Group-aware Service Discovery using Effect Ontology for Conflict Resolution in Ubiquitous Environment

Gwang-hun Kim, Do-hyun Kim, XuanTung Hoang, Young-hee Lee
Information and Communications University
{nuly17, t12t12t12, tung_hx, yhlee}@icu.ac.kr

***Abstract* — In this paper, we propose a group-aware service discovery architecture resolving conflict problem in ubiquitous computing. In the past, researches on resolving conflict have used service's QoS, user's preference, and user's intention. However, previous researches have the problem that it applies resolution scheme to non-conflicting situation because it does not consider space concept. Therefore, we propose a group-aware service discovery architecture resolving conflict problem considering interaction space that is affecting scope of task to increase user's satisfaction. In this architecture, service ontology model including interaction space as well as QoS, preference, and intention expresses service information and task information. With service ontology model, the architecture detects and resolves conflict situation. And discovery scheme that is proposed resolution method finds non-conflict service with interaction space concept. Simulation result shows that proposed architecture provides higher user's satisfaction than previous research.**

***Keywords* — conflict resolution, group awareness, semantic service discovery, ubiquitous computing**

1. Introduction

Although new computing paradigm called ubiquitous computing [1] was introduced in 1991, we are still on initial stage of ubiquitous environment. In ubiquitous environment, computer is harmonized with our daily life and it makes our daily life convenient. To realize ubiquitous computing paradigm, one of the most important and challenging issue is how to provide relevant information and / or service to the user and we call it context awareness [2].

Context awareness is the key feature to make the paradigm successful. To make context awareness system, we collect physical information with heterogeneous physical sensor and infer high level. We can use the context to adapt application or discover the most appropriate service in such environment. In real world, users usually interact with others when they perform their task. Thus, we need to not only consider the context of individual user but also put them together into group context, a set of context of individual users [3].

During the interaction, user's intention may conflict. For example, Alice turned the light off to sleep and Bob wants to turn the light on while he is entering. In this situation, their intentions make conflict. System should support a conflict resolution ability to make human's life convenient.

In conflict resolution, it is important to maximize the satisfaction of the involved users as much as possible [5]. Many researches try to address conflict problem. CARISMA [4] selects one of resolution choices that maximize user's satisfaction based on service's QoS and user's preference. However, it has limitation that targets cooperative application having same intention. Park et al. [5] considers not only service's QoS and user's preference but also user's intention. It detects conflict with action semantic ontology having intention information and finds a negotiation value that can maximize user's satisfaction. It can target different applications because it considers user's intention.

However, Park et al. has possibility to apply conflict resolution in spite of no conflict situation that can be judged based on intuition. In the example of above, if we use main light that affects entire bedroom space, it is conflict situation and system should resolve conflict problem. However, if we use stand light that affects only entering person and stand light does not affect sleeping person, it is not conflict situation and system does not need to react. Park et al. applies conflict resolution in case of both of main light and stand light.

Therefore, we need conflict resolution model similar with intuition. To realize conflict resolution system, we need 'interaction space' concept that is the affected range of service's task. If we use interaction space concept, we can discover the service that affects to wanted user and does not affect to non-wanted user. Ultimately, the consideration of interaction space concept increases user's satisfaction. To make this system, we construct a service ontology model considering service's QoS, user's intention, user's preference, and service's interaction space. System detects conflict situation based on service ontology model. And system uses 'service discovery', that is the component of ubiquitous middleware, to discover non-conflict service. The simulation results show that the proposed approach provides higher satisfaction than previous work. It is optimized at the situation when there are many services having various interaction space.

The rest of the paper is organized as follows. In Chapter 2, we discuss our approach with related work. Chapter 3 covers design considerations and assumption. Chapter 4 describes how to design our proposed approach in detail. Chapter 5 presents implementation and evaluates our research. Finally we describe our conclusions and suggest future work in Chapter 6.

2. Related work

In this chapter, we first introduce research about conflict resolution to understand how they address conflict problem and show what weak point is. And we make comparison table among them and our paper. Then, we introduce service discovery research that is a solution of this paper.

