International Journal of Innovative
Computing, Information and Control
Volume **7**, Number **5(B)**, May **2011**

ICIC International ©2011 ISSN 1349-4198
pp. **2987–3005**

A SMART-GATE BASED COMPOSITION METHOD TO PROVIDE SERVICES BY SOLVING CONFLICT USING DYNAMIC USER PRIORITY AND COMPROMISE POLICY[1]

Junbo Wang[1], Zixue Cheng[2], Lei Jing[2], Kaoru Ota[1] and Mizuo Kansen[3]

[1]Graduate School of Computer Science and Engineering
[2]School of Computer Science and Engineering
[3]University-Business Innovation Center
University of Aizu
Aizu-Wakamatsu, Fukushima 965-8580, Japan
{ d8101202; z-cheng; leijing; d8102104; Kansen }@u-aizu.ac.jp

Received February 2010; revised August 2010

Abstract. *Context-aware service is a hot topic in the research filed of ubiquitous computing. Composition of smart objects/context provides a flexible way to compose context-aware services, which means related smart objects around a user automatically being integrated together to provide the appropriate services. However, (1) there is no effective mechanism to coordinate the work of multiple smart objects and solve conflict problem in composition of smart objects; (2) priority is an effective method to solve conflict problem, however, the current priority based policies are static and thus not flexible; (3) when a conflict occurs for a numeric action of a smart object, e.g., changing of volume of a TV, it is not reasonable to just follow all/nothing policy, i.e., satisfy one user's requirement and ignore another one's, however, to find a compromised value to satisfy both users as much as possible. To solve the above problems, in this paper, a smart-gate based composition method is proposed. The smart gate has uniform interface to communicate with multiple smart objects and modules to coordinate the work of multiple smart objects and resolve conflict of services. Moreover, two novel policies are proposed in the smart gate for solving the conflict problem. One is a dynamic user priority based policy. The other is a compromise policy to minimize the required tolerance of users. Finally, we evaluate the proposed method through an experiment. The experiment result shows that the composition method works well and effective for resolving conflict problem.*
Keywords: Context-aware services, Service conflict, Composition/coordination of smart objects, Smart gate, Conflict resolution, Dynamic user priority policy, Compromise policy

1. **Introduction.** The phrase "ubiquitous computing" has been widely used in the research fields after it appeared [1]. One of the hot topics in ubiquitous computing is context-aware service/intelligent system [2-5], which provides users personalized services, by considering users' location, time, available devices, favorites, etc. For example, a mobile phone may provide information of restaurants automatically, which are close to the user's current location and matched to the user's favorites around the time for lunch, say 12:00 AM. Another ubiquitous context-aware system proposed in [4], can get person actions such as standing up/down, walking and running by embedding acceleration sensors in a mobile phone, and provides services adaptive to the user's context.

The rapid progress of smart objects augmented by the abilities of sensing, computing, and wireless communicating makes the development of context-aware possible and

[1]A preliminary version of this work was presented in the 1st International Workshop on Aware Computing (IWAC09), Aizu, Japan, 2009

convenient. For example, the Mediacup proposed in [6] can record the movement and temperature of a cup and the Pin&Play noticeboard proposed in [7] can automatically alert the users before a deadline Effective and efficient coordination of smart objects [8-10] is becoming key problem when providing services, since a service is often provided by integrating various ubiquitous devices [11]. Composition [12-15] of *smart objects/context* provides a flexible way to consider communication/coordination between smart objects, which means related smart objects around a user automatically being integrated together, to provide appropriate services with deep consideration of status, activities, properties of the users, etc. A formal ontology-based conceptual model was proposed to represent composeable smart objects in [14,15]. Based on that, each composeable smart object can be abstracted as a formal virtual entity, which has plugs to communicate with others to provide services. However, it is not flexible when there are multiple smart objects to compose a service, since there is no effective mechanism to coordinate the work of the multiple smart objects and solve the conflict problem in composition of smart objects.

Conflict is a key problem in context-aware services. Without conflict resolutions, it becomes difficult to deploy context-aware services in a smart space, since the system will get confused and cannot tolerate while there are conflict requirements of different users on the same device.

One approach to solve conflict problem is based on mediation, which lets users discuss and select a service based on the discussion result. O'Hara et al. proposed *Jukola* in [16] to allow customers democratically choose music in a public place. It solves conflict problem through voting for music by customers and then selecting the music with the highest vote rate. This approach is effective in reflecting the dynamic requirements of users with the variation of time and place, even for a temporary requirement. However, it also brings a big burden and stress to users who have to actively participate in solving conflict.

Another approach [17-21], is to automatically resolve conflict or partially combining with mediation, with support of various parameters preset by developers/users. Priority is an effective method to solve conflict problem, since the priority can be easily assigned based on preferences and profiles of users. There are various priority based policies in conflict resolution. Haya et al. proposed a mechanism to compute priority based on various parameters, e.g., who needs the services environment factors and etc in [17]. Some kinds of priority based conflict resolution policies were proposed in [18], e.g., preemptive role based priority, preemptive priority, non-preemptive priority and time-slice based priority proposed. However, there is a problem that they are static and thus not flexible. Therefore, when there is a user with a higher priority, another user with a lower priority is very hard to use services, even waiting for a long time, since the mechanisms to assign the priority does not take factor of times of user's occupying/acquiring the service into consideration.

Moreover, when a conflict occurs for a numeric action of a smart object, e.g., changing volume of a TV or temperature or brightness in a room, it is not reasonable to just follow one user's requirement and ignore another one's, however, to find a compromised value to satisfy both users as much as possible. Park et al. proposed a conflict resolution in [20], to find an appropriate value to satisfy the users. A cost function was designed to reflect the difference between users' intensions and resolutions. Then a resolution is determined to minimize the above cost function of all users involved in conflicts. However, the method did not consider tolerable degree of the users when computing the appropriate value. The tolerable degree of the users reflects how much the user can tolerate a resolution, which plays an important role in solving the conflict problem; since it can help the system find an appropriate resolution to encourage more users accept the resolution. Without

considering the tolerable degree of the users, the system cannot provide suitable services to effectively reduce the unhappiness of the users.

To this end, a smartgate based composition method is proposed to coordinate the work of multiple smart objects and solve conflict problem. First, a virtual entity is designed to represent individuals including smart objects, users and environment. It has ports to connect to the smart gates. Then, a smart gate is proposed, which has interface ports to connect and communicate with V-individuals, and two modules to compose/coordinate the work of V-individuals and solve conflict problem, respectively. Moreover, two novel conflict policies are proposed in the smart gate for solving the conflict problem. One is a dynamic user priority based policy where the priority of each user is dynamically changed with the times of user's occupying/acquiring the service. The other is a compromise policy, which finds an appropriate value to minimize the unhappiness of the users. Finally, an experiment is performed by implementing the method in an indoor ubiquitous environment and investigating the satisfaction of users. The experiment result shows the composition method works well and the conflict policies are effective in context-aware systems.

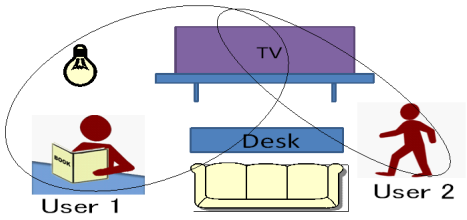
By using the method, multiple smart objects can be automatically integrated together to provide services with a uniform composition mechanism. And the conflict resolution will be more flexible and reasonable by considering times of user's occupying/acquiring service and finding a compromised resolution to satisfy both users with consideration of lowest required tolerance. The method can be used as a building block for development of various context-aware systems, e.g., in a danger-aware system or an intelligent home, the ubiquitous devices can be integrated together to detect the situations and provide services to users. The proposed conflict policies also can be implemented in various applications, e.g., in an indoor ubiquitous environment or in a public context-aware system, where the conflict resolution is necessary for conflict requirements in ubiquitous devices, e.g., controlling TV, light, radio, air-condition, information board, etc.

