A Software Infrastructure for Wearable Sensor Networks

Kensuke Hanaoka Ayako Takagi Tatsuo Nakajima
Department of Computer Science, Waseda University
{*kensk,ayako,tatsuo*}*@dcl.info.waseda.ac.jp*

Abstract

In ubiquitous computing environments, context-awareness is one of the most important research topics. Computers embedded in our surrounding can extract information about a user, and the information makes it possible to offer personalized services according to the user's preference. To extract a large amount of context information, wearable sensor devices will become more important in the near future. However, it is not easy to develop context-aware services that use context information from wearable sensor devices because of the gap between low-level sensor information and high-level context that the services require.

In this paper, we propose a software infrastructure for wearable sensor networks. We first discuss the requirements to retrieve context information from wearable sensor networks. Then, we introduce our software infrastructure named Cinnamon that extracts high-level context from low-level context information retrieved from wearable sensor networks. The software infrastructure makes it dramatically easy to develop context-aware services for wearable sensor networks. We present the design and implementation of Cinnamon, and discuss our current prototype implementation.

1. Introduction

To realize the vision of ubiquitous computing environments, various kinds of sensors and processors will be embedded in our daily lives. These embedded computers can offer the most appropriate services to a user according to context information in our surrounding [17]. For example, a context-aware mirror service offers weather or traffic information to a person in front of it, based on context information such as his/her today's schedule [2]. A service customized by context information is referred to a context-aware service. In the future, we will need to process a large amount of information, and context awareness is effective to reduce complexities and stresses in our daily lives [1][13][14].

Understanding a user's situation is very effective to develop context-aware services. Wearable sensor devices are important means to retrieve a user's situation, because the wearable devices can be attached to our bodies, and retrieve context information at anytime and anywhere without the user' notice. In the past study, we had experienced with a mobile terminal with fifteen types of sensors named *Muffin* to retrieve context information [19]. In the course of the study, we found that such a mobile terminal style had serious limitations in terms of retrieving accurate context information, because the sensors on Muffin only work if a user holds it correctly in his/her hand. On the other hand, wearable sensor devices remove the limitations, and allow services to take into account continuous context information retrieving.

The paper proposes a software infrastructure for building context-aware services for wearable sensor networks. Our software infrastructure named *Cinnamon* extracts high-level context from low-level context retrieved from wearable sensor devices. The context-aware services can take into account only high-level context information, and it is possible to make it easy to build the services. Our software infrastructure assumes a wearable sensor network that consists of two kinds of devices. One is wearable sensor device that contains a number of sensors, and the other is a host mobile terminal that has enough power for data analysis to extract high-level context. In a wearable sensor network, several wearable sensor devices are attached to a user's body, and they transmit the data to a host mobile terminal carried by a user, and analyzes the data to detect the user's current situation.

We have currently implemented a prototype system of *Cinnamon*. In the current prototype system, we use *Cookie* and *Muffin* that have been developed in the collaboration research with Nokia Research Center, Tokyo. *Cookie* is a wearable sensor device, and *Muffin* is a powerful host terminal that uses embedded Linux as an operating system. *Cinnamon* is running on *Muffin* and *Cookie* transmits sensor information through a bluetooth wireless interface.

The organization of the paper is as follows. Section 2 describes some related work, and we show the requirements to

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

offer software infrastructures for wearable sensor networks in Section 3. In Section 4, we present the design and implementation of *Cinnamon*. We also introduce *Cookie* that is a wearable sensor device used in our prototype system. Then we show a sample service developed on *Cinnamon*, and the evaluation of the service and *Cinnamon* in Section 5. Section 6 discusses some experiences with building the current prototype system, and we summarize the paper in Section 7.

2. Related Work

There is a large number of research on context-aware systems [3][10][12]. Context Toolkit is the best known software infrastructure for building context-aware services [13]. Context Toolkit provides *context widgets* that are the abstractions of sensor devices. Winograd proposes a framework based on the blackboard architecture [18], and Hong proposes a client-server system infrastructure where privacy issues are taken into account [7]. However, those systems do not support frameworks to analyze raw sensor data to generate high-level context information explicitly. On the other hand, *Cinnamon* offers a couple of methods to analyze raw sensor data to generate higher level context information. This is very important to build context-aware services for wearable sensor networks in an easy way.

