Journal of Universal Computer Science, vol. 15, no. 12 (2009), 2330-2352
submitted: 15/12/08, accepted: 25/6/09, appeared: 28/6/09 © J.UCS

Service Conflict Management Framework for Multi-user Inhabited Smart Home

Choonsung Shin
(GIST U-VR Lab., Gwangju, S.Korea
cshin@gist.ac.kr)

Woontack Woo
(GIST U-VR Lab., Gwangju, S.Korea
wwoo@gist.ac.kr)

Abstract: In this paper, we propose a service conflict management framework for detecting and resolving conflicts of multi-users who share context-aware applications within a smart home. For supporting a general solution to deal with the multi-user conflicts, the framework utilizes an ontology that describes applications and their services, an approach determination tree that assigns an appropriate resolution strategy to the conflict, and a set of resolution strategies. Based on this ontology, it dynamically detects conflicts associated among multiple users who are using various applications affecting each other, or the same application with different preferences. An appropriate resolution method is assigned to the conflict according to the properties involved, their relationship, and users' preferences. The detected conflict is resolved either by an automatic decision, based on either priority or preferences, or by a user decision. Through implementing and evaluating the framework to a smart home test-bed, we found that the proposed framework dynamically detected and flexibly resolved multi-user conflicts which occurred among the services of multiple applications, as well as within a single application.

Keywords: multi-user conflict, conflict management, intelligent space, ubiquitous computing
Categories: H.5.2, H.5.3

1 Introduction

Interest in context-aware applications for a smart home is rapidly growing, with a number of research activities focusing on intelligent spaces. The aim of the context-aware applications is to provide customized and personalized services by utilizing a variety of contexts obtained from users and their environments. In this aspect, a great deal of work has focused on describing contexts [Schilit et al. 1994c][Dey 2001b], developing frameworks for context-aware applications [Dey, Abowd 2000a] and inferring high-level contexts from a set of low-level contexts for delivering personalized services [Ranganathan et al. 2004b]. This research has contributed to developing context-aware applications for future intelligent environments. In addition to these activities, a more challenging problem needs to be considered when context-aware applications are deployed in an intelligent space occupied by multiple users. That is, when recognizing sporadic and different types of services for multiple users, intended services can interfere with other users' expectations of shared applications.

Thus, dealing with services to multi-users with different preferences and coordinating interrelated services have become crucial factors for developing real smart spaces.

Much research has dealt with multi-users in specific application domains and several research groups are currently trying to deal with the multi user conflict over media applications. However, in real smart homes, a variety of applications coexist and their services are frequently interrelated, including media applications and other applications controlling environmental conditions and required resources. Furthermore, the decision over the shared applications affects not only other applications, but also multiple users. Therefore, identifying and resolving the conflicts of multi-users are complicated and crucial problems which need to be considered.

For dealing with services for multiple users, we propose a service conflict management framework for multi-user inhabited smart homes. For incorporating various types of conflicts from a single application and among different applications, it first utilizes an ontology that describes applications and associated users' services in terms of service profile and user preferences. The framework is therefore able to detect and classify conflicts of users based on a change in shared attributes of the user's services, such as properties, shared resources and conditions. Second, an appropriate resolution strategy is dynamically assigned to each of the detected conflicts. In this determination phase, contexts such as priority, types of attributes and dependency are used. Finally, the selected resolution strategy resolves the conflict either by the automatic decision of our framework or by the users' decision. When the result of the automatic resolution is deterministic or similar to what all users want, the conflict is resolved automatically. Otherwise, conflict resolution requires users to participate in making an explicit decision over the conflict.

This work makes several important contributions. First, we introduce a service ontology to cover a variety of conflicts both between different applications and within an application. This ontology is extended from pervious well-know ontologies and can be effectively used in conflict management because it describes shared attributes of the user's services, such as properties, shared resources and conditions. Second, based on the ontology, we identify different types of conflicts according to their properties and relationship. Third, we have extended the approach determination tree [Shin et al. 2008a] to cover conflicts from multiple applications by adding criteria of dependency and interference of applications. The conflict among applications can also be resolved either by an automatic decision or through user participation. Finally, we evaluate the proposed framework in terms of its performance and user interaction in a smart home test-bed. Based on this evaluation, we can determine a variety of situations in which the proposed framework could be applied.

The rest of this paper is organized as follows. In section 2, we describe works related to conflict management. Service conflict and ontology are described in section 3. Section 4 describes how the proposed framework manages conflicts of multi-users which occur in multiple applications in terms of detection, approach determination and resolution. Details of implementation and evaluation are illustrated in section 5. We then conclude by outlining future research.

2 Related works

A multitude of work has been conducted to identify and resolve conflicts of multi-users in smart spaces such as smart homes or offices. One group of research is focused on independent applications. For example, MusicFX is an arbitrator for selecting music stations for multiple residents in a fitness center and automatically generates a sequence of preferred music stations for customers by merging the users' preferences of the music stations [McCathy, Anagnost 1998a]. Another example, MavHome, creates a context-aware resource management framework which automatically adjusts the level of services to resolve conflicts of resources, such as lights or air-conditioning, by finding an equilibrium value in each device based on the predicted users' locations [Das et al. 2006b]. However, these approaches deal only with an automatic resolution of conflicts within an application shared by multiple users. Although this mixed-initiative approach resolves conflicts of context-aware applications by exploiting the user's participation, it assumes that applications are independent of each other [Shin et al. 2008a]. Consequently, these approaches are insufficient in dealing with conflicts of interrelated applications within a smart space.

Another approach has dealt with multi-user conflicts by using multiple devices or applications. Reactive behavioural system (ReBA) is a context-aware application for an office environment which reacts to the devices in the office and resolves conflict by prioritizing them [Kulkarni 2002a]. The centralized queuing mechanism detects conflicts as an inconsistency of shared resources and resolves them by assigning priority based on the service profile [Haya et al. 2006b]. Thus, the highest priority in a queue is selected to use the shared resources when conflicts occur. However, these approaches always automatically select one of the applications based on priority, meaning that other users with lower priority are unable to use the applications or provide any input over making the decision. Even though this scheme automatically resolves conflict by dividing the resources to achieve effective utilization [Park et al. 2005a], such division is rarely applicable and cannot satisfy all users with different preferences.

Other works take into account the user's participation in conflict resolution over dependent applications. This user-centric conflict management approach manages conflict of multiple users accessing media services by making recommendations to the user for possible solutions [Shin et al. 2007b]. Similarly, the decisions in the group recommendation systems are based primarily on the users. Travel Decision Forum, for example, recommends and mediates group decision issues based on visual avatars for remote users. [Jameson 2004]. TV program recommendation strategy recommends choices for TV programs to a group of users by merging their profiles [Yu et al. 2006b]. GroupLens predicts users' interest in news articles by exploiting the ratings provided by other similar users [Konstan 1997]. These approaches encourage users to participate in conflict resolution, but it requires user intervention whenever conflicts occur in the applications.

Furthermore, other work has tried to resolve conflict by reasoning over ontology. In order to protect user privacy, the context obfuscation method controls the level of details by reasoning over ontology [Wishart 2005]. The location conflict resolution also resolves conflict from different contexts through ontological reasoning [Niu, Kay 2008]. Although this ontology-based reasoning is useful to obtain a generalized

solution, the real solution in multi-user conflicts still comes from negotiation between users.

