2005－UBI－8（79）
2005／6／10

wear-UCAM: A Toolkit for Wearable Computing*

Dongpyo Hong and Woontack Woo
GIST U-VR Lab, Gwangju 500-712, S. Korea
{dhong, wwoo}@gist.ac.kr

Abstract

In this paper, we propose wear-UCAM, which is a toolkit for Unified Context-aware Application Model in wearable computing. As the rapid developments of mobile technologies and relevant technologies, the interests in wearable computing also become popular in both academic and industrial fields. Unfortunately there are few research activities on the application toolkit for wearable computing to support application developers not to take care the low level issues. In this regard, we focus on how to manipulate user relevant context from sensors or services, how to manage it, and how to provide services based on the managed user relevant context because personal data (health, preferences, activities etc) are crucial for the personalized services. For the sake of rapid prototyping of the proposed toolkit, we first utilize PDA with wireless LAN as a wearable computer and experiment it in ubiHome, a smart home environment test-bed in GIST U-VR Lab.

Key words: Rapid Development Toolkit, Wearable Computing, Ubiquitous Computing, Context

1. INTRODUCTION

In recent the deployment of ubiquitous computing enabling technologies and their rapid development, the interests on wearable computing has been drawing attentions in research and commercial realms [1]. In general, it is required for ubiquitous computing to install various sensors to serve users in environmental infrastructure [2]. Consequently, it is not easy to manage the personal information as well as to protect privacy from the environmental monitoring [3].

In wearable computing, however, personal information can be retrieved from various sensors, analyzed and used in services through explicit controls of a user rather than environment [3][4]. In another word, personal information is controlled by wearable computer itself rather than by servers or other devices in environment such as a centralized server. In this regard, wearable computing should support users' personal information to be protected from environmental sensors as well as provide quality of services to users based on the preference of the user. This is the reason that wearable computing becomes focused in ear of ubiquitous computing environment. In addition, wearable computing complements the inherent limitation of ubiquitous computing such as cost for infrastructure construction, privacy protection and so on. Nonetheless, there have been a few research activities on application development toolkit for wearable computing compared to ubiquitous computing [5-7]. Furthermore, it is burdensome for developers to adopt toolkits for ubiquitous computing into wearable computing because toolkits for ubiquitous computing include many mechanisms for sensors to detect who we are. Therefore, it is necessary of a toolkit to support required features in wearable computing.

We propose wear-UCAM, which is a toolkit for rapid development application in wearable computing. The proposed wear-UCAM is to provide developers with acquisition methods of personal information from various sensors, manipulation and analysis of the acquired information, and implementation flows of applications for wearable computing in ubiquitous computing environment. The proposed wear-UCAM has the following 3 characteristics: 1) Acquisition and analysis of user's physiological signal, 2) User profile management from extracted and analyzed information, and 3) Privacy protection from other wearable computers or environmental sensing equipment.

As a toolkit, the proposed wear-UCAM provides procedural abstraction when processing the user's context in sensor or service. To provide procedural abstraction of processing for the user's context, wear-UCAM exploits various design patterns among sensors and services when they are dealing with context [8][9]. Moreover, wear-UCAM splits application into wearSensor and wearService in order to support independence of sensors and services. This separation of sensors and services is also a key feature of ubiquitous computing like distributed computing environment [5]. Along with the separation, wear-UCAM includes networking interface to support dynamic connections between sensors and services, and it also exploits the formatted context so as to guarantee the integrated data form among sensors and services. Likewise, there are many components in wear-UCAM to support extracting and analyzing user's context from various sensors. Therefore, the proposed wear-UCAM is to be a rapid development toolkit for developers who want to implement user centered applications for wearable

* *This work is supported by Seondo project of Ministry of Information and Communication in Korea*

computing in ubiquitous computing environment. In this paper, we reveal the fundamental architecture and design of wear-UCAM as well as the empirical implementation issues and experimental results. To show the effectiveness of wear-UCAM, however, we utilized wireless LAN and Bluetooth enabled PDA as a wearable computer.