There are some general purpose sensor devices for realizing ubiquitous computing visions. Smart-Its [5] is a typical example of the devices. *Smart-Its* is a small sensor device that can be embedded into daily objects or put on our body. The device has a wireless network interface to communicate to computers on which application services run. There are many studies to develop context-aware services with Smart-Its, but there is no software infrastructures to analyze sensor data in a generic way. *Smart-Its* is very similar to *Cookie*. However, it is not easy to use *Smart-Its* in our prototype implementation. Thus, we decided to develop *Cookie* with NRC, Tokyo. We believe that our work is unique to offer software infrastructures to analyze sensor data for wearable sensor networks, and our work compliments the previous work.

We have developed a personal mobile terminal called Muffin that contains fifteen types of sensor devices to retrieve context information [19]. Since a personal terminal is used in our daily lives, it is desirable to offer context-aware services customized according to a user's current situation. We could retrieve some environmental contexts such as temperature and humidity, a user's contexts such as the speed at which a user is walking, and a mobile terminal's contexts such as the orientation of the terminal from Muffin. However, it is difficult to retrieve context information anytime since Muffin assumes that a user holds it in his/her hand, and it is sensitive to the positions of fingers and the style of the holding of Muffin. Also, one hand is inevitably disabled because the hand is used to hold the terminal. Thus, the types of context information that can be retrieved are limited.

Korpipaa also proposes a software framework to develop context-aware services for mobile devices, and its framework is very similar to ours [9]. The framework supports the transformation of a continuous sensor data stream into abstract context information. The framework also supports ontologies to describe high-level context information. The difference of the framework and *Cinnamon* is that our software infrastructure can use both sensors embedded in a mobile terminal and wearable sensor devices. This approach makes it possible to extend to use sensor information from sensor devices embedded in our surrounding in an easy way. Our group is also working on sentient artefacts [6] that are enhanced everyday artefacts that contain sensors and computers. *Cinnamon* extract context information from sentient artefacts, and integrated with context information from wearable sensor devices.

3. Wearable Sensor Network

In this section, we introduce wearable sensor networks and discuss the requirements to offer software infrastructures for wearable sensor networks. We also show how to satisfy the requirements in our software infrastructure.

Figure 1 is an example of a wearable sensor network. In a wearable sensor network, two types of devices are necessary. One is a wearable sensor device that has several sensors and it transmits the data on a wireless network. The other is a host mobile terminal such as a cellular phone or a PDA that has enough power for analyzing context data. In a wearable sensor network, a number of wearable sensor devices are attached to a user's body directly, and transmit the sensor data to the host mobile terminal that is mounted to a user's waist or puts in a user's hand bag. The powerful mobile terminal will become popular and they are carried by most of people everyday in the near future [11][16].

From the experiences with building a couple of context-aware services using wearable sensor devices, we found that the following three requirements should be satisfied to offer software infrastructures for wearable sensor networks.

- There is a strong relationship between the position of wearable sensor devices on our bodies and the quality of context information. Thus, software infrastructures for wearable sensor networks needs to take into account the deployment of the sensor devices.
- It is necessary to analyze sensor data from multiple wearable sensor devices simultaneously to generate high-level context information in a hierarchical manner.

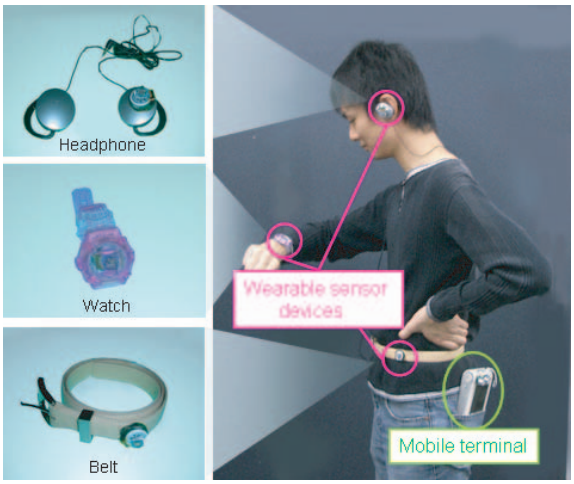

Figure 1. Wearable Sensor Network

- Context information should be stored for a long time because a history of context information is useful to build various types of context-aware services.

The following sections present the details of the requirements and how to satisfy the requirements in our software infrastructure.