In order to address the above-mentioned limitations, the proposed conflict management framework detects conflicts among services offered from different types of applications and resolves them by exploiting automatic decisions and users' participation. To cover various types of conflicts among different applications, as well as within an application, it utilizes an ontology which describes the services and associated users. It then takes advantage of an approach determination tree which controls the types of user participation based on the contextual information related to the detected conflict for making the best resolution. During conflict resolution, a conflict is thus automatically resolved when its resolution has a deterministic solution or could easily coincide with the other users. Mediation allows users to negotiate about the resolution over the conflicts when the decision has multiple and differing preferences, or is complex with multiple applications.

3 Service Conflict and Ontology

3.1 Service conflict

Before describing conflict management, it is important to clarify what conflict is in a multi-user inhabited smart home. Previous research has defined conflict as involving shared resources, conditions of an environment or shared properties of applications. Similarly, we adopt and extend such definitions into the area of services. Therefore, in this paper, we define service conflict as an inconsistency of services due to access of more than one user, each with different preferences. The service conflict is further classified into two types: 1) an internal conflict in shared properties of an application and 2) an external conflict in environmental conditions and shared resources. The external conflict can also include internal conflicts when multiple users access the shared attributes. The conflict properties refer to variables, such as the TV channel and volume or light level, and are subject to conflict within a single application. The resources are the physical elements involved, like a speaker and display in the case of a TV device. The conditions are the environmental statues within a smart space, such as temperature and illumination and, along with the resources, which are subject to conflict among applications. An external conflict, then, involves these attributes. Every service provided is directly related to a property and, depending on the service, may also involve the conditions and resources.

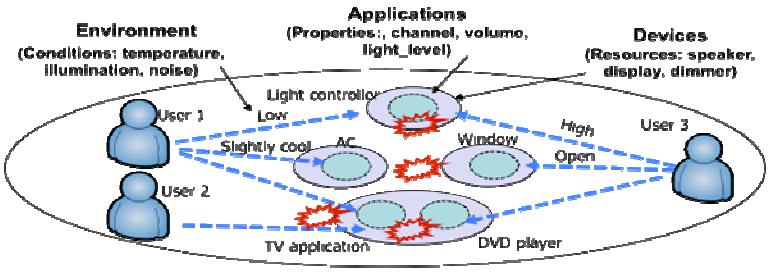


Figure 1: Service conflict scenario.

Figure 1 illustrates the conflicts occurring from three users accessing applications in a smart space. As can be seen, users' services conflict with others due to the shared attributes. In the case of the light controller, the preferred light levels are different for each user, creating a conflict. As users 1, 2 and 3 try to access the DVD player and TV applications, their services require the same resources and cause another conflict, even though each user employs different applications. The users also access applications deployed in different devices, in this case, trying to use the air conditioner (AC) and window to change the room temperature. However, the applications' effects are contradictory, resulting in yet another conflict. Consequently, none of the users are free to use their desired services due to the conflicts in the smart home.

3.2 Service ontology

In order to manage conflicts, we need to first describe services in a smart home. A great deal of research has been done to develop ontology to support interoperability and sharing contextual information in ubiquitous computing. Standard ontology for ubiquitous and pervasive applications (SOUPA) [Chen et al. 2004a], Context Ontology Language (CoOL) [Strang et al. 2003a], CONtext ONtology for modelling context in pervasive computing environments (CONON) [Wang et al. 2004a] and Web Ontology Language for Semantic web services (OWL-S) [W3C 2004] are well-established works for describing services for ubiquitous and pervasive computing. The Bayesian approach is designed for dealing with uncertain context in an intelligent space [Gu et al. 2004a]. The ontology-based generic context management (GCoM) model describes context by using a set of rules and their semantics [Ejigu et al. 2007a].

For this study, however, the aspects of ontology related to service conflict are of greater interest. Among them, parts of CONON and OWL-S are used since they are well formalized in describing users and services based on Web Ontology Language (OWL). OWL-S represents a service profile based on Input, Output, Precondition and Effect (IOPE). Similarly, we describe services based on their relative information. CONON well describes a user and his/her situation along with entities within the environment. In particular, the classes describing computing entities, such as services, applications and devices, and the status classes related to a room were extracted. We combined those classes by using OWL and extended them by adding a property class and its related classes. In our extension, we describe mainly the shared attributes, such as properties, resources and conditions. The properties describe the status of applications used by a service when it starts. The conditions are the environmental conditions of a smart home and are affected by the service. The resources belong to a particular device and are needed in order to execute the service. For simplifying service ontology, we assume that the conditions can be accurately measured in context-aware applications and do not change when affected by other elements.

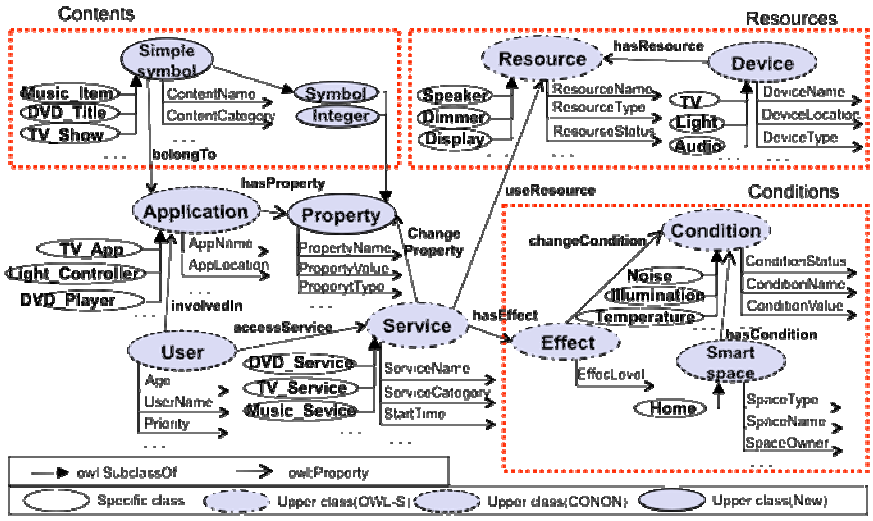


Figure 2: Ontology for managing service conflict.

Figure 2 outlines the ontology for managing service conflicts, which consists of upper classes, related properties and specific classes. Among the upper classes, the service class is the main class and is defined by its related properties, such as a property, a resource class, and an effect class. The property is associated with an application, but can also be extended to symbolically represent media content. The resource class is included in a device and each effect on it changes a certain condition of the smart space. In addition to the attribute classes related to a service, a user class, representing a user accessing the service, is associated with the service class. Based on the ontology, a user's service can be represented by details of its contents, resources and conditions. For a particular smart space, a set of subclasses, such as a TV service and DVD service, can be inherited from the upper classes and their instances classes are used for managing conflicts within the space.

