Context-aware Deployment of Services in Public Spaces

Ichiro Satoh

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
E-mail: ichiro@nii.ac.jp

Abstract. This paper presents the context-aware deployment of user-assistant services in public spaces, e.g., museums. Using location-sensing systems, it detects the locations of users and deploys user-assistant services, e.g., visitor guides, at computers near to the their current locations. When users move between exhibits in a museum, it enables agents to follow users to annotate the exhibits in personalized form and navigate them to the next exhibits along their routes. To demonstrate the utility and effectiveness of the framework, we constructed and operated a location/user-aware visitor-guide service in a museum as case studies in the development of context-aware services in public spaces.

1 Introduction

The use of user/location-aware services in public spaces, including cities, stations, and museums, has attached much attention from researchers over the past few years. This paper addresses context-aware services to guide visitors in a museum as case studies for developing ubiquitous computing systems for city-wide public spaces. Few visitors in museums have sufficient knowledge about the exhibits. Therefore, they need annotations on these. However, their knowledge and experiences are varied so that they may become puzzled (or bored) if the annotations provided to them are beyond (or beneath) their knowledge or interest. To solve this problem, we construct a context-aware system for providing visitors with services to annotate exhibits in their personalized forms at nearby computers, even when they move between exhibits.

There have been several academic or commercial 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 small-scale laboratory-based experiments. They have been designed in an ad-hoc manner to provide specific single services in particular spaces, i.e., research laboratories and buildings. As a result, they are not suitable for public spaces or for applications that they were not initially designed to support. In addition, they implicitly or explicitly assume centralized management systems, so their scalability could be a serious problem.

We construct a framework for providing context-aware services in a real museum with real users. It provides 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.

2 Approach

Our final goal is to construct a general-purpose infrastructure for providing context-aware services in large public spaces, e.g., building-wide and city-wide spaces. It was inspired by real requirements of museums rather than our academic interests.

2.1 Background

There have been many academic and commercial attempts to provide context-aware services to visitors in public museums. A typical approach has been to provide visitors with audio annotations from portable audio players. These have required end-users to carry players and explicitly input numbers assigned to exhibits if they wanted to listen to audio annotations about the exhibits in front of them. Many academic projects have provided portable multimedia terminals or PDAs to visitors. These have enabled visitors to interactively look at and operate annotated information displayed on the screen of their players, e.g, the Electronic Guidebook [3] and Museum Project [2]. They assume that visitors car carrying portable terminals, e.g., PDAs and smart phones and they are required to explicitly input their positions, the identifiers of exhibits, and items of interest by using user interface devices, e.g., buttons, mice, or touch panels of terminals. However, such operations are difficult for visitors, particularly children, the elderly, and handicapped people, and often prevent them from viewing the exhibits.

To solve this problem, several projects have used sensing systems to detect the positions of visitors, e.g., the Hippie [6], the ImogI [5], and the Rememberer [3]. Portable smart devices, including PDAs, with location sensing systems may be popular in academic projects, but museums tend to avoid using such devices because they are too expensive to lend to visitors and also require regular maintenance, e.g., replacing or recharging the batteries every day. In fact, cost issues are one of the most serious problems in deploying context-aware services at public spaces, including museums. Therefore, several existing approaches, which assume to make use of expensive or delicate devices, will not always be used in real museums, even though they are interesting within academic research communities. In addition, one of the most serious problems associated with portable smart devices in museums is that they prevent visitors from focusing on the exhibits because they tend to become interested in the device rather than the exhibition itself and therefore concentrate on operating the PDA buttons or touch panel instead of looking at the exhibits.

2.2 Requirements

To solve these problems, we should support visitors from stationary sensors and computing devices. We discuss the requirements of visitor guides in museums.