3.1. Deployment of Wearable Sensor Devices

It is not easy to generate context information from wearable sensor devices because they can be put at any positions in a user's body, and extract various types of context information [10], but there is a few research projects on discussing the relationship between the physical position of the wearable sensor devices and the types of context information that can be extracted. In our early study, we examined the relationship by using a single wearable sensor device.

As a result, for example, we found that a fingertip was the best place to detect the touching motion of a user. A wearable sensor device attached to a user's waist is useful to detect sitting and standing position. In our research, wearable sensor devices are embedded in wrist watches, belts and head phones as shown in Figure 1 because a user does not want to put devices that are not used in daily lives.

However, it is still difficult to generate even simple context information in an accurate manner by using a single sensor device. Although we found that there is a relationship between the position of the wearable devices and the context that can be extracted, we need to attach a number of sensor devices, and an analysis logic such as majority vote to determine the context from multiple devices is required to improve the accuracy of the context.

3.2. Hierarchical Context Refinement

It is possible to generate high-level context information in one step, but we consider that the approach is not flexible and desirable. In our approach, a lower level context is abstracted to a higher level context, and the context in each different level should be abstracted in a gradual way. The approach makes an analysis algorithm simple because each analysis to abstract context information is fine-grained. Also, it becomes easy to compose existing analysis algorithms to extract new abstract context information.

Figure 2 illustrates an example of the hierarchical context refinement to generate a context of reading a book. Wearable sensor devices attached to a user's waist, and right and left hands are used to obtain context information. For example, the wearable sensor networks that are put in both hands contain 3-axis linear acceleration sensors and light sensors to detect the orientation of a user's hand and their movement. Also, the sensor device on a user's waist detect whether a user is sitting or standing.

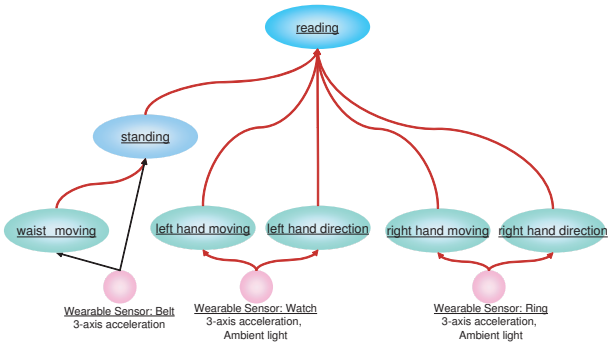

Figure 2. Example of Hierarchical Abstraction

3.3. Storing Context History

It is important to store context information in a wearable sensor network because the use of a context history expands the possibility of context-aware services dramatically [15]. For example, a user may remind what he/she has done today according to the context history, and such a service can encourage a user to write a diary in a joyful way. Also, storing context history makes it possible to predict future context. The prediction may be useful to detect the anomalies in our lives, and to increase our daily safety.

Wearable sensor devices are also effective to store context information. From our experiment of Muffin, personal devices could retrieve a user's context correctly only if the user holds them in his/her hand. However wearable sensor devices are always put to a user's body. Thus, it enables a system to retrieve accurate context at anytime.

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

Therefore, it is useful to store context information in wearable sensor networks. The most feasible way to store the information is to use a database system because a query language offered by a usual database system makes it easy to retrieve context information stored in the database. Also, some current mobile phones are already equipped a relational database system. In our software infrastructure, we also use a relational database system to store context information.

4. Design and Implementation

This section describes the design and implementation of *Cinnamon* that is a software infrastructure for wearable sensor networks and makes it easy to build context-aware services. In this section, first, we show the wearable sensor devices that are used in our prototype system. Next, we show the details of *Cinnamon*

4.1. Wearable Sensor Devices

Cookie is a wireless sensor device that can be extended with various kinds of sensor extension boards (Figure 3). *Cookie* consists of a main board, a sensor board, and an extension board. The main board contains a 2-axis linear acceleration sensor, a compass and an ambient light sensor (both visible-light and infrared sensor). The main board has a Bluetooth module for transmitting sensor data to a host such as a mobile terminal. The sensor board includes a heart rate sensor, a galvanic skin response sensor and a skin temperature sensor. Four types of the extension boards are currently available, and only one of the boards is connected to the sensor board at a time. Each extension board includes either a 3-axis linear acceleration sensor, an ambient light sensor (RGB color and UV), a pressure sensor, or a vibration motor.