The rest of the paper is organized as follows. In Section 2, an example and a composition model abstracted from the example are shown. The detail design of the composition method is presented in Section 3. The implementation and evaluation are discussed in Section 4. Finally, the paper is concluded in Section 5.

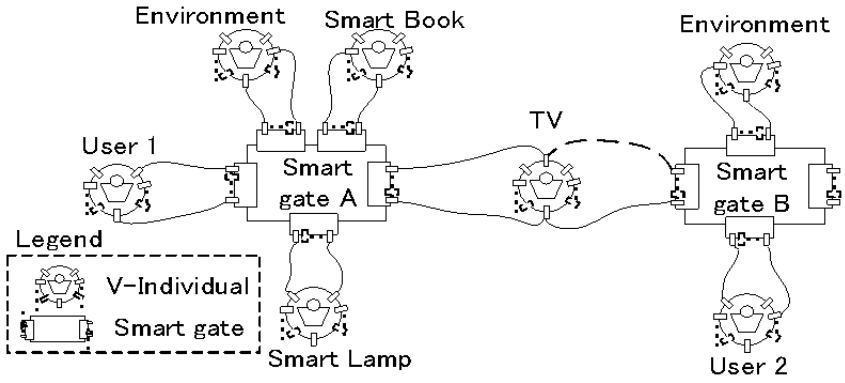
2. **Model.** Figure 1(a) is an example showing the composition of smart objects/contexts. We assume that (1) all the objects (textbook, lamp, TV, etc.) in the example are smart objects (2) the ID of each user and every object can be recognized by sensors, and (3) an smart object can sense its status. For example, the textbook can sense status of the book (e.g., opening/closing), and who opens it.

There are two users in one room as shown in Figure 1(a). User 1 is reading a textbook on the desk. When he opens the textbook, the activity of user 1 is recognized as *learning*, and a service will be provided to him, e.g., the lamp will be switched on and the TV will be set in forbidden mode of use automatically. To effectively provide the service to the user, coordination of these smart objects is very important. Without an effective coordination mechanism, the system may get confused when detecting situations and providing services based on various smart objects. Moreover there may be other requirements for the service from the user 1, such as setting the appropriate temperature of air-condition for reading, playing music based on the content of textbook, and so on, which are not considered in the example for simplicity.

Meanwhile, user 2 comes in, and sits down on the sofa. Then, a context-aware service should be provided to him, e.g., turning on the TV and selecting the channel based on the user's preferences. Then the conflict based on the different requirements of two users



(a) An example of two users using home appliances



(b) Composition model using smart gate

FIGURE 1. An example and a composition model using smart gate

happens, i.e., TV should be set as forbidden mode of use due to the user 1's context, and should be turned on due to the user 2's context.

Figure 1(b) shows a composition model abstracted from the example in Figure 1(a), by providing services based on the smart gates. The two basic concepts used in this model are shown as follows.

- **V-individual:** V-individual (virtual individual) is a formal and abstract representation for a smart object or a user, which is augmented by various sensors and other computing, communication components, so that it has the abilities of sensing, processing, communicating, etc. A V-individual consists of ID, properties and ports, which will be discussed in Subsection 3.1 in detail.
- **Smart gate:** Smart gate is a virtual connecting device which can be used for enabling multiple V-individuals to compose together and triggering services. It mainly consists of ports for connecting smart objects, a coordination module and a conflict resolution module, which will be discussed in Section 3 in detail.

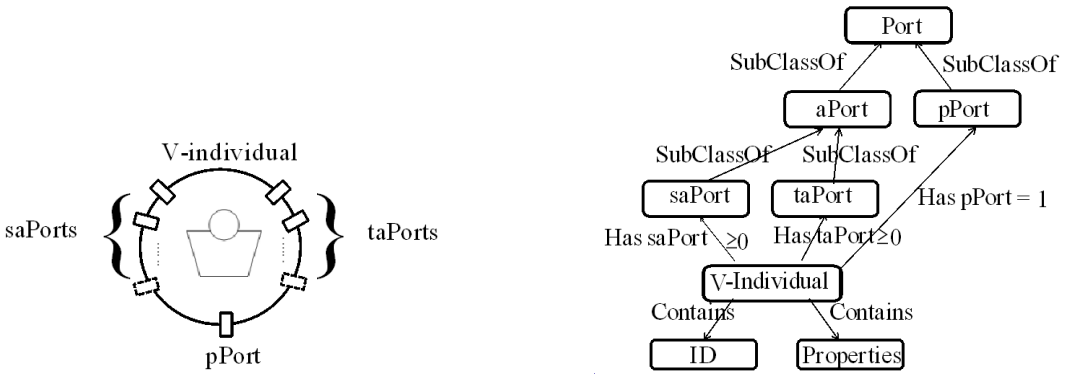
In Figure 1(b), there are five V-individuals (i.e., textbook, user 1, TV, lamp and user 2) and two gates (i.e., smart gate A and smart gate B). User 1, textbook, lamp, TV and environment compose together through the smart gate A. User 2, TV and environment compose together through the smart gate B. The environment includes the information such as brightness of a room, temperature, etc.

3. The Smartgate based Composition Method.

In this section, we present the detail of the method.

3.1. The structure of V-individual and smart gate.

Coordination among V-individuals is very important, since a service is often provided by composition of various smart objects. For coordinating the work of smart objects, first a uniform communication interface of smart object should be designed to exchange information. There are two



(a) The structure (b) Ontology based representation

FIGURE 2. The structure of V-individual

kinds of information for providing services, which are actions and properties in this study. For the actions, we can further classify them into two categories, which are sensed actions and triggered actions. Sensed actions are used to detect situations around the user and triggered actions are used to provide services by activating smart objects. With these considerations, we define the structure of a V-individual as shown in Figure 2(a).

First, it includes two kinds of ports to exchange information, i.e., *aPort* and *pPort*. *aPort* means action port and *pPort* means property port to exchange the action and property information respectively. *aPort* is further classified into *saPort* (sensed action port) and *taPort* (triggered action port), to exchange sensed actions and triggered actions respectively. For the same example in Section 1, when user 1 opens the book, the lamp will be switched on. The action of that *the book is being opened* is sent through the *saPort* of the smart book, since it is detected by sensors And the action of *switching on the lamp* is received through *taPort* of the smart lamp, since it is triggered for the service. *pPort* is used for receiving the properties information from the V-individual, e.g., the age of a person. Besides the ports, a V-individual also has ID and properties. We use ontology to represent the V-individual as shown in Figure 2(b), since it is easily to express and share knowledge [22].

When there are multiple smart objects to provide service, a mechanism is necessary to coordinate the work of smart objects, since the system may get confused when detecting contexts and providing services in composition of smart objects. Meanwhile, conflict is a key problem when providing context-aware services when there are multi-users. Therefore, a uniform mechanism, i.e., smart gate is proposed to coordinate the work of multiple smart objects and solve conflict of services.

Figure 3 shows the structure of the smart gate. It mainly includes *ports* and *processing unit*. *Ports* are used to connect with V-individuals. Processing unit consists of coordination module and conflict resolution module. The coordination module and the conflict resolution module will be presented in detail, in Subsections 3.2 and 3.3 respectively. We see the environment as a special individual with properties of physical parameters such as temperature of a room.

