A Dynamic Context Conflict Resolution Scheme for Group-aware Ubiquitous Computing Environments

Insuk Park, Kyungmin Lee, Dongman Lee, Soon J. Hyun, and Hee Yong Yoon

***Abstract*—In this paper, we propose a dynamic conflict resolution scheme for group-aware ubiquitous computing environments. The proposed scheme incorporates users' intention as well as user preferences into conflict resolution. Conflicts are resolved differently according to the situations of involved users even in the same conflicting situation. For multi-user environments, the proposed scheme dynamically resolves the conflicts occurring between different types of context-aware applications. It allows context-conflict management to become more transparent to application programmers.**

***Index Terms*—**conflict resolution, group context, context awareness, ubiquitous computing middleware.

I. INTRODUCTION

Conflict management in context-aware computing is getting more significant attention from researchers as ubiquitous computing environments take into account multiple users [4]. For multi-user ubiquitous computing environments, conflicts among users' contexts need to be detected and resolved dynamically [1, 3]. For this, application developers or end-users specify conflicts situations, and the underlying ubiquitous computing middleware detects and resolve conflicts between applications when one of the conflict situations arises.

In dynamic conflict resolution, it is important that conflicts are resolved such that the satisfaction of the involved users with the result of the resolution is maximized as much as possible. To support this, conflicts should be resolved differently depending on the intentions of the involved users. For example, suppose that a user expects the light to be turned off when she falls asleep and the light turned on when she enters the bedroom. If a conflict arises when a user is asleep, the user will be satisfied with the result such that the light is turned on as dimly as possible. On the other hand, when a user enters the bedroom, the user will be satisfied as the light is turned on as brightly as possible.

To our knowledge, CARISMA [1] is the only approach that supports dynamic conflict resolution. It selects one of resolution choices that maximize users' satisfaction based on user preferences. Users specify how much they prefer each resolution choice in terms of QoS parameter settings for conflict resolution. However, CARISMA is designed to handle conflicts among the users of cooperative applications where the intention of all the users is assumed to be the same. Thus, it cannot effectively resolve the conflicts where the satisfaction of users with resolution results varies depending on the intention of users. In the example above, if the satisfaction of the involved users is maximized by the brightness level 1 of the light, that is, minimum brightness, then the brightness of the light is always set to 1 regardless that the user is asleep or enters the bedroom. The user is satisfied with the result when sleeping, but he/she is not when entering the bedroom.

In this paper, we propose a new conflict-resolution scheme which dynamically resolves conflicts by incorporating the intentions of the involved users as well as their preferences. Our scheme determines a resolution choice such that the intentions of the involved users are preserved as much as possible. User intentions are modeled as the value assigned to a context by the actions requested from applications on behalf of users. The differences between a resolution choice and user intentions are represented by a set of distance functions. User preferences are expressed as cost functions to reflect the level of users' reluctance to the differences between their intentions and resolution choices. Based on cost functions, a resolution is determined to minimize the reluctance of all users involved in conflicts. The conflicting applications then adapt themselves to the resolution result. We implement the proposed conflict resolution scheme as part of Active Surroundings [4], our group-aware ubiquitous computing middleware.

This paper is organized as follows. The design of the proposed conflict resolution scheme is given in Section 2. Implementation details and system architecture are described in Section 3. Section 4 discusses the advantages and the remaining issues of our approach and evaluation results. Conclusion fol-

This research is partially supported by the Ubiquitous Autonomic Computting and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program and the Digital Media Lab in Korea.

I. Park is a Ph.D. candidate in School of Engineering at Information and Communications University, Daejeon, Korea (e-mail: ispark@icu.ac.kr).

K. Lee is a Ph.D. candidate in School of Engineering at Information and Communications University, Daejeon, Korea (e-mail: kmlee@icu.ac.kr).

D. Lee is a Professor in School of Engineering at Information and Communications University, Daejeon, Korea (e-mail: dlee@icu.ac.kr).

S. J. Hyun is an Associate Professor in School of Engineering at Information and Communications University, Daejeon, Korea (e-mail: shyun@icu.ac.kr).

