uBlocks: Enabling Hand-made Distributed Application Among Ubiquitous Embedded Devices

Masayuki Iwai[1], Jin Nakazawa[1], and Hideyuki Tokuda[2] [1]
[1]Graduate School of Media and Governance, [2]Faculty of Environmental Information, Keio University
5322 Endo, Fujisawa, Kanagawa 252–8520, Japan, tailor@ht.sfc.keio.ac.jp

Abstract

Increase of future appliances at home environment causes confusion with less computer-skilled users. Consequently, embedded-hardware could prevent us from recognizing all of them by ourselves. Furthermore, wireless connection among embedded devices also makes users not have a clear grasp of which kind of application is running among them. This kind of user's unawareness should make the speed of embedded ubiquitous systems to the home slow.

To overcome above mentioned issues, we have developed uBlocks that enables visual hand-made composition of software component application among heterogeneous devices by users themselves.

1. Introduction

Over the past few years, we have seen an evolution of techniques for networked and Java-enabled devices that are aimed for use in home and office. The age in which PC's dominate all other devices and networked appliances will come to the end in a few years. As one expect, network-ready tiny hardware has accelerated this trend. Furthermore, software for embedded devices such as light-weight Java and embedded Linux facilitates combinations of the software that is distributed over a network. In the near future, there is no doubt that more and more appliances will have networked connectivity and Java-enabled Runtime: an IP cellular phone, door, a networked TV, even a networked cup.

Increase of these future appliances at home environment, however, causes confusion with less computer-skilled users. Consequently such a number of embedded-hardware should prevent us from recognizing all of them by ourselves. Furthermore, wireless connection among embedded devices also makes users not have a clear grasp of which kind of application is running among them. This kind of user's unawareness should make the speed of embedded ubiquitous systems to the home slow.

To overcome above issues, we have developed uBlocks that enables visual **hand-made** composition of software components among heterogeneous devices by users themselves. Hand-made means that distributed applications are not created by developer as ready-made one but by a user by herself/himself.

The uBlocks, unlike other distributed components visual programming tools, is distinguished in three major features: multimodal visualization mechanism, multiple modeling mechanisms, and high semantic event communication mechanism. In this paper we mainly describe multimodal visualization mechanism. This feature enables various visualization tools which can construct hand-made distributed application among embedded software components.

We developed Java 2D/3D based GUI , command-line based user interface, and HTTP-based interface through cellular phone. This multimodal feature can provide suitable environment for any skill-level user in any occasion. Then, the cost of management and construction of distributed application can be reduced.

The remainder of this paper is organized as follows. In section 2, we describe requirements of creating application in a ubiquitous computing environment. The multimodal interface of uBlocks is mentioned in section 3, which include an architecture which can consist distributed model. Related works are addressed in section 4, and summary in section 5.

2. Future Embedded Ubiquitous Application Requirements

In a next few years, embedded devices will be come smaller and smaller. Therefore, every appliances and sensors must have a network connectivity and controllable through the network. As shown in Figure1, we experimentally constructed such a room, in which many sensors and appliances exit, named Smart Space Lab[9].

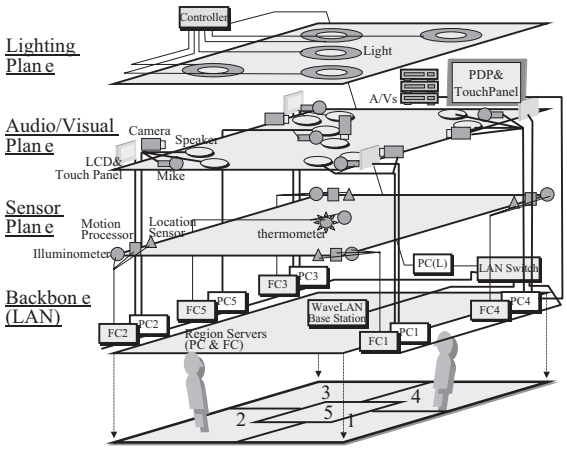

Figure 1. Smart Space Lab with heterogeneous, numerous embedded devices

From everyday experience in this Smart Space Lab, we have become aware of tree requirements of application creation for ubiquitous environment.

