A Location Model for Pervasive Computing Environments

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

Abstract

This paper presents a world model for location-aware and user-aware services in ubiquitous computing environments. It can be dynamically organized like a tree based on geographical containment, such as user-room-floor-building, and each node in the tree can be constructed as an executable software component. The model is unique to existing approaches because it can be managed by multiple computers in an ad-hoc manner and it can provide a unified view of the locations of not only physical entities and spaces, including users and objects, but also computing devices and services. A prototype implementation of this approach was constructed on a Java-based mobile agent system. This paper presents the rationale, design, implementation, and applications of the prototype system.

1 Introduction

Location is an essential part of contextual information, which has turned out to be useful in many applications, particularly those for determining position, navigation, routing, tracking, logistics, and monitoring of pervasive computing devices. In fact, a variety of location-based services have been investigated thus far, but most existing services inherently depend on particular sensing systems, such as GPSs and RFID-tags, and have inherently been designed for their initial applications.

A solution to this problem would be to provide a location model for pervasive computing services. Although several researchers have explored such models, most existing models are not available for all pervasive computing, because these need to be maintained in centralized database systems, whereas the environments are often managed in an ad-hoc manner without any database servers. Although they have been only aimed at maintaining the locations of people and objects in the physical world, the locations of computing devices and software that define services are often required in pervasive computing. Therefore, we need a general location model that can be used in pervasive computing environments and that can specify logical and physical entities in a unified manner. This paper focus is on discussing the construction of such a model, called *M-Spaces*, as a programming interface between location-sensors and application-specific services in pervasive computing environments.

In the remainder of this paper, we outline an approach to building and managing location-based and personalized information services in pervasive computing environments (Section 2), the design of our framework (Section 3), and an implementation of the framework (Section 4). We describe some experience we have had with several applications, which we used the framework to develop (Section 5). We briefly review related work (Section 5), provide a summary, and discuss some future issues (Section 6).

2 Background

This paper proposes a location model for managing location-based and personalized services in indoor settings, e.g., building and houses, rather than outdoor ones.

2.1 Requirements

Pervasive computing environments have several unique requirements as follows:

- **Mobility:** Not only entities, e.g., physical objects and people, but also computing devices can be moved from location to location. Our location model is required to be able to represent mobile computing devices and spaces as well as mobile entities. Furthermore, it needs to be able to model mobile spaces, e.g., cars, which may contain entities and computing devices.
- **Heterogeneity:** A pervasive computing environment consists of heterogeneous computing devices, e.g., embedded computers, PDAs, and public terminals. Location-based and personalized services must be executed at computing devices whose capabilities can sat-

isfy the requirements of the services. The model is required to maintain the capabilities of computing devices as well as their locations.

- **Availability:** Pervasive computing devices may have limited memories and processors, so they cannot support all the services that they need to provide. Software must be able to be deployed at computing devices only while they are wanted. The model should be able to manage the (re)location of service-provider software.
- **Absence of centralized databases:** Since pervasive computing devices are organized in an ad-hoc and peer-to-peer manner, they cannot always access database servers to maintain location models. The model should be available without database servers enabling computing devices to be organized without centralized management servers.

There have been many attempts to construct location models for pervasive computing environments, e.g., NEXUS [12, 2], Cooltown [13], RAUM [4] Sentient Computing [11], EasyLiving [5], and Virtual Counterpart [19]. However, unfortunately most of these have been inherently designed for particular location sensing systems or application-specific services and need centralized database servers outside them to maintain their location models. Therefore, we need to construct a new location model for pervasive computing environments.

2.2 Design Principles

Existing location models can be classified into two types: physical-location and symbolic-location [3, 4, 15]. 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: place. Generically, place is the human-readable labeling of positions, e.g., the names of rooms and buildings. An object contained in a volume is reported to be in that place. This paper addresses symbolic location as an event-driven programming model for pervasive computing environments. For example, when people enter a place, services should be provided from their own portable terminal or their own stationary terminals should provide personalized services to assist them. Our model also introduces a containment relationship between spaces, because physical spaces are often organized in a containment relationship. For example, each floor is contained within at most one building and each room is contained within at most one floor. Our model also has the following features:

Virtual counterparts: It introduces the notion of counterparts as digital representations of physical entities or spaces. An application does not directly interact with physical objects and places, but with their virtual counterparts. The model spatially binds the positions of entities and spaces with the locations of their virtual counterparts and, when they move in the physical world, it deploys their counterparts at proper locations within it.

Unified view: The model can maintain the locations and capabilities of computing as well as those of physical entities and services. It also manages the deployment of application-specific services according to changes in the locations of physical entities, spaces, and computing devices. That is, the model does not distinguish between physical entities, spaces, computing devices, including self-maintaining computers, or application-specific services.

Sensor-independence: 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. As the model can have a mechanism for managing location-sensors outside itself, it has been designed independently of sensors.[1]

Extensibility: The model can be managed by one or more computers, which may also offer application-specific services. It provides a demand-driven mechanism, which was inspired by ad-hoc mobile networking technology [17], that discovers the computing devices and services that are required. It also enables service-provider software to be dynamically deployed at computing devices, but only when the software is wanted.