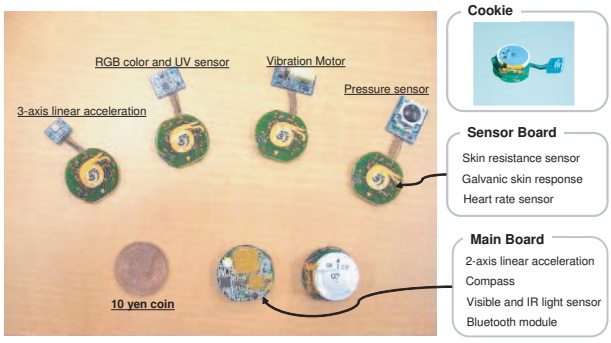

Figure 3. Cookie and Sensors

4.2. Cinnamon: A Software Infrastructure for Wearable Sensor Networks

This section presents a prototype implementation of *Cinnamon*. We, first, show the overview of *Cinnamon*, and major components in *Cinnamon* are described. Finally, we show the programming interface offered by *Cinnamon*.

4.2.1 System Architecture

Cinnamon consists of three components (Figure 4). The *worker* analyzes sensor data, and extracts high-level context information. The *database* stores the context information. The *daemon* provides the context to an application, and manages the data flow between the *worker* and the *database*. *Cinnamon* is based on the blackboard architecture, where the *database* is a shared space to store context information as a blackboard. Our approach is different from the blackboard model described in [18], and is more similar to the original blackboard model that contains workers to generate higher level abstract concepts [3].

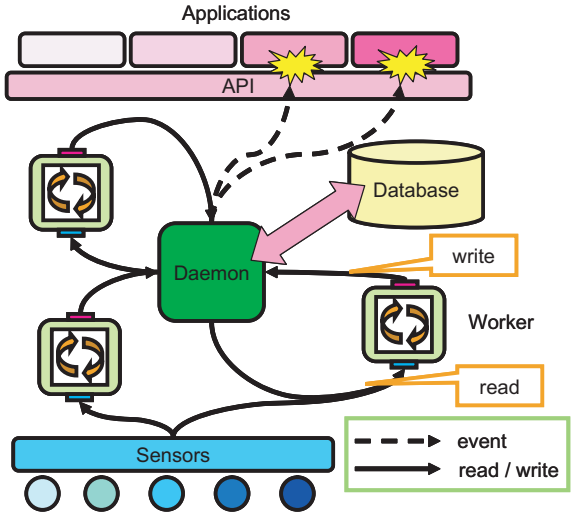

Figure 4. Structure of Cinnamon

The important characteristics of *Cinnamon* are the followings:

- Each *worker* runs as an independent process and they share context information stored in the database. Therefore, it is easy to support the hierarchical refinement of context information.
- Our system uses the *database* to store context information for a long time by defining the syntax of context information in a clear way.
- Our system does not assume multi-hop routing because multi-hop routing is not necessary in most of

services for wearable sensor networks. Thus, our software infrastructure hides details about communication in device drivers, and most parts of our software infrastructure focus on the analysis of context information.

- Sensor data is analyzed by a host mobile terminal to simplify the design of the entire system. In the approach, all sensor information can be used to analyze context information. It is important to offer a software infrastructure that is not specialized for some types of services.

Cinnamon has been implemented in the C language, and runs on the Linux operating system. We employ Muffin as the host mobile terminal that contains the ARM processor and executes the Linux operating system. We can also use a small standard PC with Bluetooth connection instead of Muffin.

4.2.2 Worker and Context Information Representation

Workers analyze sensor data to generate context information. However, it is difficult to generate higher level context information only from sensor data, *Cinnamon* offers a mechanism to analyze lower level context to generate it. Generated context information are stored in the database via *daemon*.

Every context information contains metadata that consist of six fields, *Subject*, *Predicate*, *Time*, *Worker's ID*, *Time Lag*, and *Confidence*.

Subject is a name of context information that a worker generates. For example, *Subject*, "User:Activity:Reading" indicates the context that a user is reading a book.

Predicate includes possible values of each *Subject*. *Subject*, "User:Activity:Reading" has two predicates "not reading" and "reading", where "not reading" indicates that a user is not reading a book, and "reading" indicates that the user is reading the book.

Time represents the time when the corresponding context is retrieved.

