An On-site Programming Environment for Wearable Computing

Shotaro Akiyama
Grad. School of Engineering, Kobe University
1-1, Rokkodaicho, Nada, Kobe, Hyogo 657-8501, Japan
s-akiyama@stu.kobe-u.ac.jp

Tsutomu Terada
Grad. School of Engineering, Kobe University and JST PRESTO
1-1, Rokkodaicho, Nada, Kobe, Hyogo 657-8501, Japan
tsutomu@eedept.kobe-u.ac.jp

Masahiko Tsukamoto
Grad. School of Engineering, Kobe University Institute for Clarity in Documentation
1-1, Rokkodaicho, Nada, Kobe, Hyogo 657-8501, Japan
tuka@kobe-u.ac.jp

ABSTRACT

In wearable computing environments, it is difficult for users to prepare applications that are used beforehand since there are various situations and places. Therefore, they want to define new services by themselves. In this study, we present a development framework and several tools for developing services in wearable computing environments. The framework consists of an event-driven rule processing engine and service implementation tools, which enable users to program services easily and quickly. The proposed system shows elements of event-driven rules as chips, and we can program services by selecting chips on graphical user interfaces. In addition, the proposed system has two functions considering programming features on wearable computing: genetic-algorithm-based programming and social-network-based programming.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous

General Terms

Human Factors

Keywords

Wearable Computing, Programming, Context Awareness

1. INTRODUCTION

The downsizing of portable computers has attracted a great deal of attention in the field of wearable computing. Wearable computing provides effective and attractive services compared with the use of conventional desktop/mobile devices. People's daily lives can be improved by wearable computing technologies because wearable computers have detailed information about their users. However, to provide services that are highly personalized, the system needs new models for constructing services and for programming.

We focused on real-world end-user programming in this research. Users of wearable computers frequently want to construct trivial services. For example, when someone in a bookstore wants to buy a book but does not have enough money, he/she wants to construct a reminder service, e.g. "Alert me to buy it when I'm next near the bookstore". In this definition, the condition (near the bookstore) and the action (alerts to buy the book) should be defined dynamically at the site where he/she came up with the idea of the service. Therefore, wearable systems need a mechanism that will describe services on site. Moreover, for these services to be advantageous to the general public who are not familiar with programming, the system needs to provide an interface enabling them to implement services easily.

In this study, we propose an On-site Development Tool for defining context-aware services easily and quickly. The proposed system employs a simple model where the user selects elements of event-driven rules with simple operations to define a new service. We named the elements "chips". Moreover, the proposed system has three functions considering the characteristics of programming in wearable computing. One is a function to predict chips that will be selected based on the users' input history and change the presentation order of chips. Another is a function to present new services using a genetic algorithm from the existing services. The other is a function to get services from other users who are friends on social networks.

The remainder of this paper is organized as follows. We introduce related work in Section 2, and Section 3 describes the design of the proposed system. Section 4 describes the implementation of the system and Section 5 presents the conclusion and planned future work.

2. RELATED STUDY

There are already many proposed methods, called visual programming, that do not require a large amount of text input. For example, ESPranto SDK[7] enables the user to program the system including the tangible interface by operation of the GUI. LEGO MINDSTORMS[1] is a visual programming environment that enables the user to define

AH '12, March 8-9, 2012, Megève, France.

the behaviors of robots. Viscuit[4] enables users to program animations by rules graphically drawn. Scratch[2] is an environment to learn programming for children, and Squeak eToys[3] is a programming system using tiled scripts. These studies, however, aim for convenience by visualization, ease of operation by GUI, and teaching programming. They have a different purpose from ours, which is discussing programming in wearable computing environments.

3. SYSTEM DESIGN

In this study, programming a service anywhere and anytime is called "On-site programming". Below are examples of programming on site.

- At a conference, there are many unacquainted people. I define a service that detects my shaking hands gesture and takes a photo with a wearable camera automatically to help put faces to names.
- Someone wants to display a map for a destination on an HMD when they go on a trip. Since it is annoying to always display the map, they construct a service where the map is only shown on the HMD when they are standing.
- Since sitting for many hours is not healthy, someone can implement a service that reminds him/her to have a brisk workout every two hours.
- Since people often forget where they put their cell phone, they can construct a service to remember where and when they last removed it from their pocket.