Local Interaction: People often want to communicate with someone in front of them rather than with people in another room. Services running on a device should be valid within the bounding region surrounding the device, limiting their presence in space. An entity, including a person or the computing device's surrounding scope can be a medium, e.g., visual, audio, or hand-manipulation, which enables it to interact with devices. The model enables each entity to specify the scope within which it can receive services running on the device according to the media between the entity and the device.

3 World Model

This section presents a symbolic model for location-based and personalized services in pervasive computing environments.

[1]The model transforms geometric information about the positions of objects into corresponding containment relations.

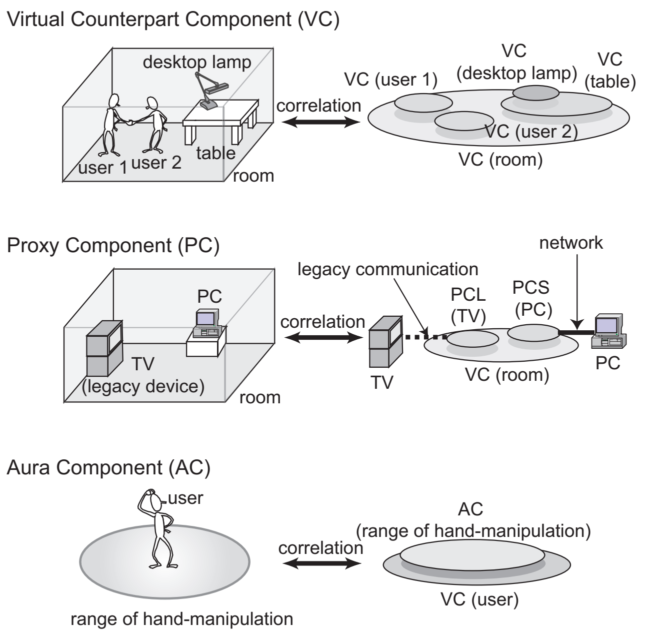

Figure 1. Four types of components (VC, PC, AC, and SC)

3.1 Containment Relationship Model

Our model consists of software elements, called components, which are not only digital representations of people, objects, or spaces in the physical world but also proxies of the computing devices and services themselves.

- **Virtual counterpart:** Each component is a virtual counterpart of a physical entity, place, or computing device and maintains its target's attributes.
- **Containment relationship:** Each component can be contained within at most one component according to the containment relationships between entities, places, and computing devices. It can move between components as a whole with all its inner components.
- **Movement range:** Each component can confine the range of its movement within a specified component that nests it and cannot move beyond this range.

Components are organized within an acyclic-tree structure, like Unix's file-directory. When a component contains other components, we call the former component a *parent* and the latter components *children*. When a component carries another component beyond the latter's range, the latter remains within the range or terminates. Each component can explicitly have a substitute or representation of it within its descendants, like Unix's symbolic link. The substitute is still a component but has no attributes. When it receives components or control messages, it automatically forwards the visiting components or messages to its original component.

3.2 Component

Components can be classified into four types (Figure 1).

- **Virtual Component (VC)** is a digital representation of a physical entity or space in the physical world.
- **Aura Component (AC)** is a virtual or semantic scope surrounding a physical entity or computing device.
- **Proxy Component (PC)** bridges the world model and computing device, and maintains the subtree of the model or executes services located in the VC.
- **Service Component (SC)** is a software module that defines application-specific services associated with physical entities or places.

Containment relationships between components reflect on the structural containment relationships among entities, e.g., people and things, spaces, i.e., rooms and floors, and computing devices by using location-sensing systems, as we can see from Figure 2. For example, when a person moves from the coverage area of one sensor to the coverage area of another sensor, the model detects and moves the VC corresponding to the moving person from the VC corresponding to the source location and then the VC corresponding to the destination location. If the person has a computing device, the VC corresponding to the person carries the PC corresponding to the device. When one or more spaces, e.g., the coverage areas of sensors, geographically may overlap, our model simply treats these spaces as coexistent components.[2]

Virtual Component (VC)

A person, physical object, or place can have more than one VC and each VC can contain other VCs, ACs, and PCs according to spatial containment relationships in the physical world. Unlike other existing location models, our model does not distinguish between entities and places in the physical world; some entities can be viewed as spaces, e.g., cars and desks, in the sense that they can contain other entities inside them. It also permits places to be mobile. For example, a car carries two people and moves from location to location with its occupants. The VC for the car contains two VCs corresponding to the two people and migrates from the VC corresponding to the source location to the VC corresponding to the destination location as a whole with its inner components.

[2]When the model detects such redundancy, it allows one of the two VCs corresponding to the spaces to contain a component bound to the entity and the other to contain a substitute for the component.

