Device Context Discovery System for Context-aware Services in Ubiquitous Device Collaboration Environment

Jinwoo Park[1], Jong-Kwon Lee[1], YoungSang Paik[1], MyungChul Lee[1], and Chandra Narayanaswami[2]
[1]*Ubiquitous Computing Lab, IBM Korea, Seoul, Republic of Korea*
[2]*IBM T. J. Watson Research Center, NY, U.S.A.*
{jwp, jkwlee, yspaik, mclee}@kr.ibm.com, chandras@us.ibm.com

Abstract

This paper describes a system for obtaining and describing device context information such as device capabilities and user preferences for context-aware services in the ubiquitous device collaboration framework called Celadon. The proposed system is designed to gather and manage device properties and user profiles from devices, and to provide generalized access methods for manipulating static and dynamic properties of the devices. We present the architecture of the proposed system within the overall Celadon framework and describe the functions of each component in more detail and present our current prototypes.

1. Introduction

Context-aware services that exploit information about user and device context are becoming one of the core components in a ubiquitous computing environment. Device context information includes device capabilities such as display resolution, processor speed and available memory, network bandwidth, user location, etc. and user context includes preferences, age, and gender, etc. Using the context information at hand, a context-aware system can provide mobile users with appropriate services according to the device/user context.

In order to enable context-aware services, it is a prerequisite to gather contextual information about the device capabilities and the user preferences, and to transform them into user/device properties that can be used in a systematic way for providing context-awareness. For this purpose, we need access methods for manipulating static and dynamic properties of a device so that any contents delivered to the device can be adapted to a particular device context. Static properties refer to data that remain constant for the session duration, while dynamic properties can change during the session.

Several efforts have proposed and recommended as frameworks for dealing with the device capabilities. CC/PP (Composite Capability/Preference Profiles) [1] is an XML representation of device characteristics based on Resource Description Framework [2]. CC/PP provides a means to transfer device characteristics from one device to another. It is also the basis for UAProf (User Agent Profile) [3], which is used to express the capabilities of many mobile devices. UAProf has been defined by OMA (Open Mobile Alliance) as an implementation of CC/PP for WAP-enabled mobile terminals. UAProf describes three classes of static device characteristics: hardware, software, and user preferences. More recently, W3C has defined platform and language-neutral interfaces, called DCI (Delivery Context Interfaces) that provide web applications with access to a hierarchy of dynamic properties representing device capabilities, configurations, user preferences and environmental conditions [4].

In this paper, we describe a system for recognizing and describing such device properties for use in a context inference engine on a context management server. Our system has been developed based on the DCI framework so that it can deal with changes in dynamic properties of mobile devices, especially the location of the mobile device. This system which gathers and interprets device/user context is one of the building blocks of the *Celadon* ubiquitous device collaboration framework [5, 6]. The Celadon project aims at enabling seamless collaboration between a wide range of heterogeneous mobile and environmental devices within Celadon zones using a service-oriented architecture. The Celadon zones are collaborative environments in public areas and equipped with wireless access points of Bluetooth or 802.11 and with environmental devices such as displays, printers, and servers (Figure 1). Thus, the system for gathering and managing context information plays an important role in providing context-aware services in the Celadon framework.

The remainder of this paper is organized as follows. In Section 2, several existing studies of the ubiquitous environments are briefly summarized. In Section 3, the overall architecture of the Celadon framework and the structure of device context discovery components are described. Each of the device context discovery components is described in more detail in Section 4. In Section 5, we illustrate example context-aware services based on the implemented device context discovery system. Finally, we conclude this paper in Section 6.