Note that we define services as parts of applications on a wearable computer. To accomplish this kind of service programming in wearable computing environments, the system needs to fulfil four requirements:
(1) Event-driven programming model
In the above programming, generally we use some events (e.g. when the user shakes hands or when the user is near a book store) as a trigger and do some action (e.g. take a photo or show us a reminder). This means that event-driven architecture is suitable for describing services for wearable computing. Since complex descriptions are not suited to on-site programming, we should use a simple but flexible model.
(2) Dynamic modifications to services when system is running
It is important to continue the system even when new services are installed. Users in on-site programming environments frequently add, delete, or modify services. However, there are important services running in the system, such as a service for sensing vital information.
(3) Simple Operation
Since we suppot end-user programming for wearable computing, the user may not be familiar with programming and hardware management. It is important to minimize user input and simplify the programming.
(4) Service development using existing services
Services in wearable computing are strongly related to human daily life, and many users need similar but slightly different services. This means most services are similar to existing services made by the user or other people.

To fulfill these requirements, our system consists of Wearable Toolkit[6] and an On-Site Development Tool.

```
DEFINE TakePhoto
WHEN CONTEXT˙RECOGNIZED
IF GLOBAL.CONTEXT==’shakehands˙start’
THEN DO MM˙GET˙IMAGE()
```

Figure 1: An example of ECA rule

Figure 2: Rule definition mode

3.1 Wearable Toolkit

Wearable Toolkit fulfills Requirements 2 and 3. The rule engine, a core part of the toolkit, works as middleware for the OS, and it manages wearable devices via plug-ins to enable flexible configuration of devices. Services are described as a set of event-driven rules to make services autonomous and simple.

All services are represented as a set of ECA(event, condition, and action) rules, which is a behaviour-description language in database systems. Each ECA rule consists of three parts: an event, a condition, and an action. The first describes an event occurring in the system, the second describes the conditions for executing actions, and the third describes the operations to be carried out. By using rules, we can configure services flexibly and simply by dynamically adding, deleting, or modifying rules. Moreover, since actions can generate new events, complex behaviours can be achieved by chaining ECA rules. These characteristics fulfill requirements 1 and 2.

3.2 On-site Developing Tool

On-site Developing Tool shows chips, which mean an event, condition, or action, or a combination of them. The user programs a service by selecting the necessary chips on the GUI as shown in Figure 2. The tool provides easy and intuitive operations for rule definition, and also has the function of defining rules using genetic algorithms and social networks. These characteristics fulfill requirements 3 and 4. The tool has three modes: rule definition mode, genetic algorithm mode, and social network mode. We can switch mode by a long press on the up or down cursor keys.

Figure 2 shows a rule definition interface. The chip currently selected is shown a little bigger than others (① in the figure). Chips that can be used with the rule are listed at the bottom of the screen (②). We can change the type of chips (event, condition, or action) with the up or down

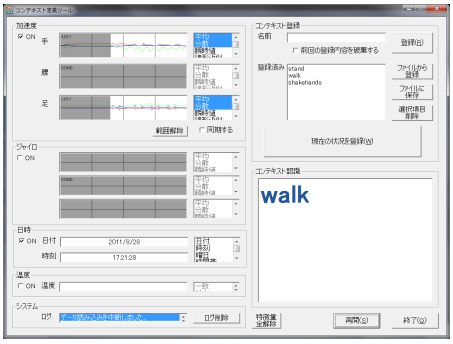


Figure 3: Context definition tool

cursor keys, and choose a chip with the right or left cursor keys. The Enter key confirms the chip (③), and we modify the parameters by cursor keys or character keys (④) if the selected chip has the parameters. After selecting all the chips we need, we define the ECA rule by a long press of the Enter key. At the same time, the defined rule is shown in the line of the event chip as a combination of chips, to be used or modified afterwards. The operation of arranging a chip or outputting a rule can be undone or redone by a long press of the left or right cursor keys.

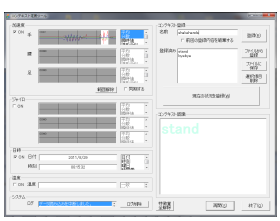
To implement the services on site, the system should provide a function where a user can define a new context as an event. We present a context definition tool to resolve the difficulty with defining contexts. Figure 3 shows a snapshot of the tool, where the acquired sensor data are shown at the left in real time. The user directly selects the window size in this area when defining a context, and he/she selects one or multiple characteristic values from the list on the right-hand neighbor. The defined context automatically appears as a new chip on the tool.