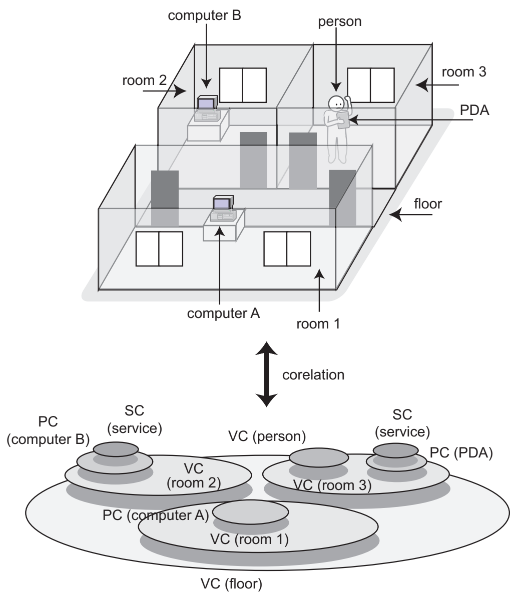

Figure 2. Rooms on floor in physical world (left) and their counterpart components in location model.

Aura Component (AC)

Several researchers on virtual-reality (VR) have provided the notion of virtual scope, often called *aura*, where interactions between two objects in a VR become possible only when the object scopes collide or overlap [6, 10]. In pervasive computing environments, interaction between people and computing devices should be possible when they are within a specified scope.

- Each entity or computing device can have more than one virtual scope, called an *aura*, surrounding it. The scope is implemented as a derivation of VC, called AC (Aura Component).
- An entity can only receive services running on a device while it is within the device's aura and the device is within the entity's aura.

An AC is a virtual bounding-scope depending on media, e.g., visual, audio, and hand-manipulation, between the entity and device. Each aura surrounding an entity or computing device is a child of the component corresponding to the entity or device, although the aura may spatially contain the entity or device. The model allows each aura to define its own shape and size. For example, a person may have a half-meter sphere so that he or she can directly manipulate devices and a computing device may have an aura of a few-meters corresponding to the range of visibility on its screen. Each AC surrounding an entity can deploy (or fetch) software to define application-specific services at devices (or from VCs or PCSs) within it.

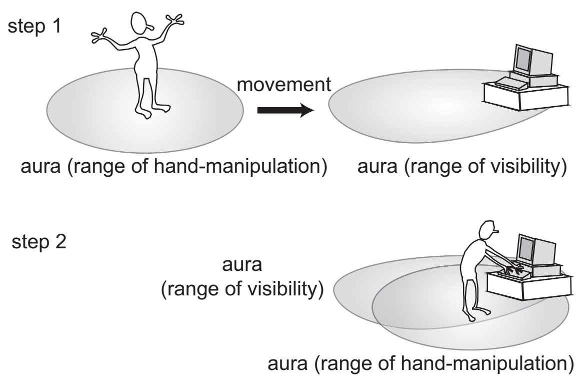
step 1
movement
aura (range of hand-manipulation)
aura (range of visibility)
step 2
aura
(range of visibility)
aura (range of hand-manipulation)

Figure 3. Auras for range of user's hand-manipulation and for range of computer's visibility.

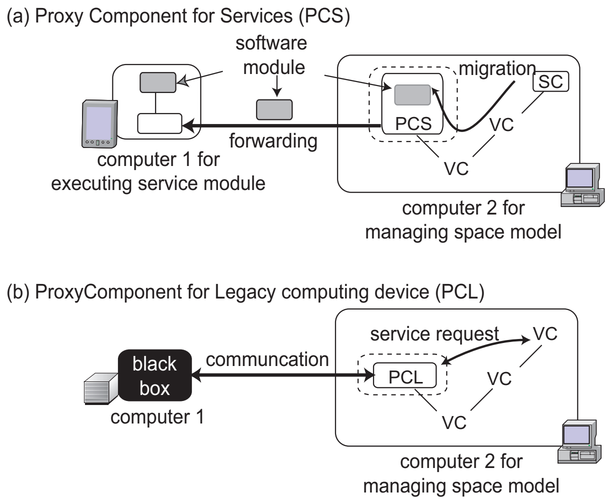
(a) Proxy Component for Services (PCS)
software module
migration
SC
PCS
VC
forwarding
VC
computer 1 for executing service module
computer 2 for managing space model
(b) ProxyComponent for Legacy computing device (PCL)
service request
VC
black box
communcation
PCL
VC
computer 1
VC
computer 2 for managing space model

Figure 4. Two types of proxy components

Proxy Component (PC)

VCs and ACs can have software to define the context-dependent services inside them. However, they may not have the ability for executing the software inside them, because all the computing devices that maintain those do not have unlimited computational resources. Instead, there are two facilities by which services can be provided. The first is to deploy such a service at a computing device embedded in or visiting a space and execute it on the device. The second is to directly use the service provided by a computing device within a space. Therefore, the model treats computing devices as the following two subtypes of PCs to naturally maintain the location of computing devices and use the devices as service providers.

- PCS (PC for Service provider) is a proxy of a computing device that can execute services (Figure 4(a)). If such a device is in a place, its proxy is contained in the VC corresponding to the space. When a PCS receives software for defining services, it forwards the software to the device that it refers to.
- PCL (PC for Legacy device) is a proxy of a computing device that cannot execute SCs (Figure 4(b)). If such a device is in a space, its proxy is contained in the VC corresponding to the space and communicates with the device through the device's favorite protocols.

These components are unique to other existing location models and are useful in naturally maintaining and using computing devices.