3.2. **Coordination module.** The coordination module coordinates the work of V-individuals, and decides what services should be provided. In this paper, we employ an ECA rule based approach to design this module. It can also be implemented by other methods, e.g., rule-based/case-based reasoning. ECA rule consists of three parts, i.e., *event*, *condition* and *action*. We describe *event* by using the signals from *saPort* of the V-individual, and describe *condition* by using the information from the *saPort* and *pPort* and describe

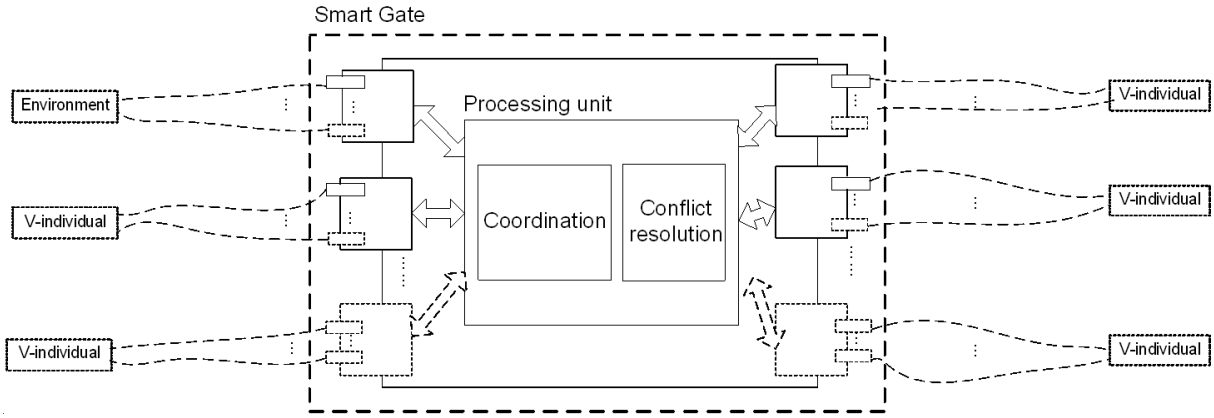


FIGURE 3. The structure of the smart gate

TABLE 1. ECA rule component form

Ports		ECA rule component		
		Event	Condition	Action
aPort of smart object	saPort	◯	◯	×
	taPort	×	×	◯
pPort of smart object		×	◯	×
aPort of enviroment	saPort	◯	◯	×
	taPort	×	×	◯
pPort of enviroment		×	◯	×
aPort of person	saPort	◯	◯	×
	taPort	×	×	×
pPort of person		×	◯	×

action by using the information from *taPort* of smart objects and environment. Table 1 shows the relation of *event, condition* and *action* with ports of V-individuals.

We use the following two examples of rules to show the coordination of the smart objects for providing services. The example **rule 1** is used for providing services when someone is learning. The meaning of **rule 1** is that the lamp is switched on, and TV mode is changed to forbidden use, when the smart book is being opened under the condition that light is not sufficient. *Textbook is being opened* is a sensed action of textbook detected by sensors, and will be sent to the smart gate through *saPort* of the textbook. After that, the smart gate gets the environment information through *saPort* of environment to judge whether the light is sufficient or not. Finally, the smart gate provides services by sending the action of *turning on the lamp* and *setting TV mode to forbidden use* to the lamp and the TV respectively, through the corresponding *taPorts*. The example **rule 2** is used for providing TV watching service to the user when the user is in leisure time.

Example rule 1:
Event: Textbook is being opened // It is a sensed action of smart book
Condition: Light is not sufficient // It is a sensed action of environment
Action: Switching on the lamp, and setting TV mode as forbidden use
//They are triggered actions of smart lamp and smart TV respectively.

Example rule 2:
Event: Someone is sitting on the sofa // It is a sensed action of smart sofa
Condition: Leisure time // It is a sensed action of environment

Action: Turning on the TV // It is a triggered action of smart TV

3.3. **Conflict resolution module.** As presented in the Sections 1 and 2, conflicting requirements for services often occur, and should be resolved when providing services to users. In this subsection, we will discuss how to resolve conflict problem by the conflict resolution module. First, we define conflicting services and show detection of conflict services in Subsection 3.3.1. Then, we present an outline and two conflict resolutions in Subsection 3.3.2.

3.3.1. *Conflict detection.* In this subsection, we will present how to detect conflicting services in the conflict resolution module.

Definition 3.1. Action.
An action is an augmented ability of a smart object to sense the situation around the object or to be triggered for providing services. It is denoted by a as follows,

$$a = \langle name, op \rangle,$$

where name and op mean name and operation of the action. Two kinds of actions are considered in this paper, i.e., sensed action and triggered action as discussed in Subsection 3.1. We use sa to denote a sensed action and use ta to denote a triggered action. Furthermore, two types of actions are considered. One is called ***enumerated action*** *which is a discrete action, e.g., turning on/off TV. The other is called* ***numeric action*** *which is to adjust a continuous variable to be a certain value, e.g., adjusting the volume of TV.*

Definition 3.2. V-individual.
V-individual is a formal and abstract representation for a smart object or a user as discussed in Section 2, which is denoted by v-i as follows,

$$v\text{-}i = \langle name, SA, TA, P \rangle$$

where,

- *name is the name of v-i*
- $SA = \{sa_1, sa_2, \ldots, sa_i, \ldots, sa_m\}$ *is a set including the sensed actions of v-i detected by sensors.*
- $TA = \{ta_1, ta_2, \ldots, ta_j, \ldots, ta_n\}$ *is a set including the triggered actions of v-i for providing services.*
- P *is a set including the properties of v-i.*

sa_i *and* ta_j *represent sensed action and triggered action respectively, and they have the same structure as the action in Definition 3.1.*

Definition 3.3. Service *(i.e., context-aware service). A service is to automatically trigger various ubiquitous devices to meet the requirement of users based on detected situations/contexts around the user. Here, a service is denoted as s as follows,*

$$s = \langle type, O, u, e \rangle$$

where,

- *type is the type of the service,*
- $O = \{o_1, o_2, \ldots, o_k, \ldots, o_w\}$ *is a set including all the smart objects to provide the services,*
- *uis the user whom the service s is provided for,*
- *e is the environment.*

o_k *and e have the same structure as the V-individual in Definition 3.2.*

Definition 3.4. Conflict of actions. *For each pair of actions, a_i and a_j, we say they are conflicting if and only if the names of the actions are same and the operations are different, shown as follows,*

Conflict_A(a_i, a_j),

iff

- $name_i = name_j$ *and*
- $op_i \neq op_j$

/** *It means that, if and only if* $name_i = name_j$ *and* $op_i \neq op_j$*, the two action* a_i *and* a_j *are recognized as conflicting actions.* **/

Function 1: Function to detect conflicting services. In this paper, we consider two kinds of conflict situations. The first is conflict for the same smart object with different requests of users. For example, the system needs turn up the volume of a TV for a user, however, be requested to turn down the volume of the TV for another user. The other is conflict for the environment due to different smart objects. For example, a user needs higher temperature and another user needs lower temperature. These requests may be implemented by the different devices, e.g., increasing the temperature by an air condition and decreasing the temperature by an electric fan.

We use function *Conflict_S*(s_i, s_j) to detect two conflicting services s_i and s_j. For simplified description, we suppose the two services are in the same time and same place.