This paper is organized as follows. In section 2, we explain the overall architecture and design issues of wear-UCAM and its components. In section 3, it is explained that the experimental setup and result with detailed analysis of wear-UCAM. Finally, we discuss the proposed wear-UCAM as a toolkit for wearable computing application and future works.

2. wear-UCAM

The fundamental architecture of wear-UCAM is conceptually illustrated in Figure 1.

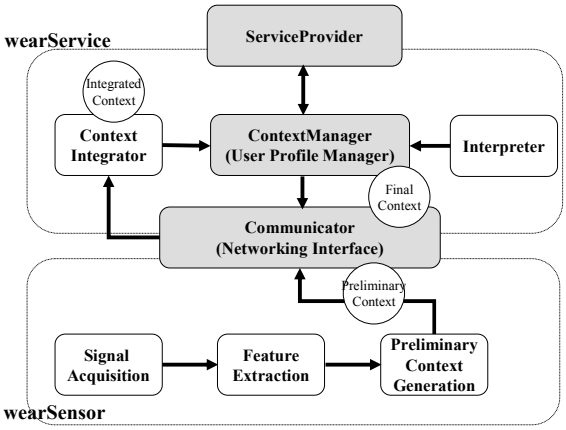

Figure 1: Conceptual wear-UCAM Architecture

As shown in Figure 1, wear-UCAM forces application developers to split wearSensor and wearService in an application as well as each wearSensor and wearService to have communicator (networking interface). This separation in the application as sensor and service is a needed mechanism for resource restricted computing environment like wearable computing because we can utilize not only wearable sensors but also other high powered sensors in environment. Furthermore, wear-UCAM keeps data exchanges in the formatted context between each component and between sensor and service, which is the same structure as ubi-UCAM [5]. However, personal context which is highly relevant to personal privacy is protected from other wearable computers and environmental monitoring by the explicit controls of the user.

2.1 Software Architecture

In order to explain the architecture of wear-UCAM, we need to introduce the UCAM[1] briefly. For context-aware applications in the future computing, application developers are able to manipulate user's context in order to provide user friendly applications. However, the application developers should not know the very low detail of context-aware mechanism while they implement context-aware applications. Therefore, it is necessary to guide developers the way to implement context-aware applications.

wear-UCAM is a part of UCAM, and also a toolkit to support developers who want to develop a prototyping application rapidly in wearable computing environment. Figure 2 illustrates hierarchical structure of software components in UCAM.

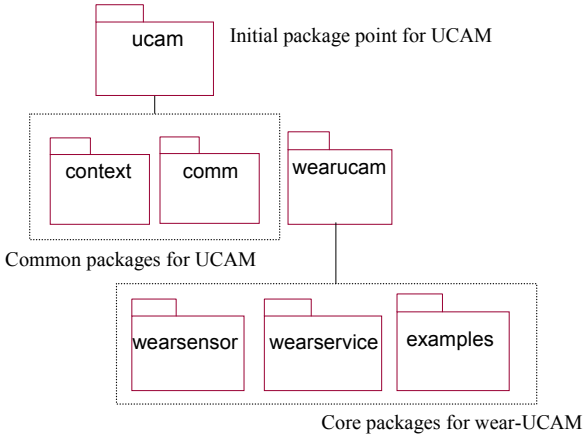

Figure 2: Software Package for wear-UCAM

As illustrated in Figure 2, wear-UCAM exploits the common packages: **context** package for the formatted context representation and **comm** package for networking interface like in UCAM package. Specifically, sensors and services are able to communicate each other with context as the common data representation due to **context** and **comm** packages. The key packages of wear-UCAM are **wearsensor**, **wearservice**, and **examples**, and they are packed in **wearucam**. Like Figure 2, **wearucam** includes various classes to support sensors and services which are applicable in wearable computing environment, and also most of the classes in **wearucam** are applied procedural abstraction. Therefore, developers are able to implement their applications in wearable computing without knowing the details of context-aware mechanism because wear-UCAM provides a guideline to implement. For instance, wearSensor extracts signals from sensors and makes preliminary context by analyzing the retrieved signals. Therefore, sensor developers are

[1] **UCAM** stands for *Unified Context-aware Application Model*. We have been developing 3 different kinds of UCAM since 2002. For example, ubi-UCAM is for ubiquitous computing environment, wear-UCAM is for attentive wearable computing, and vr-UCAM is reactive virtual environment.

forced to implement the required functionalities in wear-UCAM. Furthermore, we separate types of sensors as we support various sensors in wearable computing applications.