- Visitor-guide services for exhibits should be selected and customized according to the behaviors of users, e.g., the exhibits they looked at, how long they stayed around specific exhibits, and their current locations in addition to knowledge and interest.
- User-assistant services, including visitor-guide services in public spaces, are likely to be accessed often by users. Such services should be executed at nearby computers to minimize communication delays between user-interface devices and server-side computers.
- Visitors move between exhibits in a museum. When he/she moves to another exhibit, his/her agent should be deployed at a computer close to his/her destination by using location-sensing systems.
- Visitor-guide services should be personalized, even when they are provided in public spaces. Services should be provided and interact with users in a personalized manner adapted to individual needs.
- Computers in ubiquitous computing environments often 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 that defines services 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. Our system consists of multiple servers, which are individually connected to other servers in a peer-to-peer manner. Each server only maintains up-to-date information on partial contextual information instead of on tags in the whole space.

2.3 Approach

To meet these requirements, our system uses mobile-agent technology.

- Each mobile agent is a self-contained autonomous programming entity. Our system itself is independent of application-specific services. Instead these services are defined and performed within mobile agents.
- 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.
- Each agent can migrate from computer to computer. When an agent moves to another computer, both the code and the state of the agent are transferred to the destination. After arriving at its destination, an agent can continue working, e.g., on a user-assistant task, without losing results, such as the content of instance variables in the agent's program, at the source computer.
- 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.

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 Deployable Context-aware Agent Platform

Our context-aware system consists of three subsystems: (1) an agent host, (2) context-aware directory servers, called CDSs (Fig. 1), and (3) service-provider agents. The first can execute service-provider agents, where we assume that the computing devices are located at specified spots in public spaces. The second is an autonomous entity that defines application-specific services for visitors. The third is responsible for reflecting changes in the real world and the location of users when services are deployed at appropriate computers. 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 and other agents, which are simultaneously running to provide different services.

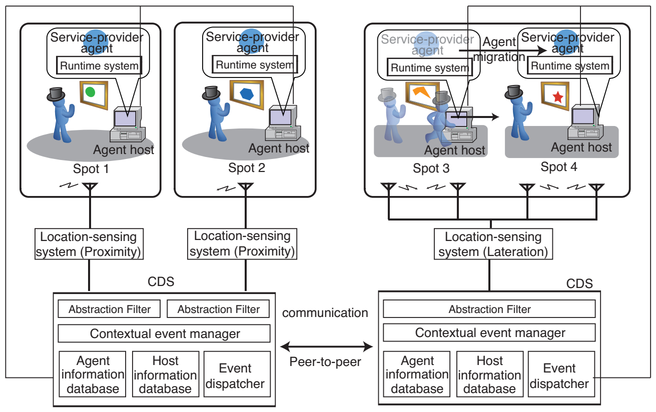

Fig. 1. Architecture of Context-aware Service Provider Agent System.

3.1 Agent host

Each agent host is a computer that can provide visitor-guide services through user-interface devices, e.g., display screens and loudspeakers. It provides a runtime system for executing and migrating agents to other hosts. Each runtime system is built on the Java virtual machine (Java VM), which conceals differences between the platform architectures of the source and destination hosts. 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 host, the runtime system issues specific events to the agent. Some navigation or annotation content, e.g., audio-annotation, should be played without any interruptions. It can exchange agents with another runtime system on a different host through a TCP channel using mobile-agent technology. 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. The host on the receiving side receives and unmarshals the bit stream. Agents may have to acquire various resources, e.g., video and sound, or release previously acquired resources.

3.2 Context-aware agent deployment

Each CDS spatially binds an agent to a user. It maintains two databases. The first stores information about each of the agent hosts 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.

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 provide visitors with tags. These tags are small RF transmitters that periodically broadcast beacons, including the identifiers of the tags, to receivers located in exhibition rooms. The receivers locate the presence or position of the tags. To abstract away differences between the underlying location-sensing systems, CDS maps geometric information measured by sensing systems to specified areas. We assume such areas contain exhibits and computing devices to play annotations. We call the areas *spots*.