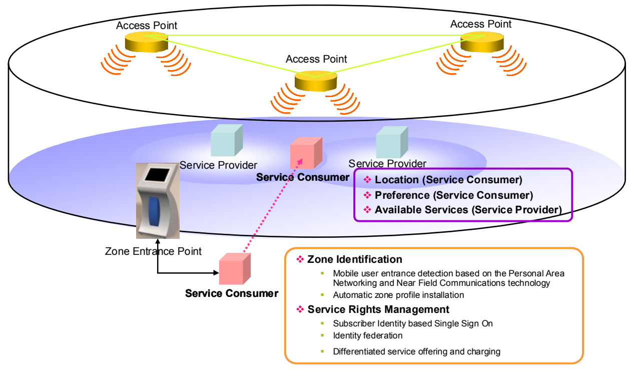

Access Point
Access Point
Access Point
Service Provider
Service Provider
Service Consumer
Location (Service Consumer)
Preference (Service Consumer)
Available Services (Service Provider)
Zone Entrance Point
Service Consumer
Zone Identification
Mobile user entrance detection based on the Personal Area Networking and Near Field Communications technology
Automatic zone profile installation
Service Rights Management
Subscriber Identity based Single Sign On
Identity federation
Differentiated service offering and charging

Figure 1. Concept of Celadon zone

2. Related work

A number of so-called ubiquitous environments or ubiquitous spaces supporting context-aware computing have been developed with various objectives. Especially regarding mobile devices environments and collaboration, the following projects have been studied and developed.

The Pebbles (PDAs for Entry of Both Bytes and Locations from External Sources) project [7] has studied how computing functions and related user interfaces can be spread across all computing and input/output devices available to the user, forming multi-machine user interfaces, or MMUIs. It focuses on how personal handheld computers interoperate with desktop and built-in computers seamlessly in real time.

Hydrogen [8] is an architecture and a software framework which supports context-awareness for the special requirements of mobile devices regarding particularly the limitations of network connections, limited computing power and the characteristics of mobile users. It comprises three layers to separate the concerns of interacting with the physical sensors, storing and maintaining the context from the applications itself. Due to the fact that applications do not deal with remote servers but only with a local server, which provides any kind of available contextual information, the architecture is robust with respect to frequent disconnections. Clearly, Hydrogen focuses on the context architecture in order to overcome the shortcomings of existing approaches in dealing with mobile scenarios and mobile devices.

The Gaia project [9] is a middleware infrastructure that prototypes the resource management and provides the user-oriented interfaces for physical spaces populated with network-enabled computing resources. It applies the concepts of a conventional operating system to middleware to manage the resources, devices and distributed objects in a physical space by driving context-sensitivity into its data storage mechanisms. Gaia exports services to query and utilize existing resources and to access and use current context, and it provides a framework to develop user-centric, resource-aware, multi-device, context-sensitive and mobile applications [10].

SOCAM (Service-Oriented Context-Aware Middleware) [11, 12] is an architecture for the building and rapid prototyping of context-aware mobile services. It targets a similar environment to the Celadon project since it is implemented on top of the OSGi (Open Service Gateway Initiative) framework, running on mobile devices and using ontology-based context model. It is made for applications to easily adapt to the changing contexts by dividing context ontologies into a common high-level ontology and domain-specific ontologies. Also SOCAM converts various physical spaces where contexts are acquired into a semantic space where context-aware applications can share and access them easily.

All the aforementioned projects address well-defined ubiquitous infrastructures and describe their own uniqueness and efficiency. However, these projects are focusing on mobile user interfaces or middleware components that support context-aware services. None of these are considering the ubiquitous collaboration between devices and the management of acquired device/user context.

3. Celadon system architecture

Figure 2 describes simplified Celadon architecture and briefly shows how the mobile device interacts with the broker in the Celadon zone that helps mobile device to find possible services. Figure 3 shows the software architecture of the device context discovery components. The system is divided into two parts: the broker part (server side) and the device part (device side). The broker part functions as a control center to manage the contexts for the devices in the local zone. The device part gathers the information related to the device and user context and reports it to the broker part. After that, the broker resolves the service availability and reserves the available services for the device to use. These two parts communicate with each other via an underlying common web services communication stack. Especially, to deal with dynamic changes in the device context, the data traffic between the two parts is transmitted as an event stream. For this purpose, the discovery components send and receive data by using the Asynchronous Messaging Engine.

