Dynamic Conflict Detection and Resolution in a Human-Centered Ubiquitous Environment

Haining Lee[1], Jaeil Park[1], Peom Park[1], Myungchul Jung[1], and Dongmin Shin[2]

[1] Industrial & Information System Engineering, University of Ajou, Suwon, Korea
[2] Hanyang University, Ansan. 1217, Kyunggi-do, Korea
{leehaining, jipark, ppark, mcjung}@ajou.ac.kr,
dmshin@hanyang.ac.kr

Abstract. In this paper, a Conflict Control Manager (CCM) for a ubiquitous services system is presented to prevent the mode confusion of humans. CCM consists of a lock-based conflict detection module and a D-PRI (dynamic priority)-based conflict resolution. By means of CCM, the mode confusion can drastically be reduced, and, as a result, CCM can assist in designing and implementing a human-centered ubiquitous environment. Through a case study, it is observed that CCM can successfully detect and resolve the runtime conflicts caused by multiple devices interconnected in a ubiquitous environment. It can also be used to detect the potential conflict risk during the service registering phase so that computerized devices are deployed to improve the human interactions with them.

Keywords: Conflicts detection, resolution, human-centered ubiquitous environment, human interaction.

1 Introduction

As a high degree of automation is achieved with rapid development of computer technologies, the way a human works in a system has drastically changed. Two separate approaches to research related to human-involved complex system have been pursued – automatic coordination and human-computer interactions. In spite of the significant work conducted in both areas, very limited crosscutting research has been conducted in the important integration area. Furthermore, as the interaction between human and computerized systems becomes complicated, the importance of the coordination of a system and conflict management has been emphasized. As the degree of computerization increases, a large number of topics have become critical in terms of human-computer interaction. Especially, the clumsy automation without considering the human factors view cannot achieve the desired benefits of the system automation that most of information technology nowadays is associated with [1].

In the ubiquitous service system, one of the most crucial research areas from the human-computer interaction perspectives, conflicts are frequently incurred at runtime when several services are requested for a device and/or the environment variables of

C. Stephanidis (Ed.): Universal Access in HCI, Part II, HCII 2007, LNCS 4555, pp. 132–140, 2007.

different types of devices are set by different users at the same time. These conflicts may result in the deadlock of the service system. Far more importantly, they may cause the degradation of situation awareness of humans involved in a ubiquitous environment where a variety of devices are interconnected. Since a ubiquitous environment needs to be designed in a human-centered manner, it is crucial to ensure that humans can be aware of the system when an unanticipated situation occurs due to conflicts among devices.

For this reason, this paper presents a conflict detection and resolution mechanism for improving the human interaction with devices that provide services to humans in a ubiquitous environment. The main purpose of this paper is to develop a reliable and robust ubiquitous service system by detecting and resolving runtime conflicts incurred when two or more services are requested for ubiquitous access in the ubiquitous service system. A test bed ubiquitous environment is implemented with WS-ECA framework that exploits Web service technologies, and the effectiveness of the proposed conflict management system is assessed. Through a case study, it is observed that CCM can successfully detect and resolve the runtime conflicts caused by multiple devices interconnected in a ubiquitous environment.

The remainder of paper is organized as follows. Section 2 presents the related works. Section 3 analyzes the conflicts in ubiquitous services system and proposes a conflict detection and resolution mechanism. The practical implementation is presented in Section 4. Section 5 concludes the paper and discusses future work.

2 Related Work

Recently, the importance of human-computer interaction has been emphasized and several researches have been conducted. Especially, human-robot interaction has been one of active research and has received much attention from various perspectives. Several approaches toward the development of intelligent robot for man-machine interaction have been pursued [2] [3].

Situation awareness is also critical in a dynamic environment since it plays an important role in dynamic human decision making [4]. It is desirable to ensure that a ubiquitous environment can be of use to humans such that humans can enjoy benefits of services provided by devices in the system.

Recently, the coordination of multiple devices has received a large amount of attention to realize a ubiquitous environment. Especially, conflicts among devices are known to have the significant influence on the system running in consideration with humans. While a significant attempt at static conflict detection has been made by Imperial College [5], the very crucial and complex issue of dynamic conflict detection in ubiquitous service management has yet to be resolved in an effective way.

Current advances in conflict detection and resolution have been primarily focused on static, compile-time environments [5]. There exists, however, a compelling requirement to examine the very complex and unresolved issues surrounding dynamic (or run-time) conflict detection and resolution.