Worker ID is a unique ID of each system, and assigned to a each *worker*.

Time Lag represents the time for analyzing context. The information is necessary to show the freshness of context information when the analysis of context information takes a long time.

Confidence specifies the accuracy of each value of context information. The value of *Confidence* is determined by developers. For example, one may define the value by testing his/her worker's algorithm with some sample data. Usually, *Confidence* is a lower value when only one sensor is used, but it becomes more accurate value when the majority voting worker that receives data from multiple sensors is used. The *daemon* offers the context information whose confidence in the highest to an application.

Table 1 shows twelve different *workers* currently available. In the table, *Subject* is the name of context information that each worker generates. The name is defined as a hierarchical name. For example, "Wrist:Activity:Moving" indicates that the activity of a user's wrist is *moving*. We adopted similar syntax to describe context information proposed in [8]. *Predicate* shows the possible values that each Subject can be set. *Method* indicates the algorithm used in each worker. For example "User:Activity:Reading" is retrieved from the other two context "RightHand:Activity:Moving" and "LeftHand:Activity:Moving" by a rule based algorithm, and if each of their states indicates "moving", the predicate of "User:Activity:Reading" shows "reading". The last field is input values to each worker. The values are either context information or sensor values.

4.2.3 Database

The *daemon* stores the context information generated by a *worker* in the *database*. We are using SQLite as the *database*, that is a relational database system for embedded systems. The *database* allows context-aware services to use not only the present context information but also a history of context information.

4.2.4 Daemon

The following shows the two functionalities of the *daemon*. The *daemon* provides two functionalities such as:

- Controlling the data flow between *workers* and the *database*.
- Delivering context information to applications.

All *workers* should store context information extracted by them in the *database* without considering the behavior of other *workers*. However, it is necessary to synchronize to write context information to the *database* when multiple *workers* write context information simultaneously. *Workers* communicate with the *daemon* by using Linux IPC, and the *daemon* receives and stores data in a FIFO manner. There is another advantage of the approach since the *daemon* can observe the data flow in the entire system completely.

Table 1. Workers Implemented in The Current Implementation

Subject	Predicate	Method	Inputs
Wrist:Activity:Direction	"X High", "X Low", "Y High" "Y Low", "Z High", "Z Low"	threshold	3D_Acceleration Sensor
Wrist:Activity:Moving	"stopping", "moving"	variance	3D_Acceleration Sensor
User:Activity:Walking	"stopping", "walking"	variance	Cookie:Direction Wrist:Activity:Moving
Waist:Activity:Moving	"stopping", "moving"	variance	3D_Acceleration Sensor
Waist:Activity:RestStyle	"sitting", "standing", "unknown"	variance	3D_Acceleration Sensor Waist:Activity:Moving
Head:Activity:Moving	"stopping", "moving"	variance	3D_Acceleration Sensor
Head:Activity:Style	"face_up", "face_normal" "face_down", "sideway", "nod"	variance threshold	3D_Acceleration Sensor
RightHand:Activity:Moving(In a ring)	"stopping", "moving"	variance	3D_Acceleration Sensor Visible Light Sensor
LeftHand:Activity:Moving(In a wrist watch)	"stopping", "moving"	variance	3D_Acceleration Sensor
User:Activity:Reading	"not reading", "reading"	rule based	RightHand:Activity:Moving LeftHand:Activity:Moving
User:Activity:Sleeping	"not sleeping", "sleeping" "unknown"	rule based	Head:Activity:Style User:Activity:Reading
User:Activity:Idle	"not idle", "idle" "unknown	rule based	User:Activity:Sleeping User:Activity:Reading

The second functionality is to offer context information to applications. Applications can retrieve context information by requesting a query whose key is the predicate of context information to *Cinnamon*. The *daemon* offers a function to select the context information whose confidence is the highest from the *database*. Applications can also receive context information through an event-based mechanism. If the *daemon* detects the change of context information, an event handler registered by an application is executed.

Since *Cinnamon* offers two ways to retrieve context information, applications can select the most suitable method to change its behavior according to the current situation. The first way is a query-based interface and the second is an event-based interface. The following functions are offered by *Cinnamon* as the programing interfaces.

get_context() is used to request a query whose parameter is the subject of context information. For example, if *get_context(User:Activity:Reading)* is invoked to know whether a user is reading a book or not, either *not reading* or *reading* is returned.

add_event_handler() is invoked, if an application wants to know the change of context information. The first parameter indicates the subject of context information that an application wants to observe, and the second parameter is a handler function that is invoked when the observed context information is changed.

remove_event_handler() removes the handler.