H. Y. Youn is a Endowed Proferssor in School of Electrical and Computer Engineering, Sungkyunkan University, Suwon, Korea (e-mail: youn@ece.skku.ac.kr).

TABLE I
POSSIBLE CONFLICTING SITUATIONS IN CONTEXT-AWARE APPLICATIONS

System	Single User		Multiple Users	
	Same App.	Diff. Apps.	Same App.	Diff. Apps.
Gaia	O	X	O	X
CARISMA	O	X	O	X
Active Sur-rouundings	–	O	O	O

lows in Section 5.

II. RELATED WORK

The definition of conflict varies from context-aware application to application as shown Table 1.

Gaia deals with conflicts that occur among simultaneously triggered rules in the same application [2]. It uses priority to resolve a conflict, that is, if there are conflicting rules, the rule with the highest priority wins. A priority-based static resolution approach is simple and powerful but its limitation has already been mentioned in other works like [5]. That is, conflicts are resolved in the same manner regardless of other contexts. The priority of each rule introduces a conflict to the application programmer at the time of designing a context-aware application. This makes the development of applications difficult since it is almost impossible for application programmers to consider all rules defined in other applications in their design. Furthermore, the conflict resolution policy in Gaia, that is, selecting the rule with the highest priority may not satisfy all users at the same time.

CARISMA proposes a runtime conflict-detecting and resolving mechanism [1]. It also defines a conflict as two or more enabled policies that occur at the same time. CARISMA provides a technique for two kinds of conflicts, i.e., a conflict within an application for a single user (intra-profile); and a conflict by cooperating applications for multiple users (inter-profile). It employs a particular type of sealed-bid auction to get a resolution policy which maximizes "social welfare" based on the QoS parameters of a service and user preferences.

However, Gaia and CARISMA cannot properly handle conflicts when multiple users are served by different applications. The proposed scheme aims to resolve a conflict in such a situation without an explicit description of the conflict. For the conflict of a single user in the same application, we just adopted the existing approach [1].

III. THE PROPOSED SCHEME

In this section, we present the proposed conflict resolution scheme. We first discuss some design issues on conflict resolution between context-aware applications. We then provide a brief overview of our context-conflict management scheme: modeling, detecting, and resolving context-conflicts between context-aware applications. Finally, we describe our conflict resolution scheme: how to model and express user preferences; how to resolve a conflict with user preferences.

A. Design Consideration

Conflicts in context-aware applications can be regarded as logical inconsistencies between users' intentions. For instance, suppose that a light in a bedroom is off since a user falls asleep. Later, another user enters the bedroom and tries to turn the light on. In this example, the latter user's intention, 'the light being on', is inconsistent with the former user's intention, 'the light being off'. To detect and resolve such conflicts, we need a model that represents user intentions.

User intentions can be captured easily by what actions applications perform on behalf of users. Existing conflict management schemes [1, 2] detect and resolve conflicts at this level. However, the approach based on application actions is difficult to handle conflicts between different types of applications. This is because the applications are likely to have different sets of actions and may not share common actions. To address conflicts between independently developed different applications, we need to model user intentions at higher level than actions such as effects on context attributes and detect and resolve conflicts [3].

In conflict resolution, user intentions should be reflected such that all the users' intentions are preserved as much as possible. In other words, a conflict resolution result should not be different from users' intentions as much as possible. For that, we need a model for measuring differences between user intentions and resolution results.

The impact of the differences on users' satisfaction may be different from one user to another. Some users are more tolerant of the differences than others. For instance, a user is insensitive to brightness when he/she is sleeping. Then the user's satisfaction decreases at a slower rate as increases brightness. That is, the user is less reluctant to brightness's difference. On the other hand, if the user is very sensitive to brightness, then the contrary is the case. Moreover, the degree of a user's reluctance varies with different context attributes. For instance, a user is sensitive to loudness while he/she is not sensitive to brightness. A conflict resolution scheme should reflect these differences.

B. Context-Conflict Management

Context-aware applications behave differently based on the current context. We assume that a context-aware application specifies what context conditions it is interested in. When one of the context conditions is met, the application is notified of that by the underlying context manager [3]. Then the application performs a corresponding task. For example, a context-aware light application turns a bedroom's light on when someone enters the bedroom or turns the light off when someone falls asleep.