Service Component (SC)

We should reuse existing location-based and personalized services as much as possible. The model introduces several typical software components, e.g., Java Beans and Java Applets as service provider programs. However, such existing components may not be suitable in our model. Each SC is a wrapper for software modules to define application specific services and each specifies the attributes of its services, e.g., the requirements that a device must satisfy to execute those services. The model maintains the locations of services by using SCs.

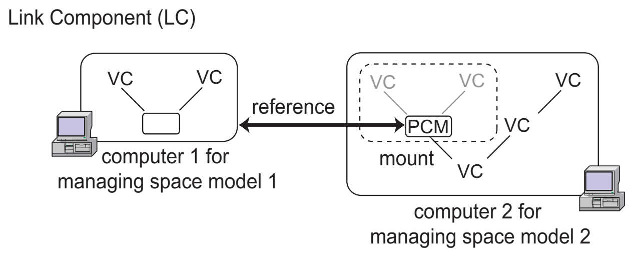

Figure 5. Attachment of location model to another location with LC

4 Prototype Implementation

This section presents a prototype implementation of our location model. Although the model itself is independent of any programming languages, the current implementation is built on a Java-based mobile agent system, called MobileSpaces [21].

4.1 Component Management System

The MobileSpaces system enables agents to be organized in a tree structure and to migrate to other agents, which may be on different computers, with their inner agents. Therefore, it can naturally and easily support a component hierarchy and migrate components between computers. After this, we will explain how to implement other features of the model in the current prototype system.

Distributed Model Management

Our model can be maintained in more than one pervasive computing device. It introduces a component, called Link Component (LC). Each LC is a proxy for a subtree that its target computing device maintains and is located in the subtree that another computing device maintains. As a result, it attaches the former subtree to the latter (Figure 5). When it receives other components or control messages, it automatically forwards them to the device that it refers to (and vice versa). Therefore, even when the model consists of subtrees that multiple computing devices maintain, it can be treated as a single tree.

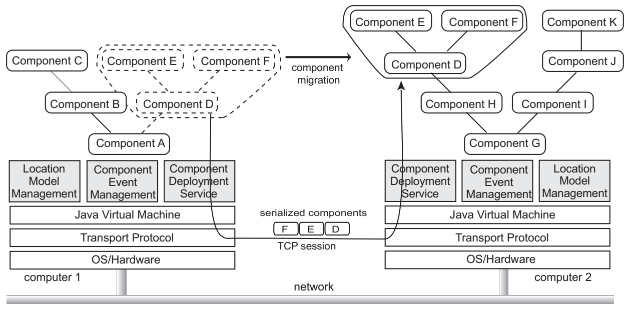

Figure 6. Component migration between computers

Component Deployment Management

Component migration in a tree maintained by a computer is done merely as a transformation of the tree structure of the hierarchy (Figure 7). When a component is moved to other components on different computers, a subtree whose root corresponds to the component and its descendent components are marshalled into a bit-stream with digital signatures for authentication and are transmitted to the destination through a TCP connection.[3] After they arrive at the destination, they can continue with their processing, because not only their program codes but also their states are

[3]The current implementation marshals components by using Java object serialization and supports the notion of weak migration [14].

transferred to the destination like mobile agents. If components have a reachable range for their movements, the system disallows them from traveling beyond the range. The system send events, e.g., changing, entering, and leaving, to components when they enter or leave other components, or another component enters or leaves them.

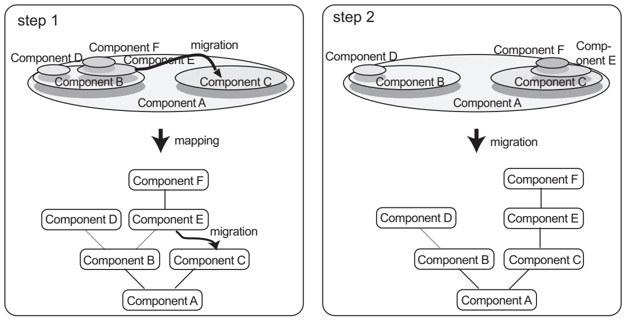

Figure 7. Component containment and migration

4.2 Location-Sensor Management System

To bridge location-sensors and devices that maintain subtrees, the model introduces location-management systems, called LSMs, outside the component management systems. Each LSM manages location sensors and exchanges information between other LSMs in a peer-to-peer manner. This is a lightweight system because it can be operated on embedded computers initially designed to manage sensors, or computers that can maintain the model or execute application services. Hereafter, we will assume that entities and computing devices have been attached with radio-frequency (RF), infrared, or ultra-sonic tags that can passively or actively notify their own identifiers to sensors.

Monitoring Location-Sensors

When an LSM detects changes in the position or presence of a tag in the coverage area of the sensor that it manages, it tries to resolve the tag (Figure 8). There are two possible scenarios, either the tag may be attached to an entity, or it may be attached to a computing device. The LSM detects VCs or PCs bound to the entity or device in the subtree that contains the VC or AC bound to the area in a breadth-first-search (BFS). This is because such new tags often emanate from one of their neighboring spaces or their surrounding space. If the LSM cannot discover any VCs or PCs in the first step, it multicasts a query message with the identifier of the tag to other LSMs and computing devices that are maintaining their subtrees. The LSMs or devices that know where these the VCs or PCs are located send reply messages to the multicasting LSM. If LSMs manage sensors that can measure geometric locations, they can define one or more virtual spaces within the coverage areas of their sensors and transform the geometric positions of entities within the areas into qualitative information concerning the presence or absence of entities in the spaces.