Conflict_S(s_i, s_j)

iff

- $u_i \neq u_j$ and
- $O_i \bigcap O_j \neq \varphi$ and
- $\exists o_1 \in O_i, \exists o_2 \in O_j, \exists ta_x \in o_1.TA, \exists ta_y \in o_2.TA, ((o_1.name = o_2.name) \wedge Conflict\_A(ta_x, ta_y))$

/** $o_1.TA$ and $o_2.TA$ represent the TA set in the o_1 and o_2 respectively. $o_1.name$ and $o_2.name$ represent the name of o_1 and o_2 respectively. **/

/** It means the users in the two services are different and there is at least a common smart object in the two services. Meanwhile, there are at least two conflicting triggered actions, i.e., ta_x and ta_y, in the common same object. **/

or

- $u_i \neq u_j$ and
- $\exists ta_a \in e_i.TA, \exists ta_b \in e_j.TA, Conflict\_A(ta_a, ta_b)$

/** e_i and e_i are requirements for environment in the service s_i and s_j respectively. **/

/** $e_i.TA$ and $e_j.TA$ represent the sets of TA in the e_i and e_j respectively. **/

/** It means the users in the two services are different. Meanwhile, there are at least two conflicting triggered actions, i.e., ta_x and ta_y, in the environment e_i and $e_j$of the two services. **/

3.3.2. *Conflict resolution.* Priority is an effective method to solve conflict problem. In this paper, we mainly employ two kinds of priorities. They are service priority and user priority. Service priority is assigned based on the importance of the service, which is represented by an integer from 1 to 3, where 1 means low level priority, 2 means middle level priority, and 3 means high level priority. For example, the service for calling helpers when someone is in an emergency should have the service priority 3, the highest priority. However, we assume there is no conflicting requirement for the same device in some special services, e.g., two services for emergency or two care services for patients. Otherwise, there should be multiple devices to provide the service for each user at the same time.

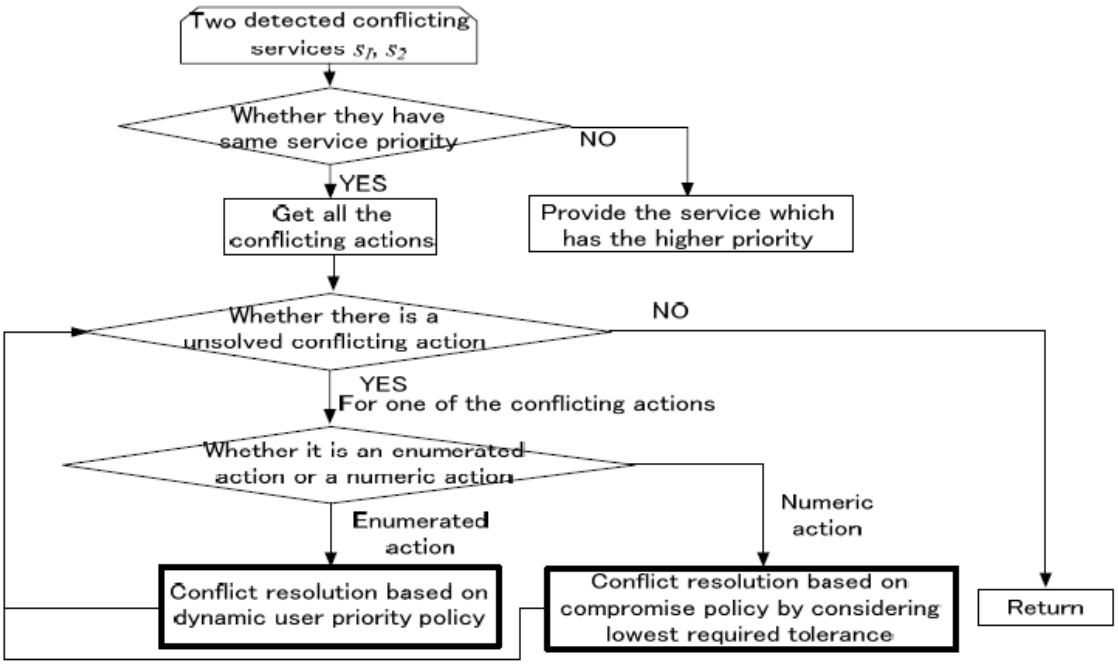


FIGURE 4. Outline of conflict resolution

(1) Outline of conflict resolution. The outline of conflict resolution is shown in Figure 4. First, the system judges whether the two conflicting services have the same service priority. If they have the same service priority, the system further considers the conflicting actions. Otherwise, the system provides the service which has higher service priority. For each conflicting action, first the system judges whether it is an enumerated action or numeric action based on Definition 3.1. Then for enumerated actions, the system solves conflict problem based on dynamic user priority policy; otherwise, the system solves it by using compromise policy considering lowest required tolerance.

(2) User priority. For enumerated actions, the system solves conflict based on user priority, since user priority can be easily assigned based on user's preferences and is effective in conflict resolution. The system decides which action should be executed based on the user priority. If the user priority is the same, a mediate mechanism will be used to decide the service.

Definition 3.5. ***User Priority.*** *We define a user priority as an integer from 1 to 10 to represent the priority of a user to occupy a service. 10 is the highest and 1 is the lowest priority for using service. It is denoted as* $p(u_i, s_j)$*, i.e., the priority of user* i *using service* j*, and can be represented by one of the following two types.*

- Static priority. The static priority is to represent the priority which cannot be adjusted by usage times. For example, learning services, working services or elderly people assisting services in this paper.
- Dynamic priority. The dynamic priority is mainly for describing the services which should be considered with the number of usage times. For example, we can use dynamic priority to represent amusement activities, such as watching TV, listening music or play games, since users normally share them based on the number of usage times. The key idea for dynamic priority is to reduce a user's priority, with increasing of times of the user's occupying the service, which means, the more times of using s_j, the lower priority of using s_j.

Function 2: Dynamic priority function. $f_p : D_t \to D_p$ is the dynamic priority function for the user$i$ and service j, where D_t is the domain of $t(u_i, s_j)$ and D_p is the domain of $p(u_i, s_j)$, such that $f_p(x)$ is monotone decreasing function. $t(u_i, s_j)$ is used to represent how many times the user u_i has already used service s_j in the specified period.

- $0 < f_p(x) \leq 10$.
- $f_p(x_1) f_p(x_2) iff x_1 < x_2$.

The dynamic priority function can be any functions satisfying the above condition. For example, it can be

$$p(u_i, s_j) = a \times t(u_i, s_j) + p(u_i, s_j) \quad (\text{where } a < 0) \tag{1}$$

$$p(u_i, s_j) = \left((t(u_i, s_j) + b)^{-a} + c\right) \quad (\text{where } a > 1 \text{ and } b \neq 0) \tag{2}$$

or

$$p(u_i, s_j) = (-(t(u_i, s_j) + b)^a + c) \quad (\text{where } a > 1 \text{ and } b \leq 0) \tag{3}$$

$p(u_i, s_j) = 1$ if $p(u_i, s_j)$ is less than 1 and $p(u_i, s_j) = 10$ if $p(u_i, s_j)$ is larger than 10 in the above three functions. We use $p_0(u_i, s_j)$ to represent the initial priority of user i using service j, and the dynamic priority will be reset within a period of time. In functions (2) and (3), $p_0(u_i, s_j)$ should equal to $b^{-a} + c$ and $-b^a + c$, respectively.

Function (1) can be used in the simple situation, e.g., the reducing speed of $p(u_i, s_j)$is fixed, and functions (2) and (3) can be used when considering more complex situation, e.g., the reducing speed of $p(u_i, s_j)$ also varies with the changing $t(u_i, s_j)$

(3) Compromise policy based on lowest required tolerance. For numeric actions, we solve conflict based on compromise policy considering lowest required tolerance, since it is not reasonable to just follow one user's requirement and ignore another one's for these actions.