2.1. I:@User Customizable Hand-made Application Needs

Any attractive distributed application, as it looks at first, could be old-fashioned and mismatched. Because members of a community and embedded products change dynamically home environment, the regularly fixed ubiquitous applications will become cheap and useless very soon. The person who updates applications in the home area must not be developers but users from the view point of running cost. You must not expect too much of special skilled system engineers at home area.

2.2. II:@Consideration of Multiple Users

In our Smart Space Lab at Keio University, we have meeting, demonstration and work. In such a public and multipurpose space, there is a big possibility that many people control and setup ubiquitous application from different places. It is needed that middleware can detect collisions of the different setting from different users.

2.3. III:@Adaptation for Various User's Circumstance

In home area network, we can not always assume that users always use same computer at a same place. In the same way, users' computer skill levels have a large difference. To adapt to these user's various circumstances, Multimodal user interfaces are required for maintain the usability.

3. Multimodal User Interface Architectures

uBlocks provides four supporting GUI Browsers that can configure the event paths for distributed services visually.

uBlock is implemented on Jini Architecture[8] and event delivering middleware:Dragon[3, 2]. Dragon is a framework for constructing flexible inter-service event-driven application systems. Dragon includes a soft real-time event delivering architecture and an composite-service framework. Dragon support service to notify specific information asynchronously from an object to another within a deadline time in a best effort way.

As users may not stay at same place for a longtime, a user's circumstance is changing hourly. It is needed to provide user-interfaces which can adapt to users' various environment.

Then, we developed various user interfaces. Java 2D browser(Figure2), 3D based browser, web-based interface, and shell type UI. Each user interface is developed on a fundamental module named "uBlockBase". uBlockBase holds service name, service status, and connection vector. Using uBlockBase, every multimodal interface can run simultaneously.

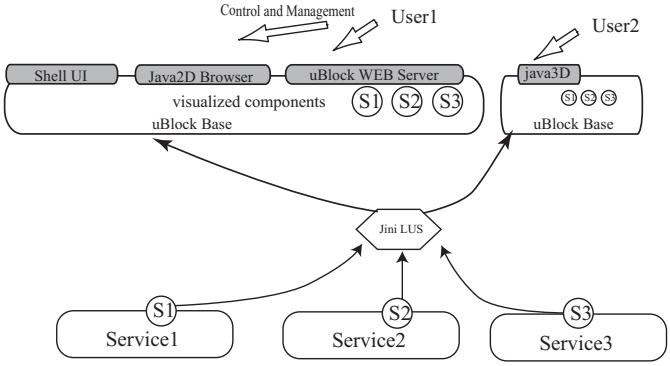

Figure 2. multiple user-interface

3.1 Java2D Browser

Figure 3 shows a screenshot of Java2D uBlock Browser. This browser is aimed for supporting users to create ad-hoc event-driven distributed application. Users who have no programming skills can see which type of services exist on the network and can setup the event-driven system that they want to use by using this browser.

When a user brings a mouse cursor to an icon, the icon shows the service detail information: name, location, vender name, and serial number. When the user clicks the icon, the icon shows the interface and

the line to connect. Then, the user moves the mouse cursor to another service icon. If the interfaces match each other, this editor repaints the icon and the line in blue and shows the details of the service. If the interfaces do not match each other, this editor repaints the icons in red. The user clicks blue colored icons to connect them. This action is transmitted immediately to their target distributed services and each target service tries to register itself to upper service of event stream. In this way, the user can configure event propagation path and can make flexible ad-hoc event-driven system easily.