SubClassof(TV_Application1, TV_Application).
hasProperty(TV_Application1, TV_Program1).
SubClassOf (TV_Service1, TV_Service).
accessService(User1, TV_Service1).
changeProperty(TV_Service1, TV_Program1).
PropertyValue (TV_Program, 'Prison_break').
useResource(TV_Service1, TV_Speaker1).
useResource(TV_Service1, TV_Display1).
hasEffect(TV_Service1, TV_Effect1).
changeCondition(TV_Effect, Noise1).
ConditionState(Noise1, 'HIGH').

Figure 3: An example of a user's service.

A sample description of a user accessing a TV service in a smart home is shown in Figure 3. This specific situation describes "User 1" accessing a TV service from a TV

application located in a TV device and he has selected his preferred program to watch. The property of the "TV application1" is "TV_program1" and "Prison_Break" is User 1's preferred item. In addition to his preference, the service also requires two resources and produces one effect. The "TV_Display1" and the "TV_Speaker1" are the shared resources belonging to the "TV_Device" and "Noise1" is the noise level of the room. To provide this TV service, there was no instance of the "TV_Program" property with different preferences, as the required resources are available and there is no service changing the condition. A variety of services can be similarly represented by this ontology and service conflicts of multi-users are managed by monitoring such properties, resources and conditions. The following section presents details of how conflict is detected and resolved based on the ontology.

4 Service Conflict Management Framework

The main goal of our conflict management framework is to detect and resolve conflicts of multi-users and is based on several assumptions. First, we assume that a user's services are not in conflict with themselves. This assumption is trivial since each user generally accesses a set of services simultaneously that are not in conflict. We also assume that user's priority can be assigned to each service rather than being applied to all the services the user accesses. This further assumes that each conflict is independent to other services if they are not directed related to property, resources or conditions. It is also assumed that the applications whose services affect the shared conditions are physically located in away from other applications. This assumption raises some possible situations where the services causing a conflict can be operated together when their effects are slightly decreased, such as an AC and a window controller installed in different locations within a smart space.

With these underlying assumptions in mind, we propose a general framework for managing multi-user conflict in a smart home. In order to manage various kinds of conflict occurring in context-aware applications within the home, our framework utilizes the ontology describing their properties, effects and required resources. Based on this, the framework is able to dynamically detect conflicts of multi-users who are using different applications or/and the same application with different preferences. The framework then resolves the conflict with an assigned resolution strategy. It tries to automatically resolve the conflict when the resolution is deterministic and close to user expectations. Otherwise, it recommends a set of choices for resolution that allow users to discuss and select their proper setting.

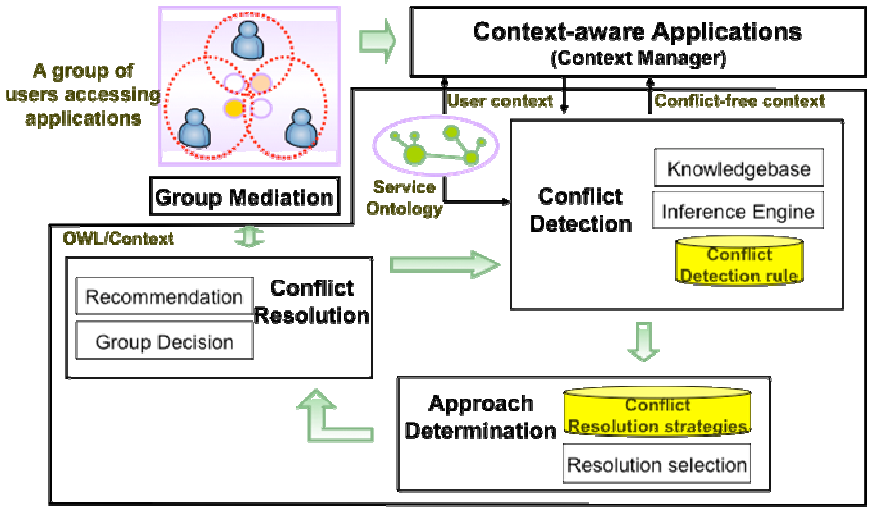


Figure 4: Service conflict management framework.

Figure 4 shows the overall architecture of the proposed conflict management framework. The proposed framework gathers contexts from context-aware applications and tries to detect conflicts with an inference engine that infers conflicts from the contexts based on a set of detection rules. It then determines an appropriate resolution approach to the detected conflicts. After determining a resolution approach, each of the conflicts is resolved either by an automatic resolution with a priority or group profile, or a mediated resolution according to the types of conflict and its properties. In the case of automatic resolution, the conflict is resolved by an assigned resolution mechanism and a conflict-free context is then delivered to the application directly. Otherwise, the conflict requires users to participate in the resolution and it tries to mediate a conflict resolution by recommending information to users and gathering their feedback. The conflict is resolved when feedback is provided for the given recommendation.

4.1 Conflict detection

In conflict detection, conflicts within an application and among applications are detected based on their context. For describing rules of conflict detection while preserving the OWL structure, we utilize Semantic Web Rule Language (SWRL), which is based on OWL and RuleML [W3C 2004]. Based on SWRL, the rules for detecting conflicts from properties of applications, resources of devices and conditions of a room are determined. In the case of a conflict within an application, the accessibility of shared properties among services is detected, including symbolic and integer properties. In both cases, conflict is detected when users have different values over the attributes. After detecting a conflict, the property is marked as "INTERNAL" with related services for further resolution. Figure 5 shows the detection rule for shared properties.

[Shared_Property:
SubClassOf(?S1, ?S) ∧ hasProperty(?S1, ?P1) ∧ SubClassOf(?P1, ?P)
∧ SubClassOf(?S2, ?S) ∧ hasProperty(?S2, ?P2) ∧ SubClassOf(?P2, ?P)
∧ PropertyValue(?P1, ?PV1) ∧ PropertyValue(?P2, ?PV2)
∧ differentFrom(?PV1, ?PV2)
⇒ hasConflict(?P, INTERNAL) ∧ conflictWith (?S1, ?P) ∧ conflictWith(?S2, ?P).]

Figure 5: Detection rule for shard properties.

In the case of the external conflict among multiple applications, the conditions and resources the services require are monitored along with collected user contexts. In the case of shared resources, those required by services are checked to determine that they are not simultaneously shared and needed for other services, as shown in Figure 6. Similar to the above-mentioned detections, the resource is marked as "EXTERNAL" and the services are related to the resources.

[Shared_Resource:
SubClassOf(?S1, ?AS1) ∧ SubClassOf(?S2, ?AS2) ∧ differentFrom (?AS1, ?AS2)
∧ useResoruce(?S1, ?R1) ∧ SubClassOf(?R1, ?R)
∧ useResoruce(?S2, ?R2) ∧ SubClassOf(?R2, ?R)
⇒ hasConflict(?R, EXTERNAL) ∧ conflictWith (?S1, ?R) ∧ conflictWith(?S2, ?R).]

Figure 6: Detection rule for shared resources.

In order to check for conflicts related to environmental conditions, the effects of the services are considered. A conflict is detected when users' services change the conditions with contrasting levels. As shown in Figure 7, the conflict is detected when the effects of the services are contradicting and when two or more mutual effects occur simultaneously. Upon detection, the condition is marked as "EXTERNAL" and the services are related to the condition.

