A Middleware for Supporting Context-Aware Services in Mobile and Ubiquitous Environment

Nam-Shik Park, Kang-Woo Lee, Hyun Kim
Electronics and Telecommunications Research Institute
{nspark, kwlee, hyunkim}@etri.re.kr

Abstract

Ubiquitous computing has developed with mobile computing technology. With the recent advances in mobile computer technology and wireless network, the nature of the services proposed to the users is moving towards mobile and context-aware services. On account of such moving, applications and services in mobile and ubiquitous environment must be aware of their changing environments and adapt according to changing contexts. However, developing such systems providing context-aware applications or services is still a complex and time-consuming task. Therefore, we introduce architecture of Context-Aware Service Middleware (CASM) that will provide a middleware-level support for building and rapid prototyping of context-aware services and describe interactions based on the architecture.

1. Introduction

With the advances in information technology, there has been a lot of change in human life and industry. Recent mobile computing and wireless network technologies have made diverse applications and services available in the wired and wireless network environment. With the rising popularity of mobile services, it is expected that mobile services will be expanded significantly in terms of complexity and variety. Moreover, convergence trends in communication, broadcasting, and finance may promote the expansion of mobile services and increase complexity further. In order to avoid increasing complexity, and allow the users to concentrate on their tasks, mobile services must be aware of their contexts and automatically adapt to their changing contexts. By context, we refer to any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computation object. For example, contexts can be a person's current location, the user profile, the current time and date, or the temperature, etc. Context-aware computing is a mobile computing paradigm in which a user's applications discover and react to changes in context [1]. Applications may deal with a wide variety of information stemming from communication, environment, and location dynamics. Context-aware applications use environmental context inputs to provide information to the user or to enable services for him/her. Also information to the application and the network can be provided to adapt the application infrastructure to the actual conditions. There have been many researches on context awareness in recent years. However, context-aware services have never been widely available to everyday users and developing such systems is still a complex and time-consuming task. Context-aware systems such as Context Toolkit [8], HP's Cooltown [7], and CoBrA [9] have been developed to demonstrate the usefulness of context-aware computing technology. Many of them are designed in an application-specific fashion and lack of an infrastructure support for sharing context knowledge and context reasoning. We introduce architecture of Context-Aware Service Middleware (CASM) that will provide a middleware-level support for building and rapid prototyping of context-aware services. Our middleware provides supports for most of the tasks involved in dealing with context such as context acquisition from various sources, context interpretation and context dissemination. Through CASM, high-level implicit contexts can be derived from low-level explicit contexts and those contexts can be given applications so that they operate in context-aware.

The rest of this paper is organized as follow. Section 2 describes middleware requirements that are necessary to be considered for context awareness in ubiquitous environments. Section 3 shows the middleware architecture for providing context-aware services. In section 4, we also describe on how CASM interacts for context-aware services in mobile and ubiquitous environment. Finally, we draw conclusions in section 5.

2. Middleware Requirements

Many people agree that middleware plays a vital role in hiding the complexity of distributed applications. These applications typically operate in an environment that may include heterogeneous computer architectures, operating systems, network protocols, and databases. It is unpleasant for an application developer to deal with such heterogeneous. Middleware's primary role is to conceal this complexity from developers by deploying an isolated layer of APIs. This layer bridges the gap between application program and platform dependency. Middleware systems have emerged in recent years to support applications in heterogeneous and ubiquitous computing environments. According as computing paradigm and environment have changed, environmental factors have affected in designing middleware systems. Network communication, coordination, reliability, scalability, and heterogeneity have been common requirements of traditional middleware such as Message Oriented Middleware (MOM), Transaction Processing Monitors (TPMON), Remote Procedure Calls (RPC), Distributed Computing Environment (DCE), and Object Oriented Middleware (ORB). During the system lifetime, the application behavior may need to be altered due to dynamic changes in infrastructure facilities, such as the availability of particular services. Therefore, some requirements different from traditional middleware have to be considered for the middleware supporting applications and services in the ubiquitous environment.