CARISMA is context-aware middleware system. It addresses conflict problem that occurs when different policies can be used in the same context. There are two types of conflicts in CARISMA such as intra-profile conflict and inter-profile conflict. Intra-profile conflict occurs on an application for a single user and inter-profile conflict occurs among applications for multiple users. It proposes sealed-bid auction mechanism that collects bid from conflicted applications and selects policy that maximizes social welfare based on service's QoS and user's preference.

Park et al. addresses a conflict problem between different context-aware applications. It considers not only service's QoS and user's preference for conflict resolution but also user's intention for different applications. It detects conflict if the effects of each user's action are contradictory. The action semantic ontology has information to infer the contradictory effect. It uses cost minimization method to resolve conflict problem. It considers user's non-satisfaction as a cost. It tries to find the value that can minimize user's non-satisfaction. It means that the negotiation value can maximize user's satisfaction.

Table 1 summarizes the consideration of existing conflict resolution researches such as CARISMA, Park et al., and this paper. CARISMA considers service's QoS and user's preference to resolve policy conflict among applications. However, it targets to same kinds of applications. Park et al. considers service's QoS, user's preference and user's intention. Thus, it can be used for various kinds of application because it considers user's intention. However, in real world, service has interaction space that is the affected range of service's task. If we consider interaction space, system can discover the service that affects the wanted user and does not affect the unwanted user. Consequently, we can maximize user's satisfaction with interaction space concept.

Service Discovery is an important component in ubiquitous middleware. It suggests the most appropriate service without user's distraction. It has two issues such as context-awareness and semantic search ability.

Context-aware Service discovery uses context information to discover service for the user. For example, when user wants to use printer service, system suggests the printer that is the nearest one from user. In that situation, system uses location as a context. System can use other contexts such as load of printer, QoS, and queue of printer as well as location context. To realize context-aware service discovery, discovery server should get context information of service provider and service requester that can help system to discover service.

Semantic Service discovery can discover service without exact information such as service's name and type. Early works in service discovery such as UPnP [6], SLP [7], and Salutation [8] try to find/match service with user queries based on exact information such as service's name and type. However, these kinds of simple syntactic matching for discovery may lose its flexibility owing to strict requirement on common agreement of all service's syntactic. Moreover, syntactic matching decreases the possibility to discover to most appropriate service because it just depends on described syntactic information, not service' semantics. With these limitations, there has been much work to discover services based on the semantics of services. Similar to semantic web [9], semantic service discovery schemes [10,11] usually deploy ontology [12] as their common knowledge repository storing service information.

System	QoS	Preference	Intention	Interaction Space
CARISMA	o	o	x	x
Park et al.	o	o	o	x
Proposed scheme	o	o	o	o

Table 1. Comparison of Conflict Resolution Research

In this paper, we use service discovery as a solution for conflict resolution. Service discovery of this work has context-aware feature because it considers location context. And service discovery of this work also has semantic search feature because it uses ontology to infer relevant service without exact service's information.

3. Design considerations and Assumption

This chapter provides some consideration points in design of proposed conflict resolution system. And this chapter shows some assumptions.

Firstly, we consider overall conflict addressing architecture. Discovery scheme that is proposed conflict resolution scheme runs in the situation that has non-conflict service. Unless there is non-conflict service, Discovery scheme does not work and we should run Negotiation scheme that is previous work's scheme. Two schemes are complementary relation. Therefore, we should consider conflict addressing architecture that two schemes harmonize on.

Secondly, we consider interaction space model that is newly added concept. We want to model interaction space concept to make conflict resolution model similar with intuition. Interaction space's design should increase user's satisfaction and be similar with human's intuition. We make service ontology that has not only interaction space, but also other considerations such as QoS, preference, and intention.

Finally, we consider that how we detect conflict situation and how we resolve conflict problem. Conflict detection runs with service ontology model. We consider how proposed Discovery scheme discovers non-conflict service with interaction space concept.

The proposed approach has these assumptions as following.
- Two persons are related with conflict
- Conflict is contradictory situation of two persons' effect
- Service's QoS can be quantified
- System needs location system that has hierarchy ontology.

4. Proposed approach

Our proposed approach consists of two parts: conflict detection, conflict resolution. System detects conflict with user's intention in service ontology model. And system resolves conflict with two schemes such as Discovery scheme and Negotiation scheme. Discovery scheme discovers the service that does not make conflict with intention, preference, and interaction space. Negotiation scheme calculates negotiated value with intention and preference. Firstly, system tries Discovery scheme. Unless there is non-conflict service, system tries Negotiation scheme.