In short, the architecture of wear-UCAM enables wearable computing application developers to exploit context-aware mechanism as well as supports communications among heterogeneous applications through the formatted context and networking interface in heterogeneous computing environment.

2.2 Procedural Abstraction

As we mentioned briefly in the previous section, it is necessary that a sequence of context processing in order to make user's context from various sensors and utilize the context. In this regard, wear-UCAM adopts various design patterns [8] for wearable computing application developers to implement context-aware applications effortlessly. The proposed wear-UCAM exploits a concept of procedural abstraction in the sequence of processing context [10].

Figure 3 depicts UML representation of the source code and how the procedural abstraction is applied to the processing context in service.

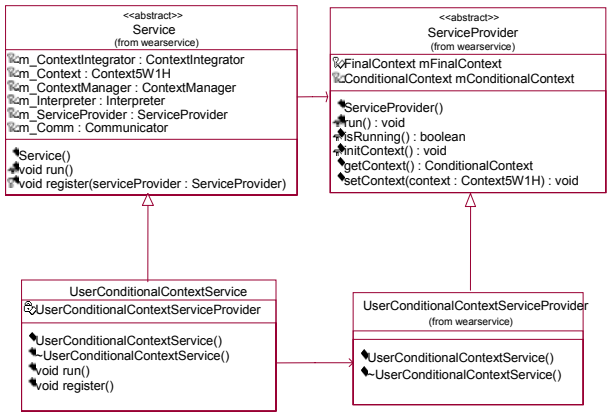

Figure 3: An Example of Procedural Abstraction

Figure 3 shows an example of procedural abstraction in the implemented *Service* and *ServiceProvider* of **wearservice** package. In wear-UCAM, service includes context processing components and is always in the waiting state to send and receive user's context. Moreover, *Service* hides the detail implementation of context processing procedures from application developers. However, wear-UCAM forces application developers to implement *ServiceProvider* specifically because it is the only interface to be open for application developers and it is automatically registered into *Service*. Therefore, *ServiceProvider* must be implemented if developers want to supply a specific service, for example, music service based on user's stress level. Those requirements must be satisfied to provide service execution, and thus service is designed as shown in Figure 3. As the designed service in wear-UCAM, the explicit implementation of *Service* and *ServiceProvider* can be different in each application domain, but the procedure of using context is the same in all the services. This inspires us to exploit procedural abstraction in sensor and service. The advantage of procedural abstraction is to share the same procedure in applications, but is able to implement various methods in the applications. Meanwhile, wear-UCAM categorizes various services in wearable computing applications by extending abstraction concept [11]. This makes that a service should not restricted in one category, but should have multiple characteristics. Abstraction is a necessary technique in not only wear-UCAM, but also other application development toolkit.

2.3 wearSensor and wearService

As shown in Figure 1, wear-UCAM requires sequences of context processing in order to exploit user's context in applications, in which user's context is acquired from various sensors and it is utilized in services. Table 1 shows the core components for user's context processing with respect to sensor and service.

Table 1. Components' Functionalities in wear-UCAM

wear-UCAM	Component	Functions
wearSensor	*Sensor**	Basic functions of wearSensor (signal processing, feature extraction, preliminary context)
Networking	Communicator	Dynamic connection among sensors and services
wearService	*Service**	Basic functions of wearService (service classification, register ServiceProvider)
	ContextIntegrator	Integrated context by analyzing preliminary context
	ContextManager	Final context by manipulating user's preference and the integrated context
	Interpreter	User's conditional context provision
	*ServiceProvider**	Basic function for developers who develop context-aware application