Within the Celadon architecture, this paper is especially describing the components on the device side which is indicated with red dotted line in the figure 2, how to acquire and manage device and user context and how to transfer the information to the server efficiently in such architecture.

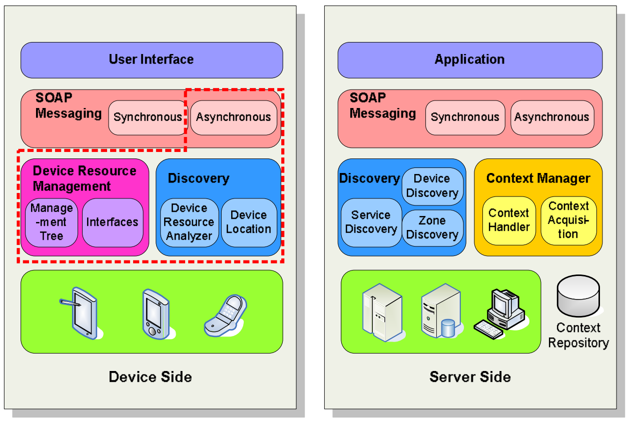


Figure 2. Celadon: zone-based, context-aware framework based on device context inference engine

3.1. Component on device side

Our discovery technology focuses on how to gather context information and manage the service availability based on the acquired device and user context. The device and user contexts, we divided these into three categories as follows:

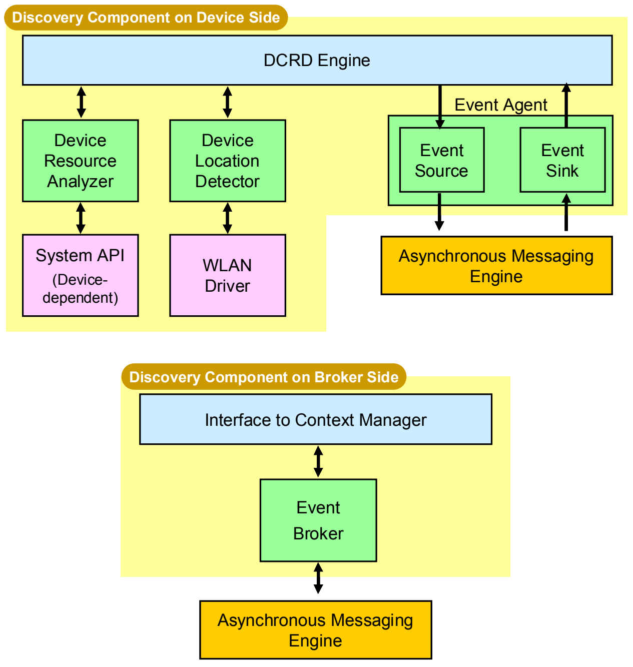
- Device capabilities (e.g., display resolution, available memory, network support, etc.)
- Device location (captured by other method than device capabilities, but treated as one of the device capabilities)
- User profile (e.g., user identity information, user preferences, etc.)

The core sub-component of the discovery engine is the DCRD (Device Context Recognition and Description) engine that is designed for managing the device capability information and for delivering it to the collaboration broker. The Device Resource Analyzer gathers device-specific information through system programming APIs. It provides device capabilities to the DCRD engine, and the DCRD engine transforms it into device properties that can be used for context inference on the broker side.

The Device Location Detector is designed for monitoring the device position by measuring the signal strength from WLAN access points. The result of captured device position is actually treated as one of the device capabilities, but the reason we need another module other than the Device Resource Analyzer is that we exported different kinds of system DLLs (Dynamic Linked Libraries) and implemented a localization algorithm in this module.

