals or groups took part in the experiment. 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 this paper does not aim at user evaluation, we presented 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 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 at watching the specimens of animals instead of the visual effects, e.g., animations, of agents.

## 6.2 Discussion

From the above experiment, we learned the following:

- 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.
- The experiment provided visitors with two different visitor-guide services. It could provide more than 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 an advantage of our system in comparison with other existing agent-based context-aware systems, because they explicitly and implicitly assume their agents to be 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 the spots that visitors previously visited. However, the durations that they spent at sports were not used in the experiments unfortunately, because the sensing system occasionally missed the presence of visitors. Nevertheless, curators used information about such user behaviors to evaluate the attention of visitors on exhibits, because the information was recorded in a spread-sheet format.
- The experiment provided pre-defined parts. When a visitor registered at our experiment, his/her agent was composed of three pre-defined parts according to his/her initial preference by using by using a GUI-based manipulation based on a compound document-based framework for building pervasive computing presented in our previous paper [13]. Although it enabled developers to defining agents by overwriting built-in classes for these parts, the experiment could continue the experiment without redefining any classes.
- Existing location-sensing systems are not perfect in the



sense that errors in their measurement are unavoidable. The ranges and rates of measurement errors tend to depend on location-sensing systems. There are two typical errors. The first is to miss a tag attached to him/her. It cause for a visitor to stand in front of the displays without receiving any services. The second is to estimate the positions of tags a long distance from their real positions. Proximity-based systems, including active RFID tag systems, may occasionally lose the positions of tags but they seldom estimate incorrect positions. Therefore, the experiment used the RF-Code's Spider system, which is a proximity-based system, instead of any lateration-based tracking systems.

Our experiment at a museum is a case study in our development of ambient computing services in large-scale public spaces. Nevertheless, we could not experiment the scalability of the system in the museum, because it consisted of only four terminals. Nevertheless, we could have positive impression on the availability of the system for large-scale public services. This is because the experiment system could be operated without any centralized management system. The number of agents running or waiting on a single computer was bound to the number of users in front of the computer.

## 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. 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. As a result, they are not suitable for public spaces or for applications that they were not initially designed to support. In fact, there is a 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 public museums, one of the most typical approaches has been to provide with audio annotations from portable audio players. These have required end-users to carry players and explicitly input numbers attached to exhibits in front of them if they 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, [2], the Museum Project [1], the Hippiie system [6], Imogl [5], and Rememberer [2]. 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. Most museums have avoided using such devices because they are too expensive to lend to visitors and they also require regular maintenance, e.g., replacing or recharging the batteries every day. In fact, cost issues represent one of the most serious problems in deploying context-aware services in the real world.

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 and therefore concentrate on operating the PDA buttons or touch panel instead of looking at the exhibits.

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 applicable in practical applications. Of these, the PEACH project [7] has developed a visitor-guide system for use in museums and has evaluated it 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 for enabling agents to migrate between computers [4] by displaying an image of an avatar or character corresponding to the agent on remote computers, but could not migrate agents themselves to computers. Like the PEACH project, several existing systems introduced the notion of agent migration, but they supported only the images of avatars or the codes with a piece of specified 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 Virtual Anatomy Assistant Ritchie [17] 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 any multiple user settings.

We 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 [8].

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. The model represents spatial relationships between physical entities (and places) as containment relationships between their programmable counterpart objects and deploys counterpart objects at computers according to the positions of their target objects or places [12]. 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 [14], whereas we address an implementation of agent-based visitor guide systems and some early experiences on the system in this paper. We also presented a location-aware communication between agents [16], but the system presented in this paper does not use any inter-agent communication.

## 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 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. To support large-scale context-aware systems, it can be managed in a non-centralized manner unlike other existing similar systems.