* *represents Abstract class*

In case of wearBioSignalSensor which is a specific instance of Senor and exploits components in wearSensor of wear-UCAM, it acquires physiological signals, extracts features for the signals, and transforms

the features into values. The transformed values are used to make preliminary context. The preliminary context is sent to sensors or services through networking interface [12]. In **comm** package, Communicator provides dynamic connections among sensors and services to deliver the formatted user's context. Communicator includes context filtering, dynamic multicasting group creation, and trivial networking functionalities to deliver contexts [13]. Then, service provides appropriate service to a user through the process and manipulation of contexts from sensors and other services. ContextIntegrator in wearService integrates user's physiological signal information from sensors or context from other services, and then make integrated context. ContextIntegrator makes the integrated context by applying integration algorithm as explained in [14]. On the other hand, ContextManager provides final context through the comparison of the delivered context from Interpreter (user's conditional context) and the integrated context [15]. Interpreter in wear-UCAM take responsible to update and manage user's profile [16].

In this paper, we reuse the user's context representation and communication method in ubi-UCAM, where components send and receive their data as the formatted context [5]. However, the presentation of context in ubi-UCAM is modified to fit into wearable computing environment. For example, contextual information relevant to user like physiological signal is added.

3. EXPERIMENTAL RESULTS

Given services in ubiHome, we simply replace current PDA-based user interface with wear-UCAM embedded PDA [15]. Figure 4 depicts the experimental setup.

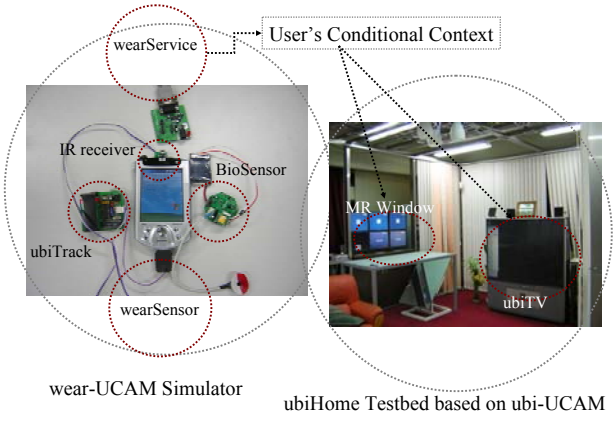

Figure 4. Experimental Setup

As shown in Figure 4, we exploit the processed physiological signal, which tell whether a user is stressful or not and location sensor, called as ubiTrack [17], which tracks user's location in ubiHome. Furthermore, we used 2 services in ubiHome such as MR Window and ubiTV, and also we followed the experimental scenario according to [11].

Table 2 shows the specifications of wear-UCAM simulator as well as the exploited sensors and services.

Table 2. Hardware and Software for wera-UCAM

Hardware		
Device	**Function & Role**	**Specs**
iPAQ h5550	wear-UCAM Simulator	PocketPC 2003 RAM: 128MB ROM: 48MB
BioSensor	Extract physiological signal	Bluetooth
ubiTrack	Track user's location	Serial communication
Software		
Java	Desktop development toolkit	J2SE (Java1.5)
Personal Java	PDA development toolkit	Java1.1.8 compatible
wear-UCAM	Toolkit for wearable computing	v0.9a1 (118KB .jar)

To show the effectiveness of the proposed wear-UCAM as a toolkit, we simulated it with various virtual sensors but actual services. However, we only focused on wear-UCAM itself rather than whole scenario which includes ubi-UCAM. The processing time for making context in wear-UCAM from sensor to service is follows.

$$FC(\Delta T) = PC(\Delta T_1) + Network(\Delta T_2) + IC(\Delta T_3) + UCC(\Delta T_4) \quad (1)$$

where final context processing time, FC, is mostly related to preliminary context (PC), network delivery (Network), integrated context (IC), and user's conditional context (UCC) generation time.