[Shared _Condition:
SubClassOf(?S1, ?AS1) ∧ SubClassOf(?S2, ?AS2) ∧ differentFrom (?AS1, ?AS2)
∧ hasEffect(?S1, ?CC1) ∧ changeCondition(?CC1, ?C1)
∧ hasEffect(?S2, ?CC2) ∧ changeCondition(?CC2, ?C2)
∧ SubClassOf(?C1, ?C) ∧ SubClassOf(?C2, ?C))
∧ ConditionStatus(?C1, ?ST1), ConditionStatus(?C2, ?ST2), differentFrom(?ST1, ?ST2)
⇒ hasConflict(?C, EXTERNAL) ∧ conflictWith(?S1, ?C) ∧ conflictWith(?S2, ?C).]

Figure 7: Detection rule for shared conditions.

For an example, in the scenario in Figure 1, a conflict is detected from the light controller accessed by the two users. There is an external conflict between the AC and window due to the temperature. In addition to these conflicts, the conflicts in the TV and DVD are considered as external conflicts because they are a shared resource and property. Therefore, this external conflict consists of a resource conflict and a property conflict and is marked as an external conflict. After detecting conflicts with

these rules, information about each conflict is organized as a set of trees that represents the conflicts and their relationship to other conflicts as well as their applications and users. Figure 8 provides an example of conflict detection.

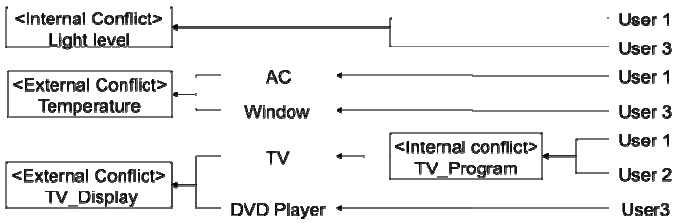


Figure 8: An example of conflict detection.

The root of each tree is a tag node indicating whether the conflict is internal or external and its child includes the related attributes. The root node of the internal conflict has child nodes for conflicting attributes followed by the users' profiles. The root node of the external conflict not only has conflicting services followed by users' profiles, but also has internal conflicts.

4.2 Approach determination

In the approach determination, an appropriate resolution approach among a set of resolution strategies available is assigned to each of the detected conflicts. Overall, the resolutions are divided into three types of resolution schemes according to the level of user participation: automatic resolution with priority, automatic resolution with users' profiles and mediated resolution. Each resolution is also related to the types of attributes. Automatic resolution with priority requires no user participation because the user already has control over the applications. In the case of automatic resolution with users' preferences, the users' preferences are used in obtaining resolution solutions. In both cases, the conflict is automatically resolved with the solution reflecting all the users' preferences, meaning that all users are implicitly involved in conflict resolution. However, users explicitly participate only in the mediation resolution. These types of conflicts are complex due to the existence of multiple solutions or interference with other attributes, forcing users to decide on one particular setting.

For deciding the appropriate resolution approach, an approach determination tree is used along with users' services to detect conflict. One fundamental principle is to resolve the conflict automatically as much as possible, minimizing the users need to participate in conflict resolution. The strategy used in single applications is similar to the Mixed-initiative approach [Shin et al. 2008a], but is extended by adding dependency among the applications. Therefore, various contexts described in users' services, such as priority, types of attributes and preferences, are used in the approach determination.

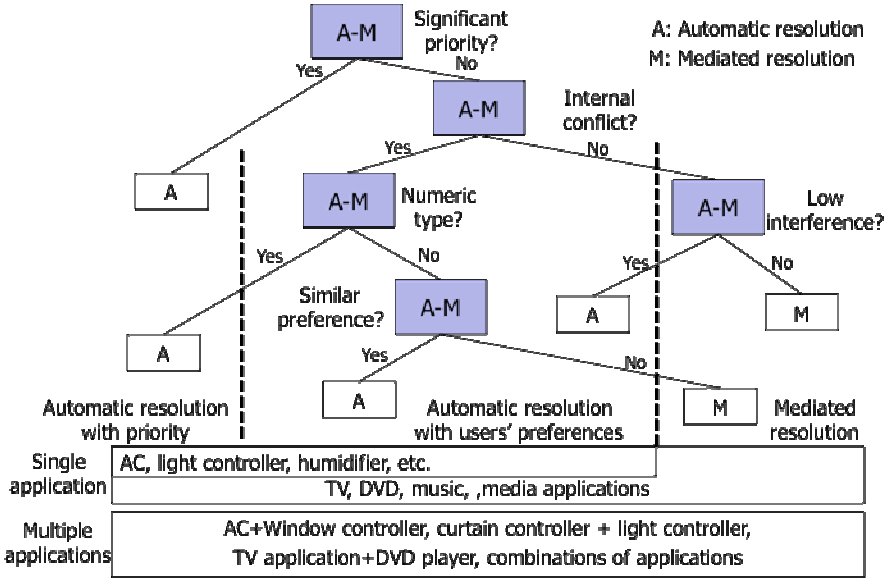


Figure 9: Approach determination tree.

Figure 9 shows how conflict is assigned to a particular resolution approach based on contextual information such as priority, types of attributes, dependency, preference and level of interference, which specify the types of conflict resolution. Among the contexts, priority is the first criteria in determining a resolution approach. In this case, outlined in Figure 9, the conflict is resolved automatically because it can be established that one user has the highest priority. In other cases where the conflict is from an application, determination is divided into two parts. The numeric attributes, are also resolved automatically because their resolution can have a single optimal solution. The symbolic attributes involved in similar preferences are resolved automatically since the solutions to these attributes are close to all of the users' preferences. Therefore, the conflicts of applications involving numeric attributes, such as an air conditioner, a light controller, and a humidifier, are automatically resolved, regardless of the users' preferences. The conflicts among media applications involving symbolic attributes, such as a TV application, a DVD player, and a music player, are resolved either by an automatic decision, or by a decision of users depending on the users' preferences.

In the case of conflicts among applications, the level of interference is used to determine conflict resolution. Thus, automatic resolution is assigned to the conflict when the level of interference is low and mediation is assigned to the conflict when shared resources or conditions require mutual execution. Accordingly, conflicts among applications that use contradictory effects are automatically resolved when their effects are relatively small. Environmental conditions like sound and temperature, such as the simultaneous use of the AC and Window in Figure 1, can have a small effect, depending on the users and their situations. Other conflicts, like

accessing both a DVD player and a TV application (in Figure 1) or a telephone and music player, require the users' participation because they are able to start only when their required resources are free to use.

4.3 Conflict resolution

In conflict resolution, the detected conflict is resolved based on the assigned resolution approach. The assigned approach can be an automatic resolution using priority or users' preferences, or a mediated resolution. Automatic resolution with priority is the most determinant situation, since one particular user has priority over the services. Thus, when conflict occurs within an application, the users' preferences dictate the resolution. Similarly, when dealing with conflicts among applications, the service with the highest priority has control over the resources and conditions within a space. Therefore, resolution occurs based on the highest priority.