5. Using Cinnamon to Develop a Context-Aware Service

In this section, we describe a sample application service and the experiences with building the service. We also show an experiment to measure the accuracy to retrieve context information in *Cinnamon*.

5.1. A Sample Application Service

We have developed a sample service to show the effectiveness of our software infrastructure for wearable sensor networks. The service offers an appropriate information to a user who does nothing on a train. We have observed people in a Japanese train, and found that the behavior of the people is categorized in the five situations, "walking in the train", "sleeping", "reading a book", "using a mobile terminal", and "do nothing". When the service detects that a user does nothing at the moment, it offers some information to make it possible to spend their boring time in an effective way.

The wearable sensor network consists of Muffin and four Cookies (Figure 5). Since it is not easy to detect that a user does nothing, we classify possible situations into five situations in the service. The service detects the four situations,

”walking in the train”, ”sleeping”, ”reading a book”, and ”using a mobile terminal”, and if a user’s current situation is not categorized into the four situations, the service considers that a user does nothing. The situation ”using a mobile terminal” is retrieved from Muffin, and other situations are retrieved from Cookies. Table 2 shows workers and context information that are used for building the service.

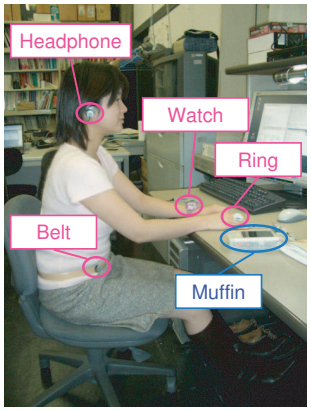

Figure 5. Device Deployment

Table 2. Worker and Context in Sample Application Service

Worker	Predicate of Context
Walking	”walking”, ”stopping”
Sleeping	”sleeping”, ”not sleeping”, ”unknown”
Reading	”reading”, ”not reading”
Watching	”under watch”, ”free”
Idle	”idle”, ”not idle”, ”unknown”

Figure 6 shows how the service works. The upper left pane in the window shows the current situation of a user. In this case, the user is currently walking in a train. If the service detects that the user is watching a mobile terminal or has nothing to do, the screen of the mobile terminal is changed as Figure 6 (b).

The service can correctly detect the contexts shown in Table 2 by using Muffin and Cookies. *Cinnamon* separates an application logic from the algorithms used to extract context information. We believe that the approach makes it dramatically easy to develop context-aware services.

5.2. Accuracy of Context Extraction

We have developed a couple of workers that detect whether a user is walking or not by using multiple wearable sensor devices, and compare the accuracy of the retrieved context information. As shown in Figure 7, we have developed four workers in the experiment. The first one is a

Figure 6. Sample Application Service

worker analyzing the data from a 3-axis acceleration sensor on a pair of head phones, the second one is a worker analyzing the data from a 3-axis acceleration sensor on a belt, and the third one is a worker analyzing the data from a 3-axis acceleration sensor embedded in a wrist watch. The fourth worker is a majority voter that compares the result of the above three workers and selects the value of the majority of the variants. All workers return either ”walking” or ”stopping”.

In our experiment, a user takes a sequence of the following three actions. The first action is ”walking” for 30 seconds. The next action is ”stopping” for 10 seconds. The last action is ”walking” again, for 30 seconds. Then, the retrieved context information and a user’s actual behavior are compared as the recognition accuracy. The upper row of the Table 3 shows the average recognition accuracy of each worker’s extracted context information. The recognition accuracy is calculated by the following formula.

$$\textit{Recognition Accuracy}(\%) = \alpha / \beta * 100$$

α: *The time while the retrieved context is correct*

β: *The time while measuring the context*

Table 3 shows the recognition accuracy to detect a user’s activity when wearable sensor devices are contained in a head phone, waist best, and wrist watch. In the table, *multi* means the results when a majority voting worker is used. The upper row shows the result when a user is working and has nothing in his/her hand. The lower row shows the result when a user uses a mobile phone while he/she is walking. The result in the upper row shows that it is possible to extract correct context information from respective sensors.