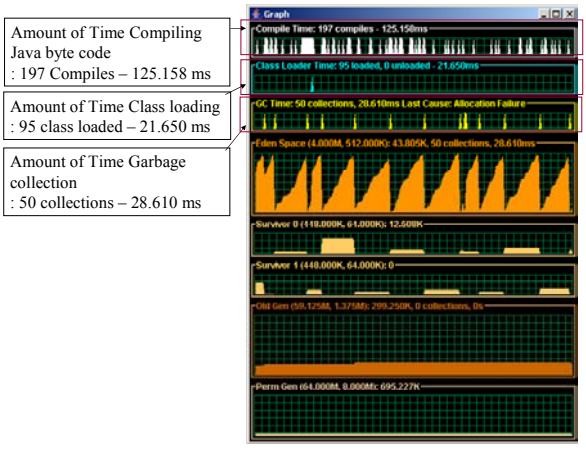

Figure 5. visualgc Monitoring Tool: Example of service

However, it is difficult to measure each component's context generation time because wear-UCAM is designed for distributed computing environment and each component is implemented as a Thread. Thus, we exploited Java monitoring tools, such as jps, jstat, and

visualgc[2] [18]. Figure 5 and Figure 6 shows visualgc monitoring tool and jstat monitoring tool respectively.

As shown in Figure 5, the compiling time to Java byte code is shown as the accumulated time during the execution meanwhile the compiling time in jstat monitoring tool indicates each time of compiling step (in Figure 6). Comparing to visualgc, jstat include more powerful options to monitor Java application, but it is a text-based software tool. Figure 6 show jstat monitoring tool.

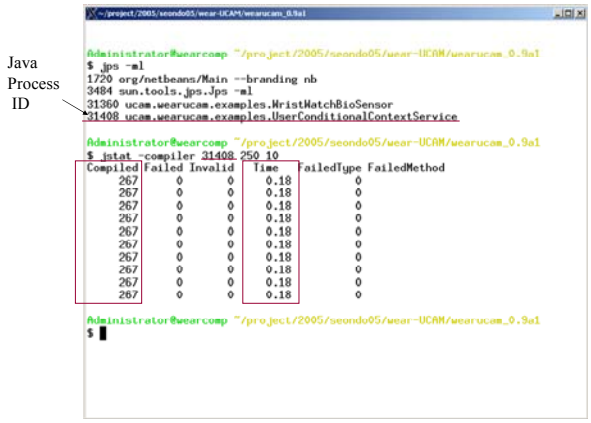

Figure 6. jstat Monitoring Tool: Example of service

Figure 5 and 6 shows a partial result of the simulation, which illustrate compile time, class loader time, memory usages and context processing performance of wear-UCAM with respect to Equation (1). With help of jstat monitoring tool, we also observed several properties on sensor and service. First of all, compile time of the service is dependent to the number of compilation tasks performed. For example, when the number of compilation tasks performed is 267, compile time is 0.18ms. However, the number of compilation tasks performed and its compiler time are converged after a period of time. In the case of class loader time, sensor requires 51 loaded classes and its time is 0.01ms. On the other hand, service requires 95 loaded classes and its time is 0.02ms. In both case, the loaded byte is 45.9KB and 91.7KB, respectively. Regarding to memory usage, we take current permanent space capacity as a clue and it is required 8192KB in this simulation.

Through the experiment, the generation of final context is closely relevant to user's conditional context in wear-UCAM. This is because final context is generated if and only if user's conditional context and integrated context, which is produced from sensors, are matched. Therefore, it is required that the matching algorithm in ContextManager should be well designed as well as user's conditional context in ServiceProvider should be provided flexibly when we want to increase the matching rate of final context. Furthermore, the integration method in ContextIntegrator should deal with unknown fields of preliminary context within a proper time window.

Lastly, we would like to point out several issues when we implement and develop context-aware applications with the proposed wear-UCAM. wear-UCAM is packed .jar file format in order to be easily distributed and keep code organized with compressed form. As shown in Table 2, the package size of the current version of wear-UCAM is 118 KB, which is compact size enough to be deployed in the restricted computing environment like wearable computing if we consider the current development of technologies in computing resources. However, it is because the current version of wear-UCAM does not include hardware drivers to be directly used in sensor or service. In the future release, we will include hardware drivers.