Automatic resolution based on users' preferences takes place when no user has the highest priority. Automatic resolution is then divided into two types: automatic resolution over conflicts within a single application and conflicts among multiple applications. Further, conflicts in a single application involve both numeric and symbolic context properties. Applications with numeric properties are able to employ a linear optimal solution [Park et al. 2005a] or an average, which automatically becomes the resolution to the conflict. Applications with symbolic properties can have multiple solutions, but when users have similar preferences, any of the optimal solutions obtained from the selection algorithms, such as multiplication, addition and least misery strategy [Masthoff, 2004b], is close to user expectations. For example, the conflict from the light controller and the AC is resolved by selecting a certain value and the conflict from a TV application can be automatically resolved with one of the TV programs. Automatic resolution over conflicts among the applications involves shared conditions and resources. In particular, the conditions and resources that can be partitioned or adjusted are subject to automatic resolution. For the conditions, services' effects can be slightly compromised as a result of resolution for the applications. For instance, the conflict between the AC and the Window in Figure 1 can be automatically resolved by both decreasing the AC temperature from 25C to 23C along with slightly opening the window. The resources can then be divided into two parts as a resolution for the applications, such as the division of a TV display for a TV and DVD player.

Automatic resolution is not assigned to the conflict in situations where the solution is different from the users' preferences, is complicated with other applications, or both. Instead, users are asked to control the conflicts. Mediation is a technique to support a decision over the conflict by providing information and gathering feedback from the users. Thus, mediation generates recommendation information with resolution strategies, collects users' selections and makes a final decision with the selections. The recommendation depends on the algorithms used for automatic resolution with the users' preferences. Rather than using a single optimal solution, it generates possible candidates close to the optimal solution to allow users to negotiate for their conflicting services. In the case of applications with numeric properties, the candidates have values close to the optimal solution. In the case of applications with symbolic attributes, the next highest items generated by the algorithm used in automatic resolution are considered as candidates. With multiple

applications, conflicts arise due to their conditions and resources and information for mediation is obtained by utilizing their properties. For contradictory services, the candidates are generated differently according to the types of dependency. The candidates are formulated by obtaining values close to the optimal solution when the services contradict. Otherwise, the services are mutually exclusive and have no optimal solution. Thus, possible candidates from each application's properties are recommended. For example, in the case of a conflict between a TV application and a DVD player, a set of contents of the two applications as used to provide recommendations, allowing users to negotiate over the selection. Based on the resolution candidates, users mediate the resolution. Similarly, in external conflicts including an internal conflict, candidates are obtained by merging the conflict with other applications in order to make recommendations to users. For example, in the conflict between the TV application and DVD player accessed by three users in Figure 1, a set of TV Programs from two users and a set of DVD contents from one user are obtained and recommend to the users for mediating their selection.

5 Implementation and Evaluation

5.1 Implementation

The proposed conflict management framework was implemented with J2SDK, incorporating conflict detection, approach determination and resolution components into the management framework. For reasoning ontology, we utilized a Jena 2 inference engine that is widely used in an OWL-based ontology [Carroll et al. 2004a]. The SWRL rules were replaced with the Jena syntax. We incorporated the applications with the framework based on the 5W1H context, which describes the user's context in terms of who, what, when, where, why and how [Hong et al. 2007a]. Among this information, we focused primarily on who and what for describing the user profile and service profile. When users access the applications, they deliver the context describing the users and their preferences to the framework. In addition the framework, we also implemented a remote control by adding a group mediation function to an existing controller already used for recommending and gathering the user's selections against conflicting applications. The overall architecture of our implemented framework is displayed in Figure 10.

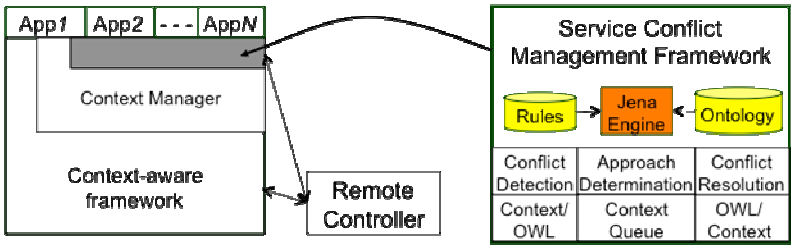
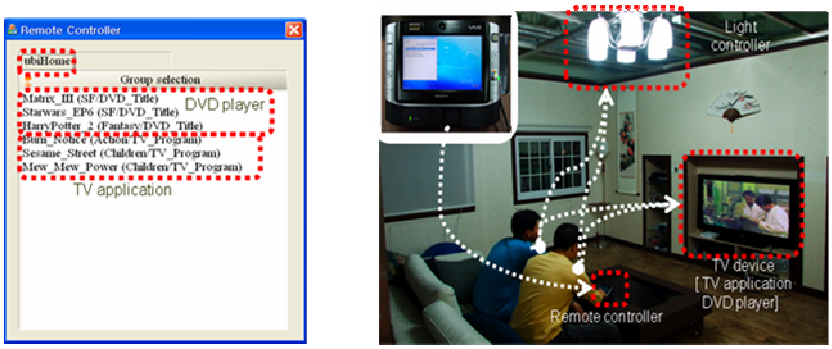


Figure 10: Implementation of the service conflict management framework.

As can be seen in Figure 10, the proposed framework is located in the context-aware application framework. The contexts are added to a context queue and the framework translates the context into an OWL description in order to detect conflict from the contexts. During the conflict resolution, the remote controller is used in recommending a set of candidates and gathering feedback from users when the framework is unable to resolve the conflict automatically. Finally, a conflict-free context is delivered to the framework after resolution. We then applied the updated framework to ubiHome [Jang et al. 2004b], a smart home test-bed where numerous sensors and applications are available for use. Among the applications, we exploited a DVD player, a music player, a TV application and a light controller for conflict management. Figure 11 shows the smart home equipped with the proposed infrastructure.



(a) (b)

Figure 11: A smart home equipped with the service conflict management framework. A recommendation on the controller (a) and ubiHome (b).

Figure 11(b), depicts two users in a smart home accessing the TV application, DVD player and light controller. In this situation, two conflicts arise due to the interferences of their services. One is between the TV applications and the DVD player deployed in the TV device and the other is from the light controller. According to the conflict management structure, the light controller automatically adjusts the level of light based on the users' preferences. However, the framework recommends a set of selections to be used for resolving the conflict between the TV application and DVD player through the remote control. Thus, users are allowed to select their preferred contents to watch together by utilizing the remote control, as shown in Figure 11(a). With their discussion and decision over the recommendation, they are able to watch one of the contents available from the two applications.

5.2 Performance Evaluation

The aim of the evaluation was to identify how efficiently the framework supports conflict management. We therefore observed the time required for detecting and resolving conflicts using the proposed framework in accordance to the number of conflicting services from different applications.

5.2.1 Method

For validating the effectiveness of the proposed framework, we first evaluated it with a general scenario and then with two different sets of service scenarios in a smart home. The first scenario was used to measure the managing time in a normal situation, as shown in Figure 1. The later two scenarios simulated situations where two users were accessing a set of conflicting applications. The performance evaluation involved both a set of applications involving an internal conflict, as well as one involving an external conflict.

1) A general scenario. Two users confront three conflicts in the smart home: the first from the light controller, the second between the TV application and the DVD player and the third between the AC and the window controller. We were interested in the compilation time of each component of the conflict management framework.