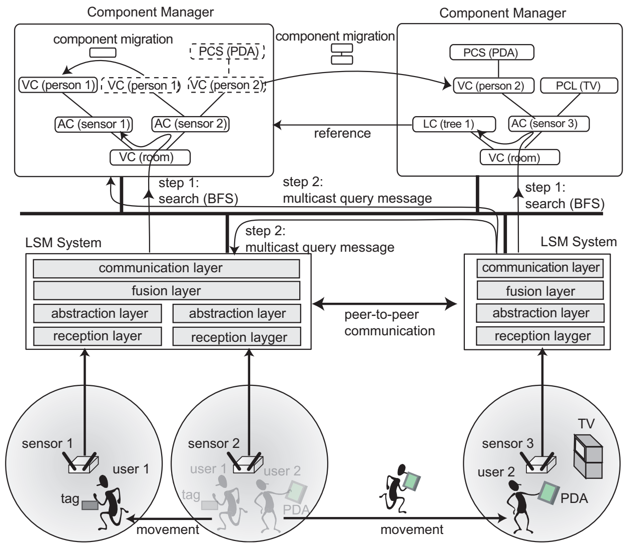

Figure 8. LSM discovers VCs or PCs bound to visiting tags

Location-based Deployment of Components

When an LSM detects the presence/absence of an entity or device in an area, it configures the model that is maintaining the VC or AC bound to the area.

- If a new tag is bound to an entity, the VC bound to the entity is deployed at the VC or AC bound to the area that contains the entity. Also, the LSM sends events to the area's VC or AC and informs the entity's VC about computing devices within the area. If the VC or AC has services and the devices can satisfy their requirements, it deploys the services at the devices.
- If a new tag is bound to a computing device, the PCS or PCL bound to the device is deployed at the VC or AC bound to the area that contains the device. The LSM sends events to the PCS or PCL and informs the area's VC or AC about the device's capabilities. If the VC or AC has services and the device can satisfy the requirements for these, it deploys them at the PCS.

4.3 Component

In the current implementation, all components are defined as a subclass of abstract class `Component`, which had some built-in methods that were used to control its mobility and life-cycle.

```
class Component extends MobileAgent {
  void setIdentity(String name) { ... }
  void add(Component comp) throws NoSuchComponent { .. }
  void remove(Component comp) throws ... { .. }
  ComponentInfo getParentComponent() { ... }
  ComponentInfo[] getChildren() { ... }
  ....
}
```

By invoking `setIdentity`, a component can assign the symbolic name of the physical entity or space that it represents. When a component invokes the `add` (or `remove`) method, it contains the component specified as `comp` inside it (or extracts the component specified as `comp` from itself).

Virtual Component (VC) Each VC is defined as a subclass of abstract class `VirtualComponent` and is bound to at least one entity or space in the physical world. By invoking `setAttribute`, each VC can explicitly record attributes about its entity or space, e.g., owner, position, shape, and size. A VC can have SCs for defining services and it also allows them to access the service methods provided by the components contained within it by invoking the `getAncestorServices` method with a keyword to obtain a list of suitable SCs.

```
class VirtualComponent extends Component {
  void setAttribute(String attr, String val) { ... }
  void registryService(SerivceProvider sp,
    VirtualComponent aura) { ... }
  ServiceInfo[] getAncestorServices(String name,
    Method meth) { ... }
  Object execService(ServiceInfo si, Message msg)
    throws NoSuchServiceException { ... }
  ....
}
```

Aura Component (AC) ACs are implemented as derivations of VCs and inherit the features of the `Virtual-Component` class. As VCs, they can carry services and deploy these at computing devices. For example, moving users may also want to constantly change the computers with which they interacts. That is, their services should be dynamically deployed by computers near them to assist or support them. The user's aura has a bounding scope and contains such services. When a user moves to another room, his or her VC migrates to a VC corresponding to a room with services. The VC deploys the services at appropriate computing devices whose computational resources can satisfy their requirements (within the scope of the aura) to execute these services on the devices. The model allows each computing device to specify its capabilities in CC/PP (composite capability/preference profile) form [28].

Proxy Component (PC) PCSs and PCLs are key elements in the model. Each PCS is a representation of a computing device that can execute SCs. When it receives a software module, a PCS automatically forwards its visiting SCs to its target device through an HTTP-based communication protocol. If the device supports Java's object serialization mechanism, the device's PCS can forward both the classes and state of the SCs to the device. Otherwise, the PCS can also extract Java classes from the modules and deploy only the classes at the device. Each PCS allows other components to fetch modules and access the methods of the SCs that are forwarded to each PCS's target device as if they were in it. Each PCL is located at a VC bound to a space that contains its target device and establishes communication with its target device through the device's favorite protocol, e.g., serial communication and infrared signals. Note that a computing device can have one or more PCs, which may be.

