Mobile Agents for Digital Signage

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 an agent-based framework for building and operating context-aware multimedia content on digital signage in public/private spaces. It enables active and multimedia content to be composed from mobile agents, which can travel from computer to computer and provide multimedia content for advertising or user-assistant services to users. The framework automatically deploys their agents at computers near to their current positions to provide advertising or annotations on objects to users. To demonstrate the utility of the framework, we present a user-assistant that enables shopping with digital signage.

1 Introduction

Digital signage has been expected to play an important role in advertising, navigation, assistance, or entertainment in public and private environments, such as public museums, retail stores, and corporate buildings. Advertising using digital signage should be a form of out-of-home advertising (OOH) in which content and messages are displayed on digital signs with the common goal of delivering targeted messages to specific consumers in specific locations at specific times. For example, several retailers offer digital signage for sales promotions. The content that can be displayed on digital signage can be anything, including text, images, animations, video, audio, and interactivity. It has frequently been argued that digital signage needs to rely on quality content if it is to work effectively. Context-awareness is useful for digital signage. For example, the content for sales promotion on digital signage should be selected according to its target customers and displayed at terminals close to where they are currently located.

- Content, which is played on digital signage, can be anything, including text, images, animations, video, audio, and interactivity. We should provide a variety of multimedia content, including interactive content, to consumers, from digital signage.
- The content for sales promotion on digital signage should be selected according to its target items and customers and displayed at terminals that are close to the customers or items.
- Digital signage has its own spatial scope where people can see or listen to its content. User-specific content should be played only when its users are in its scope.
- Digital signage should be used for not only sales promotion but also to manage product life cycles, e.g., shipment, showcasing, assembly, usage, and disposal.

– As context-aware digital signage should be provided in extensive building and city-wide areas, it cannot be managed by using traditional approaches, such as those that are centralized and top-down.

We constructed a general-purpose framework for managing context-aware annotation services [19]. The previous framework was designed for context-aware visitor assistance in museums. To support large-scale context-aware systems, it was managed in a non-centralized manner. It used mobile agent technology, where mobile agents were autonomous programs that could travel from computer to computer under their own control. Mobile agents were used as deployable services without any centralized management systems.

The framework presented in this paper is constructed based on the previous framework but it is designed for digital signage and can model the locations of people, products, and digital signage systems. This is because content displayed on digital signage needs to be selected and displayed according to people and physical entities. The framework needs to maintain a location model to manage digital signage. It introduces virtual counterpart objects for digital representations of people, physical entities, or digital signage systems, where each virtual counterpart object is a programmable entity. An application cannot directly interact with people or physical entities but with their virtual counterpart objects. Their counterpart objects interact with one another on behalf physical entities. The framework spatially binds the positions of people or entities with the locations of their virtual counterparts and, when they move from location to location in the real world, it automatically migrates their counterpart objects from the counterpart object corresponding to the source location to the counterpart object corresponding to the destination location.

Several researchers on virtual-reality (VR) have provided the notion of virtual scope, often called *aura*, where interactions between two objects in VR become possible only when the objects' scopes collide or overlap [4, 7]. Digital signage can be seen by people when the former and latter are within a specified scope. For example, a public terminal has a half-meter sphere so that a user can directly manipulate the terminal. User-aware content should be displayed at digital signage only when the target user is in front of the digital signage. The framework introduces the scope of digital signage as a virtual counterpart. When a user is within the scope, his/her virtual counterpart is located in the virtual counterpart corresponding to the scope.

There have been several commercial projects for providing context-aware content on digital signage, but they have been constructed based in an ad-hoc manner. However, several researchers have explored context-aware services independently of the literature of digital signage. Cambridge University's Sentient Computing project [8] provided a platform for location-aware applications using infrared-based or ultrasonic-based locating systems in a building. Microsoft's EasyLiving project [3] enabled services running on different computers to be combined dynamically according to contextural changes in the real world. Here, we discuss differences between the framework presented in this paper and our previous frameworks. We constructed a location model for ubiquitous computing environments. The model 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 [16]. 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 versions of this paper [19, 20], whereas this paper addresses agent-based advertising on digital signage for shopping.

2 Design and Implementation

This section describes the current implementation of our framework.