2) Internal conflict scenarios. The detection only included the users' preferences and required relatively little information. For this scenario, two users with different preferences were trying to access a set of applications, causing internal conflicts in their services over the property of the application due to their different preferences. Each user service included only 8 triples for describing the user preferences and properties of the services. In this case, only the rules for detecting the properties of the application were applied. The number of applications accessed by the users ranged from 1 to 10 and thus the number of conflicts also increased by 1.

3) External conflict scenarios. The detection required more information describing the users' preferences and service profiles. In this case, three users were trying to access a set of applications deployed in the same device. Among them, two users accessed one of the applications and the third user accessed the other, creating conflict not only in the first application, but also between the two applications. Each service included 14 triples describing resources along with user preferences and service profile. In this case, all the rules for detecting conflicts were applied and detection required much more time than in the internal conflict scenarios. The number of devices offering applications shared by the users was increased up to 10, causing the number of external conflicts to increase by 1.

To measure the performance, we evaluated the reasoning by using two lightweight platforms because context aware applications generally require less computationally powerful devices in ubiquitous computing environments. Considering the performance of currently used common set-top boxes, we selected two platforms: one platform contained a Centrino 1.5G CPU with 1G RAM and the other a Pentium III 900M CPU with 512M RAM.

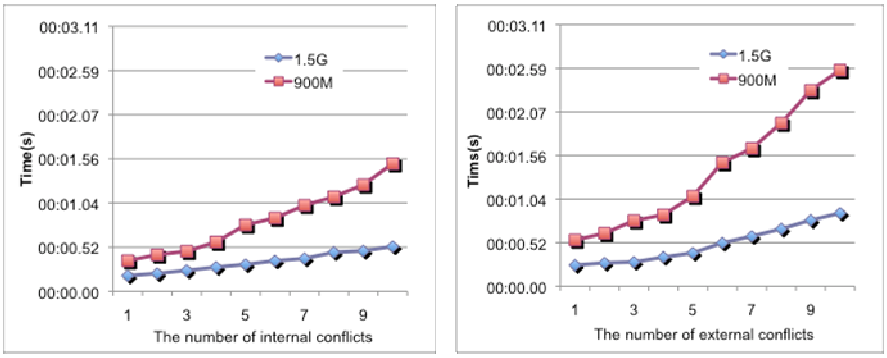
5.2.2 Results

Through the first observation, we found that time was primarily consumed in the detection process, requiring far more time than other components in the conflict management process. As seen in Table 1, the conflict detection task required much more time than the other components in both platforms. While mediation is considered to be paramount in conflict management, the quality of user negotiation is of equal importance during the mediation process.

Tasks	900M CPU		1.5G CPU	
	Time(ms)	Percentage (%)	Time(ms)	Percentage (%)
Pre-processing	56	8	43	13
Conflict detection	512	78	254	75
Approach determination	52	8	28	8
Conflict Resolution (Mediation)	38 (>1s)	6	14 (>1s)	4
Total	<658 (>1.6s)	100	<339 (>1.3s)	100

Table 1: Time spent on internal tasks of conflict management.

In the second observation, then, it is not surprising that the level of performance in conflict detection depends on the platforms and the rules applied to each situation.



(a) *(b)*

Figure 12: Time spent on reasoning service conflicts involving internal conflicts (a) and external conflicts (b).

Figure 12(a) shows the performance level of conflict management in the internal conflict scenarios. The time of conflict detection linearly increases in both platforms according to the number of internal conflicts, even though the lower platform took longer than the higher platform. Similarly, the performance in detection of external conflicts also linearly increases, but requires much more time than in situations involving external conflicts, indicated in Figure 12(b). Thus, the proposed conflict management framework seems sufficient to deal with conflicts of applications within a relatively small smart space where a limited set of applications operate.

5.3 User Study

For validating the effectiveness of the proposed method, we also conducted a user study on how well the conflict management framework assigns an appropriate resolution to conflict according to the types and situations of applications in the smart home. Since conflict is deterministically resolved when a specific user has a higher priority among users, we mainly observed conflicts within a single application and among applications involving users with equal priority.

5.3.1 Method

For the user study, we recruited 12 people, ten male and two female, aged from 24 to 40 who were members of a university community. Each experienced conflicts and resolutions of four applications in the test-bed. In addition to the real applications, a virtual living room and two virtual applications were shown in the large display in the test-bed as a flash animation to inform participants of the applications' responses. These applications included an air conditioner (AC) nearby the door to the room and a window controller. Two actors used the applications together with each participant and became involved in the conflict resolution. The user study consisted of four steps: a questionnaire, a preference setting, a main conflict experiment and resolutions with another questionnaire and a final interview.

Before the experiment, participants were first questioned about their preferred resolution method over conflicts occurring from a single application and multiple applications widely used in a smart home. Each question was about which method they would prefer to use to resolve conflicts and was answered on a 5-level *Likert scale*. They then set their preferences for the applications, which included a TV application, a music player, a light controller and an AC. The participants rated their preferences of over 40 items of the TV application and music player using an 11-level *Likert scale*, from 1, the lowest preference, to 11, the highest preference. The remaining applications used a 5-level *Likert scale* to set their preferred level of intention.

In the main study, the participants experienced three types of conflicts and their resolutions: 1) internal conflicts within an application, 2) external conflicts among multiple applications without an internal conflict and 3) external conflicts with an internal conflict. For internal conflicts of applications, the participants experienced conflicts and resolutions of three applications, the TV application, the music player and the light controller. Each of the applications had two scenarios: a scenario where the users had similar preferences, and an alternative scenario in which users had different preferences. In each scenario, the conflict was resolved both by an automatic resolution with the users' preferences and by a mediated resolution with recommendations. The multiplication algorithm, accurately reflecting both the rating and individual misery [Masthoff 2004b], was used for generating the optimal setting and candidates for the TV application and music player. The mean of the users' preferences and its close values were used for the solution and candidates for the light controller, respectively.

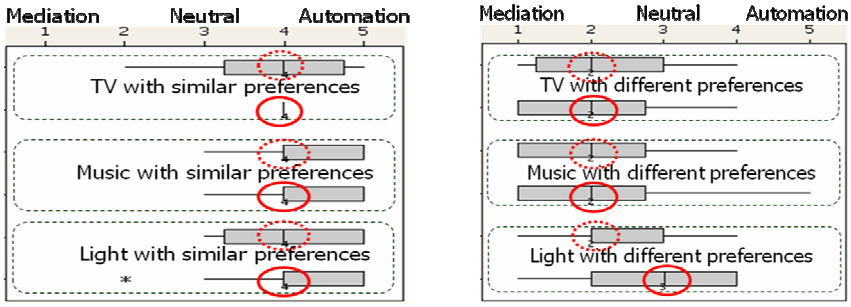
In the case of the external conflicts without any internal conflict involving the multiple applications, participants experienced conflicts and resolutions involving four sets of two applications with one actor. The conflicts were from the simultaneous access of the TV application with the DVD player, the TV application with the music player and the AC with the window controller with a small deviation and large deviation. The TV application and AC were given to each participant and the DVD player, music player and window controller were given to the actor, all having a default setting with a slightly different preference or deviation. To prove the effectiveness of the determination tree, participants experienced two resolution methods in each scenario: an automatic resolution and a mediation resolution. The automatic resolution divided the display into two parts for the TV application and DVD player, adjusted the volumes of the TV applications and the music player and

slightly decreased the openness of the window controller and the intensity of the AC. These resolution methods were considered the most objective means in situations where there was no priority among users. In contrast, a recommendation list, including selections obtained from the resolution algorithm corresponding to each of the conflicts, was given to the users through the remote control in the mediated resolution of each scenario.

