An Active Program-based Design for User-Centric System by Symbiotic Computing

Shigeru Fujita
Department of Computer Science
Chiba Institute of Technology
2-17-1, Tsudanuma, Narashino-shi, Chiba-ken, Japan
fujita@cs.it-chiba.ac.jp

Kenji Sugawara
Department of Information and Network Science
Chiba Institute of Technology
2-17-1, Tsudanuma, Narashino-shi, Chiba-ken, Japan
suga@net.it-chiba.ac.jp

***Abstract*— User-centric systems are active programs which always pay attention at users. In this case, users don't need to give explicitly their position to a system which receives it thanks to electronic devices. More, a user-centric system has to provide services to users without explicit request. We have proposed the concept of Symbiotic Computing, which can become the base technology of user-centric systems. We present a concept based on agent-oriented programming, for building systems that support users automatically.**

Keywords-user-centric; symbiotic computing;

I. INTRODUCTION

The kind of activities required to develop a human-oriented and personalized computing is an open problem of software engineering [1]. The eXtreme Programming (XP) [10] is mainly considered as a human activity in software engineering. The Service-oriented Architecture (SOA) may be used to generate user-centric services and contents [2]. XP and SOA require human-activities on system development and maintenance, but we are more concerned by automatic system adaptation to users. The title, "an active program–based design for user-centric system by symbiotic computing" claims that a user-centric system should be designed as an active system. Such a system is built around an autonomous agent. This concept was proposed in paper [11].

A middleware for user-centric systems is a new foundation of platforms that deliver services [3][4]. Infrastructure as a Service, IaaS, Platform as a Service, PaaS and Software as a Service, SaaS are developed as a cloud computing. IaaS, PaaS and SaaS are good solutions to satisfy number of users in the Internet, but a service in a cloud computing should be operated by users.

Advanced research works presented during The Fourth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services (CENTRIC 2011), identified several problems to achieve a user-centric system. The first one is a "Low user acceptance" [5]. Many devices support users to satisfied their requirements, but, in general, not all users are satisfied with those services. Users feel to be forced to use specific devices and to obey complex procedures. Users may be bored with using digital devices and networked services; they have other things to do. We defined two requirement specifications to create user-centric systems. The first one is that a user-centric system always has to be aware of the user's situation without expressed authentication. The second one is that a user-centric system has to bring services to users without explicit instruction.

In other words, an intelligent agent is necessarily required for a user-centric system. We focus on two things that are: find a proper user for a system and deliver a service without action of the user.

II. SYMBIOTIC COMPUTING

To develop a user-centric system, we use the concept of Symbiotic Computing [6]. Symbiotic Computing was proposed to bridge the gap between the real space and the digital space by creating symbiotic relations between users in the real space and information resources in the digital space. Symbiotic Computing is not ubiquitous computing as it may seem at a first glance. The key concept of Symbiotic Computing is based on Perceptual Functions and Social Functions. A Perceptual Function translates a signal from real space into a symbol in Symbiotic Space. Social Function translates a signal inside the Symbiotic Space. A signal in real space is captured by processes (**P** in Figure 1). A signal in Symbiotic Space is captured by elements of the Symbiotic Space itself.

Symbiotic Computing looks like Ubiquitous Computing, but normally Ubiquitous Computing does not care about user. A system based on Symbiotic Computing always keeps a contact with a user. This concept is *a priori* mechanism.

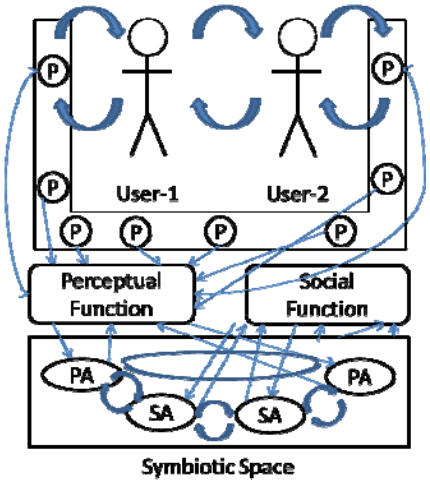

Figure 1. Symbiotic Computing

 ISBN: 978-1-61208-232-5

In Figure 1, **P** may be a computational process which provides a service to a user such as a web browser, a mailer or a smartphone application and so on, or **P** may be a sensor which captures a signal in real space such as a video camera, a GPS device, or a microphone and so on. **PA** is a Partner Agent and **SA** is a Social Agent. A partner agent is attached to a unique user in real space. On the other hand, a social agent is an entity which corresponds to a social activity in real space and digital space.

III. PASSIVE VS. ACTIVE

Most of applications in a smart phone such as iPhone or Android device achieve passive actions from user's operations. A basic action of a passive system occurs from a user's request. A little bit more active system is able to trigger some action. A simple push action may be driven by a timer in a system. A typical example is an alarm for waking up a user application. A smarter push application may use another system (for example, on the internet) to provide suitable information. A passive system is shown in Figure 2.