2.1 Basic approach

Like our previous framework [19], the framework introduces mobile agent technology. Content for digital signage is defined in mobile agents so that they can be dynamically deployed at computers close to users according to contexts in the real world, e.g., the locations of users and physical objects by using locating systems. Each mobile agent is a programmable entity with stored data. Therefore, each mobile agent-based services can define programs to play its visual/audio content and interact with users inside it. Therefore, the framework itself is independent of application-specific tasks and provides multiple kinds of multimedia content, because such tasks are performed within mobile agents.

Computers in digital signage only have limited resources, such as restricted levels of CPU power and amounts of memory. Mobile agents can help to conserve these limited resources, since each agent needs to be present at the computer only while the computer needs the content provided by that agent. After arriving at its destination, a mobile agent can continue work without losing the results of working, e.g., the content of instance variables in the agent's program, at the source computers. Therefore, users can continue to watch or listen to content from computers close to their current positions, even when the users move from location to location.

Existing location models can be classified into two types: physical-location and symbolic-location models [1, 2, 9]. The former represents the position of people and objects as geometric information. A few outdoor-applications like moving-map navigation can easily be constructed on the former. Most emerging applications, on the other hand, require a more symbolic notion, i.e., place, where place is the human-readable labeling of positions, e.g., the names of rooms and buildings. This paper addresses symbolic location as an event-driven programming model for context-aware digital signage. The model is maintained as a tree structure of virtual counterpart objects corresponding to people, physical entities, and digital signage according to containment relationships in the real world.

Digital signage has its own scope where people can see or listen to its content and it should be activated when people are with its scope. We introduce such a scope as virtual space, like the *aura* studied in virtual-reality research. Each digital signage can have a virtual counterpart object corresponding to the scope where people can see it in addition to the virtual counterpart corresponding to it. When a user is within the scope of a digital signage, his/her virtual counterpart object is located in the virtual

counterpart object corresponding to the scope and the latter activates the digital signage via the virtual counterpart object corresponding to the digital signage.

2.2 System structure

The framework consists of three parts: (1) mobile agents, (2) agent runtime systems, (3) location model management systems (called LMMSs), and (4) location information servers (called LISs) (Fig. 1). The first offers application-specific content, which are attached to physical entities and places, as collections of mobile agents. The second runs on digital signage and is responsible for executing and migrating mobile agents. The third maintains the containment relationship of virtual counterparts to model the location of people or physical entities, e.g., products. The fourth provides a layer of indirection between the underlying location sensing systems and mobile agents. Each LIS manages more than one sensor and provides the agents with up-to-date information on the state of the real world, such as the locations of people, places, and things, and the destinations that the agents should migrate themselves to.

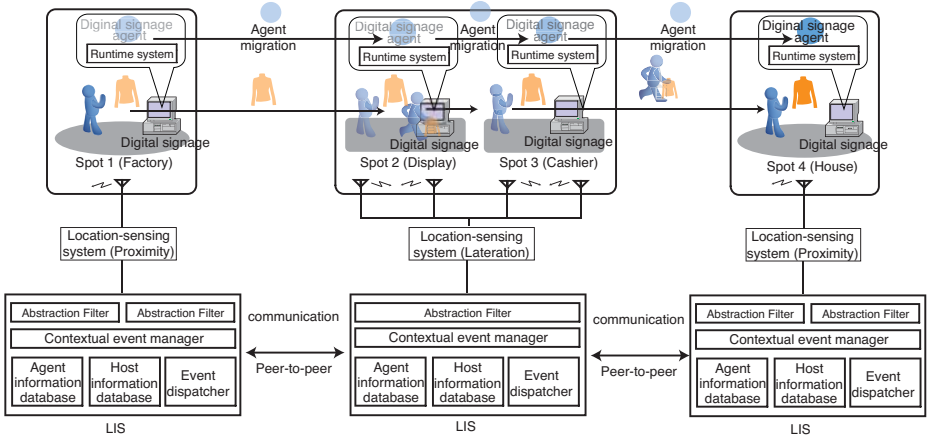

Fig. 1. Architecture

2.3 Location information server

The framework should be independent of its underlying location sensing systems. Location-sensing systems can be classified into two types: tracking and positioning systems. The former, including RFID tags, measures the location of other objects. The latter, including GPS, measures its own location. Since it is almost impossible to support all kinds of sensors, the model aims at supporting various kinds of tracking sensors, e.g., RFID-, infrared-, or ultra-sonic tags and computer vision, as much as possible. Each LIS can