Participants also experienced external conflict scenarios with an internal conflict, which involved four sets of applications with the two actors. These conflicts occurred from the simultaneous access of the TV application with the DVD player and the music player, the TV application involving two users and the music player involving one user, and the AC involving two users and the window controller involving one user. Conflict resolution methods used in these scenarios were the same as the methods used in conflicts involving multiple applications. After each scenario, the participants were asked to identify which of the two resolution methods they preferred to resolve conflict when they were together with one or two family members. After the main study, we also interviewed each participant to understand their feelings about the conflicts and resolution.

5.3.2 Results

Overall, we found that most participants preferred resolution methods directly related to their preferences in the case of internal conflicts, while they appreciated the recommendation and decision in the case of external conflicts. In order to compare the participants' opinions before and after experiencing the conflict resolution, we calculated the interquartile range (IQR) of the data from the *Likert scale*. Figure 13 shows the box plots of participants' preferred resolution methods in the internal conflict scenarios. Each application had two preferences scenarios (the left figure and right figure) and circles in the plots highlight the median of each scenario. Dashed circles and solid circles indicate the median of the preferences before and after the experience, respectively.



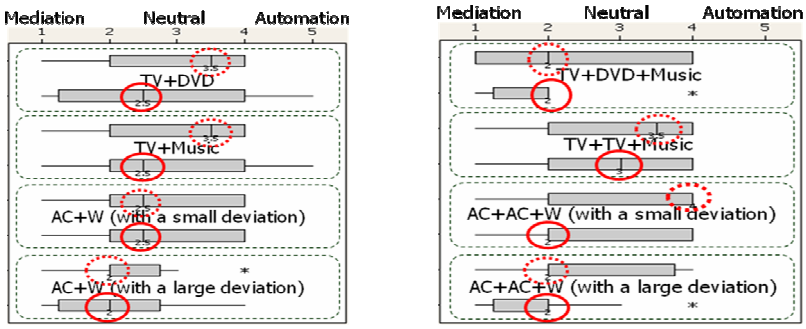
(a) *(b)*

Figure 13: Box plots of participants' preferred resolution method in internal conflicts from an application involving similar preferences(a) and different preferences(b).

As can be seen in Figure 13(a), the participants preferred automatic resolution when their preferences were similar in the three applications before and after the

experiences. On the other hand, they preferred the mediation resolution in the TV application and music player when their preferences were different, as shown in Figure 13(b). For the light controller, they had no preferred resolution method in the different preference scenarios, but more preferred an automatic resolution after experiencing the conflict resolution. Through the interviews, most of the participants reported that they liked the automatic resolution when they had similar preferences because it gave results similar to their expectations and allowed them to avoid arguments with other family members. However, they also stated that they preferred a mediated resolution when they had different preferences, except in the case of the light controller, as they could simply predict the optimum setting with no need for further recommendation or discussion.

In the conflicts and resolution schemes involving multiple applications, participants indicated a preference for mediated resolutions in all applications after the experiments.



(a) *(b)*

Figure 14: Box plots of participants' preferred resolution methods in external conflicts(a) and external conflicts involving an internal conflict(b).

As seen in Figure 14(a), the preference was for automatic resolution before the experiments, but this changed to a mediated resolution after. However, for the AC and window controller in both preference scenarios, participants preferred to discuss the controls with other family members. Not surprisingly, almost all participants were against the simultaneous operation of both the AC and window controller, mainly due to their large deviation in controlling temperature. In the interview, those preferring an automatic resolution stated the reason to be that both users could have their services after an automatic resolution, thus satisfying each user. Others, however, preferred to select their resolution with other family members because they did not like the divided display and room chosen by the automatic resolution. They commented, "The living room is an open place and people need to do common activities in this place" and "the members who want to do his/her own activity should use their own room". For the temperature control, energy consumption was an additional consideration in conflict resolution. Most of those who preferred the mediated resolution stated that the AC and window controller could automatically operate together if both users were satisfied and the controls consumed low energy. In

other words, the current temperature, or no control, could be a solution of automatic resolution.

Overall, in the external conflicts involving an internal conflict, most participants preferred a mediated resolution in all applications after the experience, as illustrated in Figure 14(b). In the situations where three users access the TV application, the DVD player and the music player or the AC and window controller with a large deviation, most participants wanted to discuss and control the applications together, rather than use the applications separately. Although most had either no preferred method or slightly preferred the automatic resolution in the TV application with two users and the music player with one user before the experience, they slightly changed to prefer the mediated resolution. Even in the case of the temperature control using the AC and window controller involving three users with a small deviation, participants largely changed their preferred resolution method after experiencing conflict resolutions.

5.4 Discussion

Through this preliminary observation and user study, we found several implications about our framework intended for a smart home. First, a variety of conflicts occurring in a smart home can be flexibly managed. Although only a subset of applications were introduced in the implementation and evaluation, we expect that the conflicts from other types of applications can also be managed with the proposed approach if their services are described with the ontology or there is a match between their description and the ontology. Second, it is also clear that users' preferences and the application type are the main factors in determining a resolution approach for a conflict of a single application. The conflict of the applications using a symbolic attribute can be automatically resolved when the users' preferences are similar. The conflict of an application using a numeric property can also be resolved automatically with no dependency in preference deviation. This result is similar to that previously found in a mixed-initiative conflict resolution [Shin et al. 2008a]; however, this study has proven the preference for resolving a conflict of a numeric type application together with symbolic type applications. Third, mediation is the preferred means to resolve conflicts among multiple applications. The simultaneous operation by partitioning a resource can satisfy individual preferences, but divides the space and separates the users. Mediation, on the other hand, allows users to coordinate their applications with the recommendations and discussion and encourages interaction with family members, making it the preferred method of resolution. Finally, although mediation is preferred in conflict resolution among multiple applications, the possibility exists of automating the resolution when the effects of conflicting services are small enough to use the applications together and the resolution does not create other problems, such as energy consumption.

6 Conclusion

In this paper we proposed a service conflict management framework for a smart home, in which multiple residents interact with various kinds of applications together and simultaneously. We introduced an ontology which describes the users' desired

services implemented in conflict detection as well as an approach determination tree which assigned an appropriate resolution method to the detected conflict. Based on the selected method, conflict was resolved either by automatic decision or through discussion between the users. Through implementation and a preliminary evaluation, we found that the proposed framework can flexibly manage conflicts of multiple users in a smart home.

This work is the first step toward developing a general conflict management framework for dealing with conflicts of multiple users in a smart home and we are planning to continue this research further. The first area of future study will focus on conflict resolution of users from different generations. Although the current framework dynamically resolves conflicts of users by exploiting contextual information, a group in smart spaces consists of a wide range of people and must incorporate different preferences. We are further interested in the social relationship of users within a smart space. The current framework assumed that the users did not contribute to changing each other's preferences, but this possibility certainly exists and needs to be explored.