In the case of implementation programming language in wear-UCAM, we adopt Java™ because it can be easily deployed into various platforms and well defined language to realize object-oriented concepts. However, there are a few difficulties when we apply it into wear-UCAM. Currently we utilize Personal Java™, which is not supported any more and restricted features comparing to currently released J2ME version for mobile applications. The other problem is that most of hardware interface and real time signal processing are implemented in C or C++. In fact, we adopt network communication to interface hardware sensor and software wearSensor. Thus, we have to take into account of hardware interface with Java features.

4. DISCUSSION AND FUTURE WORKS

In this paper, we proposed a toolkit, wear-UCAM, for rapid application development in wearable computing. The proposed wear-UCAM could provide appropriated services to users based on their preferences because wear-UCAM supports retrieving personal information from sensors, processing it, and analyzing it while taking into account of privacy in ubiquitous computing environment. Furthermore, the proposed wear-UCAM provides the abstraction in processing context from sensors and services as well as supports independence of sensors and services in applications by splitting wearSensor and wearService. However, wear-UCAM is not yet experimented on actual wearable computer. Thus, it is required to conduct experiment and analysis on wear-UCAM based on software engineering aspect with apparel type of wearable computer.

[2] In current distribution of **visualgc** tool, there is a bug on batch file named "**visualgc.cmd**" for Windows. You have to replace %JVMSTAT_JAVA_HOME% with %JAVA_HOME in line 51.

REFERENCE

[1] M. Weiser, “Some computer science issues in ubiquitous computing”, Communications of the ACM, vol. 36, no. 7, pp. 75-84, July 1993.

[2] Guanling Chen, and David Kotz. “A Survey of Context-Aware Mobile Computing Research”, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November, 2000.

[3] Bradley J. Rhodes, Nelson Minar and Josh Weaver, "Wearable Computing Meets Ubiquitous Computing: Reaping the best of both worlds", Appears in The Proceedings of The Third International Symposium on Wearable Computers (ISWC '99), San Francisco, CA, October 18-19 1999, pp. 141-149.

[4] Jennica Falk,Staffan Björk, “Privacy and information integrity in wearable computing and ubiquitous computing,” Conference on Human Factors in Computing Systems archive CHI '00, pp.177-178, 2000.

[5] S.Jang, W.Woo, "ubi-UCAM: A Unified Context-Aware Application Model," Lecture Note Artificial Intelligence, Vol.2680, pp. 178-189, 2003.

[6] Anind K. Dey and Gregory D. Abowd, "The Context Toolkit: Aiding the Development of Context-Aware Applications", In the Workshop on Software Engineering for Wearable and Pervasive Computing, Limerick, Ireland, June 6, 2000.

[7] Bill Schilit and Norman Adams and Roy Want, "Context-Aware Computing Applications",IEEE Workshop on Mobile Computing Systems and Applications,1994.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley. October 1994.

[9] Fowler, Martin, "UML distilled: applying the standard object modeling language", Addison-Wesley, 2004.

[10] Wizard Book n. Hal Abelson's, Jerry Sussman's and Julie Sussman's Structure and Interpretation of Computer Programs (MIT Press, 1984; ISBN 0-262-01077-1).

[11] C.Shin and W.Woo, "Conflict Resolution among Users for Context-aware Media Services", KHCI2005, pp. 594-599, 2005.

[12] A.Choi and W.Woo, "Feature extraction for emotion analysis based on physiological signal pattern", KHCI2005, pp. 624-629, 2005.

[13] Sj.Oh, Y.Lee, and W.Woo, "Dynamic Network Reconfiguration for Seamless Interactions between Real and Virtual Environments", KHCI2005, pp.721-726, 2005.

[14] Y.Oh, W.Woo, "A unified Application Service Model for ubiHome by Exploiting Intelligent Context-Awareness", UCS2004, pp.117-122, 2004.

[15] C.Shin, W.Woo, "Conflict Resolution Method using Context History for Context-aware Applications," Pervasive 2005 workshop, accepted, 2005.

[16] Y.Suh, W.Woo, "User Profile Management for Context-aware Applications for ubiHome environment," ubiCNS05, accepted, 2005.

[17] S.Jung, W.Woo, "UbiTrack: Infrared-based user Tracking System for indoor environment," ICAT04, pp. 181-184, 2004.

[18] http://java.sun.com/performance/jvmstat/