When the underlying sensing system detects the presence (or absence) of a tag in a spot, it sends the arrival and departure message to a CDS. The CDS 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. 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 a computing device

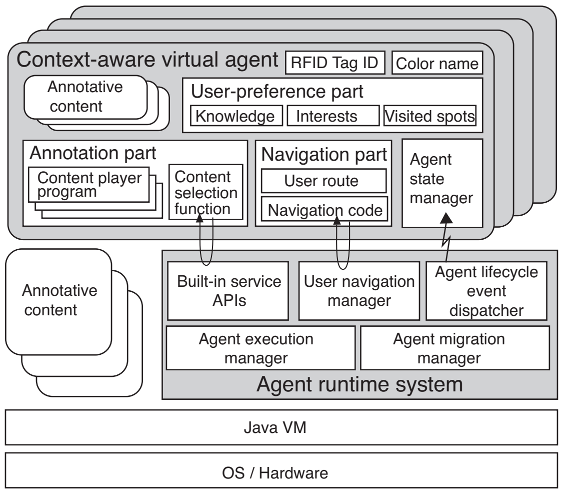

Fig. 2. Architecture of agent host.

4 Context-aware Service-provider Agent

Each agent is attached to at most one visitor and maintains the preference information for its user and programs to provide annotation and navigation to its visitor. To enable agents to be easily developed and configured agents without any professional administrators, we divided each agent into three parts:

- **The user-preference part** maintains and records information about visitors, e.g., knowledge, interests, routes, their name, and durations spent at exhibits they visited.
- **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.

When an agent is deployed at another computer, the runtime system invokes a specified callback method defined in the annotation part and then one defined in the navigation part. Although these parts are implemented as Java objects, they are loosely connected with one another through data attributes by using Java's introspection mechanism so that they can be replaced without any compilations and linkages for their programs. The current implementation uses the standard JAR file format for archiving these parts because the format can support digital signatures, enabling authentication. Each agent keeps the identifier of the tag attached to its visitor. Each agent can specify a requirement that its destination hosts must satisfy in CC/PP form and the runtime system can select an appropriate destination among multiple destination candidates through a comparison between the capabilities required by agents and the capabilities of the candidates.

4.1 User-preference part

This is responsible for maintaining information about a visitor. In fact, it is almost impossible to accurately infer what a visitor knows or is interested in from data that are measured by sensing systems. Instead, the current implementation assumes that administrators will explicitly ask visitors about their knowledge and interests and manually input the information into this part. Nevertheless, it is still possible to make a qualified guess with some probability as to what a visitor may be interested in, if we know which spots he/she visited, how many he/she visited, and how long he/she visited. Each agent has a mechanism to automatically record the identifiers, the number of visits to, and length of stays at spots by visitors. This part is implemented as a hash-table for maintaining the collection of data entries. Each entry is a pair of a name and a value, where the former is a string data and the latter is an arbitrary data structure represented as Java objects. The second and third parts can access entries with key names so that these parts can be combined loosely and replaced by compatible parts.

4.2 Annotation part

Each agent is required to select annotations according to the current spot and route in addition to the information stored in the user-preference part and play the content in its user's personalized form. This part defines a content selection function and a set of programs for playing the selected content. The function maps more than one argument, e.g., the current spot, the user's selected route, and the number of times a user has visited the spot into a URL referring to the annotative content. The content can be stored in the agent, the current agent host, 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. Such content is provided in a variety of multimedia representations, 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. The current implementation can divide the part into three sub-parts: opening, annotation, and closing, which are played by turns.

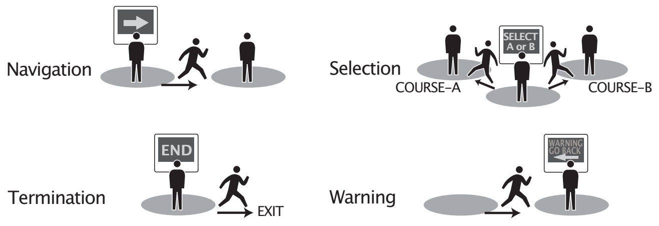

Fig. 3. User-navigation patterns.