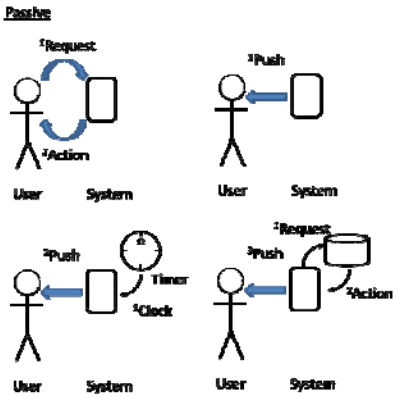

Figure 2. Interactions Between a User and a Passive System

Another problem of passive user-centric system is the incapacity to recognize a user. A user does not appear in front of a system. But, a system will still push information for user. A worse case is when a fake user uses a user-centric system. Of course, a user-centric system would have an authentication for a proper user. But, sometimes, a user does not care about a device after authentication. A fake user takes a benefit for a proper user. On the other hand, a proper user may become nervous with a user-centric system, because, the system often pushes information without considering the user's situation or context. For example, a user in an important meeting may see the system pushing information about an Olympic game score. And the more a user gets skills to use a system, the more she becomes irritated by an inappropriate system. This problem in passive system is shown in Figure 3.

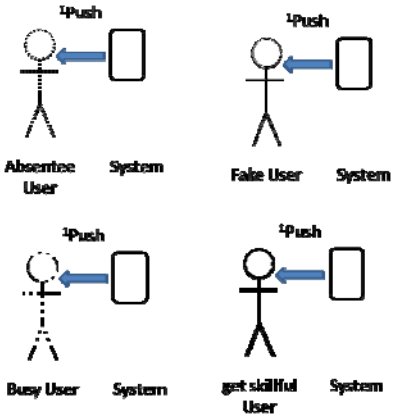

Figure 3. Passive System Fails User Cognition

To avoid problems caused by passive systems, a user-centric system must have two functions. The first one is a user cognition function and the second one is a generating service function. A simple active system model is shown in Figure 4. When using an active system, a user does not make actions for triggering effects; an active system always follows the user's behavior by using Perceptual Function and Social Function.

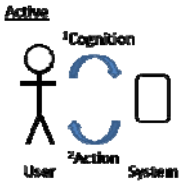

Figure 4. Active System

IV. EXPERIMENTAL ENVIRONMENT

We show in Figure 5 an experimental environment to test an active system in Symbiotic Computing. Four web cameras are capturing scenes every minute in a laboratory. The first objective is to identify a user from captured images.

The graphical image processing is supported by the OpenCV, the well-known middleware for graphical based image recognition. A captured image is an example of signal in real space and is translated into symbols by perceptual functions. When a web camera captures a room image, a system supported by the OpenCV captures a face image from this room image, and then looks for a corresponding face image in database thanks to the SIFT algorithms. Finally, a system creates a symbol which identifies Mr. A in the room image (Figure 6).

 ISBN: 978-1-61208-232-5

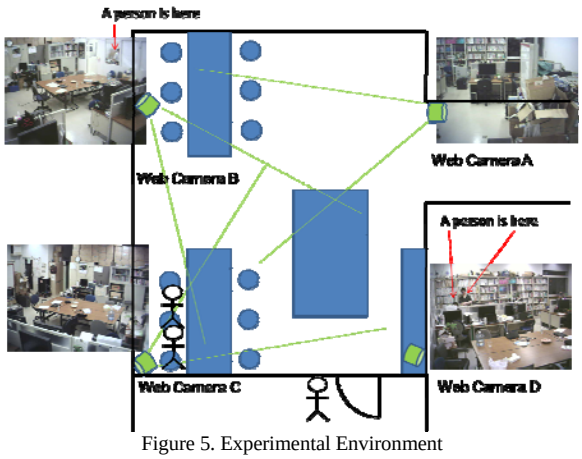

Figure 5. Experimental Environment

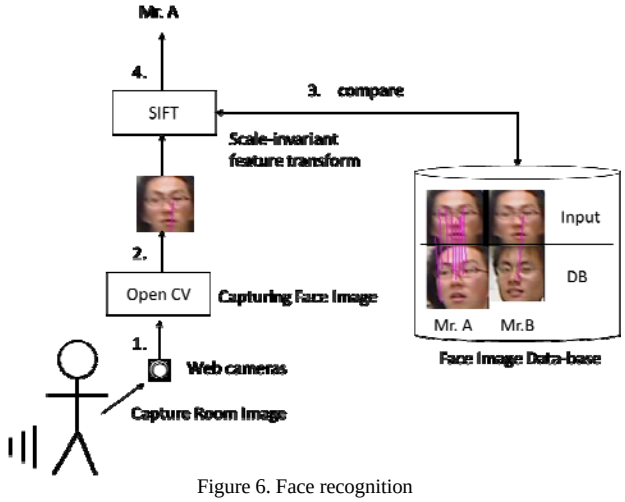

Figure 6. Face recognition

V. ACTIVE PROGRAM BASED DESIGN

The concept of active system has been presented in Section III. We make a template of active program based design for user-centric system. Agent-oriented programming [8] is a base of our template.

