Citron: A Context Information Acquisition Framework for Personal Devices

Tetsuo Yamabe*, Ayako Takagi, Tatsuo Nakajima
Department of Computer Science, Waseda University
{yamabe, ayako, tatsuo}@dcl.info.waseda.ac.jp

ABSTRACT

This paper describes a context information acquisition framework for a personal device that equips a variety of sensors. The framework captures context information about a user and his/her surrounding environment; and the information is used to adapt the behavior of applications running on the personal device. Our framework adopts the blackboard architecture to execute multiple analysis modules that analyze signals from respective sensors. Respective modules implement different algorithms to complement each other's results to retrieve more accurate and higher abstract context information.

1. INTRODUCTION

The rapid progress of computing and communication technologies enables devices to be smaller and distributed in our daily life. Today, we find these devices to be embedded in the variety of daily objects(e.g. chair[17], desk[12]) in an invisible way. As described in [11], context-awareness is one of the significant abilities for such devices to work spontaneously in a dynamic changing environment. One of the efficient methods to realize context-awareness is to perceive the real world by sensors. By analyzing raw data acquired from sensors; devices and services can recognize the context of a user, surrounding environment and itself.

Personal devices, such as PDAs and mobile phones have been developing with a perceptual ability[6, 4]. These devices are so close to a user that it is expected to work as a smart assistant of him. However, it has not been discussed sufficiently that what type of sensors are useful to acquire a user's context information and what is required in the process of context acquisition on a personal device.

In this paper, we have discussed about Muffin that is a prototype of a sensory personal device and its context information acquisition framework. Muffin has fifteen kinds of sensors and can sense several types of contextual quantity, such as acceleration, orientation, air temperature, a user's heart rate and so on. We have performed some experiments about these sensors to investigate the characteristics of each sensor for acquiring context information.

These experiments show that the validity of the sensor value and analysis algorithm changes frequently. Because a personal device is used in several styles. For example, we should monitor the state of the device, such as "Which side is top?" or "Is the person holding the device or not?". Then, we can reflect the state to select an appropriate set of sensors and change the behavior of an analysis algorithm. Furthermore, time delay is caused by each analysis algorithm in some cases and it prevents the applications from context acquisition in real time. Therefore, we should select another responsive way according to the state of the device. It can be said that multiple sensors enable applications to acquire context information flexibly by analyzing the information from multiple aspects of view. These facts make it difficult to develop context-aware applications on these devices and we feel that a middleware support should be offered to application programmers.

We have developed a framework named Citron to utilize the advantage of multiple sensory personal device like Muffin and to implement context analysis modules on it. Citron supports a parallel context analysis and a coordination among them by employing the blackboard architecture[16]. By running context-aware applications on top of Citron, we present the usefulness and future possibilities of such personal devices fabricated with multiple sensors.

This paper is organized as follows. In Section 2, we have described the characteristics and requirements for realizing context-awareness on personal devices. In Section 3, we have introduced Muffin that is a prototype of a sensory personal device. Based on our experiences with Muffin, we show some difficulties that are inherent in context acquisition on personal devices. In Section 4, Citron framework has been introduced. Citron offers a framework for developing context analysis modules and for coordinating them on a shared space. It enables context analysis modules to adapt themselves according to the state of Muffin. In Section 5, we have shown a sample application running on Citron and we have evaluated our system in Section 6. In the final section, we have summarized the effectiveness of our approach and we have presented the future work obtained from the evaluation.

2. PERCEPTUAL ABILITY IN PERSONAL DEVICES

As seen in the vision of a ubiquitous computing environment proposed by Mark Weiser[14], the recent progress of computing and communication technologies has made computers pervasive. They have been distributed and disappeared into the background of our daily life in the last twenty years. It has also been said that context-awareness is a significant ability to make services and applications more useful and comfortable for a user. Context information varies according to the subject and purpose of the usage. Therefore, many physical objects are augmented to enable applications to retrieve context information by embedding computers and sensors.

*Currently belong to Nokia Research Center, Nokia Japan Co., Ltd

Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)
1533-2306/05 $20.00 © 2005 IEEE

Personal devices have two special characteristics in ubiquitous computing environments. One is a partnership with a user. Because they are carried by the user most of the day and they are used very frequently. Considering from a sensing aspect, they could monitor the user and could recognize everything he/she does. Second one is a connectivity to a user and the augmented environment by context-aware services. The user can access to the several context-aware services running in the background through the device, because they have connectivity to the access point in the surrounding environment. On the other hand, services can access to the user through his/her personal device. Services can display information and make interactive actuation(e.g. vibration, make sound and light) through the device, because of this connectivity. Therefore, personal devices also provide a perceptual ability and play an important role in the ubiquitous computing environments.