4.1 Overall Architecture

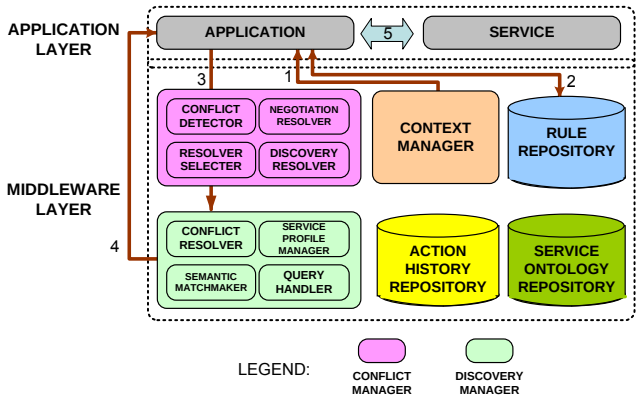

Figure 1. Overall structure of the conflict resolution architecture

The overall structure of the architecture is presented in Figure 1. There are mainly two parts in this system such as application layer and middleware layer. Application layer has application and service. And middleware layer has middleware components to support ubiquitous application such as conflict manager, context manger, discovery manager and so on. Conflict manager manages conflict situation and discovery manager finds appropriate service for user and context manager manages and generates context. We explain architecture's work step as follows.

1. Context Manager infers person's activity context with collecting physical information and notifies activity context to application.
2. Application finds matched rule in Rule Repository and retrieves task information if it exists.
3. Application requests relevant service instance with task information to Discovery Manager.
   A. Conflict Manager intercepts the request to check conflict existence.
   B. If conflict is detected, Conflict Manager executes resolution scheme.
      i. Firstly, it executes Discovery scheme.
      ii. Unless there is non-conflict service, it executes Negotiation scheme.
4. Application receives service instance (essential) and negotiation value (optional).
5. Application interacts with the service.

4.2 Service Ontology Modeling

We propose service ontology model for conflict detection and conflict resolution. This ontology contains service's QoS, user' intention, user's preference, and service's interaction space. User's intention can be used for conflict detection. User's preference, service's QoS, and service's interaction space can be used for conflict resolution. We extend Woohyun et al. [13], the ontology for context-aware and semantic service discovery, to address conflict problem. Thus, if we use proposed service ontology model, we can use two features such as conflict resolution and context-aware semantic service discovery. Newly added features are Preferredlevel and usedcount in preference ontology, haspolarity in effect ontology, InteractionSpace.

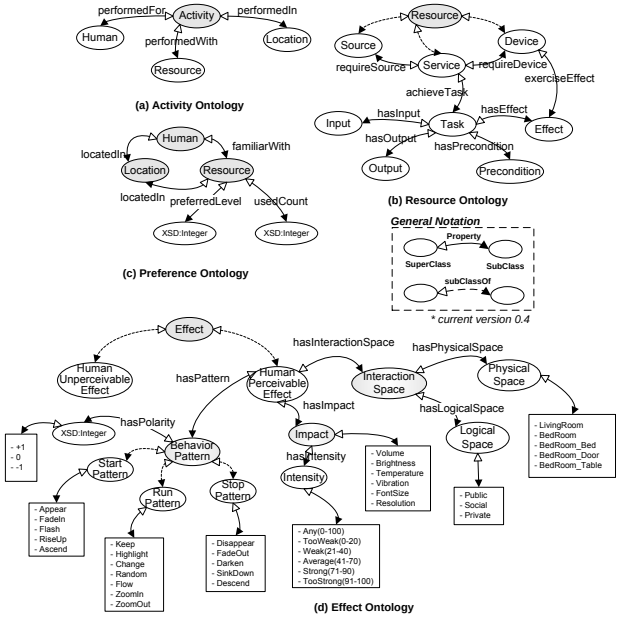

Figure 2. Service Ontology Model

Figure 2 shows an upper-ontology designed by OWL [14]. Ovals represents classes, black arrows represents characterized properties, while arrows represent inverse properties, and dotted lines represent subsumption relations among some classes. On the upper ontology, we do not show all of the properties among the classes and all of the characteristics of each property such as function, transitive, inverse, and symmetric. For example, input node can get various types of node such as account number of bank application and departure time of train application according to target environment. We just present a design principle to make use of this ontology model in various environments.