As shown in Figure 1, the Celadon zone consists of zone facilities and service actors. The service actors indicate all the devices that reside in the service zone and interact with each other for service execution. The zone facility includes a wireless networking infrastructure and an entrance point. The wireless networking reuses the existing public WLAN service infrastructures such as enterprise WLAN backbone and the WLAN Hot Spots of the mobile operators. We assume that the service zone is equipped with WLAN access points and the mobile user entering the service zone can find out his location by measuring the signal strength of all access points.

The Event Agent implements communication mechanism that allows an object to accept registrations and send published events to a number of receiver objects. The Event Agent relies on the Asynchronous Messaging Engine that constitutes the event delivery channels of the Celadon on top of common communication module.



Discovery Component on Device Side
DCRD Engine
Event Agent
Device Resource Analyzer
Device Location Detector
Event Source
Event Sink
System API (Device-dependent)
WLAN Driver
Asynchronous Messaging Engine
Discovery Component on Broker Side
Interface to Context Manager
Event Broker
Asynchronous Messaging Engine

Figure 3. Architecture of device context discovery components

3.2. Component on broker side

The broker side of the discovery component consists of the Event Broker and the interface to the context manager. The Event Broker module plays a mediator role such as event filtering, event forwarding, and event redirection among mobile devices, environmental devices, and the broker.

4. Dynamic recognition and description of device capability and user profile

Within many kinds of device capabilities, we divided device capabilities into three properties which are static properties, dynamic properties, and user properties to stay with consistency. These properties are gathered by our device resource analyzer. Static device properties are stack of device resources that once it is initialized, it never changes during the session. Device model, CPU information, total memory, MAC address, etc. will be within this property. Dynamic device properties are changing device resources during the session such as current date and time, available memory, remaining battery, screen brightness, etc. Since we treat device location as one of the device capabilities, device location information is also one of these properties. User properties retrieved from the mobile device includes the user's name, email address, phone number, home address, etc. This information can also be used as a resource of context-aware services. User properties are different from the user profile because it is not presenting user preferences but just description of user information just by capturing from the device owner information.

User profile is collected by user's direct input before using any of applications. It is decided by each application's scenario and also can be used as a resource of context-aware services. In Section 5, example content of user profile is suggested by showing the UI for gathering user profile of our scenario.

4.1. Device resource analyzer

Figure 4 shows the implementation structure of the discovery component on the device side. Static/dynamic properties and user properties are dealt with by the device resource analyzer in Figure 3, while the location analyzer corresponds to the device location detector.

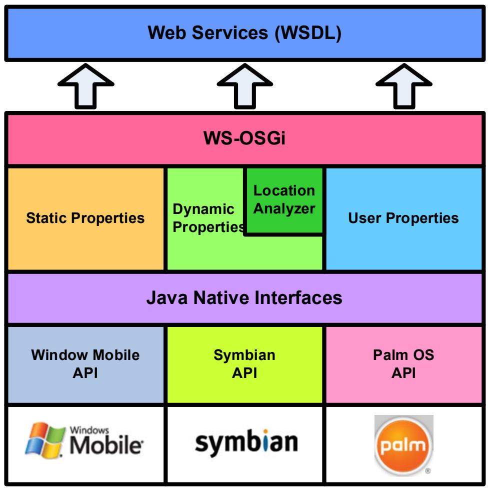


Figure 4. Implementation structure of discovery component on device side

We have experimented with the PDA (HP hx4700 series) for the user's device as it has enough system power to run our applications. Since our PDA runs on Windows CE 4.2, we used Microsoft PocketPC 2003 SDK and its system API to gather device capabilities – static/dynamic/user properties. We implemented this with JNI (Java Native Interfaces) to connect between the native code and our higher level of Java code so that we can transfer this information to our DCRD engine. We implemented all the components with Java except this device resource analyzer to meet the device heterogeneity. But because JNI is system OS dependent and for staying with this heterogeneity, we need to implement this functionality on various mobile device types, i.e., cell phone, PMP (Portable Media Player), etc., to support and each system OS should provide this information.