At the early stage of context acquisition from personal devices; location and time are the main resource of its context[1]. To identify its location, GPS is used in outdoor environments and the other location systems are used indoor environments[10]. Device activities(e.g. network connection) and static personal information(e.g. id, name, schedule) are also processed as context. To acquire more rich and various context, however, additional sensors are required. Personal devices also have two characteristics from the view point of context sensing. One is the capability for attaching sensors, another one is its capability to cope with the rapidly changing context. Context information that devices can detect heavily depends on the sensors that they are equipped with[2]. In the case of mobile devices, it is limited that how many and what type of sensors can be attached and arranged because of its portable small size. Furthermore, context that they detect(e.g. a user's activity, location, network connection) is changing rapidly and unstable. Thus, some limited sensors have been attached for specific context acquisition in traditional mobile and personal devices.

The most frequently used sensor on mobile context acquisition is an accelerometer. An accelerometer is cost efficient and sufficiently small to be incorporated into personal devices. Furthermore, the acceleration of the device could be effective to recognize the activities(e.g. walking, running, walk up/down stairs) and motion(e.g. shaking, rotating, knocking on) of a user[8, 13]. Second one is a microphone. Ambient noise is measured to detect the place where a user is(e.g. meeting room, restaurant, on the street)[6, 9]. To recognize speaking or talking, a microphone is also useful[13]. Environmental sensors like microphone is frequently used on a personal device. Light and temperature sensors are also used to acquire environmental situation and a user's context[5].

However, it has not been discussed that what types of sensors are useful and what kind of context is acquired efficiently by the personal devices because of its limitation. In the next section, we introduce Muffin that is an unique personal device in terms of the sensors that it has. Based on our experiences with the application development on Muffin, we have figured out the requirements and possibilities in such a multi sensory personal devices and context acquisition.

3. MUFFIN AND SOME REQUIREMENTS IN CONTEXT ACQUISITION

3.1 Overview of Muffin

Muffin is a prototype of a future personal device for studying on context acquisition in the ubiquitous computing environments. It has been developed by the collaboration work of Nokia Japan Corporation and our laboratory. The significant characteristics of Muffin are its sensing capability for context-awareness and its interactive motion(e.g. gesture) detection. Muffin has thirteen kinds of sensors in the PDA box and two kinds of externally attached sensors. Figure 1 shows Muffin and sensors with arrows pointing the places they are incorporated into.

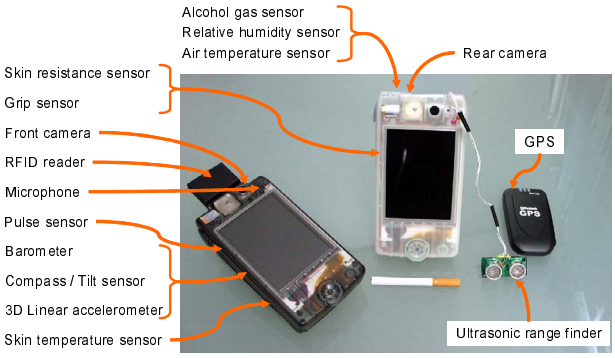


Figure 1: Sensors on Muffin terminal

Sensors on Muffin are roughly divided into four groups. First group is the environmental sensors, which include an air temperature sensor, a relative humidity sensor and a barometer. Second group is the physiological sensors, which include an alcohol gas sensor, a pulse sensor, a skin temperature sensor and a skin resistance sensor. Third group is the motion/location sensors, which include a compass/tilt sensor, a 3D linear accelerometer, a grip sensor, a ultrasonic range finder and a GPS. The ultrasonic range finder and the GPS are externally attached as optional sensors. Last group is the other sensors, which include an RFID reader, front/rear cameras and a microphone. Linux is running on Muffin, so each sensor is enabled to be accessed as a device file. To acquire the sensor value, applications should open and read the value from the device file(e.g. /dev/AccelX).

Muffin has been developed as an ordinary PDA. Therefore, it has common user interfaces and connection interfaces. To be operated by a user, Muffin provides a touch screen, a micro joy stick and a microphone. To actuate to the user, Muffin is fabricated with a speaker, an LED light and a vibration motor. Muffin also provides IrDA, Bluetooth and wireless LAN as remote connection interfaces. The other equipments are a PCMCIA card slot, a USB port and a serial connection port.