Service Component (SC) SCs are defined as subclasses of the `Component` class. The model enables SCs to specify preferable and minimal capabilities for computing devices that they may visit in CC/PP form, e.g,, device types (Desktop PC, Notebook PC, or PDA), screens, and input devices. It permit Java Beans or Applets, which are widely used, to be wrapped by special SCs to be treated as children of VCs, ACs, and PCs.

4.4 Current Status

A prototype implementation of this model was built with Sun's J2SE version 1.4.[4] It uses the MobileSpace mobile agent system to provide mobile components and supports three commercial locating systems: Elpas's system (infrared tag sensing system), RF Code's Spider (active RF-tag system), and Alien Technology's UHF-RFID tag (passive RF-tag system).

Although the current implementation was not built for performance, we measured the cost of migrating a 4-Kbyte component (zip-compressed) from the source to the destination recommended by an LSM over a network. The latency of component migration to the destination after the LSM had detected the presence of the component's tag was 390 msec and the cost of component migration between two hosts over a TCP connection was 41 msec. This experiment was done with two computing devices that maintain component tree, and source and destination computing devices, each of which was running on one of six computers (Pentium M-1.6GHz with Windows XP and J2SE ver. 5) connected through a Fast Ethernet network. We believe that this latency is acceptable for a location-aware system used in a room or building.

[4]The functionalities of the framework except for subscribe/publish-based remote event passing can be implemented on Java Developer Kit version 1.1 or later versions, including Personal Java.

5 Experience

We have had experience with this model in developing and operating several typical applications for location-based and personalized services. Since some of these have been presented in previous papers [22, 23] independent of the model, this section addresses the use and advantages of the model.

5.1 Location-based Navigation Systems

The first example is a user navigation-system application running on portable computing devices, e.g., PDAs, tablet-PCs, and notebook PCs. The initial result on the system were presented in a previous paper [23]. There has been many research or commercial systems for similar types of navigation, e.g., CyberGuide [1] and NEXUS [12]. Most of these have assumed that portable computing devices are equipped with GPSs and are used outdoors. Our system is aimed at use in a building. As a PDA enters rooms, it displays a map with its current position. We assumed that each room in the building would have a coverage of more than one RF-tag reader managed by an LSM, the room is bound to a VC that had a service module for location-based navigation, and each PDA could execute service modules and be attached to an RF-tag. When a PDA enters a room, the RF-tag reader for the room detects the presence of the tag and the LSM tries to discover the component bound to the PDA through the procedure presented in the previous section. After it has information about the component, i.e., a PCS bound to a PDA, it informs the VC corresponding to the room about the capabilities of the visiting PDA. The VC then deploys a copy of its service module at the PCS and the PCS forwards the module to the PDA to which it refers, to display a map of the room. When the PDA leaves the room, the model issues events to the PCS and VC and instructs the PCS to return to the VC. Figure 9 outlines the architecture for the system and shows the screen of a service module running on a visiting PDA displaying a map on the PDA's screen.

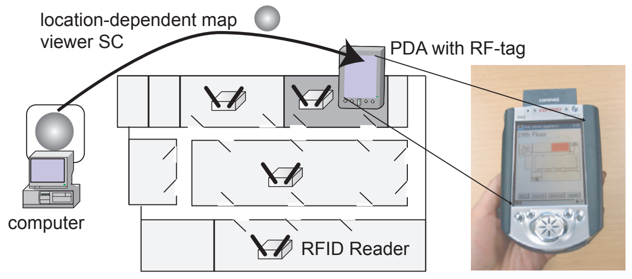

Figure 9. RF-tag-based location-aware map-viewer service.

5.2 Follow-Me Applications

Follow-me services are a typical application in pervasive computing environments. For example, Cambridge University's Sentient Computing project [11] enabled applications to provide a location-aware platform using infrared-based or ultrasonic-based locating systems in a building.[5] While users are moving around, the platform can track their movements so that the graphical user interfaces of their applications can follow them and be displayed on the screens of computers near them through the VNC system [18]. The model presented in this paper, on the other hand, enables the movement of users to be naturally represented independent of location sensing systems. Unlike previous studies on applications, it can also migrate applications themselves to computers that are near moving users.

The model binds a user with a VC and two computers with PCSs that contain two ACs surrounding both of them, where each AC defines the range of visibility as a one-meter sphere surrounding its target computer. As we can see from Figure 10, when a user comes near a computer, the model deploys the user's VC at the computer's AC. The VC then finds the PCS that refers to the computer via the AC. The user's VC deploys applications stored in the user's AC at the PCS so that the applications can execute on the computer. When the user moves to another computer, the model removes the user's VC from the AC and the VC fetches its applications from the previous computer via the PCS. It then allows the user's VC to contain the PCS that refers to the computer at the destination. The current implementation uses an active RF-tag system (RF-Code Spider system). Since the system can change the coverage of its readers, we can control the scopes of the two ACs.