An application's task is implemented through a series of actions (i.e., methods) provided by available services in the environment. Each action of a service requested by applications is interpreted as an effect on some context attributes, which is encoded in the action semantic ontology [3]. In the example above, the application requests the 'turn on' action of a light service in the bedroom when someone enters the bedroom. The

'turn on' action is then interpreted as 'increase brightness' or 'set brightness as the maximum level' in the bedroom.

Actions are monitored and conflicts among them are detected at runtime by the context manager. The manager intercepts an action request from an application and checks if that action conflicts with other actions executed by another applications. Two actions are defined in conflict if their effects on a context attribute are contradictory. The action semantic ontology has the information required to infer the contradictory effects. For example, the effects of the 'turn on' and 'turn off' actions of a light service are contradictory in that the former increases brightness while the latter decreases it.

When a conflict is detected, the context manager resolves it by determining a compromised value on the conflicting context attribute. Our scheme determines the compromised value such that the intentions of the involved users are preserved as much as possible. User intentions are modeled as the value assigned to a context attribute by the actions requested by applications on behalf of users. The differences between a compromised value and user intentions are encoded as distance functions, which represent how much one value on a context attribute is different from another. User preferences, expressed as cost functions, reflect how much users are reluctant to the differences between their intentions and compromised resolution results. Based on user preferences, our scheme minimizes the amount of reluctance of all users involved in conflicts. The involved applications behave according to the resolution result.

C. Modeling User Intentions

In modeling user intentions, we assume that user intentions are explicitly represented by actions requested by applications on behalf of users. This simplifying assumption may not be valid since an action of a service can be requested by applications with different user intentions. For example, a user would want to turn a light off when he/she either watches movie or falls asleep. With this 'turn off' action of the light service, users have different intentions. However, such fine-grained user intentions are difficult to be modeled explicitly or formally. Although we model only coarse-grained user intentions in this work, our resolution scheme can be easily applicable to fine-grained user intentions.

We define user intentions as follows:

Definition 1: (User intention). *Let A be the set of actions either executed or requested by applications on behalf of a user. Then, a set of pairs, $I = \{<c_1, v_1>, <c_2, v_2>, \ldots, <c_k, v_k>\}$ is the* **user intention** *iff:*

- *$\{c_1, c_2, \ldots, c_k\}$ is a set of context attributes that an action $a \in A$ may affect. In other words, the action a can change one of their values.*
- *For each pair $<c_i, v_i>$ in I, context attribute c_i is altered to v_i by the execution of an action $a \in A$.*

For example, suppose that a user entered a bedroom. The context-aware light application then turns on the bedroom's main light. This action makes the brightness of the bedroom level 10. He also wants to listen to music and requests a MP3 player to play music. This action will change the loudness of the bedroom to level 20. At this moment, the user's intention is represented as {<bedroom's brightness, 10>, <bedroom's loudness, 20>}.

D. Modeling Differences between User Intentions and Resolution Results

A user's satisfaction or dissatisfaction with a conflict resolution result may depend on the difference between the result and his/her intention. For instance, the intention of a user is {<bedroom's brightness, 10>}, but the brightness of the bedroom is set to 5 by the resolution of a conflict. The user may be dissatisfied with the current brightness of the bedroom (5) in proportion to the difference (|10-5|=5) between the current brightness of the bedroom and his/her intention (10).

To simplify modeling differences between user intentions and resolution results, we assume that context attributes are modeled independently for each context attribute. With this assumption, our scheme can deal with each context attribute in a user's intention one by one. Difference on each context attribute contributes independently to the overall difference between user intentions and resolution results.

Each context attribute has a distance function that gives the difference between two values on it.

Definition 2: (Distance function). *Let C denote a set of context attributes $c \in C$, let D_c be the domain (that is, value space) of c. Let R^+ be the set of nonnegative real values. For each c, a function $d_c : D_c \times D_c \rightarrow R^+$ is the* **distance function** *of the context attribute c if:*

- *$d_c(x, y) \geq 0$.*
- *$d_c(x, y) = 0$ iff $x = y$.*
- *$d_c(x, y) = d_i(y, x)$.*
- *$d_c(x, z) \leq d_i(x, y) + d_i(y, z)$.*

We assume that context attributes fall into two categories: those characterized by enumeration, and those characterized by numeric values. For each category, distance functions are differently defined.