3.2 Sensors and Context-Awareness on Muffin

Muffin has so many kinds of sensors that it can acquire several kinds of physical phenomena. We listed up and classified context expected to be acquired by Muffin. Table 1 shows the relationship between expected context and sensors on Muffin. Context is divided into three categories based on its subject(Muffin, a user and an environment). Based on this classification, we performed some experiments about these context acquisition. As the result of that, we have found some important points and issues to consider when we acquire context information from Muffin.

Muffin : The context states of Muffin can be acquired accurately because of the sensors that they depend on are very reliable and responsive in real time. Furthermore, the

Class	Description	Sensor
Context of Muffin		
State	Moving	Accelerometer,
	Top side	Ultrasonic range finder
	Direction	Compass,
		Accelerometer
	Held by a user	Skin resistance sensor,
		Grip sensor
Context of a environment		
Air	Air temperature	Air temperature sensor
	Air humidity	Relative humidity sensor
	Air pressure	Barometer
Sound	Ambient noise	Microphone
	Talking voice	

Class	Description	Sensor
Context of a user		
Activity	Standing or sitting	Accelerometer,
	Walking or running	Ultrasonic range finder,
	Going up/down stairs	Compass
Geographical information	Location	GPS
	Orientation	Compass
Physical condition	Level of stress	Skin resistance sensor, Skin temperature sensor, Pulse sensor, Grip sensor
	Alcoholic breath	Alcohol sensor
Emotion	Exciting	Skin resistance sensor, Grip sensor
	Surprising	Skin temperature sensor,
	Fearing	Pulse sensor, Accelerometer

Table 1: Relationship between expected context and sensors on Muffin

states of Muffin can be classified clearly into exclusive classes. For example, Muffin could be observed just as "held" or "free" from the view point of "Is Muffin held by a user?". Consequently, they could be analyzed by a simple algorithm, such as a threshold analysis.

User : There are three significant issues in a user's context acquisition. First issue is an immediate change of available sensors and the context analysis method. To acquire a user's activity, the user has to take Muffin with him/her in some ways. However, most of the context analysis and available sensors change according to the position or situation in which Muffin is used. For example, the analysis method to detect standing or sitting by the acceleration; changes whether Muffin is held or waist-mounted. Furthermore, if it uses a ultra range finder to correct analysis results by measuring the distance from the floor, the validity of the sensors also changes.

Second issue is the time consuming process in context acquisition, and it is important to the characteristic of sensors and a context analysis method. In the previous example, detecting "sitting" state is easy at five minutes after the "sitting" event occurs. To detect the moment when the user sat, we need to analyze a special wave patten of sensor values in a few hundred millisecond. This issue should be also discussed about physical conditions and emotion sensing. When retrieving the raw value from physiological sensors, Muffin has to be gripped. However, the validity of data from these sensors in short term is not sufficient to recognize context information. Because the value from a skin temperature sensor changes very slowly, and one from a pulse sensor changes too rapidly. To generate context, it is necessary to log the average of acquired data and compare them in the span of time.

Last issue is the complexity and ambiguity of context information. The definition of complex context, such as angriness or feeling of hunger, heavily depends on its situation and applications that they are used by. For example, the meaning of loud voice is different in respective situations(e.g. between meeting room and talking with friends). To analyze and classify the sensor value, not only main resources(e.g. microphone) but also other resources(e.g. location, activity) are required.

Environment : Context of the surrounding environment relates directly to raw sensor data, so it is not difficult to acquire them. However, Muffin gets hot internally as time goes on and environmental sensors are affected by the heat. As the result of that, sometimes the measurement is invalid. This problem arises from the design defection, so we should redesign the placement of sensors in Muffin and protect them from the internal heat.

3.3 Requirements in Context Acquisition on Muffin

We have found that there are practical difficulties in recognizing even standing or sitting. In the physical condition or emotion sensing, such difficulties become increasingly prominent because the complexity of these context increases. Therefore, we should go back to consider simple context acquisition and discuss what is significant requirements for effective and robust context acquisition on Muffin.

At first, the representation of context should be decomposed into the combination of other context processing. If complex context could be decomposed into simple one like the activity of Muffin, it becomes easy to reconstruct and represent higher abstract context with a combination of them.

Next, we should observe physical phenomena from multiple aspects of view. Additional sensor data or context are required to increase the quality of information for making a decision. Furthermore, it is considerable that an analysis method is changed to alternative one when required sensors became disabled as described in Section 3.2. This approach is also effective for avoiding a time consuming process. According to the situation, an additional sensor or another analysis method is very effective to recognize such time consuming context. For example, an ultra range finder is effective to detect "He sat just now.", if he is holding Muffin and the sensor measures the distance from the floor. Considering from the standpoint of sensors, it is also said that one sensor offers multiple meanings and context.