4.3 Navigation part

Our agents are required to navigate visitors to their destinations along routes recommended by museums or the 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 displays (or from loudspeakers) of the current agent host. For example, the agents display the 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 primitive and natural behaviors in museums. This part provides the four navigation patterns, outlined in Fig. 3.

– *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.
– *Termination* informs users that they have arrived at the final 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 for specifying the itineraries of mobile agents for network management [11] and they are stored in the user-preference part. 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 host and the locations of the destinations, as well as the direction to the current host's screen. The current implementation permits administrators to manually input the directions of possible destinations and the direction to the screen. Agent hosts provide 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 host returns the direction to the destination to it or displays the direction on the screen on its behalf.

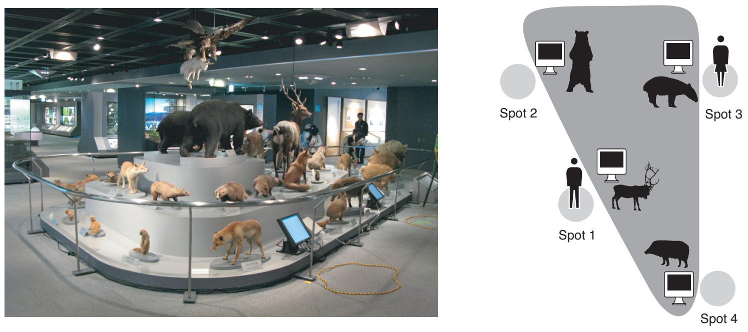

Fig. 4. Experiment at Museum of Nature and Human Activities in Hyogo.

5 Experience

To prove the utility of the propose system, we constructed and operated an experiment at the Museum of Nature and Human Activities in Hyogo, Japan, using the proposed system. Figure 4 has a sketch that maps the spots located in the museum. The experiment was carried out at four spots in front of specimens of stuffed animals, i.e., a bear, deer, racoon dog, and wild boar. Each spot could provide five different pieces of animation-based annotative content about animal, e.g., its ethology, footprints, feeding, habitat, 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. 5.

When a visitor first participated in the experiment, an operator input the point of interest and the route for the new visitor and created his/her service-provider agent. As shown in Fig. 6, an agent tied to a pendant played the opening animation and then played the annotation. It next plays the closing animation.

We simultaneously provided two kinds of routes for visitors to evaluate the utility of our user-navigation supports. Both routes navigated visitors to destination spots along the way (Fig. 7). They made each visitor go around an exhibit booth consisting of four spots two or three times, as shown on the right of Fig. 4. That is, a visitor might visit the same spots two or three times depending on the navigation of their agents. 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

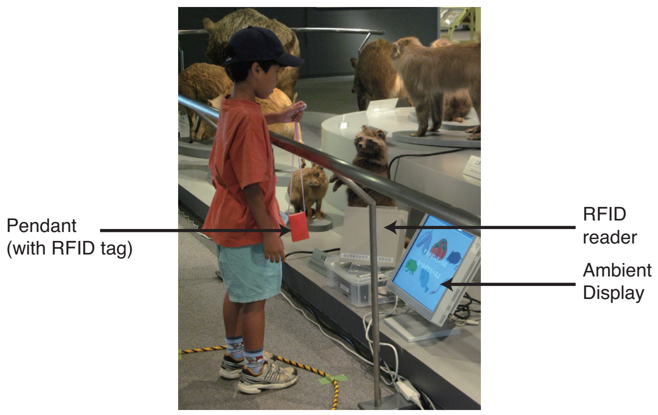

Fig. 5. Spot at Museum of Nature and Human Activities in Hyogo.