Table 3. Recognition Accuracy

status	head	waist	wrist	multi
normal walking	98.3 %	97.6 %	100 %	100 %
walking with using a phone	92.5 %	95 %	0 %	92.5 %

On the other hand, when a user uses a mobile phone while he/she is walking, the recognition accuracy of the worker from a sensor device in the wrist watch becomes 0 % because his/her arm is not down and our analysis algorithm assumes that a user is walking only when his arm is down. However, by using values from multiple wearable sensor devices, the accuracy of the context information retrieved from the majority voting worker shows 92.5 %. This means that using multiple wearable devices attached to users body improves the shortcomings of the context retrieved from the wrist watch. In this way, it is important to consider how to deploy wearable sensor devices in our body to extract accurate context information, and we believe that using the values from sensor devices attached to multiple positions are effective.

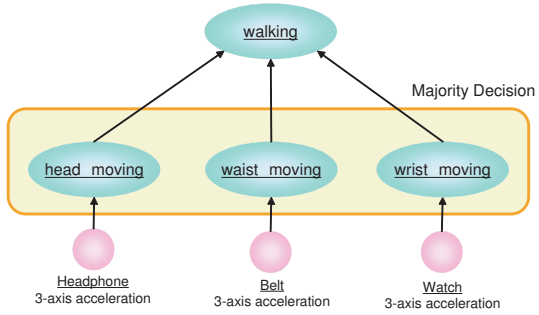

Figure 7. Majority Decision

6. Discussions

Cinnamon is very effective to develop context-aware services for a wearable sensor network. However, we still found several issues to improve our prototype system. In this section, we show four issues and describe how to solve the issues in the near future.

The first issue is the difficulty in developing workers to retrieve context information. As mentioned in Section 2, the type of context information retrieved from Muffin is limited. On the other hand, we found that the number of the type of context information is increasing because wearable sensor devices can be attached to any positions on our bodies in a flexible way, and monitor a user at anytime and anywhere. We need more workers to retrieve more context information so that the number of workers will be increased rapidly. There are two problems when the number of workers is increased. The first problem is that the performance of context processing depends on the number of workers in the current prototype implementation. The second problem is that it makes difficult for developers to understand the whole dependencies among workers. To solve the problems, each worker needs to specify which context it is interested in, and *Daemon* activates only workers that have interests to the currently changed context.

The second issue is to develop methods to analyze raw sensor data. We have adopted simple methods to use such as thresholds and variances to extract context information. Although the simple methods are easy to implement and able to retrieve context information in a short time, the accuracy of the context information is limited. For example, the accuracy of retrieved context information is good when a user is walking normally, but the retrieved values may be incorrect when a user uses a mobile phone when he/she is walking because we assume that a user's arm is down while he/she is walking. A developer can improve the accuracy by determining a threshold carefully, but the method to consider every situation manually has a serious limitation when complex context information wants to be retrieved. To solve the problem, we are considering adopting more advanced context mining techniques such as machine learning based methods to analyze sensor data to improve the accuracy of extracted context information.

The third issue is the type of sensors and the deployment method for wearable sensor devices. We have developed several workers in our current prototype implementation, but the prototype heavily relies on a 3-axis linear acceleration sensor. None of the biological sensors such as skin resistance and heart rate sensors has been used, because the skin resistance sensor only works when it is touched with a finger or an ear, and the heart rate sensor in Cookie is unstable when analyzing sensor data. There may be a problem in the current implementation of Cookie. In addition, the value of the skin temperature sensor is increased gradually due to the heat of a Cookie itself. We are currently looking for different wearable sensor devices to utilize biological sensors in an effective way.

We also found that the accuracy of retrieved context information is changed according to the positions of wearable sensor devices in a user's body. It is not easy to determine whether a user is walking when he/she puts a wearable sensor device on his/her waist. The correct context values are retrieved from most of users, but it fails to extract correct values from some users. Determining the threshold in a unique way is not easy because the suitable threshold is

changed according to each user. Thus, it may be necessary to personalize the configuration of workers for each user to extract accurate context information.

The fourth issue is how to use a history of context information. Our sample service uses only the present context information, but our software infrastructure stores context information in the database. Thus, context-aware services may use the past context information. Thus, it is possible to develop context-aware services according to a history of context information. For example, it is easy for a user to know where a user has visited today if we develop a service to draw a user's route according to the history of a user's location context. Also, we may find the location of a book according to its user's current location information while the user is reading the book recently. It is a future issue to develop context-aware services that use a history of context information in an effective way.