First, users/developers assign a desired value and a tolerable degree for a service based on Definition 3.6. The tolerable degree is to represent how much the user can tolerate at most when the value changes around the desired one. Then we introduce the required tolerance variable for each user by using Definition 3.7 with an assumption that the value of action is set as r. The required tolerance variable is used to represent how much the user needs to tolerate if the system sets the value as r. We further calculate the total required tolerance of two users by using Definition 3.8. Finally, we minimize the total tolerance of two users to get a compromised value r_{app} by using **Function 3**.

Definition 3.6. Tolerable degree. *Tolerable degree is to represent how much a user can tolerate for an action in a service, and is denoted by $\mu_0(u_i, a_k, s_c)$, i.e., tolerable degree of the user u_i for the action $a_k$in the service s_c. It is a decimal fraction from 0 to 1. It can be assigned directly, e.g., $\mu_0(u_i, a_k, s_c) = 0.2$. Furthermore, it also can be assigned based on an extreme value the user can tolerate. For example, it can be assigned as follows, $\mu_0(u_i, a_k, s_c) = \frac{|r_{ie} - r_0(u_i)|}{\varepsilon}$ where r_{ie} represents an extreme value the user u_i can tolerate. $r_0(u_i)$ represents the predefined value by the developers/users and ε represents the maximum range of the numeric action a_k.*

Definition 3.7. Required tolerance variable. *We use a required tolerance variable to represent how much the system requires the user tolerate the service when the value of the action is set to a specified one. It is denoted as $\mu(u_i, a_k, s_c) = \left(\frac{|r - r_0(u_i)|}{\varepsilon} - \mu_0(u_i, a_k, s_c)\right)^2$, i.e., the required tolerance variable of the user u_i for action a_k in the service s_c when the value of numeric action is set as r. Similarly, for user u_j, $\mu(u_j, a_l, s_d) = \left(\frac{|r - r_0(u_j)|}{\varepsilon} - \mu_0(u_j, a_l, s_d)\right)^2$. $r_0(u_i)$ and $r_0(u_j)$ represents the desired value by developers/users and ε represents the maximum range of the numeric action a_k and a_l.*

TABLE 2. Smart objects

Smart objects		**Actions transferred through *aPort* of smart objects**
Smart chair	saPort	Who is sitting on it
	taPort	
Smart lamp	saPort	The lamp is being switched on/off
	taPort	Switching on/off the lamp
Smart TV	saPort	The TV is being turned on/off
	taPort	Turning on/ off the TV, selecting program, turning up/down the volume of the TV
Smart desk	saPort	What is putted on the desk
	taPort	

Definition 3.8. Total required tolerance variable. *The total required tolerance variable is a weighted summation of the required tolerance variables of all the users in the conflicting services. For simplifying discussion, we assume there are only two users involved in the conflict services in this paper and use T to denote the total required tolerance variable of two users, i.e., u_i and u_j.*

$$T = p\left(u_i, s_c\right)\left(\frac{|r - r_0(u_i)|}{\varepsilon} - \mu_0\left(u_i, a_k, s_c\right)\right)^2 + p\left(u_j, s_d\right)\left(\frac{|r - r_0(u_j)|}{\varepsilon} - \mu_0\left(u_j, a_l, s_d\right)\right)^2$$

where $p(u_i, s_c)$ and $p(u_j, s_d)$ are user priorities of user u_i and u_j for service s_c and s_d respectively. a_k and a_l are conflicting actions in service s_c and s_d respectively.

Function 3: The function to calculate a compromised value based on lowest required tolerance.

The function is to calculate a compromised value based on lowest required tolerance. We support r_{app} is a suitable compromised value of an action. Generally, r_{app} should be between the predefined values of the two users, i.e., $r_0(u_i)$ and $r_0(u_j)$ to satisfy both users. Here, we support $r_0(u_i) < r_{app} < r_0(u_j)$. Then, the formula in Definition 3.8 should be

$$T = p\left(u_i, s_c\right)\left(\frac{r - r_0(u_i)}{\varepsilon} - \mu_0\left(u_i, a_k, s_c\right)\right)^2 + p\left(u_j, s_d\right)\left(\frac{r_0(u_j) - r}{\varepsilon} - \mu_0\left(u_j, a_l, s_d\right)\right)^2$$

Then

$$\frac{\partial T}{\partial r} = 2\frac{p\left(u_i, s_c\right)}{\varepsilon}\left(\frac{r - r_0(u_i)}{\varepsilon} - \mu_0\left(u_i, a_k, s_c\right)\right) - 2\frac{p\left(u_j, s_d\right)}{\varepsilon}\left(\frac{r_0(u_j) - r}{\varepsilon} - \mu_0\left(u_j, a_l, s_d\right)\right)$$

To minimize the tolerance of two users, we take $\frac{\partial T}{\partial r} = 0$ and get r_{app},

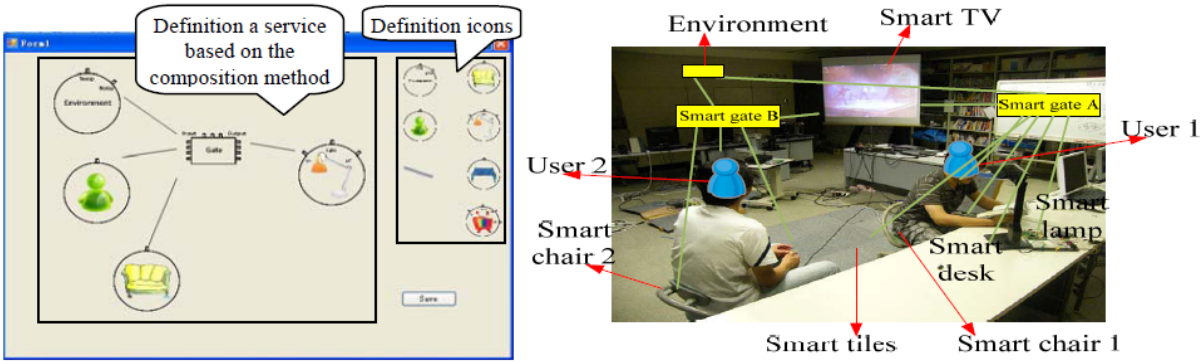
$$r_{app} = \frac{\varepsilon \times (\mu_{0i} p_i - \mu_{0j} p_j) + p_i r_{0i} + p_j r_{0j}}{p_i + p_j},$$

where for simplifying description, we use $p_i = p(u_i, s_c), p_j = p(u_j, s_d)$, $\mu_{0i} = \mu_0\left(u_i, a_k, s_c\right)$, $r_{0i} = r_0(u_i)$, $r_{0j} = r_0(u_j)$ and $\mu_{0j} = \mu_0\left(u_j, a_l, s_d\right)$.

4. **Evaluation.** In this section, we will evaluate the proposed method in the following aspects, (1) feasibility and effectiveness of the proposed method through an investigation of satisfactory degree of users, (2) the effectiveness by a questionnaire after the experiment.

(a) U-tile (b) Back of each tile (c) Smart TV (d) Smart lamp

FIGURE 5. Smart objects



(a) Definition interface (b) Implementation of smart gates

FIGURE 6. Implementation of the method

4.1. **Experiment environment.** In order to evaluate the proposed method, we implement a system based on the composition method in an indoor environment consisting of various smart objects. Table 2 shows the smart objects and their augmented abilities.

Figure 5(a) shows an implementation of u-tiles sensor network which has been presented in [23,24]. The outline of u-tiles sensor network is as follows. We divide the real floor into various pieces/tiles, and attach pressure sensors and RF-ID antenna on the back of each tile as shown in Figure 5(b). Each tile has a unique tile ID and an RF-ID antenna, connected to an RF-ID reader through a switch controlled by a micro-computer. The IDs of Tiles are through tile 0 to tile N. By getting user/objects ID, tiles ID and searching the corresponding information in a DB, we can get the position/distance/relation of individuals.