We are using the WS-OSGi for our common communication stack so that we can meet with standard JSR-172 mobile web services specification. WS-OSGi is an IBM technology that provides a method and system of mapping a web service to a OSGi service and of exposing the local OSGi service as one web service. It maps a web service with a OSGi service transparently using Java dynamic proxy technology to create a proxy bundle.

Figure 5 illustrates exposed WSDL (Web Services Description Language) by the implemented interfaces for getting the device and user properties. Celadon context acquisition module will access to this description when needed via web services and use this as a primary inference resource.

```
 … … …
 - <complexType name="UserProperties">
 - <sequence>
    <element minOccurs="1" maxOccurs="1" nillable="true"
       name="name" type="ns0:string" />
    <element minOccurs="1" maxOccurs="1" nillable="false"
       name="age" type="ns0:int" />
    <element minOccurs="1" maxOccurs="1" nillable="false"
       name="gender" type="ns0:boolean" />
    <element minOccurs="1" maxOccurs="1" nillable="true"
       name="occupation" type="ns0:string" />
  … … …
   </sequence>
   </complexType>
 - <complexType name="DynamicDeviceProperties">
 - <sequence>
    <element minOccurs="1" maxOccurs="1" nillable="false"
       name="availableMemory" type="ns0:int" />
      <element  minOccurs="0"  maxOccurs="unbounded"
       nillable="false"                  name="currentTime"
       type="ns0:int" />
      <element  minOccurs="0"  maxOccurs="unbounded"
       nillable="false"                  name="currentDate"
       type="ns0:int" />
    <element minOccurs="1" maxOccurs="1" nillable="false"
       name="batteryStatus" type="ns0:int" />
    <element minOccurs="1" maxOccurs="1" nillable="false"
       name="backlightIntensity" type="ns0:int" />
    <element minOccurs="1" maxOccurs="1" nillable="true"
       name="currentIp" type="ns0:string" />
   … … …
   </sequence>
   </complexType>
 - <complexType name="StaticDeviceProperties">
 - <sequence>
    <element minOccurs="1" maxOccurs="1" nillable="true"
       name="processorName" type="ns0:string" />
    <element minOccurs="1" maxOccurs="1" nillable="false"
       name="totalMemory" type="ns0:int" />
      <element   minOccurs="0"   maxOccurs="unbounded"
        nillable="true"          name="networkSupport"
        type="ns0:string" />
      <element   minOccurs="0"   maxOccurs="unbounded"
        nillable="true"             name="macAddress"
        type="ns0:string" />
  … … …
    </sequence>
    </complexType>
    </schema>
… … …
```

Figure 5. WSDL of exposed device and user properties

4.2. Device location detection using RSSI

There have been many studies for indoor device localization based on the networks including 802.11, Bluetooth, or RFID using signal strength. We also implemented the device location detection part using the Received Signal Strength Indication (RSSI) from 802.11-based access points and find out where the device and its owner are. Although almost all recent studies are using triangulation with probability analysis, currently we just compare the RSSI itself after filtering for avoiding unstable signal strength. The filter we used is a kind of complex filter that can reduce noise by avoiding unstable signal strength. The pseudo code of this filter is shown in figure 6 below.

```
N: # of most recent RSSI samples
RSSI_i: i-th past RSSI sample among N most recent samples (i = 1, 2, …, N)
[RSSI_i]: array of N RSSI_i's for i = 1, 2, …, N
rssi: current RSSI sample
TH: threshold for determining abrupt changes in RSSI
RSSI_C: filtered current RSSI for use in location determination

begin Complex Filter
  $RSSI_N \leftarrow RSSI_{N-1}$ , $RSSI_{N-1} \leftarrow RSSI_{N-2}$ , … , $RSSI_2 \leftarrow RSSI_1$ , $RSSI_1 \leftarrow rssi$
  if( $|RSSI_1 - RSSI_2| > TH$ )
    $RSSI_C \leftarrow$ median([$RSSI_i$] )
  else
    $RSSI_C \leftarrow$ mean([$RSSI_i$] )
end
```