3 Conflict Detection and Resolution Mechanism

3.1 Conflict Classification

In the human-centered ubiquitous environment, dynamic conflicts are frequently incurred at runtime when several services are requested for a device and/or the environment variables of different types of devices are set by different services or users at the same time. We consider the conflicts that may impose negative impact on the human-centered ubiquitous environment and identify two types of conflicts: (1) resource conflicts and (2) environment variable conflicts. The resource conflicts happen when one device which is performing a service is interrupted by other services. This device should be used exclusively and not be interrupted by other services while performing its service. For example, a TV recorder is scheduled to record Channel 1 from 1:00 to 2:00pm. If another service attempts to use the same recorder to record Channel 2 from 1:10 to 1:30, a resource conflict happens.

Some variables of devices can be used to describe the environment surrounding a ubiquitous service system, such as temperature, noise level, illumination, etc. These environment variable conflicts happen when users request to change the environment variables of devices, which are common for them. For example, while a sleeping service requested by one user sets the "noise level" of his/her house to be "silent", the other user, who likes to use a cleaner at the same time, raises the "noise level" of the house to be "noisy". This is a typical example of the environment variable conflicts.

Resource conflicts can cause the mode confusion when a human uses a device that is subject to the resource conflict. On the other hand, environment variable conflicts can degrade the quality of services that a human expects to enjoy, resulting in the poor human interactions with devices.

3.2 Conflicts Detection and Resolution Mechanism

As mentioned in the previous section, all the conflicts incurred in the human-centered ubiquitous environment can be decomposed into a collection of resource conflicts and environment variable conflicts. The detection and resolution of these conflicts can be a key to ensuring that humans can be aware of the system when an unanticipated situation occurs due to conflicts among devices. In an attempt to detect and resolve conflicts, we suggest (1) lock-based conflicted detection and (2) D-PRI (Dynamic Priority) conflicts solution.

Lock-based Conflict Detection Method. The lock-based conflict detection method is devised to manage universal access to devices and their environment variables in the human-centered ubiquitous environment. Each time a service requests an access to a device or manipulates its environment variable, the service sends a lock request to the conflict control manager, which registers the lock and checks the system for runtime conflict detection. The conflict control manager can accept the lock request for an item (a service or an environment variable) if the item is currently free or there is no lock held on the item. The conflict control manager uses the lock compatibility

database in which the compatibility of services is registered in terms of items when an item can acquire multiple services at the same time. For instance, services A and B can run on device 1 at the same time, or Services A and C can not run on device 1 simultaneously, to name a few. If an item is already locked by a service, a new lock to the item by other services can be granted only if the lock request is compatible. If the lock request is not compatible, the services wait for the lock timeout interval of the previous service to expire.

In summary, the lock compatibility database is used to check the compatibilities between different lock requests. Conflicts are incurred when the new requested lock on the item is incompatible with the previous one. Figure 1 shows the lock-based conflict detection mechanism. The lock-based conflict detection method enables the detection of resource and environment variable conflicts by checking the Lock Compatibilities.

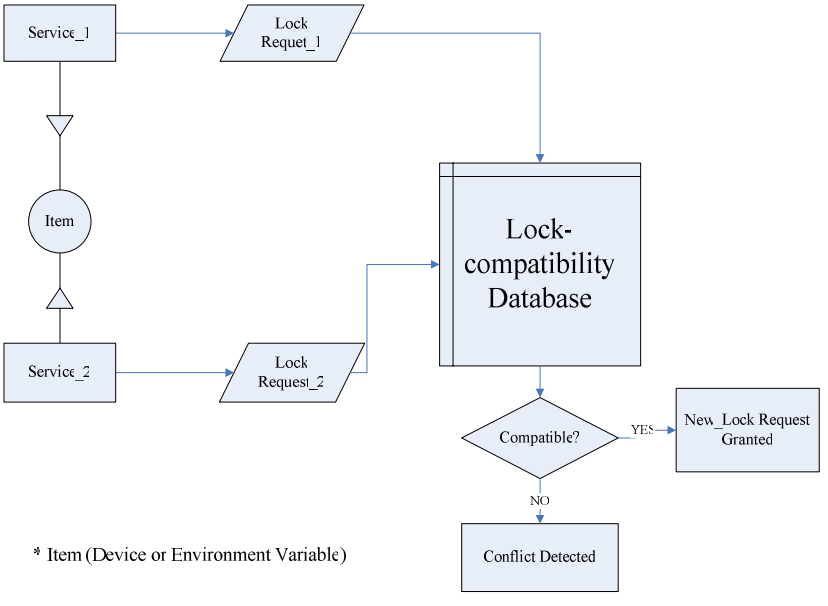

Fig. 1. Lock-based Conflict Detection Mechanism

D-PRI (Dynamic Priority) Conflicts Method. When conflicts happen, the conflict control manager decides which lock request can be granted (which service can access the devices and/or change environment variables). We suggest a D-PRI (Dynamic Priority) conflicts solution where each service is assigned a D-PRI. When conflicts happen between two services, the conflict control manager checks each service's D-PRI and allows the one of them which has the higher D-PRI to use the device and/or change environment variables. The D-PRI is decided in three ways as follows:

- Comparative way: the conflict control manager assigns each service a D-PRI by comparing its importance to the user. For example, security services have higher D-PRI than entertainment services.