Agent-oriented programming is based on "mental-state" to check a set of assumptions about agent cognition. The mental-state corresponds to an expression of real space and identifies a proper user.

Partner agents always follow a user. A partner agent captures real world elements and generates symbols. A symbol comes from one of it Perceptual Function. A symbol expresses a real world element with Vectors. Vectors are a set of symbols and value vectors. A Decision is due to a set of rules applied by an active action to serve a user. All of rules begin to check a user's symbol. If a user's symbol does not exist in Vectors, a default rule of a partner agent is fired for searching a user. A Criterion is a set of meta-rules of a partner agent. A rule in a criterion evaluates the Perceptual Functions which create a symbol. If a bad symbol comes from a Perceptual Function, a Criterion will require another Perceptual Function to create a good symbol for identifying a user. The Partner agent model is shown in Figure 7.

Currently, we are implementing the concept of active program based design for user-centric system with our multi-agent framework [9] written in Steel-Bank Common Lisp.

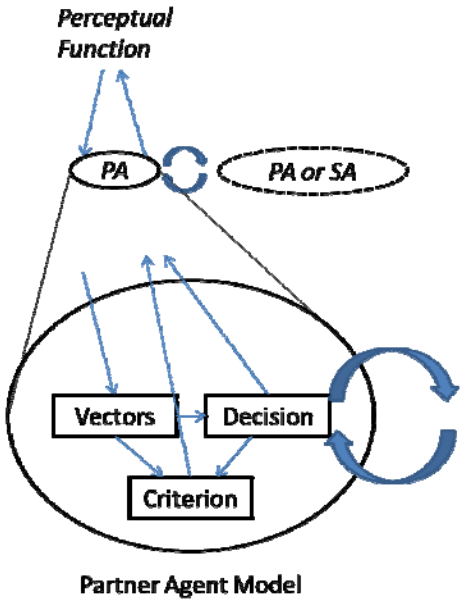

Figure 7. Partner Agent Model

VI. CONCLUSION AND FUTURE WORK

In this paper, we showed the problem of passive system which does not recognize user. We also showed the concept of Symbiotic Computing which is based on an agent-oriented programming model. A system which is based on Symbiotic Computing principles is able to recognize a user. Such a system will support a user without reconfiguration by a developer.

In order to support a user, a system always uses two functions. The first one is the function that recognizes a proper user at every time. The second one is the function that automatically delivers a service by considering the user's situation.

We showed an experimental environment in section IV. And, we currently implement a prototype system with our multi-agent framework. The prototype system will identify a proper user.

In the future, we will implement perceptual function and partner agent, and we will define a formal design model of a user-centric system based on Symbiotic Computing.

REFERENCES

[1] M. DeBellis and C. Haapala, "User-Centric Software Engineering," IEEE Expert, pp. 34–41, February 1995.

[2] S. Komorita, M. Ito, H. Yokota, C. Makaya, B. Falchuk, D. Chee and S. Das, "Loosely Coupled Service Composition for Deployment of Next Generation Service Overlay Networks," IEEE Communication Magazine, pp. 62-72, January 2012.

 ISBN: 978-1-61208-232-5

[3] R. E. Schantz, "Middleware for Distibuted Systems," http://www.agentgroup.unimore.it/didattica/ingss/Lec_Middleware/Schmidt_Middleware.pdf (last access, 2012/July/31).

[4] K. Henricksen, J. Indulska, T. McFadden and S. Balasubramaniam, "Middleware for Distributed Context-Aware Systems," http://henricksen.id.au/publications/DOA05.pdf (last access, 2012/July/31).

[5] Y. Chen and C. Leung, "A Study on the Lost Seeking Devices and Systems for Dementia-Patients," CENTRIC 2011, pp. 15-21, ISBN: 978-1-61208-167-0, October 23-29, 2011 - Barcelona, Spain

[6] N. Shiratori, K. Sugawara, Y. Manabe, S. Fujita and B. Chakraborty, "Symbiotic Computing Based Approach Towards Reducing User's Burden Due to Information Explosion," Journal of Information Processing, Vol.20, No.1, pp. 37-44, January 2012.

[7] K. Sugawara, Y. Manabe and S. Fujita, "Mobile Symbiotic Interaction between a User and a Personal Assistant Agent," ICCI*CC2012, August 22-24, 2012, Kyoto, Japan

[8] Shoham, "Agent-oriented programming," Artificial Intelligence, Vol.60, pp. 51-92, 1993

[9] S. Fujita, K. Sugawara and C. Moulin, "The Design of a Symbiotic Agent for Recognizing Real Space in Ubiquitous Environment," Advances in Cognitive Informatics and Cognitive Computing, Springer, pp. 13-32, 2010

[10] Kent Beck, "Extreme Programming Explained: Embrace Change," Addison-Wesley Professional, 1999

[11] S. Fujita, H. Hara, K. Sugawara, T. Kinoshita, N. Shiratori, "Agent-Based Design Model of Adaptive Distributed Systems," Applied Intelligence, Vol. 9, pp. 57-70, 1998

 ISBN: 978-1-61208-232-5