Figure 4 shows an example of defining a rule "When the user makes the gesture of shaking hands, the system takes a photo." To define this rule, firstly the user registers the gesture of shaking hands in the context tool by actually doing the gesture (Figure 4(a)). Secondly, he/she selects an event chip that was added by the previous operation (Figure 4(b)). Next, since the condition chip is not necessary in this case, he/she chooses an action chip of taking a photo (Figure 4(c)). Finally, he/she outputs the rule (Figure 4(d)).

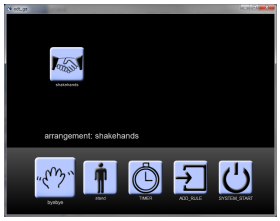
3.3 Chip sorting

At the stage of rule definition, the display order of chips strongly affects the efficiency of programming. Therefore, our system has a function to sort chips in accordance with the history of rule definitions for minimizing the operations to define a new rule. Concretely, the system has variables for storing the frequency of usage based on an n-gram model. For example, if a user registers a rule using Event E_1, Condition C_1, and Action A_1, the system increases the score of 3-gram $E_1 \rightarrow C_1 \rightarrow A_1$, 2-gram $E_1 \rightarrow C_1$, $E_1 \rightarrow A_1$, and $C_1 \rightarrow A_1$. When the user is making a new rule, the system displays chips in order of n-gram scores in accordance with the current input of the rule (e.g. when the user selects E_1, C_1 and A_1 will be displayed as high priority). This sorting is performed at every user input.

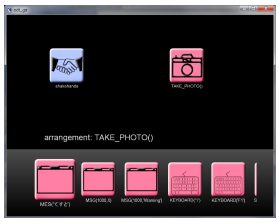
3.4 Genetic algorithm mode



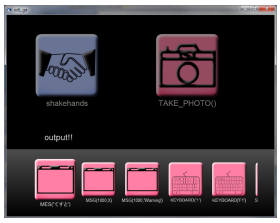
(a) register the gesture



(b) select an event chip



(c) select an action chip



(d) output an ECA rule

Figure 4: rule definition steps

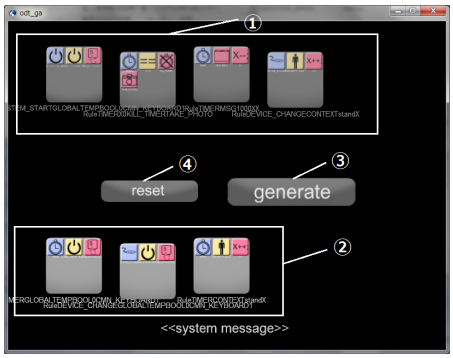


Figure 5: Genetic algorithm mode

In on-site programming, there are many cases where a new service is partially the same as the previously defined services. In other words, if a user makes a rule with a specific event, he/she tends to make another rule with the same event. For example, if a user makes a rule to take a photo on shaking hands, the rule to record the current position with the same event is also useful to him/her.

Therefore, we propose a service definition mode using genetic algorithms as shown in Figure 5. In this mode, the system shows chips that are already defined in upper area (①). By pushing button (③), the system generates new rules in the lower area based on a genetic algorithm (②). We can select and use generated rules. We use roulette-wheel selection as a selection algorithm, and a priority score as the fitness value. The priority score is increased by selecting a rule in the upper area by hand. There is a button to reset the score (④).