At last, it is required to specify the relationship and dependency among decomposed context information. Our experiments show that there are dependency relations among context information. For example, available analysis methods or sensors are changed according to the situation, because there are several styles to use Muffin. We summarize and classify these relationships of three cases shown in Figure 2. These relationships are classified based on a hierarchical context abstraction. Resources in the figure include context or raw sensor data, and they are inputs for the analysis module. The analysis module works on predetermined context acquisition and has an analysis method in it.

Case a) is a basic hierarchical context abstraction with a combination of Resource A and Resource B. Case b) and case c) are its variations. In case b), context is mainly represented as the result of processing of Resource A. Resource B enables an analysis module to adapt its behavior a[illegible]

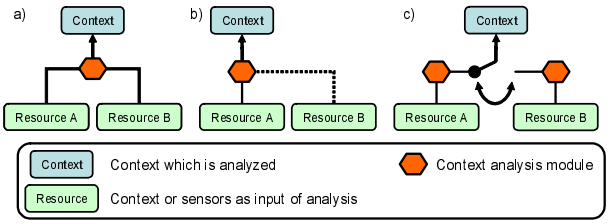


Figure 2: Relationship and dependencies among context

ing to the availability of resources. In case c), context is represented as the result of processing of Resource A or Resource B. When the availability of an analysis module is not sufficient, the current module is alternated by another.

As a result of the discussion, significant requirements for context acquisition on Muffin are pointed. They are not separated issues, but related to each other. Figure 3 shows the complication diagram of context processing with these requirements in Muffin.

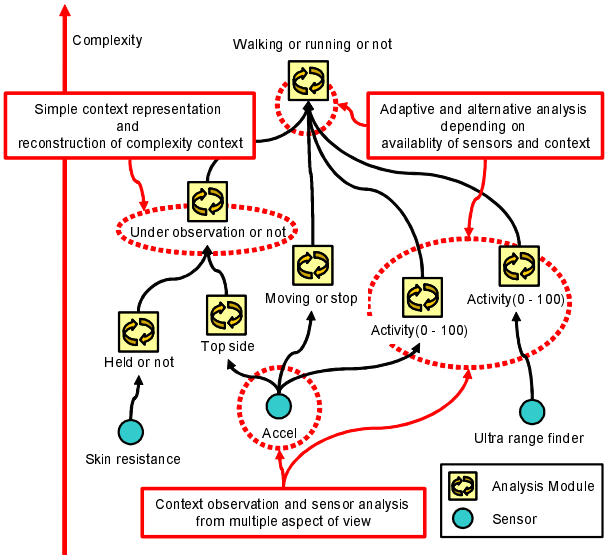


Figure 3: Context processing diagram with requirements in Muffin

There are four basic contexts of Muffin activities in the figure, "Held or not", "Top side", "Moving or stop" and "Activity". Each subject takes an exclusive state and the state of Muffin can be classified into them clearly. However, the relationship among them is non-exclusive, so more complex context information can be represented with the relationship as described above. For example, if the display of Muffin turns up("Top side") and a user holds it("Held or not"), then Muffin may be under observation("Under observation or not"). This is one of the relationship among context represented as case a). To acquire a wide variety of context, one sensor is analyzed from multiple aspects of view. For example, acceleration data are analyzed by three analysis modules and processed into different context and meanings. Furthermore, context is also observed from multiple view points. In this figure, "Activity" is recognized in two ways. They are changed according to the availability of sensors. This relationship is case c) that changes its analysis module and required resources dynamically. We can also find case b) in this case. The analysis module of "Walking or running or not" changes its threshold according to the style of Muffin taken by.

We have refined this context processing diagram into the framework of context acquisition on Muffin. Therefore, we have implemented it as a framework named Citron(Context Information acquisiTion fRamework fOr muffiN), which offers a framework for context analysis modules and coordinating them on a shared space. In Section 4, we have introduced Citron and its implementation.

4. CITRON: ARCHITECTURE AND IMPLEMENTATION

4.1 System Architecture

To implement the requirements discussed in Section 3.3, two functional capabilities are necessary. First one is a context analysis module framework that enables to specify the relationship to the other. Second one is a common space to share the acquired context among context analysis modules for a coordination.