- *Dynamic Reconfiguration*: Detecting changes in available resources and reallocating them or notify the application to change its behavior.
- *Adaptivity*: The ability of a system to recognize unmet needs within its execution context and to adapt itself to meet those needs.
- *Context-Awareness*: Disclosing the execution context to the upper layer. The context may include device characteristics, user's activities, and services.
- *Asynchronous Paradigm*: Decoupling the client and server components and delivering multicast messages.
- *Lightweight*: Minimum range of functionality used by most applications.

Context-awareness is an important requirement to build an effective and efficient adaptive system. For examples, the context of a mobile unit is usually determined by its current location which, in turn, defines the environment where the computation associated with the unit is performed. When designing the context-aware middleware, it is necessary for its architecture to consider following:

- Support sensing from a variety of sources.
- A simple modeling of the physical world.
- Processing of heterogeneous sensor data.
- A simple event delivery structure and mechanism.
- Support storing and querying contextual information.
- Extensible via pluggable and standard interfaces wherever possible.

3. Architecture

In the ubiquitous environment, various devices can be used. When a service is provided, the service needs to be bound to given environment based on the information related to user's intension. Insertion or deletion of a binding by service creation or environmental change must be considered. The CASM consists of build-time and runtime parts as Figure 1.

3.1. Build-Time

In order to provide a service, CASM needs both environment and binding information of a service at the time providing the service. As build-time is to model information to be used in providing a service, it consists of modeling components such as sensor modeler, environment modeler, user modeler, task modeler, and service modeler.

1) *Service Modeler*: As to model services to be invoked from a task, it defines services to be needed in application domain. Also, it consists of components for the implementation and management of defined services.

2) *Sensor Modeler*: As to collect the contextual information from the environment that services are performed and to deliver the information to a context manager, it is to model services to be provided actively.

3) *Environment Modeler*: The cyber space is an abstract model of the physical space. Environment is an abstract model of the place in a specific domain. Users perform their work by interacting with the space and the environment where they are located through CASM. Environment modeler models not only sensors, devices, and services in environments but also hierarchically the environment for a domain.

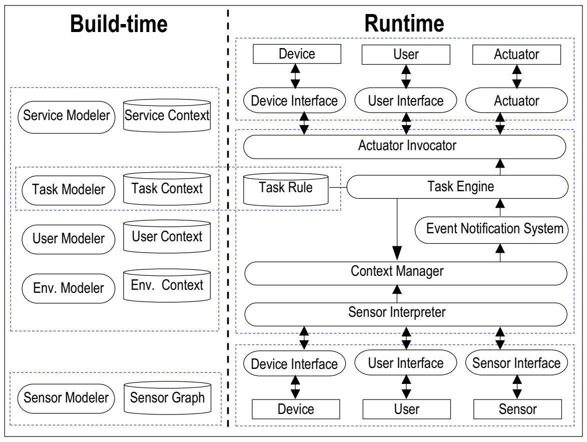

Figure 1. Middleware Architecture

4) User Modeler: As to model users, user information such as organizations, personal profile information, and user preference information is shared by all tasks.

5) Task Modeler: As to write tasks, it includes either a task context modeling or a task rule which is used in writing tasks to be executed. All the contextual information to be used in CASM is represented as Universal Data Model (UDM) that denotes all information to an association between two nodes [2, 3].

6) Task Rules: It is a set of rules defined by a task definition language that is Programming Language for Ubiquitous Environments (PLUE).

3.2. Runtime

CASM provides users with services at runtime using the contextual information modeled at build time. The runtime components of CASM consist of context manager, task engine, and event notification system.

1) Context Manager: As a module managing the contextual information that is represented in UDM, it plays a role as the information storage of CASM. It supports a mapping of data representation denoted by such ways as XML representation, File system in Unix/Windows, NamingContext in CORBA, Java objects (including JavaBeans), Tables in RDBMS, XTM, RDF/OWL[5, 6]. Also, it provides facts for an execution of Task Engine.

2) Event Notification System: It delivers events transferred by sensors or external services to tasks or listeners. In case of using Notification Service in CORBA as an event notification system, CORBA operations can be optimized since transferring only the event that an event listener needs [4].