For context attributes with an enumerated domain (for instance, a monitor's display resolution), a distance function is a matrix called *distance matrix*. A distance matrix has as its rows and columns possible (finite and discrete) values of a context attribute. Its element represents each distance between values. Diagonal = 0, symmetry,

On the other hand, for context attributes with a numeric domain (for instance, brightness or loudness), a distance function is any mathematical function satisfying conditions in Definition 2. For instance, brightness is a real value from 0 to 10. The distance function of brightness can be $d(x, y) = |x-y|$, $d(x, y) = (x-y)^2$, or $d(x, y) = e^{|x-y|}$.

TABLE II
EXAMPEL OF *USER PREFERENCE ON CONTEXTS*

	Brightness	Loundness	Display Size
User A	Very High	Neutral	Low
User B	Low	Neutral	High

E. Modeling User Preferences

User preferences are expressed formally as *cost functions*[2]. The possible values of a cost function, called *cost space*, provide a formal representation of how much users are reluctant to the difference on a context attribute. Now the problem of conflict resolution is converted into a cost minimization problem.

Formally, cost functions are defined as follows.

Definition 3: (Cost function). *Let U be the set of users and C be the set of context attributes. The* **cost function** *of a user $u \in U$ with respect to a context attribute $c \in C$ is a function $F_{u,c} : D \rightarrow [0,1]$ where D is the domain of distances of the context attribute.*

Note that the cost is a real value from 0 to 1. The cost value of one corresponds to users who would not tolerate the difference between their intentions and the compromised result. On the other hand, the cost value of zero means that the compromised result would not be different from their intentions.

F. Resolving Conflicts

We have defined user intentions, distance functions, and cost functions. Based on those models, conflict resolution in our scheme is defined as follows:

Definition 4: (Conflict resolution). *Let U be the set of users involved in a conflict. Let C be the set of context attributes relevant to the conflict. Then, the* **conflict resolution** *result is a set of compromised values, $R = \{r_c \mid c \in C\}$, such that for each $c \in C$.*

$$r_c = \arg\min \sum_{u \in U} F_{u,c}(d_c(v_{u,c}, r_c))$$

where $v_{u,c}$ is the intention value of u on c and d_c is the distance function of c.

As mentioned above, the problem of conflict resolution is deduced to a minimization problem. Note that each compromised value is computed independently.

G. Adapting Application

Definition 5: (Application adaptation). *Let a_c be an action that changes the context c. Let A is an application which consists of a set of actions $\{a_{c,1}, a_{c,2}, \ldots, a_{c,n}\}$. Then the* **application adaptation***, T is an ordered sequence of actions of A which changes the value of context c, v_c to the resolution result r_c.*

$T = \{a_{c,1}, a_{c,2}, \ldots, a_{c,k}\}$ $(k \le n)$

IV. IMPLEMENTATION

A. Case Study

In this section, we apply algebraic functions to each formal definition in the previous section to show an example. In this case study, we assume that the intention of a user *u*, I_u is represented by a single pair of a context and its value, $I_u = \langle c, v_c \rangle$. We define the distance function over the intention of a user, *c* and the resolution choices, *r* as $d_c(v_c, r) = (v_c - r)^2$. The cost function of the user *u* is given as $F_{u,c} = Pr_{u,c} \times d_c$ ($Pr_{u,c}$: preference of user *u* on a context *c*). To help users to specify their preferences, five choices are given to users which are "very high", "high", "neutral", "low", and "very low". We assign natural numbers on them from 5 to 1, respectively. Table 2 shows an example of the preference of User A and B.