Figure 6. Pseudo code of the applied complex filter for device location detector

As the primitive way of device localization, a location of the device is determined to be a section where the RSSI from one of the access points (APs) shows the strongest value. Thus we assume that a zone can be divided into several sections with the number of APs in the zone. Among a bunch of APs already deployed around, it is possible to detect section-belonging access points by only taking the RSSI from our special SSID (Service Set Identifier). Further study is under way to determine more precise and dense location and to use less access points rather than using one access points corresponding to one section. In addition, to support network heterogeneity, we are trying to implement the same functionality in other network environments such as Bluetooth.

After each device property is obtained by the Device Resource Analyzer and the Device Location Detector, the data is transferred to the DCRD engine for the effective management of the obtained properties.

4.3. DCRD engine: management of device and user properties

The DCRD should be more than just a static list of device characteristics. Rather, we need access methods for manipulating static and dynamic properties so that any contents delivered to the device can be adapted to a particular device context. The active instances – OSGi bundles, and Eclipse plugins, etc. – that can deal with device capability recognition and description should provide other active instances with dynamic access to a hierarchy of properties representing the current device capabilities. For dynamic properties such as the location of a mobile device, it is important to be able to respond to changes when they occur. Consequently, a mechanism to subscribe and unsubscribe to specific events is required. The DCRD concentrates on the following two things:

- Interfaces by which access to static and dynamic properties corresponding to the device capabilities are obtained in programming environments
- Mechanisms used to represent the static and dynamic properties of the mobile devices

A concept behind the DCRD information model is depicted in Figure 7. The information model takes full advantage of the separation between data manipulation and data access. The data manipulation means to initialize/de-initialize the properties and to raise events for notifying changes to property values. The data access includes:

- Generic read/write operations on device properties.
- A form of access control that is needed along with the means to ensure that integrity constraints are upheld. There are instances where certain components should be prevented from modifying property values or adding/removing properties. Access rights are also useful for restricting event handling.

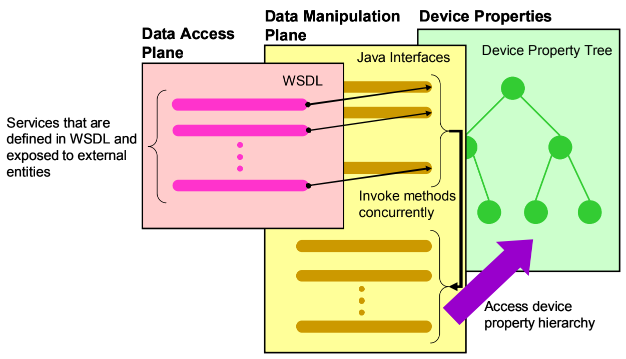

Data Access Plane
Data Manipulation Plane
Device Properties
Java Interfaces
Device Property Tree
WSDL
Services that are defined in WSDL and exposed to external entities
Invoke methods concurrently
Access device property hierarchy

Figure 7. DCRD information model

4.4. Event agent for handling changes in dynamic properties

Since dynamic device properties are changing in time, messaging method should be different from other properties. We use asynchronous messaging to handle this part. Firstly one method of the dynamic device properties is published to the web services, any module subscribes to that method (topic) can be notified by asynchronous messaging module. We defined topics for dynamic device properties and implemented asynchronous messaging part with WSRF plugins for OSGi (WSRF4OSGi) [13, 14]. WSRF4OSGi is a lightweight implementation of WS-Resource Framework (WSRF) set of specifications along with the WS-Notification and set of specifications in the OSGi environment. The WSRF4OSGi provides an environment whereby various web service clients can access stateful resources running in an OSGi environment using WSRF standards [14]. After considering the advantages and disadvantages of the WSRF4OSGi, we decided to approach SOAP optimization, lessening message overhead over the network, keeping web service available on the mobile side. Figure 8 shows our event agent based on conventional event-driven architecture.