5.3 Personal Server

The third example is similar to the second, but it was inspired by the personal server proposed by Want [27]. By using this model, we could easily implement interactions between personal servers and stationary computers, i.e., wall-mounted smart displays and public terminals. A user carried a handheld file-sharing server that had no integral user interface, but had a processor, and secondary storage, and a wireless LAN network interface and was tied to an active RF-tag. When he or her approached a stationary computer, his or her personalized services were dynamically deployed at smart TVs. The user had a VC as a digital representation of him or her in the model and this contained a PCL bound to a personal server that supported a file-sharing server for maintaining images and the SC that defined an

[5]Although the project does not report their world model, their systems seem to model the position of people and things through lower-level results obtained from their underlying location sensing systems.

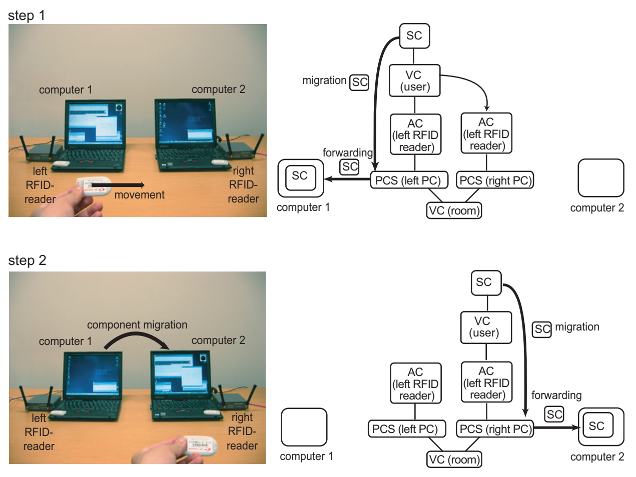

Figure 10. Follow-me desktop applications between two computers.

image viewer. A smart TV had a PCS that referred to itself and has an AC for specifying the range of visibility between the user and the TV. When he or she moved to the TV with his/her personal server, the model deployed the PCL bound to the server at the AC surrounding the TV. The PCL then communicated with the TV's PCS so that he or she viewed stored stored in the server on the screen of the TV (Figure 11). We developed a mechanism that enabled personal servers and embedded computer [16] to communicate the PCL could gathers image data from the server and then display the image data through this mechanism.

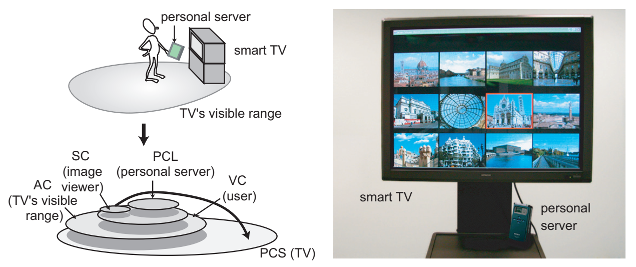

Figure 11. Smart TV executes PCL to access image data from personal server to display GUI on screen of TV.

6 Related Work

There has been a great deal of research and commercial location-based information services, e.g., GUIDE [7] Context Toolkit [20], and commercial GIS software. Most existing services, e.g., map viewer and tourist navigation, have been designed to run on portable computing devices equipped with GPSs and have been dependent on their initial application-specific services. They lack any general world model or are inherently based on geometric information measured from GPSs. Of these, NEXUS [12, 2] and Cooltown [13] can transform geometric information measured from GPSs into references in a symbolic world model so that they can determine the positions of objects by identifying the spatial regions that contain these objects. However, they do not support tracking sensors, which can measure the location of other objects, e.g., RFID. Sentient Computing [11] and EasyLiving [5] were aimed at building smart rooms and supported ultra-sonic or computer-vision-based tracking sensors instead of positioning sensors. They do not support other locating systems.

ParcTab [26], Leonhard's zone model [15], Aura [9], and RAUM [4] offer symbolic models as a set of names or references to places, independent of sensing systems. Like our model, they can represent containment relationships between entities and places. ParcTab and Leonhard's zone model were aimed at representing the location of people and physical objects. The RAUM model can represent the location of pervasive computing devices like ours. However, these existing models cannot manage the location and deployment of services and assume that their location models are maintained in a centralized database server. Virtual Counterpart [19] supports RFID systems and provides objects attached to RFID-tags with Jini-based services. Since it enables objects attached to RFID-tags to have their counterparts, it is similar to our model. However, it only supports physical entities other than computing devices and places, whereas our model cannot distinguish between physical entities, places, or software-based services.

The model presented in this paper was initially inspired by our previous work, called SpatialAgents, which is an infrastructure that enables services to be dynamically deployed at computing devices that are near people and objects [23]. The previous framework unfortunately lacked any location model and could not represent any structural relationships between physical spaces, e.g., containment relationships between rooms and buildings. We also presented a framework for dynamically deploying software components [24], but it was aimed at the self-organization of distributed systems and lacked any location model. One goal with the model presented in this paper was to provide a general-purpose location model for the previous two frameworks.

7 Conclusion

We presented a location model for developing and managing context-aware services, e.g., those that are location-aware and personalized, in pervasive computing environments. Like other existing related models, it can be dynamically organized like a tree based on geographical containment and each node in the tree can be constructed as an executable software component. It is unique to other models, because it can provide a unified view of the locations of not only physical entities including users, objects, and spaces but also of computing devices and service-provider software. It was designed to provide context-aware services for physical entities rather than model the locations of physical entities and places. We also designed and implemented a prototype system based on the model and demonstrated its effectiveness in several practical applications.