Activity Ontology has information about user's current activity. It can present user's context. Resource Ontology has information about service that supports its function. It contains service model. We defined 'Task' as a service granularity [15]. Task has IOPE (Input, Output, Precondition, Effect) [9] to register and discover service semantically. Preference Ontology expresses user's preference about service. It has

PerferredLevel that can express preferred level of impact and UsedCount expresses service's preference.

Effect Ontology is major ontology for conflict resolution. It has three sub classes such as InteractionSpace, Impact, and BehaviorPattern. InteractionSpace is defined as the affected range of service's task. We use two concepts to express interaction space such as physical space and logical space. Physical Space is geographical range of that service`s action affects. It uses location hierarchy ontology for semantic expression. For example, location ontology has bedroom class and there is sub-class such as bedroom's bed and bedroom's door in bedroom class. Logical Space is human activity range that service's action affects. We inspire logical space concept from [16]. Logical space expresses not geographical space, but human relation space. It consists of private that is for only one person, social that is for group, and public that allows freely accessible to people. For example, PDA and headphone is private, shared screen is social, and big screen in Seoul train station is public. Impact affected target's context. It has intensity that can express level of service's impact such as brightness, temperature, sound. BehaviorPattern is changing pattern of target's context. It has polarity that can simply express BehaviorPattern. It is used for conflict detection.

4.3 Conflict Detection

Conflict Manager intercepts task request from Application and check if conflict exists. To detect conflict, Conflict Manager gets a list of previous tasks from Task History Repository and check whether below condition is matched or not.

If (A ∩ B), then they are conflict
A: Impact is same
B: Behavior Pattern's polarity is -1 and +1

If task request and one of previous tasks is matched with above condition, Conflict Manager determines that it is conflict situation. We assume that conflict is contradictory situation of two persons' effect. We call the previous task to conflict task. We can show example of detecting conflict situation with Figure 3. In Figure 3, if previous task is 'turn the light off', its impact is brightness and behavior pattern's polarity is -1. And if request task is 'turn the light on', its impact is also brightness and behavior pattern's polarity is +1. Thus, it is conflict situation because impact is same and behavior pattern's polarity is contradictory.

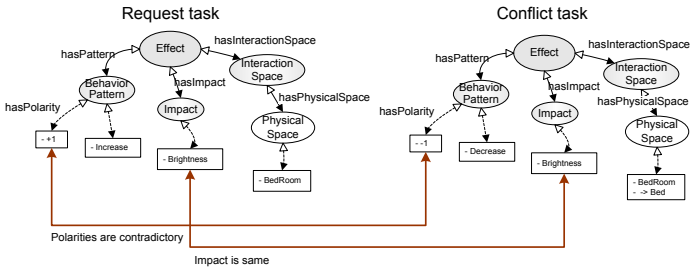

Figure 3. Example of detecting conflict situation

4.4 Conflict Resolution

If Conflict Manager detects conflict situation, it tries to resolve conflict problem. This architecture has two resolution schemes such as Discovery scheme and Negotiation scheme. Discovery scheme is proposed scheme that finds non-conflict service with interaction space and Negotiation scheme is previous scheme proposed by Park et al. Firstly, system runs Discovery scheme. Unless there is non-conflict service, system executes Negotiation scheme. The reason why executes discovery scheme first is that discovery scheme can increase user's satisfaction more than negotiation scheme.

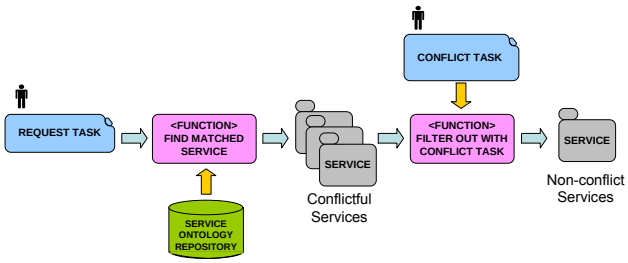

Figure 4. Discovery scheme's flow

Figure 4 shows Discovery scheme's flow. Discovery scheme finds the service that does not make conflict problem with interaction space. There are two steps in discovery scheme. In first step, system finds matched services with request task. Service should register itself before executing discovery scheme. And in step 2, with candidate services that system finds in step 1, system finds non-conflict services with conflict task in terms of interaction space. Originally, the service matched with request task makes conflict with conflict task. However, we discover non-overlapped service considering interaction space. We determine whether interaction space overlapped or not with below rule.