have a mechanism for managing location-sensors outside itself so that it is designed independently of sensors. It transforms geometric information about the positions of objects into corresponding containment relations. Each LIS is also responsible for discovering agents attached to people or physical entities. When an LIS detects a new person or physical entity by using its sensing system, it sends a query message about agents attached to the person or entity to all LMMSs in its current sub-network by using UDP multicast communication. When an LMMS returns a reply message to the LIS, the LIS sends a control message to migrate the agents to a virtual counterpart corresponding to digital signage close to the current location of the new person or entity that is visting.

2.4 Agent runtime system

The framework assumes that each digital signage supports a runtime system. Each runtime system 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 is responsible for executing agents. For example, 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.

Each runtime system running on digital signage has its own virtual counterpart agent (Fig. 2). The agent is located at a virtual counterpart corresponding to a space in a tree structure maintained by an LMMS according to the location of the signage in the real world. When it receives a mobile agent for content, it forwards the agent to its target digital signage and then the computer of the signage executes the visiting agent.

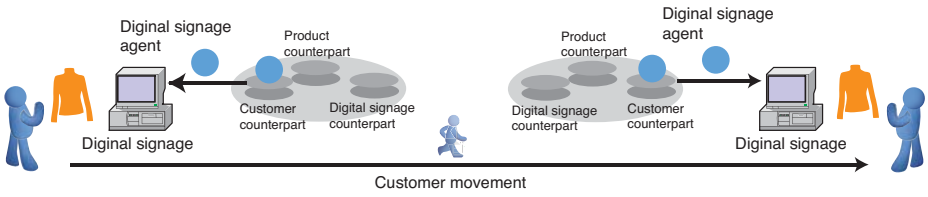

Fig. 2. Forwarding agents to digital signage when user moves.

When an agent is transferred to digital signage 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. Arriving agents may explicitly have to acquire various resources, e.g., video and sound, or release previously acquired resources.

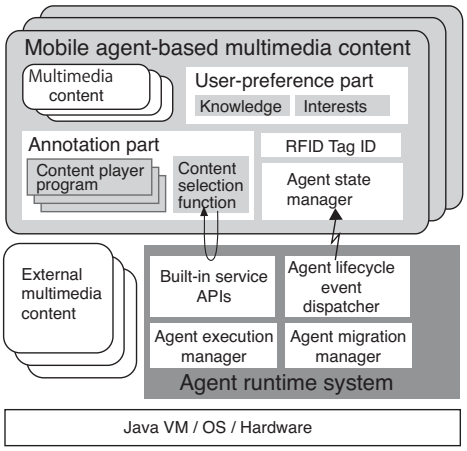

Fig. 3. Architecture of runtime system for service-provider agent .

2.5 Location model management system

To model the locations of entities or places in the real world, we introduce a tree structure of virtual counterpart objects. Each counterpart object can be contained within at most one counterpart object according to containment relationships in the real world. Each counterpart object can move between other counterpart objects. In the current implementation, each virtual counterpart object is constructed as a mobile agent with the notion of hierarchical mobile agents [14]. A mobile agent migrates to another agent, which may be running at a different computer, as a whole with all its inner agents. When physical entities move from location to location in the real world, an LIS detects their movements through location-sensing systems and changes the containment relationships of agents corresponding to moving entities, their source, and destination.

The tree structure of virtual counterpart objects can be maintained in more than one computer. Each runtime system enables counterpart objects to be organized in a subtree structure. We also introduce a link object, which is used as a proxy for a subtree that its target computer maintains and is located in the subtree that another computer maintains. As a result, it attaches the former subtree to the latter.

As mentioned previously, digital signage has its own scope where people can see or listen to its content. We introduce such a scope as virtual space, like *aura*. The framework allows each virtual space to define its shape and size. For example, a public terminal has a half-meter sphere so that a user can directly manipulate the terminal. However, there is a gap between the scope and the space where the underlying location sensing systems can detect the presence of the user. For example, such a scope may be smaller than the coverage areas of location sensing systems. Therefore, the current implementation introduces such a scope as the coverage area of a location sensing system.