There are three simple connection types: normal connection, priority connection, and disconnection. This simplicity eases users' troublesome setup step for construction of distributed services.

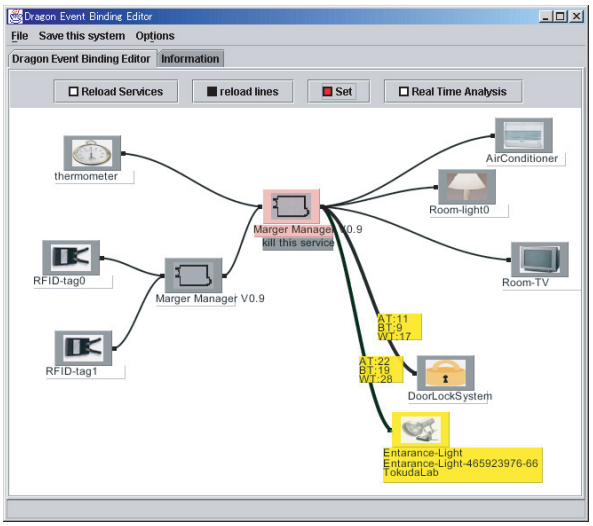

Figure 3. Java2D uBlock Browser

3.2 Java3D Browser

We developed Java3D-based browser with Yuu Furuichi. Through touch-panel, user creates sticky editor that define the control information of appliances. Figure4 shows 3D modeling of Smart Space LabD

3.3 uBlock WEB Browser

Users are always moving at ubiquitous environment. In any circumstance, users could be constructing distributed ubiquitous application through their cellular phone. This web-based uBlock Browserfs transition of pages are shown in Figure5.

However, It is difficult to represent rich context of services because of the limited resources of cellular phone. Accordingly, we designed these character-

Figure 4. Java3D uBlock Browser

based interface simply. Users can control and setup distributed ubiquitous application through cellular phone by following tree steps.

1: First, a user select the kind of services. "s" means source service which generate events and "d" means distinction services devices which consumes events. System shows available services to the user.

2: Second, users can select the service. This web-based uBlock browser, then, shows connectable list of services in a same environment.

3: In each element of list, there is a opposite connection type link. When components are connected to to each other, the system shows a disconnection link.

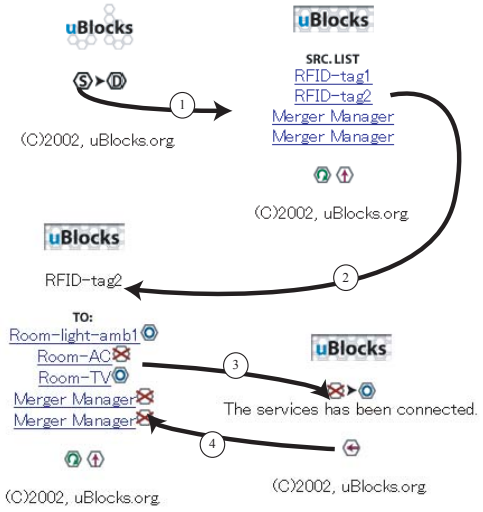

Figure 5. Page Transition of Web-based uBlock Browser

As shown in Figure 6, Users can run multiple user interfaces simultaneously. If a user create an application through a cellular phone, other users who are

looking at Java2D Browser can see the correct updated distributed application immediately.

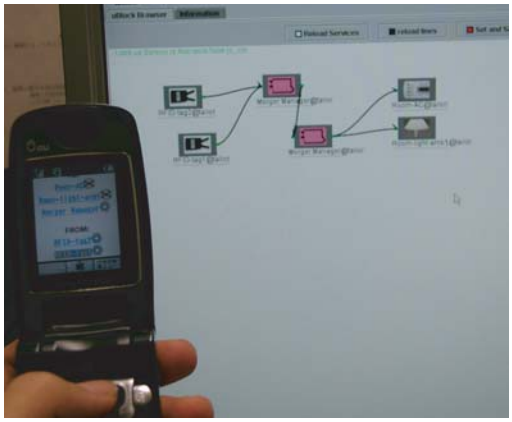

Figure 6. WEB-based uBlock Browser and Java2D Browser are working together with consistency

3.4 Shell UI

Shell UI is a character-based interface for debugging. There are four commands: *list*(show the information of each service components), *connect*(start event delivery to lower components), disconnect(stop event delivering to lower components), and kill(stop advertisement of service). In the near future, we plan to be compatible with JXTA shell[7].