- Interactive way: when conflicts between services A and B are detected, the conflict control manager sends conflict information to a user who is interested in using the services. If the user prefers service A rather than B, service A has a higher D-PRI than service B.
- User-specific way: the user-specific way uses the statistical result of the interactive way to determine user specific D-PRI. The conflict control manager examines the past records of how conflicts are resolved by individual user, and, then, statistically determines the D-PRI of each service in terms of user' preference.

It should be noted that human is considered a critical component in establishing the dynamic priorities of services. This can contribute to reflect human's preferences and behavioral history in designing a ubiquitous environment.

4 Conflict Detection and Resolution Mechanism on the WS-ECA Platform

WS-ECA is an ECA (Event-Condition-Action)-based platform for the ubiquitous services system to support human interactions among service-oriented devices in ubiquitous computing network. In this system, conflicts can happen when rules specified with WS-ECA framework are triggered by some internal or external events that are generated by devices within the environment. We incorporate the conflict detection and resolution mechanism into WS-ECA platform for improving the human interaction with devices that provide effective services to humans in the system.

4.1 WS-ECA Platform Architecture

WS-ECA platform is composed of (1) service register, (2) event simulator, (3) device simulator, (4) variables monitor, and (5) ubiquitous service management system as shown in Figure 2.

1. Service Register: services for the ubiquitous service system are described in the WS-ECA language, and the service register is used to register the service using XML-formatted files. For example, an Event is that a car is started. A Condition is whether temperature is over 32°C in summer. An Action is to turn on its air conditioner.
2. Event Simulator: the event simulator simulates the event in the ubiquitous service system by generating event messages. The event messages are sent to ubiquitous service management system to activate services.
3. Device Simulator: the device simulator activates and/or deactivates devices in the ubiquitous service system. Device's states change according to the use of the corresponding services.
4. Variable Monitor: in the ubiquitous service system, the information of environment variables such as temperature, humidity, noise level, etc. are obtained though various types of sensors. We use the variable monitor instead of sensors to collect the information of environment variables.
5. Ubiquitous Service Management System: the ubiquitous service management system is the core of the platform. It allocates and controls all the services running in the system and detects and solves conflicts.

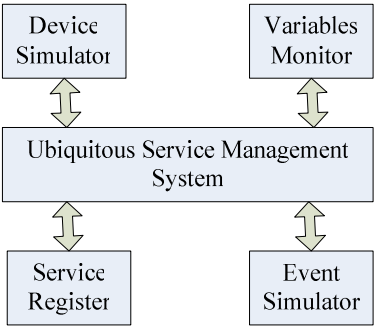

Fig. 2. WS-ECA Platform Architecture

4.2 Conflict Control Manager Architecture

The conflict control manager architecture on the WS-ECA platform is divided into four parts: (1) resource and environment variable status database, (2) lock compatibility database, (3) D-PRI database, and (4) service controller.

1. Resource and Environment Variable Status Database: the resource and environment variable status database stores the lock status of resource and environment variables.
2. Lock Compatibility Database: the lock compatibility database stores the compatibilities between lock requests.
3. D-PRI Database: the D-PRI database assigns and stores dynamic priorities which are the base of conflict resolution to different services.
4. Service Controller: the service controller grants the lock request from different services and schedules the services based on the information from the resource and environment variable status database, the lock compatibility database, and the D-PRI database.

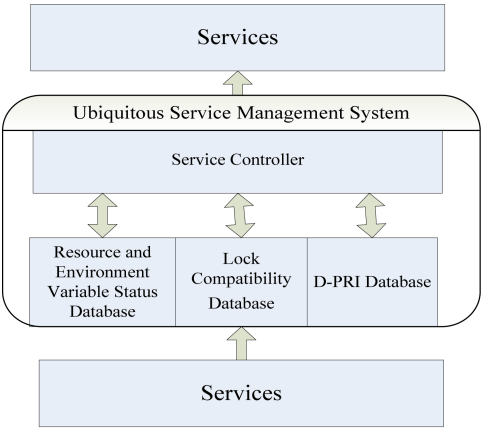

Fig. 3. Conflict-Control Manager Architecture

4.3 Implementation

Using the proposed conflict detection and resolution solution on the WS-ECA platform, we have developed a ubiquitous home services system. In this test bed, three types of services such as security service, entertainment service, and sleeping service are considered for the ubiquitous service system. Figure 4 shows the registration of ECA rule for a device on the WS-ECA platform.