Suppose that a conflict occurs because two users, U_A and U_B set the value of a certain context c to α and β $(\alpha < \beta)$ respectively, at the same time. Then, the distance functions of two users $d_{U_A,c}$ and $d_{U_B,c}$ are;

$$d_{U_A c} = (r - \alpha)^2 \text{ and } d_{U_B c} = (r - \beta)^2 \text{ where } \alpha \le r \le \beta$$

The total cost of two users, C_T weighted by user preferences, $Pr_{U_A C}$ and $Pr_{U_B C}$ is;

$$\begin{aligned} C_T &= Pr_{U_A C} d_{U_A c} + Pr_{U_B C} d_{U_B c} \\ &= Pr_{U_A C}(r - \alpha)^2 + Pr_{U_B C}(r - \beta)^2 \end{aligned}$$

Therefore, we can say that resolving the conflict is finding C_i which minimizes C_T. It is the same as the solution of the below equation.

$$\frac{dC_T}{dC_i} = 2Pr_{U_A C}(C_i - \alpha) + 2Pr_{U_B C}\beta(C_i - \beta) = 0$$

The conflict resolution r_c is;

$$r_c = \frac{Pr_{U_A C}\alpha + Pr_{U_B C}\beta}{Pr_{U_A C} + Pr_{U_B C}}$$

For example, a conflict occurs on the context, Brightness when User A turns off the light (set Brightness to 0) while User B turns on the same light (set Brightness to 10). The resolution choices are real values between 0 and 10. In this case, the resolution of the conflict is determined as below by user preferences as defined in Table 2.

$$\frac{5 \times 10 + 2 \times 0}{5 + 2} = 7.14$$

According to the result of conflict resolution, the light is dim to the brightness level given by the result. We assume that every application has enough functionality to adapt itself to the conflict resolution.

[2] User preferences are usually represented as utility functions, which indicate how much users are satisfied with something. Cost functions also can capture user preferences in terms of how much users are reluctant to something. They represent user preferences with the opposite semantics.

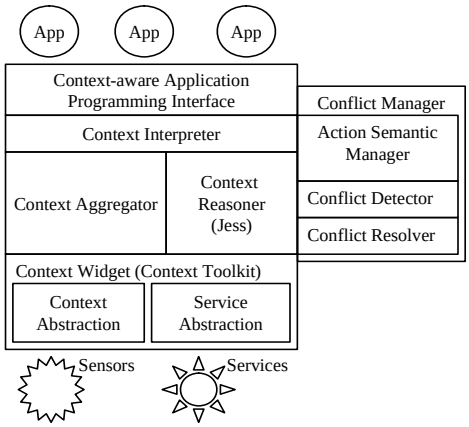

Fig. 1. The Overall Architecture for Context Awareness and Conflict Management

B. System Architecture

We implement the proposed scheme on top of the Service Interaction Broker [4], which is a communication channel between applications and services. It is a modified Java version of the Apache Axis [9] to enable dynamic adaptation. We also leverage the ContextToolkit [6], the OWLJessKB [7], and the Jess [8] on the JDK 1.5 to build context management components.

The left part of the block diagram in Figure 1 depicts the middleware components of context-awareness inspired by the ContextToolkit and the Solar [10]. Basically, we adopt the producer-consumer model [11] for gathering, aggregating, inferring, and disseminating contexts. On top of the components for context-awareness, the proposed context-conflict management scheme requires four additional components to support dynamic detection and resolution of context-conflicts between applications. As shown on the right-hand side of the diagram in Figure 1, they are the conflict manager, the conflict detector, the action semantic manager and the conflict resolver.

The conflict manager coordinates the whole process of managing context-conflicts. It consists of three steps. The first is to manage action semantics. The second is to detect context-conflicts by monitoring the action semantics. The last is to resolve them based on user preferences.

C. Context Conflict Management Procedure

Context-conflict management scheme requires four additional components to support conflict representation and dynamic detection in addition to the proposed resolution scheme. The conflict manager intercepts action invocation for capturing its information such as the name of the action, the list of arguments, and the target service before action execution. It activates the action semantics of the action through the action semantics manager in the action semantic ontology. After that, the conflict detector searches for a context-conflict caused by the action semantics activated just before. If it finds a conflict, the conflict resolver generates a new action invocation as its resolution. The action semantics activated by the old action are then invalidated and those derived by the new one are activated.