We have employed the blackboard architecture for the coordination among context analysis modules. Furthermore, we have defined each context analysis module as a worker for exclusive context acquisition. The blackboard architecture is the data centric processing architecture that has been developed for speech understanding and artificial intelligence. There are one common shared message board and multiple worker modules for data analysis. Each module reads information from the board as a resource and writes the result of processing to it. Therefore, the coordination among modules is established without the knowledge about other modules. Furthermore, acquired data are analyzed and complimented in the process of the coordination. To implement the blackboard architecture, we adopt the tuple space based programming model[3]. The tuple space is one of the implementation for interprocess communication among independent processes. The tuple space refines a common shared space with a very flexible data type called tuple and enables applications to search tuples with a template based query. The significant characteristics of the tuple space model in the ubiquitous computing environments are 1) loose coupling of worker modules, 2) information sharing among worker modules and 3) flexible data representation.

It is efficient for devices and services to enable them to work without specific knowledge(i.e. IP, port) about others in a dynamic changing environment. The shared space enables worker modules to coordinate with each other. Lastly, context information for several applications could be represented as tuples because of its flexible data type. Winograd [16] discussed the characteristics and trade off between the blackboard architecture and other frameworks for coordinating multiple processes. We have also employed the blackboard architecture for designing Citron architecture, because it is an adequate architecture to implement the requirements in context acquisition on Muffin. Figure 4 shows an overview of the Citron system architecture. Citron is consist of two software components, Citron Worker and Citron Space. We explain respective components as follows.

Citron Worker is the sensor data analysis module. Each worker retrieves sensor data from Muffin and context information analyzed by other workers from Citron Space. Each worker handles the acquisition of specified context, which could be represented exclusively. For example, a worker for recognizing "held or not" observes the val[illegible]

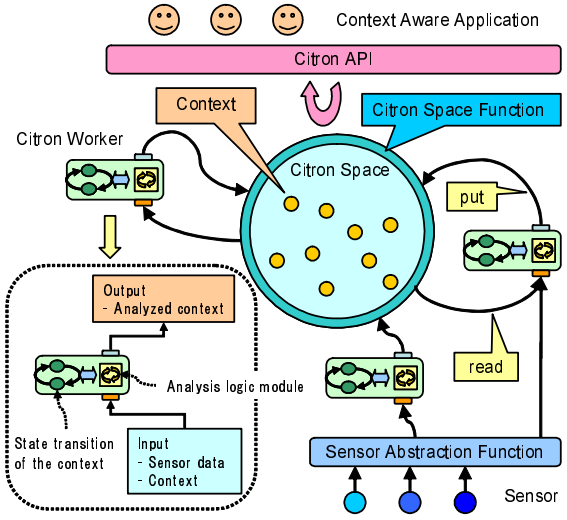


Figure 4: Citron architecture overview

skin resistance sensor and analyzes "held" or "free" by the threshold analysis. This design enables workers to adapt its analysis according to the state of Muffin.

Citron Space is the tuple space for storing tuples including context information by Citron Workers. Citron Space handles the requests for data management from Citron Worker and applications running on Citron. Context information is represented as the combination of meta information, such as subject, state and time.

There are two internal functions and one function for external applications in the system. Internal functions are the sensor abstraction function and the Citron Space function. The sensor abstraction function is just a simple wrapper function for accessing device files of sensors. Citron Worker acquires the sensor value by using this function. The Citron Space function provides the basic function to access the common tuple space, which are put, read and get. Put function inserts context into Citron Space. Read and get function read context from Citron Space with template matching. A difference between them is that get function remove context, but read function does not. Last function is Citron API for being used by applications. This API is offered for retrieving context information from context-aware applications running on Citron. More details are described in Section *4.3.2*.

4.2 Context Representation

As shown in the previous section, context in Citron is not just the representation of context information, but the medium for the communication among Citron Workers. Furthermore, context information is treated as a tuple. Thus, it is required to consider the available context representation format for searching in the read/get method. In the current implementation, context information in Citron is represented as follows.

$$Context := \{ID, Subject, State, Time, Lag, Interval\}$$

The meaning of the context is represented as the combination of a subject, state and time field. For example, the context information, which is written as "A user is not walking(just standing).", is represented as

$$\{\text{“}ID\_walk\text{”}, \text{“}walking\text{”}, \text{“}at\_rest\text{”}, 1107005245, 0, 100\}$$

in the tuple format. In this case, the state could be "walking" or "at_rest". However, a subject is fixed and Citron Worker only acquires specified context. The lag and interval field are meta information of the context, which are inherent in its analysis. Some analysis algorithms such as the FFT analysis require a certain amount of data and time to store them into a buffer. It means the analysis of context information requires some time interval. The lag field shows the time lag and applications can handle it by itself. On the other hand, the interval field shows the freshness of the context. The ID field specifies the Citron Worker which analyzed the context. Context in Citron Space is updated every its polling interval only by Citron Worker specified in the ID field. Therefore, applications can examine its freshness by handling the interval and time field.