4. Related Works

In this section, we describe some related works to create flexible distributed application.

A Reflection Based Tool for Observing Jini service (Carp@)[1] is management tool for dynamic distributed Jini systems. Carp@ allows observation of services and clients, their interconnections and the messages exchanged among them. The tool extracts an architectural component model based on components, ports and connectors. However, Carp@ is not aimed at multimodality unlike uBlocks. Furthermore, we focus on the event propagation path by connecting Event Managers linearly. Agent Anytime Anyware[4] also provides both the configuration graphic tool and notification-based mechanism. C2 [5] provides a GUI configuration tool based on JavaBeans named "aribica". To compare with uBlokcs, these above two researches have richer function of GUI tools. For example property configuration dialog and saving mechanism. However, uBlocks's GUI supporting tools dynamically detect service generation, expiry, and attribute change, because we assume that uBlocks is used at home and factory. COCA[6] can transform heterogeneous ubiquitous computing resources through a process called classification into a conceptualized representation, which allows high-level manipulation and configuration by ubiquitous computing applications. However, COCA dose not focus on user-interface and simplicity in creation of application. we have developed uBlocks for the purpose of providing not only multi-modal application but multi-modal user interface.

5. Conclusion

In this paper, we presented a new framework for constructing hand-made distributed application with visual editor by users, referred to as uBlocks Browsers. UBlocks includes an composite-service framework from multimodal interfaces. The users can create ad-hoc distributed systems without the cost of configuration by using GUI named uBlock Browsers. This browsers support users to control whole disturbed ubiquitous applications without programming intuitively.

uBlocks's latest software version, manuals, and evaluations are available at www.uBlocks.org.

References

[1] M. Fahrmair, C. Salzmann, and M. Schoenmakers. A Reflection Based Tool for Observing Jini Services. In *Reflection and Software Engineering*, pages 209–227, June 2000.

[2] M. Iwai, J. Nakazawa, and H. Tokuda. Dragon: Soft Real-Time Event Delivering Architecture for Networked Sensors and Appliances. In *The 7th International Conference on Real-Time Computing System and Applications*, pages 425–432, Dec. 2000.

[3] M. Iwai, J. Nakazawa, and H. Tokuda. Flexible Distributed Event-Driven Programming Framework for Networked Appliances and Sensors. In *The 3rd International Symposium on Distributed Objects and Applications Short Papers Proceedings*, pages 61–68, Sept. 2001.

[4] S. Lacourte and L. Bellissard. Agents anytime anyware whitepaper. http://dyade.inrialpes.fr/aaa/.

[5] D. S. Rosenblum and R. Natarajan. Supporting architectural concerns in component interoperability standards. In *IEE Proceedings Software*, 2000.

[6] S. Schubiger-Banz and B. Hirsbrunner. A Model for Software Configuration in Ubiquitous Computing Environments. In *Pervasive 2002, LNCS 2414*, pages 181–194, Aug. 2002.

[7] Sun Microsystems, Inc. JXTA v1.0 Protocols Specification. http://www.jxta.org/project/www/docs/ProtocolSpec.pdf.

[8] Sun Microsystems Inc. *Jini Architecture Specification*, Oct. 2000.

[9] Tadashi Okoshi et al. Smart space laboratoty project: Toward the next generation computing environment. In *IWNA2001*, Feb. 2001.