Finally, we would like to identify further issues that need to be resolved. Some may consider that 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 large 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 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 Sci-

ence), 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

- [1] C. Ciavarella and F. Paterno, "The design of a handheld, location-aware guide for indoor environments," *Personal and Ubiquitous Computing*, vol.8, no.2, pp.82–91, 2004.
- [2] M. Fleck, M. Frid, T. Kindberg, R. Rajani, E. O'Brien Strain, and M. Spasojevic, "From informing to remembering: Deploying a ubiquitous system in an interactive science museum," *IEEE Pervasive Computing* vol.1, no.2, pp.13–21, 2002.
- [3] M. Kruppa, L. Spassova, and M. Schmitz, "The virtual room inhabitant — Intuitive interaction with intelligent environments," *Proc. Advances in Artificial Intelligence (AI'2005), Lecture Notes in Computer Science (LNCS)*, vol.3809, pp.225–234, Nov. 2005.
- [4] M. Kruppa and A. Kruger, "Performing physical object references with migrating virtual characters," *Proc. Intelligent Technologies for Interactive Entertainment, Lecture Notes in Computer Science (LNCS)*, vol.3814, pp.64–73, 2005.
- [5] K. Luyten and K. Coninx, "ImogI: Take control over a context-aware electronic mobile guide for museums," *Workshop on HCI in Mobile Guides, in conjunction with 6th International Conference on Human Computer Interaction with Mobile Devices and Services*, 2004.
- [6] R. Oppermann and M. Specht, "A context-sensitive nomadic exhibition guide," *Proc. Symposium on Handheld and Ubiquitous Computing (HUC'2000), LNCS* vol.1927, pp.127–142, Sept. 2000.
- [7] C. Rocchi, O. Stock, M. Zancanaro, M. Kruppa, A. Kruger, "The museum visit: Generating seamless personalized presentations on multiple devices," *Proc. 9th International Conference on Intelligent User Interface*, pp.316–318, 2004.
- [8] I. Satoh, "SpatialAgents: Integrating user mobility and program mobility in ubiquitous computing environments," *Wireless Communications and Mobile Computing*, vol.3, no.4, pp.411–423, June 2003.
- [9] I. Satoh, "Itinerant agents for network management," *IEICE Trans. Commun.*, vol.E86-B, no.10, pp.2865–2873, Oct. 2003.
- [10] I. Satoh, "A component framework for document-centric networking," *IEICE Trans. Commun.*, vol.E87-B, no.7, pp.1826–1833, July 2004.
- [11] I. Satoh, "A location model for pervasive computing environments," *Proc. IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05)*, pp. 215–224, March 2005.
- [12] I. Satoh, "A location model for smart environment," *Pervasive and Mobile Computing*, vol.3, no.2, pp.158–179, 2007.
- [13] I. Satoh, "Visual components for pervasive computing management," *Proc. IEEE International Conference on Pervasive Services (ICPS'07)*, pp.19–28, July 2007.
- [14] I. Satoh, "Context-aware agents to guide visitors in museums," *Proc. 8th International Conference on Intelligent Virtual Agents (IVA'08), Lecture Notes in Artificial Intelligence (LNAI)*, vol.5208, pp.441–455, Sept. 2008.
- [15] I. Satoh, "A formal approach for milk-run transport logistics," *IEICE Trans. Fundamentals*, vol.E91-A, no.11, pp.3261–3268, Nov. 2008.
- [16] I. Satoh, "Location-aware Communications in Smart Environments," *Information Systems Frontiers*, vol.11, no.5, pp.501–512, 2008.
- [17] V. Wiendl, K. Dorfmueller-Ulhaas, N. Schulz, and E. Andre, "Integrating a virtual agent into the real world: The virtual anatomy assistant ritche," *Proc. Intelligent Virtual Agents (IVA'2007), Lecture*

Notes in Computer Science (LNCS), vol.4722, pp.211–224, Sept. 2007.



**Ichiro Satoh** received his B.E., M.E, and Ph.D. degrees in Computer Science from Keio University, Japan in 1996. From 1996 to 1997, he was a research associate in the Department of Information Sciences, Ochanomizu University, Japan and from 1998 to 2000 was an associate professor in the same department. Since 2001, he has been an associate professor in National Institute of Informatics, Japan. His current research interests include distributed and mobile computing. He received IPSJ paper award, IPSJ

Yamashita SIG research award, and JSSST Takahashi research award. He also received The Young Scientists' Prize of The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology. He is a member of six learned societies, including ACM and IEEE.