Fig. 4. ECA rule registration step

Each time a service requests an access to a device or changes its environment variable, the service sends a lock request to the conflict control manager, which is responsible for the lock registration and checking system for the runtime conflict detection. The conflict control manager can accept the lock request for an item (a service or an environment variable) if the item is currently free or there is no lock held on the item. The conflict control manager uses the lock compatibility database in which the compatibility of services is registered in terms of items, when an item can acquire multiple services at the same time.

When conflicts happen, the Conflict Control Manager should decide which Lock Request should be granted (which service can access the resources and environment variables). In this case, the D-PRI (Dynamic Priority) conflicts resolution method is applied. Each service is assigned a DPRI and when conflicts happen between two services, Conflict-Control Manager checks each service's DPRI, and allow the one who has the higher DPRI to use the resource and/or to change environment variables.

Consider the following situation.

- Service A (winter-season-rule): when the season is winter, turn on the heater
- Service B (new-season-rule): when temperature>27℃, turn off the heater

In winter season, service A is activated. Service A sends “heater-on lock request” to CCM. The heater is free at this time, CCM, thus, grants the “heater-on lock request” from Service A. When temperature becomes higher than 27℃,Service B is activated. Service B sends “heater-off lock request” to CCM. CCM checks the Lock

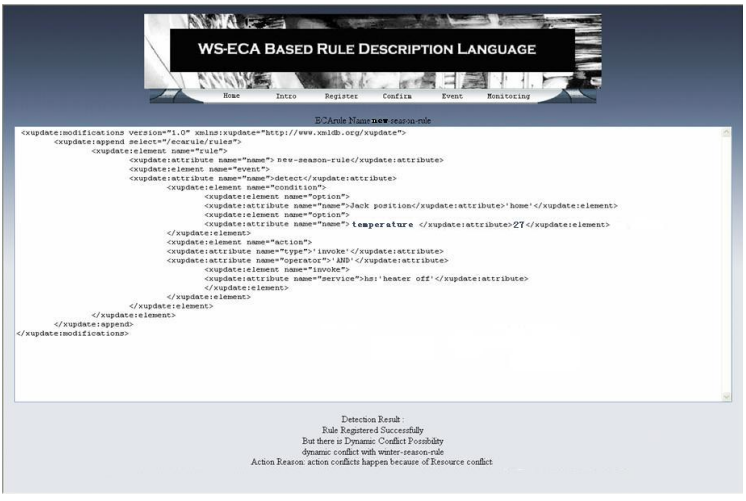

Fig. 5. Conflict detection at the registration stage

Fig. 6. Conflict detection at runtime

Compatibility Database and finds that “heater-on lock request” is incompatible with “heater-off lock request” (because “heater on” and “heater off” can not be executed at the same time). Conflict is detected. At this time, if Service B’s DPRI is lower than Service A, then Service B is aborted. Otherwise, Service A is suspended and Service B is executed.

Figure 5 show both device conflict and environment variable conflict detected in the phase of Service Registration. With Conflict Control Manager implemented, runtime conflicts which occur frequently in Ubiquitous Services System can be successfully detected and solved. By employing the lock-based detection method in the services registering phase, difficulty caused by potential conflicts between difference services can also be avoided in the early stage of the rule registration phase.

Figure 6 shows conflict detection and suggestions for resolving the conflicts that are detected at runtime of the ubiquitous service system.

5 Conclusion and Future Work

This paper presents the design and implementation of Conflict Control Manager for Ubiquitous Services System for improving the human interaction with computerized devices. The Conflict Control Manager implementation makes it possible to detect and resolve the conflicts in ubiquitous environment and can assist in providing higher quality of services to humans involved in the system.

Our future work includes the establishment of a more realistic environment for evaluation of the effectiveness of human-computer interaction. By conducting a variety of experiments, human preference on services and its impact on designing and building the human-centered ubiquitous environment can be evaluated.

Acknowledgments. This work was supported by the research fund of Hanyang University (HY-2006-N).

References

1. Christofferson, K., Woods, D.D.: How to make automated systems team players. In: E. Salas (Ed.), Advances in Human Performance and Cognitive Engineering Research, vol. 2. JAI Press, Elsevier (2002)
2. Graefe, V., Bischoff, R.: A human interface for an intelligent mobile robot. In Proc. Robot and Human Communication, 6th IEEE International Workshop on (RO_MAN ’97), pp. 194–199 (1997)
3. McGuire, P., Fritsch, J., Steil, J.J., Rothling, F., Fink, G.A., Wachsmuth, S., Sagerer, G., Ritter, H.: Multi-modal human-machine communication for instructing robot grasping tasks. In: Proc. Intelligent Robots and System, 2002. IEEE/RSJ International Conference on, vol.2, pp. 1082–1088 (2002)
4. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems: Situation awareness. Human Factors 37(1), 32–64 (1995)
5. Lupu, E.C., Sloman, M.S.: Conflicts in Policy-Based Distributed Systems Management. IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management (1999)