7. Conclusion

We have proposed a software infrastructure for wearable sensor networks, that makes it easy to develop context-aware services for mobile users. In a wearable sensor network, a host mobile terminal obtains context information from wearable sensor devices and analyzes the information. We have developed a prototype implementation of the software infrastructure to evaluate the effectiveness of our approach. We also show a sample service that uses our software infrastructure, and present some future issues that should be improved in the future.

We believe that wearable sensor networks can enhance previous systems that use the personal server model such as personal servers [16] and personal home servers [11]. Also, context information retrieved from wearable sensor devices can be used to personalize various services that are available on smart information appliances to reduce complexities and stresses in our daily lives. The complexities and stresses will become more serious issues in our future daily lives so that our approach will be useful to make ubiquitous computing visions more practical.

References

[1] 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.

[2] Kaori Fujinami, Fahim Kawsar, Tatsuo Nakajima, "Aware Mirror: A Personalized Display using a Mirror", 3rd International Conference on Pervasive Computing, May 2005.

[3] Hans-W. Gellersen, Albrecht Schmidt, Michael Beigl, "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts", ACM journal Mobile Networks and Applications(MONET), Vol. 7, No. 5. October 2002.

[4] B. Hayes-Roth, "A Blackboard Architecture for Control", Artificial Intelligence 26, pp.251-321, 1985.

[5] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl and Hans-W. Gellersen, "Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts", Proc. Ubicomp 2001.

[6] Fahim Kawsar, Kaori Fujinami, and Tatsuo Nakajima; "Augmenting Everyday Life with Sentient Artefacts", In Proceedings of Joint Conference on Smart Objects and Ambient Intelligence (sOc and EuSAI), 2005.

[7] Jason I. Hong, "An Architecture for Privacy-Sensitive Ubiquitous Computing", PhD Thesis, University of California at Berkeley, Computer Science Division, Berkeley, 2005.

[8] Panu Korpipaa, Jani Mantyjarvi, Juha Kela, Heikki Keranen, and Esko-Juhani Malm, "Managing Context Information in Mobile Devices", IEEE Pervasive Computing, pp.42-51, 2003.

[9] Panu Korpipaa, "Blackboard-based software framework and tool for mobile device context awareness", VTT Publications 579, 2005.

[10] Spyros Lalis, Alexandros Karypidis, Anthony Savidis and Constantine Stephanidis, "Runtime Support for a Dynamically Composable and Adaptive Wearable System", ISWC pp.18-21, 2003.

[11] T.Nakajima, I.Satoh, "A Software Infrastructure for Supporting Spontaneous and Personalized Interaction in Home Computing Environments", Personal and Ubiquitous Computing, Springer, 2006.

[12] Anand Ranganathan and Roy H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments", In Proceedings of the ACM/IFIP/USENIX International Middleware Conference, June 2003.

[13] Daniel Salber, Anind K. Dey, Gregory D. Abowd, "The Context Toolkit: Aiding the Development of Context-Enabled Applications", CHI, pp.434-441, 1999.

[14] Daniel P. Siewiorek, Asim Smailagic, Junichi Furukawa, Andreas Krause, Neema Moraveji, Kathryn Reiger, Jeremy Shaffer, Fei Lung Wong, "SenSay: A Context-Aware Mobile Phone", ISWC, 2003.

[15] Toshihiro Takada, Satoshi Kurihara, Toshiharu Sugawara, and Toshio Hirotsu, "Proximity Mining: Finding Proximity using Sensor Data History", In Proceedings of 5th IEEE Workshop on Mobile Computing Systems and Applications, 2005.

[16] Roy Want, Trevor Pering, Gunner Danneels, Muthu Kumar, Murali Sundar, John Light, "The Personal Server: Changing the Way We Think About Ubiquitous Computing", Ubicomp, 2002.

[17] Mark Weiser, "The computer for the 21st century", Scientific American, 94-104, September 1991.

[18] Terry Winograd, "Architectures for Context", Human-Computer-Interaction, vol.16, 2001.

[19] Tetsuo Yamabe, Ayako Takagi, Tatsuo Nakajima, "Citron: A Context Information Acquisition Framework for Personal Devices", RTCSA, pp489-495, 2005.