4.3 Implementation

Citron is written using the C language and provides interface for C applications. Moreover, a framework for Citron Worker development is also provided. In this section, we introduce the framework and API offered for developers.

4.3.1 Citron Worker Framework

This framework offers the high level abstraction for programming a context analysis module in Citron. It provides common functions among context analysis modules, such as connection management with Citron Space and sensor data retrieval from Muffin. Therefore, developers who create Citron Worker to acquire context have to consider the analysis logic module implementation only. The specification of the analysis, such as the type of the sensor, subject of the context, time lag in the analysis and polling interval are declared in the initialization of Citron Worker.

At first, a developer has to set value to these fields and initialize his/her Citron Worker. After the initialization, the analysis function is invoked at every specified interval. This function invokes an analysis logic module with retrieving specified sensor values and context. The result of the analysis is returned as context and is put into Citron Space.

4.3.2 Citron API

Applications running on Citron access to Citron Space by invoking this API. Citron Space is implemented based on LinuxTuples, which is the tuple space implementation running on Linux [15]. The query processing to search tuples is wrapped as a simple function, which only use the subject field as a key to search. Because this design is sufficiently useful for many applications and it decreases the load of template matching on Citron Space. Furthermore, Citron API also provides the callback function management interface. Developers can register/remove callback functions to handle state change events that occur in Citron Space.

We have developed several Citron Workers and context-aware applications to evaluate the Citron architecture and framework. In the next section, one sample application and Citron Workers required by it are shown.

5. A SAMPLE APPLICATION

We have developed a sample context-aware application named "StateTracer", which uses a user's context acquired by Muffin. This application displays the track of a walking route with a user's state in real time. StateTracer shows not only the track of the route but also walking spe[illegible]

how long a user had stayed at the same point. The walking speed is divided into five levels and distinguished by using different colors. The point where the user had stopped is represented as a circle and its radius become larger based on the stopping interval.

From the view point of the application, three kinds of context are required. They are "walking state", "walking direction" and "walking speed". To draw the track of the route, at least "walking direction" and "walking state" are required. Context representing "walking speed" is optional, however, it is necessary to speculate the distance and create a more accurate map. "Walking speed" is measured by the level of the activity of Muffin. Based on our examination, we have found that the activity is correlated with the walking speed of a user, while he is walking and while watching Muffin. It follows that context representing "watching" is required. To acquire these context, six Citron Workers work in this application. They are "orientation", "walking", "activity", "watching", "holding" and "top side". Figure 5 shows the relationship among context information required by StateTracer and Citron Worker.

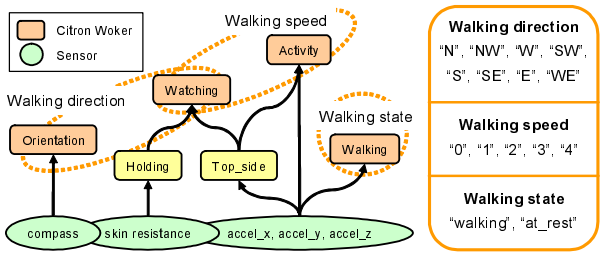


Figure 5: Required context and Citron Worker in StateTracer

The "orientation" worker retrieves data derived from a compass and analyzes which orientation Muffin is looking forward to. This orientation can be regarded as the direction of a user walking, when a user is watching at Muffin. The "watching" worker recognizes whether the user is watching at Muffin or not by a result of "holding" and "top side" worker analysis. If the user holds Muffin and displays or the head of Muffin turns up, the "watching" worker recognizes the state as "The user is watching Muffin". The "holding" worker retrieves and analyzes data derived from a skin resistance sensor. The "top side" worker analyzes the gravity acceleration. The "activity" worker also retrieves the acceleration and analyzes the motion of Muffin by using the FFT analysis. This worker requires context generated by the "top side" worker, because the axis, which is mainly measured, changes according to the top side of Muffin. As described above, an analyzed activity is divided into five levels and treated as the walking speed. The "walking" worker decides whether a user is moving or at rest by analyzing data derived from an accelerometer. This context is as same as the "walking state" in StateTracer.