If (A ∩ B), then their interaction space is overlapped
A: Physical space is overlapped
B: Service's logical space is public or social

If user's interaction space where user can sense environmental change is not overlapped with service's interaction space where service affects to others, we determine that it is not conflict. And although user's sensing area is overlapped with service's affecting range, we also determine that it is not conflict if service's logical space is private. For example, in bedroom, a user is sleeping on the bed. If turning the main light on, it is conflict because light service's physical space is overlapped and light service's logical space is public. However, if turning the stand light on, it is not conflict because stand light service's physical space is not overlapped. We made a feasible scenario to show how discovery scheme works in chapter 5.

If Conflict Manager failed to discover non-conflict service, it executes Negotiation scheme. It calculates negotiated value that can minimize user's dissatisfaction with QoS and preference. We skip detail explanation of Negotiation scheme.

5. Evaluation

5.1 Scenario

To illustrate how this work address conflict situation, we now describe a feasible scenario.

Bob arrives at home after finishing his work and sits down at sofa. System plays the music because system knows that he always play the music. But Bob commands stop the music to take a rest because of hard work in his office. Sometime after, Alice who is Bob's wife enters the home. Alice has listened to the music that is song by her favorite singer. Alice commands play the music to listen to the music continually. Then system detects conflict between Bob's task and Alice's task, it suggests a portable music service to Alice that can avoid conflict. Thus, Bob does not affect and Alice can listens to the music with a portable music service.

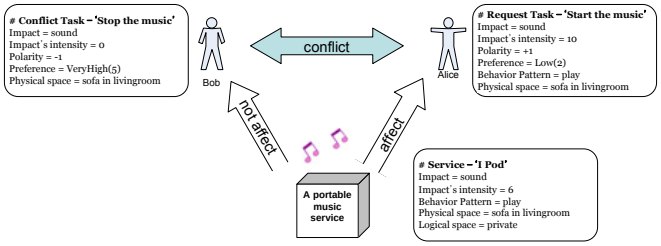

Figure 5. Scenario – conflict situation between user who wants to listen to music and user who does not listen to music in living room

In Figure 5 scenario, Bob's task and Alice's task make conflict problem because their impact (sound) is same and polarity is contradictory (-1 and +1). Thus, system detects conflict situation and it executes discovery scheme. System found a portable music service as a result of discovery scheme. In step 1 of discovery scheme, a portable music service is matched with Alice's task because impact (sound) is same and behavior pattern (play) is same and physical space (sofa in living room) is overlapped. And in step 2 of discovery scheme, system determines that a portable music service does not affect to Bob because its logical space is private even if its physical space is overlapped. Thus, with this process, system addresses conflict problem.

5.2 Simulation

We have built a simulator to validate the proposed approach. In this simulation, we compare difference between previous work and proposed approach and we show that our approach has better performance in terms of user's satisfaction.

We use total cost of two users as dissatisfaction degree. If cost is high, user's satisfaction is low. We use Table 2's equation. We modify Park et al.'s equation to make Table 2. In case of discovery scheme, it uses part of cost function because it does not affect to the user that does not want to use service.

	Negotiation scheme	Discovery scheme
r factor	if QoS > negotiataion point r = negotiation point else r = QoS	r = QoS
Cost function	$C_T = \Pr_{U_AC}(r-\alpha)^2 + \Pr_{U_BC}(r-\beta)^2$	$C_T = \Pr_{U_AC}(r-\alpha)^2$