Despite its limitations, we expect this study will play a vital role not only in smart homes, but also in smart offices, by utilizing both the ability of the applications and the users in resolving conflict of multiple users.

Acknowledgements

This research is supported by KOCCA of MCST, Korea, under the CT R&D Program 2009.

References

[Carroll et al. 2004a] Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. Wilkinson, K.: "Jena: implementing the semantic web recommendations"; *Proceedings of the* 13th international World Wide Web conference, New York, NY, USA (2004), 74-83.

[Chen et al. 2004a] Chen, H. Perich, F. Finin, T. Joshi, A.: "SOUPA: standard ontology for ubiquitous and pervasive applications"; Proc. of 1st Annual International Conference on Mobile and Ubiquitous Systems (*MOBIQUITOUS*), *IEEE CS Press*, Boston, Massachusetts, USA (2004), 258- 267.

[Das et al. 2006b] Das, S.K., Roy, N., Roy, A.: "Context-Aware Resource Management in Multi-Inhabitant Smart Homes: A Nash H-Learning based Approach"; Pervasive and Mobile Computing, *ELSEVIER*, 2, 4(2006), 372-404.

[Dey 2001b 2001b] Dey, A.K.: "Understanding and Using Context"; *Journal of Personal and Ubiquitous Computing*, 5, 1(2001), 4-7.

[Dey, Abowd 2000a] Dey, A.K., and Abowd, G. D., "The Context Toolkit: Aiding the Development of Context-Aware Applications"; Proc. of the Workshop on Software Engineering for Wearable and Pervasive Computing (*SEWPC*), Limerick, Ireland (2000).

[Ejigu et al. 2007a] Ejigu, D., Scuturici, M. Brunie, L.: "An ontology-based approach to context modeling and reasoning in pervasive computing"; In Proc. of Pervasive Computing and Communications Workshops, New York, USA(2007), pp.14-19.

[Gu et al. 2004a] Gu, T., Pung H. K., Zhang D. Q.:"A Bayesian Approach for Dealing with Uncertain Contexts"; In Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive '04), Advances in Pervasive Computing, Austrian Computer Society, vol. 176, Vienna, Austria (2004).

[Haya et al. 2006b] Haya, P.A., Montoro, G., Esquivel, A., Garcia-Herranz, M., Alaman, X.: "A Mechanism for Solving Conflicts in Ambient Intelligent Environments"; *Journal of Universal Computer Science*, 12, 3(2006), 284-296.

[Hong et al. 2007a] Hong, D., Schmidtke, H.R., Woo, W.: "Linking Context Modelling and Contextual Reasoning"; In Proc. of the 4th International Workshop on Modeling and Reasoning in Context *(MRC)*, Roskilde, Denmark (2007), 37-48.

[Jameson 2004] Jameson, A., "More than the sum of its members: challenges for group recommender systems, Proceedings of the working conference on Advanced visual interfaces, Gallipoli, Italy (2004), pp. 8-54.

[Jang et al. 2004b] Jang. S., Shin, C., Oh, Y., Woo, W.: "Introduction of "UbiHome" Testbed"; *Information processing society of Japan (IPSJ)*, 2005, 60(2005), 215-218.

[Kulkarni 2002a] Kulkarni, A.: "A Reactive Behavioral System for the Intelligent Room"; M. Eng. Thesis for MIT, Cambridge, Massachusetts, USA (2002).

[Masthoff 2004b] Masthoff, J.: "Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers"; User Modeling and User-Adapted Interaction, *Kluwer Academic Publishers*, 14, 1(2004), 37-85.

[McCathy, Anagnost 1998a] McCarthy, J.F. and Anagnost, T.D.: "Music FX: An arbiter of group preferences for computer supported collaborative workouts"; Proc. of *CSCW*, *ACM Press*, Seattle, Washington, USA (1998), 363-372.

[Niu and Kay 2008] Niu, W., Kay J.: "Location Conflict Resolution with an Ontology"; The proceeding of Pervasive 2008, Sydney, Australia (2008), pp. 162–179.

[O'Hara et al. 2004a] O'Hara, K., Lipson, M., Jansen, M., Unger, A., Jeffries, H., and Macer, P.: "Jukola: Democratic music choice in a public space"; Proceedings of *DIS*, *ACM Press*, Cambridge, Massachusetts, USA (2004), 145-154.

[Park et al. 2005a] Park, I., Lee, K., Hyun, S., Yoon, H.: "A dynamic Context Conflict Resolution scheme for Group-aware ubiquitous computing"*;* Proc. of *ubiPCMM* at *ubiComp2005, CEUR*, Tokyo, Japan (2005), 42-47.

[Ranganathan et al. 2004b] Ranganathan, A., Al-Muhtadi, J., Campbell, R.H.: "Reasoning about Uncertain Contexts in Pervasive Computing Environments"; *IEEE Pervasive Computing*, 3, 2(2004), 62-70.

[Resnick et al. 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: "GroupLens: An Open Architecture for Collaborative Filtering of Netnews"; Proceedings of ACM Conference on Computer Supported Cooperative Work, *ACM Press*, Chapel Hill, North Carolina, USA(1994), pp. 175-186

[Schilit et al. 1994a] Schilit B., Adams, N., Want R.: "Context-Aware Computing Applications"; *Proc. of the 1st* International Workshop on Mobile Computing System and Applications, *IEEE CS Press*, Santa Cruz, California, USA (1994), 85-90.

[Shin et al. 2008a] Shin, C., Dey, A.K., Woo, W.: "Mixed-Initiative Conflict Resolution for Context-aware applications"; Proc. of ubiComp2008, *ACM Press*, Seoul, Korea (2008), 262-271.

[Shin et al. 2007b] Shin, C., Yoon, H., Woo, W.: "User-Centric Conflict Management for Media Services Using Personal Companions"; *ETRI Journal*, 29, 3(2007), 310-321.

[Strang et al. 2003a] Strang, T., Linnhoff-popien, C., Frank, K.: "CoOL: A Context Ontology Language to enable Contextual Interoperability"; Proc. of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (*DAIS2003), Springer-Verlag*, Paris, France (2003), 236-247.

[Wang et al. 2004a] Wang X.H., Zhang, D. Q., Gu, T., Pung, H.K.: "Ontology based context modelling and reasoning using OWL"; Proc of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (*PERCOMW'04*), *IEEE CS Press,* Orlando, Florida, USA (2004), 18-122.

[W3C 2004] OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/ SUBM-OWL-S/.

[W3C 2004] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/.

[Wishart et al. 2005a] Wishart, R. Henricksen, K., Indulska1, J.: "Context Obfuscation for Privacy via Ontological Descriptions"; The Proceedings of the First International Workshop on Location and Context-awareness, Oberpfaffenhofen, Germany (2005), pp. 276~288.

[Yu et al. 2006b] Yu, Z., Zhou, X. Hao, Y., Gu,J.: "TV program recommendation for multiple viewers based on user profile merging"; User Modeling and User Adapted Interaction, 16, 2(2006), pp. 62-82.