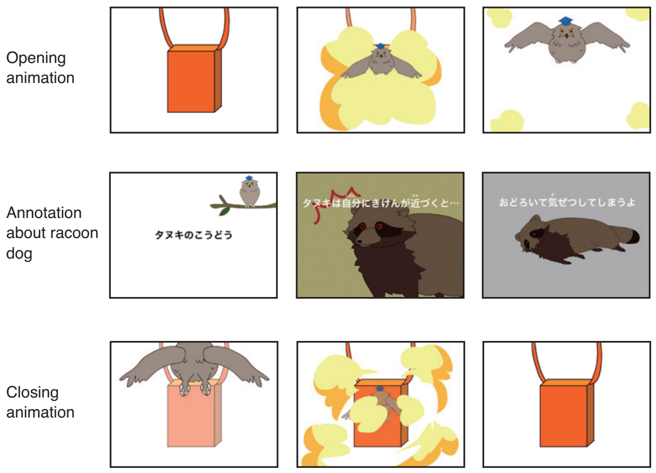

Fig. 6. Opening animation, annotation animation, and closing animation for orange pendant

to their answers in specified spots at specified timings. Both the experiments offered visitors animation-based annotative content about the animal in front of them so they could learn about the animal while observing the corresponding specimen.

The experimental system consisted of one CDS and four agent hosts. It enabled curators to configure annotation content through a GUI-based monitoring and configuration for agents (right of Fig. 8) and to operate the assignment of annotation to visitors by using a Web browser running on a portable terminal (Apple iPod Touch) equipped with a WiFi interface (left of Fig. 8).

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 an agent host contained in the spot. After arriving at the host, the runtime system invoked a specified callback method defined in the annotative part of the agent. The method first played the opening animation defined in 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 of an agent and starting its opening animation at the destination

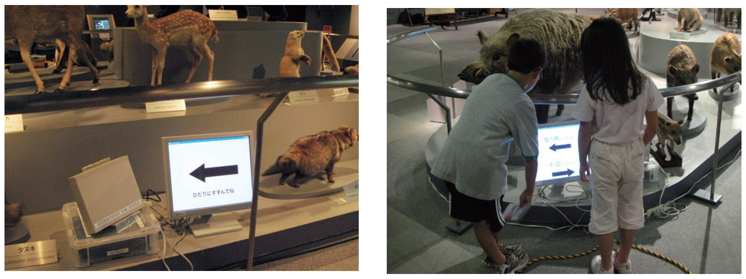
Navigation to one destination
Selection from two destinations

Fig. 7. Navigation patterns for user navigation at Museum of Nature and Human Activities in Hyogo.

after visitors arrived at a spot was within two seconds, so that visitors could view the opening animation soon after they stood in front of 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 agent host. 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, which was maintained within it. Furthermore, if a user left out his/her route, the navigation part invoked a method to play warning content to return him/her to his/her previous spot.

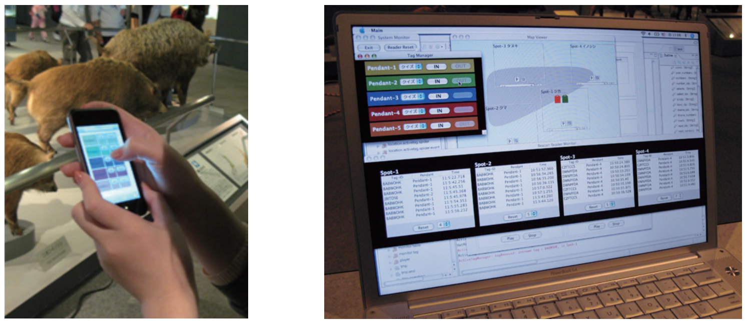
Portable administration terminal (iPod touch)
Terminal for monitoring the positions of visitors and customizing agents

Fig. 8. Portable management terminal (left figure) and GUI-based system for monitoring and configuring agents (right figure).

We operated 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 genders and ages. Almost all the participants (more than 95 percent) had positive feedback on the system.