It is also possible for the StateTracer to recognize the walking state only by "walking speed". However, the performance in the analysis is not good, because the "activity" worker has to take about 6.4 seconds to recognize the context. It is the time to collect 128 samples into buffer and analyze at every 50 msec. On the other hand, the "walking" worker analyzes a user walking or not by simple 0 cross detection in real time. It is expected that the parallel analysis using these two Citron Workers enables StateTracer to be more responsive to recognize the walking state. In the next section, we evaluate Citron by measuring the overhead in invoking API and comparing the accuracy of the map drawn with different workers.

6. EVALUATION

6.1 Performance

We have measured the overhead in accessing Citron Space by invoking the Citron Space functions described in Section 4.1. The dependency relation between the time and number of running Citron Workers is also examined. The execution time is measured as the average of whole workers with changing a number of Citron Worker. Each worker invokes each Citron Space function ten times. Figure 6 shows the result of the examination.

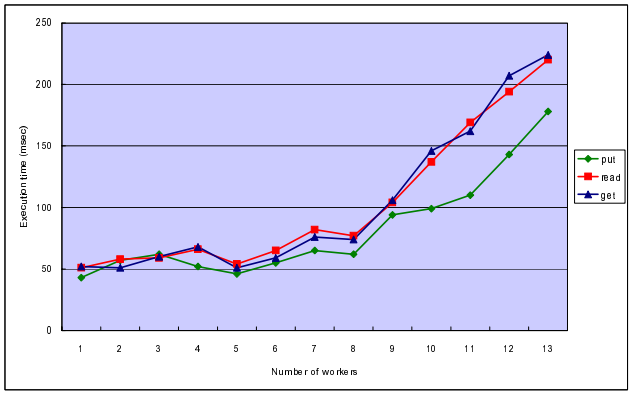
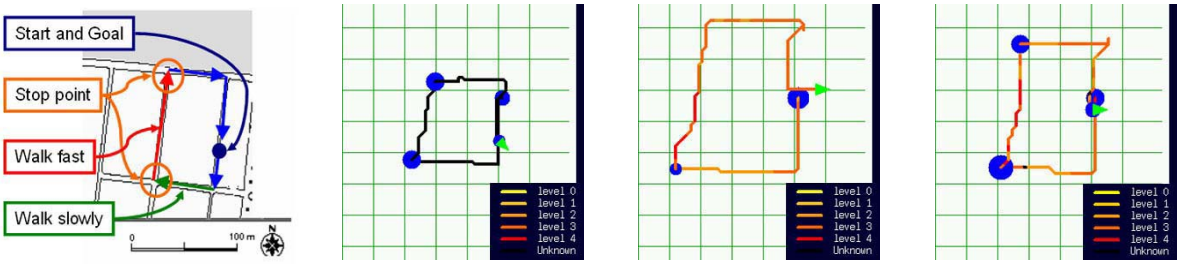


Figure 6: Relationship between execution time and number of running workers

Figure 6 clarifies that the overhead in accessing Citron Space increases as the number of Citron Worker increases. Especially, the execution time increases remarkably when the number of Citron Worker goes over eight. It is assumed that the limitation of workable Citron Worker heavily depends on the implementation of Citron Space. As described in Section *4.3.2*, Citron Space is implemented as a wrapper function on LinuxTuples. It can be said that the performance will be enhanced if the implementation is optimized for processing context information. In the current implementation, Citron Worker only puts the result of its analysis, when the state of context is changed, to reduce the load of Citron Space.

6.2 Experiences with the Sample Application

In this section, we evaluate Citron by measuring the accuracy of context by StateTracer. The accuracy is measured by comparing drawn maps in three cases. We defined the case with retrieving only walking state as case 1 and only walking speed as case 2. Requiring both context for a hybrid analysis is defined as case 3. Figure 7 shows the walking route for the examination in the left side. An examinee walks the route with holding Muffin in his/her hand. However, the examinee does not pay an attention to the display to eliminate the effects of the intentional map creation. To examine the difference of performance among Citron Workers and the effect to analyze context information, the examinee changes walking speed intentionally. The maximum walking speed in this examination is about five kilometers per hour and normally it is about three kilometers per hour. Mo[illegible]



Walking route a) case 1 b) case 2 c) case 3

Figure 7: Walking route and drawn maps by StateTracer in each case

there are two stop points on the route. The examinee stops for ten seconds when he comes to the point.