2.6 Mobile agent for digital signage content

The content for digital signage and its virtual counterpart are implemented as mobile agents. Each mobile agent for providing content is attached to at most one product or user and defines programs that provide annotation and navigation to him/her (Fig. 3). To support user/location-dependent content, each agent is dynamically assembled from the *content* and *user-preference parts*.

Content part: This part is responsible for selecting and playing annotations according to users and products in addition to the information stored in the user-preference part and it plays the content in the personalized form of its user. It is defined as a set of a content-selection function and programs for playing the selected content.

The function maps more than one argument, e.g., the users and products into a URL referring to the annotative content. The content can be stored in the agent, the current 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.

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, 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.

User-preference part: This part 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 have been 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.

2.7 Current status

This section describes the current implementation of our system. It was implemented using Sun's Java Developer Kit version 1.5 or later versions.

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.

Each LIS for the RFID tag system manages multiple sensors that detect the presence of tags and maintains up-to-date information on the identities of tags that are within the zone of coverage of its sensors. This is achieved by polling sensors or receiving the events issued by the sensors themselves. An LIS does not require any knowledge of other LISs. To conceal the differences among the underlying locating systems, each LIS maps low-level positional information from each of the locating systems into information in a symbolic model of location. An LIS represents an entity's location, called a *spot*, e.g., a space of a few feet, which distinguishes one or more portions of a room or building. When an LIS detects a new tag in a spot, it multicasts a query that contains the identity of the new tag and its own network address to all the agent runtime systems in its current sub-network to discover agents tied to the tag. When there are multiple candidate destinations, each of the agents that is tied to a tag can select one destination on the basis of the profiles of the destinations. When the absence of a tag is detected in a spot, each LIS multicasts a message with the identifier of the tag and the identifier of the spot to all runtime systems in its current sub-network.

Security and privacy: The framework only maintains per-user profile information within those agents that are bound to the user. It promotes the movement of such agents to appropriate hosts near the user in response to the user's movements. Thus, the agents do not leak profile information on their users to other parties and they can interact with their mobile users in personalized form that has been adapted to respective, individual users. The runtime system can encrypt agents to be encrypted before migrating them over a network and then decrypt them after they arrive at their destination. Moreover, since each mobile agent is just a programmable entity, it can explicitly encrypt its particular fields and migrate itself with these fields and its own cryptographic procedure. The Java virtual machine can explicitly restrict agents to only access specified resources to protect hosts from malicious agents. Although the current implementation cannot protect agents from malicious hosts, the runtime system supports some authentication mechanisms for agent migration so that each agent host can only send agents to and only receive them from trusted hosts.

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 host to a destination host that was recommended by the LISs. The latency in 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 the two hosts over a TCP connection was 38 ms and 480 ms. This evaluation was operated with three computers (Intel Core 2 Duo 2 GHz with Windows XP Professional and JDK 1.5) connected via a Fast Ethernet. This cost is reasonable for migrating agents between computers to that follow visitors moving between exhibits.

3 Early Experience

We experimented and evaluated mobile agent-based active content for appliances, e.g., electric lights. This was unique among other existing active content because it did not support advertising for its target appliance but assisted users to control and dispose of the appliance. We attached an RFID tag to an electric light and provided a mobile agent as an active content for the light. The content was attached to its target item and was deployed at computers close to the current position of the item. An agent for managing active content on its target appliance is created when the appliance was shipped from its factory.

Since the agent defines programs to display three kinds of active content content inside it, it selects them according to their spaces. It supports the lifecycle of the item from shipment, showcasing, assembly, usage, and disposal.

- **In warehouse:** While the light is in a warehouse, its virtual counterpart agent is deployed at digital signage in the warehouse. It notifies a server in the warehouse of its specification, e.g., its product number, serial number, the date of its manufacture, and its size and weight.
- **In store:** While the light is being showcased in a store, its counterpart agent is deployed at a computer close to its target object to display advertising content to encourage customers who are visiting the store to buy it. Figure 4 a) and b) are two images maintained in the agent and they display the price, product number, and manufacture's name on its current computer.
- **In house:** After the light has been bought and taken to the house of its new owner, its agent migrates to digital signage in the house and illustrates how to assemble it. Figure 4 c) is the active content for the assembly manual. The agent also illustrates how to use it, as shown in Figure 4 d). When it is disposed of, the agent shows its active content to assist disposal guide. Figure 4 e) illustrates how to dispose of the appliance.