For example, when a user sits on the chair, which is placed on a predefined tile, the smart TV will be turned on and play the related program corresponding with the user's preference as shown in Figure 5(c) by checking the ID of the user and his favorites in DB. These simulate TV and sofa in the living room as mentioned in the example in Section 2. Figure 5(d) shows the implementation of smart lamp, which is controlled by an embedded MCU.

Smart gates are stored in a server. In the current implementation, when the system starts up, smart gates are created and exist in the server statically to wait for the smart object' connection requests. In the future, we plan to realize dynamical connection of smart gates for the different situations. For example, when a user opens a book, a related smart gate will be created automatically to coordinate the work of smart objects related with reading the book.

Figure 6(a) shows the definition interface to compose a service based on the method. The smart gates and V-individuals can be drawn to the left part and connected together to compose a service. Figure 6(b) shows the implementation of the smart gates. User 1, smart desk, tile 11, and smart chair compose together through the smart gate A, and the

context is recognized as User 1 is learning. Then the smart gate A switches on the lamp, and sets the TV in forbidden use mode based on rule 1 defined in Section 3. User 2, tile 15, and smart chair 2 compose together through smart gate B. Then the user context is recognized as amusement and smart gate B tries to turn on the TV.

4.2. **Experiment design.** To evaluate a conflict resolution, firstly, it should be compared with the scenario without conflict resolution. From the comparison, the necessity and the effectiveness of the conflict resolution in our daily life can be analyzed. Meanwhile, the conflict resolution should be compared with the existing ones to show the effectiveness of the proposed methods for solving conflict problem.

Satisfaction of the users is a very efficient method to evaluate the effectiveness of a conflict resolution [21,25,26]. However, there is no experiment to effectively evaluate the dynamic user priority based policy and the compromise policy. Meanwhile, in the previous experiments, they ignore analyzing subjects' satisfaction by considering the parameter of position/role in the conflicting services. The position/role of a subject is to classify the users based on their conflicting requirements, which is a very important parameter to evaluate a conflict resolution. For example, if there are two persons who have conflicting requirements when a conflict resolution increases the satisfaction of one person significantly, generally it also decrease the satisfaction of the other person in the contrary position/role, since the person in the contrary position/role cannot get the service well. Therefore, when evaluating the effectiveness of a conflict resolution, the satisfaction of the users in the both positions/roles need be analyzed. Otherwise, the analysis is not clear and completed enough to show the effectiveness of the conflict resolution, e.g., when the method increasing the satisfaction of the users in one position/role, it may decrease the satisfaction in the contrary position.

With these considerations, the experiment is designed as follows:

(1) Two scenarios are designed, one is without conflict resolution and the other is applied with the conflict resolutions, to show the effectiveness of the conflict resolution.
(2) The proposed conflict resolution is further compared with some previous ones by investigating satisfaction of users, i.e., the proposed dynamic user priority based policy is compared with static user priority based policy, and the proposed compromise policy is compared with all/nothing policy.
(3) Position/role is considered to evaluate the method more clearly and comprehensively, when analyzing the effectiveness of the proposed conflict resolution.

For the common electronic appliances, the actions/operations mainly can be classified into two types, i.e., enumerated action/operation and numeric action/operation. An enumerated action is a discrete action, e.g., turning on/off. A numeric action is to adjust a continuous variable to be a certain value, e.g., turning up/down the volume or raise/decrease the temperature. Therefore, generally the conflict occurs in the following two typical situations

The first typical situation TS1 is "There are some subjects who have different requirements for services, which are conflicting in an emulated action of a smart object".

The second typical situation TS2 is "There are some subjects who have different requirements for services, which are conflicting in a numeric action of a smart object".

In order to evaluate the proposed conflict resolutions, we evaluate the proposed method in the above two typical situations as shown in Table 3.

All/nothing policy means the system totally satisfies one user's requirement and ignore another's. All/nothing policy based on user priority is applied in typical situation TS1, since conflicting action is an enumerated action in TS1, which is hard to find a compromised resolution to solve the conflict. And the conflict resolution based on compromise

Table 3. Experiment design

Typical Situation	Action type	Conflict resolution
TS1	Enumerated action	Conflict resolution based on all/nothing policy by using user priority
TS2	Numeric action	Conflict resolution based on compromise policy considering lowest required tolerance

policy considering lowest required tolerance is applied in typical situation TS2, since the conflicting action in typical situation TS2 is numeric action, for which it is possible to find a compromised resolution. To evaluate the effectiveness of the dynamic user priority based method, it is compared with a static user priority based method in TS1.

Finally, a questionnaire is answered by the subjects to evaluate the method in the aspects of the necessity and effectiveness of the proposed method, the suitability of the selected methods in conflict resolution from the users' point of view.

4.3. **Experiment setup and process.** In the experiment, 12 participants are recruited for evaluating the method. And working service, book reading service and TV watching service, are implemented and applied in the two typical situations. We set the service priorities of TV watching, working and book reading as the same level. Moreover, the user priority for watching TV is set as dynamic, and the user priorities for book reading and working service are set as static. We use function (1) in definition of **Function 2** to compute dynamic user priority.

The typical situation 1 is implemented as *"a subject wants to work and the other subject wants to watch TV"*. The conflicting action is turning on/off the TV. The typical situation 2 is implemented as *"a subject want to read a book and the other subject want to watch TV"*. The conflicting action is setting the volume of TV.

The 12 participants are divided into pairs, each of which includes two subjects. One participant of a pair was assigned with a higher user priority and the other one was assigned with a lower user priority. Then, we asked them experience the volume of smart TV freely, and selected an appropriate volume and set a tolerable degree when they were watching TV or reading book, respectively.

In the scenario without conflict resolutions, we closed conflict resolution mechanism when providing services to let participants experience and recall the scenes including conflict in their daily life. In scenario with conflict resolution, the system provided service by solving conflict automatically based on the proposed method. In both scenarios, for each pair, one subject was asked to watch TV, and the other was asked to get a main idea of a book by reading it within some minutes and then finish a computer program including designing and coding. After that, they were told to change the position, i.e., changing the services.

4.4. **Experiment result.** In the scenario with conflict resolution, after the system provided each service with conflict resolution, we asked the 12 participants to answer their degree of satisfaction as shown in the following two tables. For the participants who gave an answer of dissatisfaction, we let them write down their feelings, comments and suggestions freely to improve our method.

(1) Effectiveness of dynamic user priority in TS1. Firstly, let us consider the conflict resolutions for TS1 which is employed with all/nothing policy by using user priority. Two cases are considered in TS1. Static user priority and dynamic user priority are applied to the two cases respectively as shown in Table 4. The users in the position

TABLE 4. Experiment result 1 to evaluate the dynamic user priority

Conflict resolution for enumerated action in TS1		**The position/role of the subjects**		**Satisfactory degree of subjects**
All/nothing policy by using user priority	Case C1 Static user priority	(a)	Turn off the TV	① 42% ② 8% ③ 50%
		(b)	Turn on the TV	① 41% ② 17% ③ 42%
	Case C2 Dynamic user priority	(a)	Turn off the TV	① 0% ② 8% ③ 92%
		(b)	Turn on the TV	① 45% ② 17% ③ 38%
※ ① Dissatisfy ② Neutral ③ Satisfy				

(a) require turning off the TV and the users in the contrary position, i.e., position (b), requires turning on the TV.