Finally, we would like to identify further issues that need to be resolved in the future. We need to develop more applications with the model. Since the prototype implementation presented in this paper is constructed on Java but the model itself is independent of the language, we are therefore interested in developing it with other languages. We plan to design more elegant and flexible APIs for the model by incorporating existing spatial database technologies. We are interesting in designing a query language for mobile entities and spaces by extending the Ambient calculus [8]. We also plan to apply the model to software testing for location-based mobile computing [22, 25].

References

[1] G.D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, “Cyberguide: A Mobile Context-Aware Tour Guide”. ACM Wireless Networks Vol. 3, pp.421–433. 1997.

[2] M. Bauer, C. Becker, and K. Rothermel, Location Mmodels from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks, Personal and Ubiquitous Computing, vol. 6, Issue 5-6, pp. 322-328, Springer, 2002.

[3] C. Becker, Context-Aware Computing, Tutorial Text in IEEE International Conference on Mobile Data Management, (MDM’2004), Januray 2004.

[4] 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.

[5] 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.

[6] C. Carlsson, and O. Hagmnd, DIVE: A platform for multi-user virtual environments Computer and Graphics. vol. 17, no. 6, pp. 663-669, 1993.

[7] K. Cheverst, N. Davis, K. Mitchell, and A. Friday, Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project, Proceedings of Conference on Mobile Computing and Networking (MOBICOM’2000), pp. 20-31, ACM Press, 2000.

[8] L. Cardelli and A. D. Gordon, Mobile Ambients, Foundations of Software Science and Computational Structures, LNCS, Vol. 1378, pp. 140-155, Springer, 1998.

[9] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, ” Project Aura: Towards Distraction-Free Pervasive Computing, IEEE Pervasive Computing, vol. 1, pp. 22-31, 2002.

[10] C. Greenhalgh and S. Benford, MASSIVE: A Collaborative Virtual Environment for Teleconferencing ACM Transactions on Computer-Human Interaction, vol 2, no. 3, September 1995.

[11] 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, 1999.

[12] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, and M. Schwehm, Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications, Proceedings of Conference on Mobile Computing and Networking (MOBICOM’99), pp. 249-255, ACM Press, 1999).

[13] T. Kindberg, et al, People, Places, Things: Web Presence for the Real World, Technical Report HPL-2000-16, Internet and Mobile Systems Laboratory, HP Laboratories, 2000.

[14] B. D. Lange and M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, 1998.

[15] 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.

[16] T. Nakajima and I. Satoh, Personal Home Server: Enabling Personalized and Seamless Ubiquitous Computing Environments, Proceedings of 2nd International Conference on Pervasive Computing and Communications (PerCom’2004), pp.341-345, IEEE Computer Society, March 2004.

[17] C. E. Perkins, “Ad Hoc Networking”, Addistion Wesley, 2001.

[18] T. Richardson, Q, Stafford-Fraser, K. Wood, A. Hopper, Virtual Network Computing, IEEE Internet Computing, Vol. 2, No. 1, 1998.

[19] K. Romer, T. Schoch, F. Mattern, and T. Dubendorfer, Smart Identification Frameworks for Ubiquitous Computing Applications, IEEE International Conference on Pervasive Computing and Communications (PerCom’03), pp.253-262, IEEE Computer Society, March 2003.

[20] D. Salber, A. K. Dey, and G. D. Abowd, The Context Toolkit: Aiding the Development of Context-Enabled Applications, Proceedings of Computer-Human Interaction (CHI’99), pp.15-20, ACM Press, 1999.

[21] 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.

[22] I. Satoh, A Testing Framework for Mobile Computing Software, IEEE Transactions on Software Engineering, vol. 29, no. 12, pp.1112-1121, December 2003.

[23] I. Satoh, Linking Phyical Worlds to Logical Worlds with Mobile Agents, Proceedings of International Conference on Mobile Data Management (MDM’2004), IEEE Computer Society, January 2004.

[24] I. Satoh, Dynamic Federation of Partitioned Applications in Ubiquitous Computing Environments, Proceedings of 2nd International Conference on Pervasive Computing and Communications (PerCom’2004), pp.356-360, IEEE Computer Society, March 2004.

[25] I. Satoh, Software Testing for Wireless Mobile Computing, IEEE Wireless Communications, vol. 11, no. 10, pp.58-64, October 2004.

[26] R. Want, B. Schilit, A. Norman, R. Gold, D. Goldberg, K. Petersen, J. Ellis, and M. Weiser, An Overview of the Parctab Ubiquitous Computing Experiment, IEEE Personal Communications, Vol 2. No.6, pp28-43, December 1995.

[27] R. Want, The Personal Server - Changing the Way We Think about Ubiquitous Computing, Proceedings of 4th International Conference on Ubiquitous Computing (Ubicomp 2002), LNCS 2498, pp. 194-209, Springer, September 2002.

[28] World Wide Web Consortium (W3C), Composite Capability/Preference Profiles (CC/PP), http://www.w3.org/ TR/NOTE-CCPP, 1999.