Figure 7 also shows the result of the examination in each case. Figure a) is drawn with the time of walking and its orientation. Turning point and time that a user has stopped can be confirmed clearly. Because the "walking" worker responds quickly when it recognizes the state change of a user's walking. However, the walking speed is not reflected and the length of each edge is determined according to only walking time. As contrasted with Figure a), Figure b) speculates the walking distance based on the walking speed. The drawn map is similar to an actual map than Figure a). Figure b) also shows that the "activity" worker could not detect the stop point, because ten seconds are not sufficient time for the FFT analysis to take the state change internally. Figure c) shows that a more accurate map than Figure a) and Figure b). This case exploits the advantage of each analysis and shows the effectiveness of hybrid context acquisition applying multiple types of analysis. The stop point is detected clearly and the shape of the map is the most accurate.

7. CONCLUSION AND FUTURE DIRECTION

In this paper, we have discussed about Muffin that has fifteen kinds of sensors and an effective framework called Citron for context acquisition on personal devices. We found two further issues in context acquisition while developing the prototype of Citron. One is the performance of Citron based on the blackboard architecture. The parallel context analysis with multiple sensors heavily burdens the personal devices. Therefore, we should optimize the performance and reduce the load by the redesign of Citron.

Another one is the limitation of context acquisition on Muffin. Many physiological sensors on Muffin requires some constraints to be used, such as the position of a finger and the style of holding, to measure valid data. Therefore, the accuracy of such sensors changes so frequently. It follows that other devices(e.g. wearable sensor) and sensors are required as an alternative resource of context analysis. Citron has a mechanism for the coordination with such remote devices, because the blackboard architecture enables a system to add/remove other knowledge resources easily.

8. REFERENCES

[1] G.D.Abowd, C.G.Atkeson, J.Hong, S.Long, R.Kooper, M.Pinkerton, "Cyberguide: A mobile context-aware tour guide", In ACM Wireless Networks, pp.421-433, 1997.

[2] M.Beigl, A.Krohn, T.Zimmer, C.Decker, "Typical sensors needed in ubiquitous and pervasive computing", *First International Workshop on Networked Sensing Systems (INSS) 2004*, pp.153-158, 2004.

[3] N.Carriero, D.Gelernter, "Linda in Context", Communications of the ACM, 32(4), 1989.

[4] G.Chen, D.Kotz, "A survey of context-aware mobile computing research", Tech Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[5] P.Fahy, S.Clarke, "Cass – middleware for mobile context-aware applications", In Second International Conference on Mobile Systems, Appli- cations, and Services, 2004.

[6] H.W.Gellersen, A.Schmidt, M.Beigl. "Multi-sensor context-awareness in mobile devices and smart artifacts", The Journal of MONET, 7(5):341-351, 2002.

[7] K.Hinckley, J.S.Pierce, M.Sinclair, E.Horvitz, "Sensing techniques for mobile interaction", In UIST, pp91-100, 2000.

[8] K.V.Laerhoven, N.Villar, H.W.Gellersen. "Multi-level sensory interpretation and adaptation in a mobile cube", In In Proc. of the third workshop on Artificial Intelligence in Mobile Systems, 2003.

[9] J.Mantyjarvi, J.Himberg, P.Kangas, U.Tuomela, P.Huuskonen, "Sensor signal data set for exploring context recognition of mobile devices", In Workshop: Benchmarks and a database for context recognition, 2004.

[10] N.B.Priyantha, A.Chakraborty, H.Balakrishnan, "The cricket location-support system", In Mobile Computing and Networking, pp32-43, 2000.

[11] B.Schilit, N.Adams, R.Want, "Context-aware computing applications", In IEEE Workshop on Mobile Computing Systems and Applications, 1994.

[12] A.Schmidt, M.Strohbach, K.Laerhoven, A.Friday, H.W Gellersen, "Context acquisition based on load sensing", In *In Proceedings of Ubicomp 2002*, 2002.

[13] D.Siewiorek, A.Smailagic, J.Furukawa, N.Moraveji, K.Reiger, J.Shaffer, "Sensay: A context-aware mobile phone", In Proceedings of 2nd International Semantic Web Conference, 2003.

[14] M.Weiser, "The computer for the twenty-first century", Scientific American, pp.94-104, Sep, 1991.

[15] W.Will, "Linuxtuples", http://linuxtuples.sourceforge.net/.

[16] T.Winograd, "Architectures for context", HCI Journal, 2001.

[17] T.Yamabe, K.Fujinami, T.Nakajima. "Experiences with building sentient materials using various sensors", In *In Proceedings of 24th International Conference on Distributed Computing Systems Workshops*, 2004