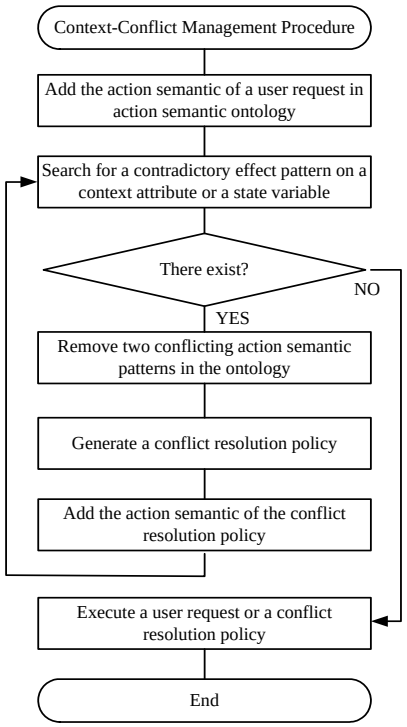

Fig. 2. Context-Conflict Management Procedure

The process is repeated until there is no more conflict in the action semantic ontology. Finally, the conflict-free action is delivered to the target service and executed. Figure 2 shows the sequence of the context-conflict management procedure for each action invocation.

V. Discussion

We assume that contexts are not correlated with each other. It makes conflict resolution simple when more than one contexts are involved in the same conflict by applying resolution scheme to them independently. Our approach needs to be extended to cover this issue.

In this paper, we show a conflict situation in which only two users are involved as an example, but the proposed scheme can be easily extended to the case of more than two users because the cost of each user's intention for conflict resolution is not related with each other.

We simply assume that the result of conflict resolution can be realized as a single application policy but, if it is not, the result can be realized with a combination of application policies. For example, if the conflict resolution says brightness has to be 7, but the light application only has *dimUp* and *dimDown*, then the resolution can be shown as the several iteration of *dimUp* or *dimDown*. It will be enhanced by the techniques of AI planning or the Semantic Web service composition in the future.

VI. Conclusion

In this paper, we proposed a dynamic context-conflict resolution scheme for resolving conflicts between different context-aware applications. It considers the intentions of the involved users as well as user preferences in the conflict resolu-

tion model. User intentions are modeled as the value assigned to a context attribute by the actions requested from applications on behalf of users. User preferences are expressed as cost functions over the distance between user intentions and the resolved value. Based on user preferences, the resolved value is determined to the one which minimizes the cost of all users involved in conflicts. The involved applications behave according to the resolution result. We plan to extend our scheme such that it resolves conflicts appropriately depending on the situation in which the involved users reside. Although we model only coarse-grained user intentions in this work, our resolution scheme can be easily applicable to fine-grained user intentions. We will deal with the more fine-grained modeling of user intentions in future work.

REFERENCES

[1] L. Capra, W. Emmerich, and C. Mascolo, "CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications," IEEE Trans. on Software Engineering, Vol. 29, Issue 10, Oct. 2003, pp. 929-945.

[2] A. Ranganathan, and R. H. Campbell, "An Infrastructure for Context-Awareness based on First Order Logic," Journal of Personal and Ubiquitous Computing, Vol. 7, Issue 6, Dec. 2003, pp. 353-364.

[3] I. Park, D. Lee, and S. Hyun, "A Dynamic Context-Conflict Management Scheme for Group-aware Ubiquitous Computing Environments," To appear in 29th Annual Int'l Computer Software and Applications Conference.

[4] D. Lee, et al.,"A Group-Aware Middleware for Ubiquitous Computing Environments" The 14th Int'l Conf. on Artificial Reality and Telexistence, 2004, pp. 291-298.

[5] J. Chomicki, J. Lobo, and S. Naqvi, "Conflict Resolution Using Logic Programming," IEEE Trans. on Knowledge and Data Engineering, Vol. 15, Issue 1, 2003, pp. 244-249.

[6] A. K. Dey, G. D. Abowd, and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," HCI Journal, Vol. 16, Issue 2-4, 2001, pp. 97-166.

[7] The OWLJessKB, "http://edge.cs.drexel.edu/assemblies/software/owljesskb/," 2004.

[8] E. Friedman-Hill, *Jess in Action: Java Rule-Based Systems*, Manning Publications, Greenwich, 2003.