Table 2. Cost function table for simulation

C_T - total cost of two users

$\Pr_{U_AC}$ - user A's preference

$\Pr_{U_BC}$ - user B's preference

α - user A's service QoS

β - user B's service QoS

r - resolution QoS

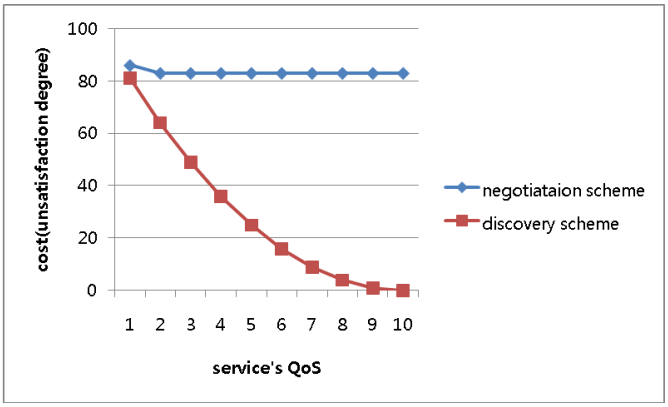

Figure 6. cost according to service's QoS with high preference of 'turn on' task

Parameter	Alice	Bob
Intention	TurnOff(0)	TurnOn(10)
Preference	VeryHigh(5)	VeryLow(1)

Table 3. Setting table of Scenario 1

Figure 6 shows the cost according to service's QoS with high preference of 'turn on task'. Its setting uses Table 3. In Figure 6, x-axis shows servcie's QoS and Y-axis is a cost (unsatisfaction value) related calculated with above equation for negotiation scheme's cost and discovery scheme's cost respectively.

Negotiation value is 1.66 values if we calculate it with Table 3's setting. Equation is shown in [5]. In Table 3's setting, TurnOffTask has more preference than TurnOnTask. Thus, negotiation value is near to TurnOffTask. In negotiation scheme's graph, if service's QoS is bigger, cost is smaller and it becomes flat from 1.66 QoS. It is because r value (resolution value) uses negotiation value if service's QoS is bigger than negotiation value. If we use QoS value as r value although service's QoS is bigger than negotiation value, negotiation scheme's cost will increase from 1.66 QoS. Figure 6 shows big cost difference between negotiation scheme and discovery scheme.

On the other hand, discovery scheme graph decreases constantly from QoS 0 and cost will be 0 at QoS 10. It is because discover scheme provides service to wanted user and it does not affect to unwanted user. And we can see that cost

difference between negotiation scheme and discovery scheme increases when QoS increases. It is because negotiation scheme affects both of users and discovery scheme affects just one user.

It is because small negotiation value (1.66) makes fast flat pattern. Figure 6 shows the best performance of discovery scheme.

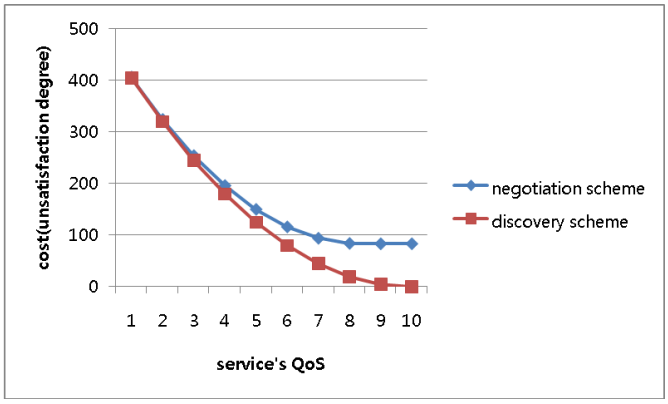

Figure 7. cost according to service's QoS with high preference of 'turn off' task

Parameter	Alice	Bob
Intention	TurnOff(0)	TurnOn(10)
Preference	Low(2)	VeryHigh(5)

Table 4. Setting table of Scenario 2

Figure 7 shows the cost according to service's QoS with normal preference of 'turn on task'. Its setting uses Table 4. Negotiation value is 8.33 values if we calculate it with Table 4's setting. In Table 4's setting, TurnOnTask has more preference than TurnOffTask. Thus, negotiation value is near to TurnOnTask. In negotiation scheme's graph, if service's QoS is bigger, cost is smaller and it becomes flat from 8.33 QoS. It is because r value (resolution value) uses negotiation value if servcie's QoS is bigger than negotiation value. If we use QoS value as r value although service's QoS is bigger than negotiation value, negotiation scheme's cost will increase from 8.33 QoS. On the other hand, discovery scheme graph decreases constantly. Figure 7 shows the worst performance of discovery scheme. However, discovery scheme's performance is better than negotiation scheme's performance.