From Table 4, we can see satisfaction of the subjects increased and dissatisfaction decreased clearly in C2(a) compared with C1(a). Moreover, for the subjects who are in the contrary position, there is no significant variation by comparing C2(b) with C1(b).

In case C1, the system solves the conflict problem based on static user priority. Among the 12 subjects, half of them originally have higher priority and half of them have lower priority. Therefore, in C1, half of them can get their expected services and half of them cannot. However, in the C2, when the dynamic user priority is applied in the system, the subjects who have lower priority also have chances to get their services. Therefore, the satisfaction increased and dissatisfaction correspondingly decreased in C2(a). Meanwhile, for the subjects who are in the contrary position, there is no significant changing by comparing C2(b) with C1(b). It means dynamic user priority does not bring big influence for the subjects in the contrary position. In other words, even though the system provides the service to the subjects with lower priority; the subjects in the contrary position who have higher priority do not show significant dissatisfaction. In the experiment, we performed a short interview after the subjects used the system. Some subjects reported that they were satisfied with the resolution, since they can work well even with a lower user priority. Some subjects who wanted to watch TV reported that they can accept the resolution of turning off TV, since they have watched TV many times. However, there were also some subjects who were dissatisfied with the resolution. One subject reported that he can accept a lower volume but cannot accept turning off the TV even though he had a lower priority or had watched TV many times. One subject reported the system should at least show the screen to him, even though without sound. In the future, we will enhance the resolution to further improve the satisfaction of users.

From the above discussion, we can see that the dynamic user priority is effective for resolving conflict problem in typical situation TS1 in the aspects of significantly increasing the satisfaction of users in position (a) and also not decreasing the satisfaction of the users in the contrary position (b).

(2) Effectiveness of compromise policy based on lowest required tolerance in TS2. To evaluate the effectiveness of compromise policy based on lowest required tolerance, two cases C3 and C4 are considered for typical situation TS2 as shown in Table 5. All/nothing policy by using static user priority is applied to C3 and compromise policy considering lowest required tolerance is applied to C4. The users in the position (a) need low volume and the users in the position (b) need high volume. In case C3, the system totally satisfies one user's requirement and ignore another's, i.e., turning on/off the TV. In C4, the system finds a compromised resolution to satisfy both users.

TABLE 5. Experiment result 2 to evaluate the compromise policy based on lowest required tolerance

Conflict resolution for numeric action in TS2	**The position/role of the subject**		**Satisfactory degree of subjects**
Case C3 All/nothing policy by using static user priority	(a)	Low volume	① 42% ② 8% ③ 50%
	(b)	High volume	① 41% ② 17% ③ 42%
Case C4 Compromise policy considering lowest required tolerance	(a)	Low volume	① 50% ② 17% ③ 33%
	(b)	High volume	① 17% ② 25% ③ 58%
※ ① Dissatisfy ② Neutral ③ Satisfy			

From Table 5, we can see that the satisfaction increased and the dissatisfaction decreased in C4(b) by compared with C3(b). Meanwhile, there is no significant changing in C4(a) compared with C3(a).

From the experiment result, we can see that, the compromise policy based on lowest required tolerance can increase satisfaction of the subjects in the position (b). It is because the system can consider the users in both positions to set the volume to a compromised one, rather than just turn off the TV for the subjects in position (a). Therefore, the satisfaction of subjects in the position (b) increases clearly. Meanwhile, for the subjects in the position (a), it does not bring big influence. It means for the subjects who want to work/read a book, they also can accept the conflict resolution by finding a compromised volume instead of completely turning off the TV. In the interview after the experiment, some subjects reported that they enjoyed the conflict resolution since they can use their service, i.e., watching TV even though with a small volume. Some subjects reported that it is very good to consider the tolerance of them for finding a compromised value. Some subjects reported that the volume does not affect them so much when they are reading/working. However, there are also some subjects who are dissatisfied with the resolution. Some subjects reported the volume is not so low enough that they cannot concentrate on reading. Some subjects reported the volume is a little low for watching TV. In the future, we will consider more factors to enhance the proposed method for improving the satisfaction of users. For example, adjusting the volume based on the content of the book and TV to enhance the satisfaction of users.

From the above discussion, we can see that the compromise policy based on lowest required tolerance is effective for resolving conflict problem in typical situation TS2 in the aspects of increasing the satisfaction of the users in position (b) significantly, and also does not decrease the satisfaction of the users in the contrary position too much.

(3) Effectiveness of the method from the questionnaire. Just using parameter of satisfaction of the users is not enough to evaluate the method, since it cannot fully reflect the opinions of users, e.g., the necessity and effectiveness of the proposed method, and the suitability of the selected methods in conflict resolution from the users' point of view. Therefore, after the experiment, we asked the subjects to answer a questionnaire for evaluating the method based on the above aspects. The experiment result is shown in the following tables.

To evaluate a conflict resolution, firstly, it should be compared with the scenario without conflict resolution. From the comparison, the necessity and the effectiveness of the conflict resolution in our daily life can be analyzed. Therefore, two scenarios without and with

Table 6. The experiment result for Q.1 and Q.2

Question	Answer
Q. 1. In scenario without conflict resolution, do you think you were disturbed by TV when you were working?	① 8% ② 0 ③ 92%
Q. 2. Compared with the scenario without conflict resolution, do you think you are more comfortable in the scenario with conflict resolution?	① 0 ② 25% ③ 75%
⁎ ① Disagree and partially disagree ② Neutral ③ Agree and partially agree	

Table 7. The experiment result for Q.3 to Q.5

Question	Answer
Q. 3. Do you think the priority can be used to decide who can use devices?	① 17% ② 0 ③ 83%
Q. 4. Do you think the user priority should be decreased by considering number of usages times when watching TV?	① 17% ② 8% ③ 75%
Q. 5. Do you think the value, i.e., volume of TV, was suitable for you?	① 25% ② 17% ③ 58%
⁎ ① Disagree and partially disagree ② Neutral ③ Agree and partially agree	

conflict resolutions have been set up in the experiment. In the questionnaire, two questions are asked focusing on comparison of the two scenarios to evaluate whether the conflict resolution is necessary and effectiveness in the real life as shown in Table 6.

From Table 6, we can see that, the conflict resolution is necessary and effective in our daily life since most of subjects were disturbed by TV in the scenarios without conflict resolution, and most of subjects thought they were more comfortable in the scenario with conflict resolution. In the short interview after they answered questions, some subjects reported that they could not concentrate on the work due to the sound of TV. Some subjects reported they could not finish the program in a noisy environment. Most of them reported they felt comfortable in the scenario with conflict resolutions. However, there were also some subjects who could work/read well in such an environment. However, they also reported that it will be better if the system can automatically solve conflict problem.

In the proposed method, the conflict resolution is based on user priority and a compromise policy considering lowest required tolerance of the users. To evaluate the proposed conflict resolutions from the users' point of view, Q.3, Q.4 and Q.5 are answered by the subjects. For the subjects to well understand and answer the questions, we have simply explained the main idea and flowchart of the conflict resolution to them as follows. Firstly, the system judges who can occupy the service based on the service priority, which is based on the importance of the service. Then, the system judges the user priority when the service priorities are the same. And in some services, the user priority can be decreased with the times of users' occupying/acquiring services, e.g., the user priority for watching TV. After the explanation, Q.3, Q.4 and Q.5 are answered by the subjects.