Their typical feedback were “We were very interested in or enjoyed the system.”, “We could easily answer to the quizzes by our moving between the spots.”, and “We gained detail knowledge about the animals with our watching them in front of our standing positions.” As application-specific services could be defined and encapsulated within the agents, we were able to easily change the services provided by modifying the corresponding agents while the entire system was running and more than two different visitor-guide services could also be simultaneously supported for visitors. Even while visitors were participating, curators with no knowledge of context-aware systems were able to configure the annotative content by doing drag-and-drop manipulations using the GUI-based configuration system. Such dynamic configuration is useful, because museums need to provide and configure services with visitors without any stopping.

6 Related Work

As we discussed in Section 2, there have been many attempts to provide visitor-guide systems in museums, but most existing projects assume that visitors carry smart terminals. On the other hand, there have been several research attempts on smart spaces equipped with stationary sensors and terminals. Cambridge University’s Sentient Computing project [4] provides a platform for location-aware applications using infrared-based or ultrasonic-based locating systems in a building. Using the VNC system [7], the platform can track the movement of a tagged entity, such as individuals and things, so that the graphical user interfaces of the user’s applications follow him/her while he/she moves around. Although the platform provides similar functionality to that of our framework, its management is centralized and their services are executed in centralized servers. Microsoft’s EasyLiving project [1] enabled services running on different computers to be combined dynamically according to contextural changes in the real world, but aimed at private spaces, e.g., living rooms. It could not deploy services at different computers.

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 [9]. 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 some basic evaluation on the usability of mobile agent-based services in public museums in our another paper [13].

7 Conclusion

We designed and implemented an agent-based system for building and operating context-aware visitor-guide services in public 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 watched. His/her agent can also navigate him/her to exhibits along his/her route. To support large-scale context-aware systems, the system is managed in a non-centralized manner. Using the system, we constructed and operated location/user-aware visitor-guide services at a museum as case studies in our development of ambient computing services in public spaces.

References

1. B.L. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer: EasyLiving: Technologies for Intelligent Environments, Proceedings of International Symposium on Handheld and Ubiquitous Computing, pp.12-27, 2000.
2. 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.
3. M. Fleck, M. Frid, T. Kindberg, R. Rajani, E. O'BrienStrain, E. 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.
4. A. Harter, A. Hopper, P. Steggeles, A. Ward, and P. Webster: The Anatomy of a Context-Aware Application, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 59-68, ACM Press, August 1999.
5. K. Luyten and K. Coninx, ImogI: Take Control over a Context-Aware Electronic Mobile Guide for Museums, In 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, Proceedings Symposium on Handheld and Ubiquitous Computing (HUC'2000), LNCS vol.1927, pp.127-142, Springer, September 2000.
7. Richardson T, Stafford-Fraser Q, Wood K, Hopper A. Virtual Network Computing. IEEE Internet Computing 1999; 2(1):33-38.
8. C. Rocchi , O. Stock , M. Zancanaro , M. Kruppa , A. Kruger: The Museum Visit: Generating Seamless Personalized Presentations on Multiple Devices, Proceedings of 9th international conference on Intelligent User Interface, pp.316-318, ACM Press, 2004.
9. 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, John Wiley, June 2003.
10. I. Satoh: A Location Model for Pervasive Computing Environments, Proceedings of IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05), pp,215-224, IEEE Computer Society, March 2005.
11. I. Satoh: Building and Selecting Mobile Agents for Network Management, Journal of Network and Systems Management, vol.14, no.1, pp.147-169, Springer, 2006.
12. I. Satoh: A Location Model for Smart Environment, Pervasive and Mobile Computing, vol.3, no.2, pp.158-179, Elsevier, 2007.
13. I. Satoh: Context-aware Agents to Guide Visitors in Museums, to appear in in Proceedings of 8th International Conference on Intelligent Virtual Agents (IVA'08), LNCS, September 2008.