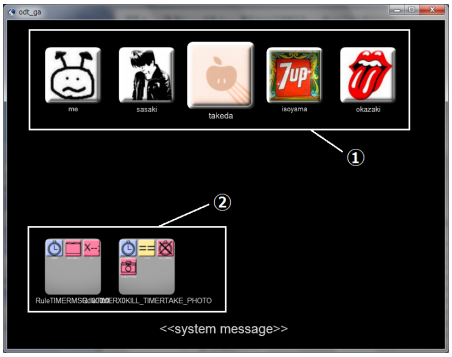


Figure 6: Social network mode

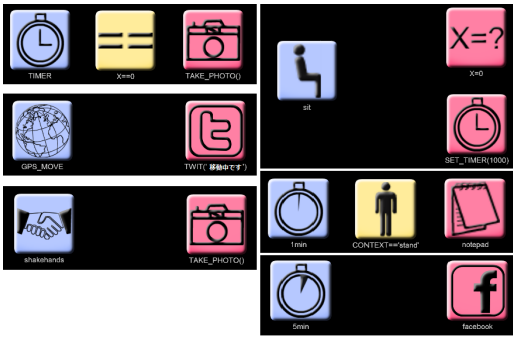


Figure 7: Examples of defined rules

We show a concrete example of ECA rule generation. We assume that there are four rules: "when the user makes the gesture of shaking hands, it takes a photo", "when the user is running, it pauses the system", "when the user has one minute to kill and he/she is standing still, it shows a notepad", "when the user has five minutes to kill, it shows the user's facebook page". By using our mechanism, the system generates new rules such as "when the user has one minute to kill, it shows a weather forecast", "when the user has five minutes to kill, it takes a photo." It may help users to use new services.

3.5 Social network mode

In on-site programming, there are cases where rules that people under similar situations have are also useful to the user. For example, if the rule "when the user arrives at an event venue, it sends a message to Twitter" is used by a person, the rule is also useful for aother person who is on the way to the same place. Moreover, the rules of a person who has similar characteristics to the user are more useful than the rules of strangers.

Therefore, we propose a service definition mode using a social network as shown in Figure 6. The system shows friends in the upper area in order of their similarity to the user (①). The rules are shown in the lower area by selecting a friend (②), and we can use/modify any displayed rule. The similarity is calculated based on the rules that the users have.

4. IMPLEMENTATION

We implemented a prototype of the proposed system. The On-site Developing Tool is implemented as an Adobe AIR application running on Windows 7. We can operate the system with only four cursor keys and the enter key of a small keypad or joystick. As a preliminary evaluation of the sorting function, we measured the number of operations to define eight rules that are actually likely to be used in daily-life. Using our sorting method, it takes 85 steps to define all rules while 242 steps are required when not using our sorting method. In addition, we made a preliminary evaluation of the genetic algorithm mode. We actually made 20 rules from the 4 rules described in Section 3.4. Subjectively, 5 rules out of 20 are useful.

Figure 7 shows examples of defined rules. From top left, they mean "when the timer is started and the count is 0, it takes a photo", "when the position changes, it posts 'I am moving now' to Twitter", "when the user makes the gesture of shaking hands, it takes a photo", "when the user sits, it initializes the count and measures the duration he/she is sitting", "when the user has one minute to kill and he/she stands still, it shows a notepad", and "when the user has five minutes to kill, it shows a his/her facebook page". The last two services are examples of that the system helping the user to use time efficiently.

5. CONCLUSION

In this paper, we proposed an on-site programming environment for wearable computing. The proposed system helps users to define rules with few keys and few operations. The functions of the social network mode and genetic algorithm mode are specialized for programming in wearable computing environments and have the possibility to make the programming drastically easier for the end-user.

On the other hand, we did not evaluate the proposed mechanism in actual environments and will evaluate it by feedback from end-users and investigate further improvement of our tools. The effectiveness of implementing services in ECA rules is confirmed in our previous researches[5]. End-users usually cannot think of the correct idea of services they want. Our tools present several hints to think of concrete services and motivates users to implement services.

6. REFERENCES

[1] Lego mindstorms. http://www.legoeducation.jp/mindstorms/.

[2] Scratch. http://scratch.mit.edu/.

[3] Squeak etoys. http://www.squeakland.org/.

[4] Y. Harada. Learn programming with viscuit. Journal of the Institute of Electronics, Information and Communication Engineers, 87(8):674–677, 2004.

[5] M. Miyamae, T. Terada, M. Tsukamoto, S. Nishio, K. Hiraoka, and T. Fukuda. An event-driven wearable system for supporting motorbike racing teams. Wearable Computers, IEEE International Symposium, pages 70–76, 2004.

[6] T. Terada and M. Miyamae. Toward achieving on-site programming. Wearable Computers, IEEE International Symposium, pages 3–10, 2009.

[7] R. van Herk, J. Verhaegh, and W. F. Fontijn. Espranto sdk: An adaptive programming environment for tangible applications. ACM Conference on Human Factors in Computing Systems, 2009.