Fig. 4. Digital signage for supporting appliance

In a house-setting, we can define agents that control appliances, which may not have any network interfaces. In both of the approaches we have described here, the lights are controlled by switching their power sources on or off through a commercial protocol, called X10.

The first can autonomously turn room lights on with a tagged user is sufficiently close to them. The agent attached to the light can also work as our X10-based server's client and runs on the stationary runtime system in the room. When a tagged user approaches a light, an LIS in the room detects the presence of his/her tag in the cell that contains the light. The LIS then moves the agent that is bound to his/her tag to the runtime system on which the light's agent is running. The user's agent then requests the lights' agent to turn the light on through inter-agent communication.

The second allows us to use a PDA to remotely control nearby lights. In this system, place-bound controller agents, which can communicate with X10-base servers to switch lights on or off, are attached to places with room lights. Each user has a tagged PDA, which supports the runtime system with WindowsCE and a wireless LAN interface. When a user with a PDA visits a cell that contains a light, the framework moves a controller agent to the runtime system of the visiting PDA. The agent, now running on the PDA, displays a graphical user interface to control the light. When the user leaves that location, the agent automatically closes its user interface and returns to its home runtime system.

4 Conclusion

We designed and implemented a context-aware infrastructure for building and managing mobile agent-based content displayed on digital signage, where mobile agents are autonomous programs that can travel from computer to computer under their own control as virtual counterpart objects for people or physical entities. It provides users and physical entities with mobile agent-based content to support and annotate them. Using location-tracking systems, it can migrate content to stationary or mobile computers near the locations of users and physical entities to which the agents are attached. That is, it allows a mobile user to access its personalized services in an active computing environment and provides user/location-dependent active content to a user's portable computer or stationary computer. It is managed in a decentralized manner. In addition, the system is managed in a non-centralized manner to support large-scale context-aware systems. Using the system, we constructed and operated two applications of location/user-aware multimedia on digital signage as case studies in our development of ambient computing services in public spaces.

Acknowledgement This research is supported by Promotion program for Reducing global Environmental loaD through ICT innovation (PREDICT), Ministry of Internal Affairs and Communications of Japan.

References

1. C. Becker, Context-Aware Computing, Tutorial Text in IEEE International Conference on Mobile Data Management, (MDM'2004), Januray 2004.

2. M. Beigl, T. Zimmer, C. Decker, A Location Model for Communicating and Processing of Context, Personal and Ubiquitous Computing, vol. 6 Issue 5-6, pp. 341-357, Springer, 2002.
3. 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.
4. C. Carlsson, and O. Hagmnd, DIVE: A platform for multi-user virtual environments Computer and Graphics. vol. 17, no. 6, pp. 663-669, 1993.
5. 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.
6. 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.
7. C. Greenhalgh and S. Benford, MASSIVE: A Collaborative Virtual Environment for Teleconferencing ACM Transactions on Computer-Human Interaction, vol 2, no. 3, September 1995.
8. 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.
9. U. Leonhardt, and J. Magee, Towards a General Location Service for Mobile Environments, Proceedings of IEEE Workshop on Services in Distributed and Networked Environments, pp. 43-50, IEEE Computer Society, 1996.
10. 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.
11. 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.
12. Richardson T, Stafford-Fraser Q, Wood K, Hopper A. Virtual Network Computing. IEEE Internet Computing 1999; 2(1):33-38.
13. 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.
14. I. Satoh, MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, Proceedings of International Conference on Distributed Computing Systems (ICDCS'2000), pp.161-168, IEEE Computer Society, April 2000.
15. 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.
16. 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.
17. 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.
18. I. Satoh: A Location Model for Smart Environment, Pervasive and Mobile Computing, vol.3, no.2, pp.158-179, Elsevier, 2007.
19. I. Satoh: Context-aware Deployment of Services in Public Spaces, in Proceesings of 6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS'2008), Lecture Notes in Computer Science (LNCS), vol.5287, pp.221-232, September 2008.

20. I. Satoh: A Context-aware Service Framework for Large-Scale Ambient Computing Environments, in Proceedings of ACM International Conference on Pervasive Services (ICPS'09), pp.199-208, ACM Press, July 2009.