6. Conclusion and Future work

In this paper, we proposed conflict addressing architecture considering interaction space. We considered not only service's QoS, user's intention, user's preference, but also service's interaction space. Service's interaction space expresses the affected range of service's task. Proposed approach detects conflict with user's intention in service ontology model. And system resolves conflict with two schemes such as Discovery scheme that is proposed scheme and Negotiation scheme that is previous scheme. Discovery scheme discovers the service that does not make conflict with intention, preference, and interaction space. Negotiation scheme calculates negotiated value with intention and preference. The simulation results showed that the proposed approach provides higher satisfaction than previous work. The difference increases if 'turn off' task's preference is higher. In the future, we plan to extend proposed approach to urban space environments where many people interact with each other.

7. Acknowledgement

This work was supported in part by MIC & IITA through IT Leading R&D Support Project

References

[1] M. Weiser, "The Computer for the 21st Century," Scientific American Special Issue on Communications, Computers, and Networks, Sep, 1991.
[2] Anind K. Dey. "Providing Architectural Support for Building Context-Aware Applications," PhD thesis, College of Computing, Geogia Institute of Technology, Nov, 2000.
[3] D. Lee, S. Han, I. Park, S. Kang, K. Lee, S. Hyun, Y. Lee, and G. Lee, "A Group-Aware Middleware for Ubiquitous Computing Environments," in Proceedings of the 14th International Conference on Artificial Reality and Telexistence (ICAT 2004), pp. 291 - 298, Seoul, Korea, Nov-Dec, 2004.
[4] L. Emmerich, C.Mascolo, "CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications," IEEE Transactions on Software Engineering, vol. 29, no. 10, pp. 929-945, Oct, 2003.
[5] I. Park, K. Lee, D. Lee, S. Hyun, and H. Yoon, "A Dynamic Context-Conflict Resolution Scheme for Group-aware Ubiquitous Computing Environments," in Proceedings of the 1st International Workshop on Personalized Context Modeling and Management for UbiComp Applications (ubiPCMM'05), in conjunction with the 7th International Conference on Ubiquitous Computing (ubicomp'05), pp. 42-47, Sep, 2005.
[6] Universal Plug and Play Forum. Universal Plug and Play Device Architecture. Version 0.91, http://www.upnp.org, March 2000.
[7] Guttman, E., Perkins, C., Veizades, J., and Day, M. Service Location Protocol, Version 2. IETF RFC 2608, June 1999.
[8] Salutation Consortium. White Paper: Salutation Architecture : Overview, http://www.salutation.org/whitepaper/originalwp.pdf, 1998.
[9] W3C. Semantic Web. http://www.w3.org/2001/sw/
[10] Broens, T., Pokracv, S., van Sideren, M., Koolwaaij, J., and Costa, P. D. Context-aware, Ontology-based Service Discovery. EUSAI 2004: 72-83.
[11] Toninelli, A., Corradi, A., and Montanari, R. Semantic Discovery for Context-Aware Service Provisioning in Mobile Environments. 9th MCMP 2005, Ayia Napa, Cyprus.
[12] Ontology. http://en.wikipedia.org/wiki/Ontology
[13] Kim, W., Kang, S., Lee, Y., Lee, D., and Ko, I. Activity Policy-Based Service Discovery for Pervasive Computing. Lecture Notes in Computer Science, Vol. 4254, pp 756-768, 2006.
[14] OWL. http://www.w3.org/2004/OWL
[15] Masuoka, R., Parsia, B., and Labrou, Y. Task Computing - the Semantic Web meets Pervasive Computin. 2nd International Semantic Web Conference (ISWC2003), 20-23 October 2003, Sanibel Island, Florida, USA.
[16] V. Kostakos, E. O'Neill, and A. Penn, "Designing Urban Pervasive Systems," Computer, vol. 39, no. 9, pp. 52-59, Sep, 2006.
[17] D. Lee, S. Han, I. Park, S. Kang, K. Lee, S. J. Hyun, Y. H. Lee, and G. Lee, "A Group-Aware Middleware for Ubiquitous Computing Environments," ICAT 2004,
[18] Protégé. http://protege.stanford.edu/