From Table 7, we can see, most of subjects agree that priority can be used to resolve conflict problem, and user priority also can be decreased with the times of occupying the service. Meanwhile, most of users think the value calculated based on compromise policy are suitable for them. In the short interview after they answered questions, some subjects reported priority especially service priority, is very important in conflict resolution. And

most of the subjects reported that generally they think the volume was suitable. However, some subjects reported that user priority should not be assigned based on age, e.g., elder has a higher priority. And some subjects also reported that the volume of TV when they were working is not so suitable that they could not concentrate on working/reading. In the future, we will enhance the conflict solutions to satisfy more users. Moreover, user priority should be assigned based on different applications for different families.

As a summary, from the experiment we can see that, (1) the smartgate based composition method work well when providing context-aware services; (2) the conflict resolution is effective to resolve the conflict problem in the typical situation TS1 and TS2, in the aspects of increasing the satisfaction of the users in one position/role and not decreasing the satisfaction of the user in the contrary position/role; (3) the method is necessary and effective from the user's point of view. However, for improving satisfaction of the users in the both positions/roles, a mediate mechanism is necessary to let the users discuss shortly to select a policy in conflict resolution. In future, we will perform the research to design this kind of mediate mechanism to satisfy the users in the both positions/roles at the same time.

5. Conclusion.

In this paper, a smartgate based composition method was proposed to coordinate the work of multiple smart objects and solve the conflict problem in composition of smart objects. First, the structures of V-individual and smart gate were designed to exchange the information and provide services. In the smart gate, there are two modules, i.e., coordination module and conflict resolution module. Then, we proposed two novel policies for solving the conflict problem. One is based on dynamic user priority and the other is compromise policy considering the lowest required tolerance of users. The experiment results show that the composition mechanism works well and the conflict resolutions are effective in increasing the satisfaction of the users in the context-aware systems through an investigation of satisfactory degree of users and a questionnaire after the experiment.

The method can be used as a building block to ease the development of context-aware computing systems, since it can let developers develop each of composeable units, e.g., a smart objects separately, and they compose together to provide services by solving conflict problem automatically. The proposed conflict resolution can be used in various applications, e.g., in a ubiquitous home/office or in a public context-aware systems. In the future, we will enhance the composition mechanism by dynamically generating smart gates and the conflict resolutions by increasing more parameters to satisfy the users.

Acknowledgment. The authors would like to say thanks to all subjects who attended our experiment and gave us useful comments and suggestions. The authors also gratefully acknowledge the helpful comments and suggestions of the anonymous reviewers.

REFERENCES

[1] M. Weiser, The computer for the 21st century, *American Science*, pp.94-104, 1991.

[2] B. N. Schilit, N. Adams and R. Want, Context-aware computing applications, *Proc. of the Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, pp.85-90, 1994.

[3] A. K. Dey, *Providing Architectural Support for Building Context-Aware Application*, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, 2000.

[4] Y. Kawahara, H. Kurasawa and H. Morikawa, Recognizing user context using mobile handsets with acceleration sensor, *Proc. of IEEE International Conference on Portable Information Devices*, Orlando, FL, 2007.

[5] O. Castillo and P. Melin, Intelligent systems with interval type-2 fuzzy logic, *International Journal of Innovative Computing, Information and Control*, vol.4, no.4, pp.771-784, 2008.

[6] M. Beigl, H. Gellersen and A. Schmidt, MediaCups: Experience with design and use of computer-augmented everyday objects, *Computer Networks*, vol.35, no.4, pp.401-409, 2001.
[7] K. V. Laerhoven, A. Schmidt and H. Gellersen, Pin&Play: Networking objects through pins, *Proc. of Ubicomp*, Gothenburg, pp.219-229, 2002.
[8] J. Beutel, O. Kasten, F. Mattern, K. Römer, F. Siegemund and L. Thiele, Prototyping wireless sensor network applications with BTnodes, *Proc. of the 1st European Workshop on Wireless Sensor Networks*, Berlin, Germany, pp.323-338, 2004.
[9] M. Beigl and H. Gellersen, Smart-Its: An embedded platform for smart objects, *Proc. of Smart Objects Conference*, 2003.
[10] C. Decker, A. Krohn, M. Beigl and T. Zimmer, The particle computer system, *Proc. of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2005.
[11] K.-L. Su, S.-V. Shiau and C.-C. Wang, Module based security system applying in intelligent home, *ICIC Express Letters*, vol.3, no.4(A), pp.1167-1172, 2009.
[12] T. Buchholz, M. Krause, C. Linnhoff-Popien and M. Schiffers, CoCo: Dynamic composition of context information, *Proc. of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, Boston, MA, 2004.
[13] Y. Koumoto, H. Nonaka and T. Yanagida, A proposal of context-aware service composition method based on analytic hierarchy process, *Proc. of the 1st KES International Symposium on Intelligent Decision Technologies IDT'09*, Himeji, Japan, 2009.
[14] A. Kameas, S. Bellis, I. Mavrommati, K. Delaney, M. Colley and A. Pounds-Cornish, An architecture that treats everyday objects as communicating tangible components, *Proc. of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, Fort Worth, Texas, 2003
[15] E. Christopoulou, C. Goumopoulos, I. Zaharakis and A. Kameas, An ontology-based conceptual model for composing context-aware applications, *Proc. of the 6th International Conference on Ubiquitous Computing*, 2004.
[16] K. O. Hara, M. Lipson, M. Jansen, A. Unger, H. Jeffries and P. Macer, Jukola: Democratic music choice in a public space, *Proc. of DIS 2004*, pp.145-154, 2004.
[17] P. A. Haya, G. Montoro, A. Esquivel, M. García-Herranz and X. Alaman, A mechanism for solving conflicts in ambient intelligent environments, *Journal of Universal Computer Science*, vol.12, no.3, pp.284-296, 2006.
[18] G. S. Thyagaraju, S. M. Joshi, U. P. Kulkarni, S. K. NarasimhaMurthy and A. R. Yardi, Conflict resolution in multiuser context-aware environments, *Proc. of CIMCA 2008, IAWTIC 2008 and ISE 2008*, pp.332-338, 2008
[19] C. Shin and W. Woo, Conflict resolution method using context history for context-aware applications, *Proc. of Pervasive 2005 Workshop*, Munich, Germany, pp.105-110, 2005.
[20] I. Park, K. Lee, D. Lee, S. J. Hyun and H. Y. Yoon, A dynamic context conflict resolution scheme for group-aware ubiquitous computing environments, *Proc. of ubiPCMM*, pp.42-47, 2005.
[21] C. Shin, A. K. Dey and W. Woo, Mixed-initiative conflict resolution for context-aware applications, *Proc. of UbiComp*, Korea, pp.262-271, 2008.
[22] H. Park, Y. Lee, B. Noh and H. Lee, Ontology-based generic event model for ubiquitous environment, *International Journal of Innovative Computing, Information and Control*, vol.5, no.11(B), pp.4317-4326, 2009.
[23] J. Wang, M. Kansen, Z. Cheng, T. Ichizawa and T. Ikeda, Designing an indoor situation-aware ubiquitous platform using rule-based reasoning method, *Proc. of 2008 International Computer Symposium*, Taiwan, 2008.
[24] T. Ichizawa, M. Tosa, M. Kansen, A. Yishiyama and Z.Cheng, The attribute and position based ubiquitous development environment using antennas with an automatically switch, *IPSJ SIG Technical Reports*, vol.2006, no.14, pp.109-114, 2006.
[25] Y. Wang, Y. Shen, M. Guo and D. Zhang, A context conflcit resolution with optimized mediation, *Prof. of the 24th IEEE International Conference on Advanced Information Newtorking and Applications Workshops*, Perth, Australia, pp.842-847, 2010.
[26] C. Shin, H. Yoon and W. Woo, User-centric conflcit management for media services using personal companions, *ETRI Journal*, vol.29, no.3, pp.311-321, 2007.