[9] The Apache <Web Services/> Project, "http://ws.apache.org/axis," 2003.

[10] G. Chen and D. Kotz "Solar: A pervasive-computing infrastructure for context-aware mobile applications," Dartmouth College Technical Report, TR2002-421, 2002.

[11] T. Zimmer, "Towards a Better Understanding of Context Attributes," Proc. of 2nd IEEE Int'l Conf. on Pervasive Computing and Communications, March 2004, pp. 23-28.

Insuk Park received the B.S. degree in computer engineering from Kyungpook National University (KNU), Korea, in 2000 and the M.S. degree in engineering from Information and Communications University (ICU), Daejeon, Korea, in 2002.

He is currently a Ph.D. candidate at ICU. His research interests include context management and data management in ubiquitous computing.

Kyungmin Lee received the B.Eng. degree in mechanical engineering from Pohang University of Science and Technology (POSTECH), Korea, in 2000 and the M.S. degree in engineering from Information and Communications University (ICU), Daejeon, Korea, in 2003.

He is currently a Ph.D. candidate at ICU. His research interests include dynamic reconfiguration and seamless communications in ubiquitous computing.

Dongman Lee received the BS degree in Computer Engineering from Seoul National University, Korea in 1982, and the MS degree and Ph.D. degree in Computer Science from KAIST, Korea in 1984 and 1987, respectively.

From 1988 to 1997, he worked as Technical Contributor at Hewlett-Packard. He is currently Professor in School of Engineering at Information and Communications University (ICU), Daejeon, Korea. He is also Associated Director of Digital Media Laboratory, ICU. He has been actively participating in Korean Internet address and name committee since 1998. He received a Prime Minister Award as the recognition of the advancement of the Korean Internet in 2000. His laboratory, Collaborative Distributed Systems and Networks Lab, has been appointed as National Research Laboratory in 2001. He has published more than 50 papers in international journals and conference proceedings. His research interests include distributed systems, computer networks, mobile computing and pervasive computing.

Dr. Lee is a member of IEEE Computer Society, IEEE Communication Society, and ACM.

Soon J. Hyun received the BS degree in electrical engineering from the Kyungpook National University, Korea; the ME degree in electrical engineering from the Katholike Universitate Leuven, at Havelee, Belgium; and the PhD degree in electrical and computer engineering from the University of Florida, in 1981, 1987, and 1995, respectively. His PhD dissertation was on parallel query processing in objectoriented temporal database systems.

He is an Associate Professor of the School of Engineering, Information and Communications University (ICU), Korea. From 1983 to 1997, he worked with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea, where he was leading a digital library development project in an effort to build the Korea National Information Infrastructure. From 1984 to 1986, he was a visiting member of the research staff at Bell Telephone/ITT (presently, Bell/Alcatel), Antwerp, Belgium, where he worked on the development of telecommunications network protocol systems. His recent research interests include context management, ubiquitous computing, temporal database management, multimedia databases and knowledge-based information search over information highway, and digital library systems, and applications development.

Hee Yong Youn received the BS and MS degree in electrical engineering from Seoul National University, Seoul, Korea, in 1977 and 1979, respectively, and the PhD degree in computer engineering from the University of Massachusetts at Amherst, in 1988.

From 1979 to 1984, he was on the research staff of Gold Star Precision Central Research Laboratories, Korea. He had been Associate Professor of Department of Computer Science and Engineering, The University of Texas at Arlington until 1999. He was also a faculty of School of Engineering, Information and Communications University, Daejeon, Korea from 1999 to 2000. He is presently Chairperson (Endowed) of Computer Engineering Dept., School of Information and Communications Engineering, Sungkyunkan University, Suwon, Korea. His research interests include storage system, distributed computing and networking, Internet and mobile computing, and fault-tolerant computing. He has published more than 150 papers in int'l journals and conference proceedings, and received Outstanding Paper Award from the 1988 International Conference on Distributed Computing Systems, 1992 Supercomputing, and 2001 Korean Society Internet Information Spring Symposium, respectively. He also served as a lecturer of the ACM Lectureship Series from 1993 to 1997.

Dr. Youn is a senior member of the IEEE Computer Society.