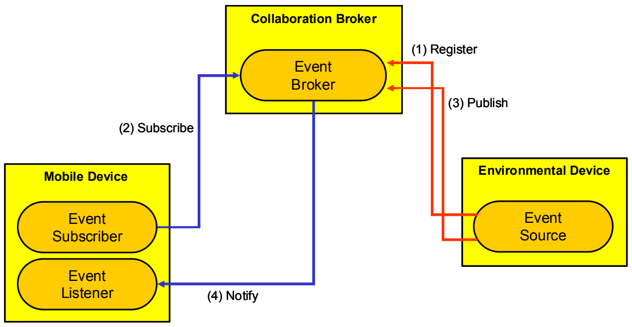


Figure 8. Event agent diagram

4.5. Advantages of suggested DCRD engine

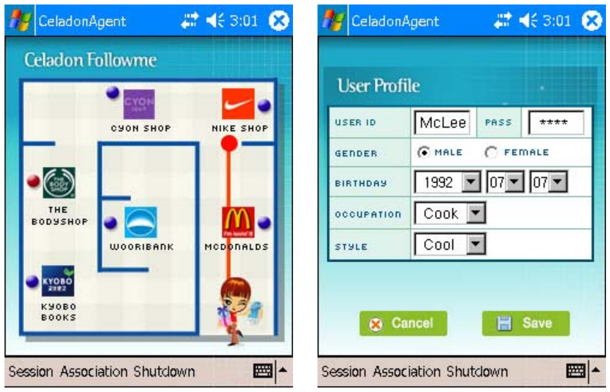
We do not focus on how we use context information to accomplish inference or to match service instances, but propose how we gather and manage device/user context. Since device resources should be managed efficiently in the mobile ubiquitous environment, efficient resource management and communication can be achieved by well-defined infrastructure in the mobile ubiquitous environment. Followings can be advantages of our proposed DCRD architecture.

- Using OSGi standard makes it possible to easily manage of services with platform independence.
- Using JNI enables us to achieve device heterogeneity because of using Java on the top of it.
- Exposing device/user context into the web services makes easy access to a context-aware system at any location, and using various contexts from system OS provides richer context-aware services.
- Property management by DCRD enables efficient management of device/user context, especially in the mobile environment.
- Event manager lets us to use less network resources and the information can be updated in real time whenever it's needed.

5. Implementation: context-aware services based on device/user context

5.1. Implemented scenario: Location-aware services

Device location is one of the important information for the context-aware services. It is applicable for many fields such as location guide services, personal preference analysis, mobility analysis and many others. For our Celadon project, we have built a demo test bed on the scenario including location guide service with context awareness. A captured scene on the mobile device of the location guide service is depicted in Figure 9(a). We divided our demo room into three parts, each of which belongs to a shopping spot with a corresponding access point. (We consider only 3 shopping spots of the map in the screen of Figure 9(a).) If a user enters the Celadon zone, the device (currently PDA) is registered to the Celadon broker and it pushes all the device and user properties to the context engine on the broker. Based on the Celadon context module using IODT (Integrated Ontology Development Toolkit) [15], the user preference is inferred and a recommendation is provided. The location guide service is provided based on our device location information after the user selects what he wants to purchase. For example, the final destination of shop and the shortest path to the shop is expressed on the map of user's mobile device, and current location is also indicated. While user is on his way, personalized advertisement is popped up on the environmental device around when the user approaches to some specific area.



User Profile

USER ID	McLee	PASS	****
GENDER	[✓ MALE] FEMALE		
BIRTHDAY	1992 07 07		
OCCUPATION	Cook		
STYLE	Cool		

(a) Location guide service (b) User profile configuration

Figure 9. Captured screenshot of Celadon context-aware services

5.2. Further implementation: User-aware services

Other than device properties, user properties from the mobile device which the user has already provided are also applicable to many ways. Pushing these properties to the context engine, we can provide the user more user-specified and more convenient services. For example, by knowing user's home address, the items can be shipped without asking the user to enter the home address. By knowing time of day, we can provide user the restaurant menu for the appropriate time, e.g., the lunch menu or the evening special menu. By sharing company name the user might be able to get company discount price for a museum instead of regular price.