3) Task Engine: A task is a set of actions to be performed according to requests and situations of users. It is a unit of work for CASM. Task engine boots tasks and manages task processes being executed and its roles are as below:

- Core of task control: The task engine manages various components consisting of services, sensors, actuators and user interactions.
- Creation of work item for a situation: It analysis contexts to be required in executing a work, acquires a right context from context manager, and prepares to execute the work.
- Reasoning: It collects contextual information to be required for a work execution and decides a conditional branch to control begin, end, and branch of the work execution.

4. Interactions

From a viewpoint of CASM, the real world consists of the physical space and the cyber space. The physical space is composed of physical resources which persons work by interacting with. As the cyber space is an abstraction of the physical space, physical objects can be modeled to electronic resources by mapping. CASM works for a person by interacting with cyber space. There are various physical objects residing in the physical world such as a mobile terminal, a home server and a robot. Each of them can be mapped to an object in the cyber space to be controlled by CASM. CASM can decide a specific service that a physical object will provide based on internal/external contextual information of the object. It receives a state of the world and changes the state by executing a work. All these processes are done by task executions. In the real world, there are various data sources such as devices, users, and sensors. A sensor interpreter changes lower level information, which is collected by using a sensor graph created from the sensor modeler, to higher level information and transfers it to a service. To access sensor information through a sensor object, context manager uses the pull method which accesses sensor information by request or the push method which requests to transfer a changed sensor value. The sensor information accessed by the context manager is transferred to a task managed by the task engine. There are work items in a task described by task rules. The task selects a work item matched with a given situation and actuator invocator of the task invokes a service defined in a task action part according to its own task rules. Device, User, and Sensor have their own actuators and some of actuators can be triggered from their actuator invocator to take action for a service. For example, a home server or physical robot in mobile and ubiquitous environment can be actuated by a command from an external device with a microphone such as smart phone if CASM is built on the device. In this case, CASM sends the voice signal from the external device to a remote server to process the signal and receives the data processed by the server. Then CASM triggers an action specified in a task to actuate a service with the data.

5. Conclusions

Ubiquitous computing is to provide invisible computation anytime and anywhere. To reduce user's distraction and environmental complexity and allow the user to concentrate on her tasks, applications must be capable to operate in highly dynamic environments. In mobile and ubiquitous environment, context-awareness is an important requirement to build an effective and efficient adaptive system. Applications and services in ubiquitous environments must be aware of their changing environments and adapt according to changing contexts. However, developing such systems providing context-aware applications or services is still a complex and time-consuming task. Therefore, we introduced the architecture of Context-Aware Service Middleware (CASM) that will provide a middleware-level support for building and rapid prototyping of context-aware services. For building and developing context-aware applications and services, our middleware provides supports for most of the tasks involved in dealing with context such as context acquisition from various sources, context interpretation, and context dissemination. Context-aware services using our middleware are provided by executing tasks in according to the changes of internal/external contextual information of physical objects in the real world. To make an experiment on the prototype of our middleware, simple services such as context-aware presentation and wireless remote home control will be tested in an experimental room sensing RFID tags.

10. References

[1] G. Chen, and D. Kotz, “A survey of context-aware mobile computing research”, Tech. Rep. TR2000-381, Dartmouth, November 2000.
[2] L. Silverston, "Physically Implementing Universal Data Models to Integrate Data", Data Management Review Magazine, September 2002.
[3] L. Silverston, "A Universal Data Model for Relationship Development", Data Management Review Magazine, March 2002.
[4] IBM, “CORBA notification services : Getting to know CORBA 3.0” , http://www-106.ibm.com.
[5] W3C, “Web Ontology Language”, http://www.w3.org.
[6] W3C, “Resource Description Framework”, http://www.w3.org.
[7] T. Kindberg and J. Barton, "A Web-based Nomadic Computing System", Computer Networks (Amsterdam, Netherlands: 1999), 35(4):443–456, 2001.
[8] Dey, A.K., Salber, D. Abowd, G.D., "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Anchor article of a special issue on Context-Aware Computing, Human-Computer Interaction (HCI) Journal, Vol. 16(2-4), pp. 97-166, 2001.
[9] Harry Chen and Tim Finin, "An Ontology for a Context Aware Pervasive Computing Environment", IJCAI workshop on ontologies and distributed systems, Acapulco MX, August 2003.