Figure 9(b) illustrates a captured screen of configuring user profiles on the mobile device. The user can input at any time his/her profile such as personal information and preferences. This user context is also delivered to the context manager as soon as the user with the mobile device enters the Celadon zone, and can be exploited for provision of context-aware services.

6. Conclusions and future work

In this paper, we addressed device context recognition and description system for context-aware services. The system is one part of the Celadon framework and implemented within the architecture. The proposed system gathers device and user properties, and then manages and handles these properties with the DCRD engine based on DCI. Location context of the device is also included and treated as one of the dynamic device properties. All properties are exposed to the web services and especially for the dynamic device properties, asynchronous messaging is used.

Further important development is being considered including privacy and security issues, role policy by credentials, following the international standard for example JSR-179 for the localization, etc. The enhancement of localization algorithm is also under way.

7. Acknowledgment

This work is supported by the IT839 project from the Institute of Information Technology Assessment and Ministry of Information and Communication in Republic of Korea.

8. References

[1] Graham Klyne, et al., "Composite Capability/ Preference Profiles (CC/PP): Structure and Vocabularies 1.0," http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/.
[2] Dan Brickley and R.V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
[3] "Wireless Application Group User Agent Profile Specification," http://www.wapforum.org/what/technical/ SPEC-UAProf-19991110.pdf.
[4] Keith Waters, et al., "Delivery Context: Interfaces (DCI) Accessing Static and Dynamic Properties," http://www.w3.org/TR/2005/WD-DPF-20051111/.
[5] C. Narayanaswami, M.T. Raghunath, M.C. Rosu, H.K. Jang, S.E. Jin, S.Y. Kim, M.C. Lee, S. Lee, and Y.S. Paik, "Device Collaboration for Ubiquitous Computing: Scenarios and Challenges," The First Korea/Japan Joint Workshop on Ubiquitous Computing and Networking System, 2005.
[6] M.C. Lee, H.K. Jang, S.Y. Kim, Y.S. Paik, S.E. Jin, S. Lee, C. Narayanaswami, M.T. Raghunath, and M.C. Rosu, "Celadon: Infrastructure for Device Symbiosis," The Seventh International Conference on Ubiquitous Computing, 2005.
[7] Brad A. Myers, "Using Handhelds and PCs Together," Communications of the ACM, vol. 44, no. 11, pp. 34-41, 2001.
[8] Tomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, and Josef Altmann, "Context-Awareness on Mobile Devices – the Hydrogen Approach," Proc. Of the 36th Hawaii Int. Conf. System Sciences (HICSS), 2002.
[9] Manuel Roman, Christopher Hess, Renato Cerqueira, Anand Ranganat, Roy H. Campbell, and Klara Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," IEEE Pervasive Computing, pp. 74-83, 2002.
[10] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg, "A Survey on Context-Aware Systems," Int. Journal of Ad Hoc and Ubiquitous Computing, 2004.

[11] Tao Gu, Hung Keng Pung, and Da Qing Zhang, “Toward an OSGi-Based Infrastructure for Context-Aware Applications,” IEEE Pervasive Computing, pp. 66-74, 2004.
[12] Tao Gu, Hung Keng Pung, and Da Qing Zhang, “A Middleware for Building Context-Aware Mobile Services,” Proc. of IEEE Vehicular Technology Conference (VTC), 2004.
[13] http://www.alphaworks.ibm.com/tech/wsrf4osgi
[14] Manu Kuchhal and Umesh Joshi, “WSRF4OSGi Programming Guide,” Technology Incubation Center IBM Software Labs, India.
[15] http://www.alphaworks.ibm.com/tech/semanticstk