Context-based User Profile Management for Personalized Services

Youngjung Suh, Dongoh Kang and Woontack Woo

***Abstract*— Frameworks for context-aware services in a ubiquitous computing environment have been developed so far to allow the application services in the environment to monitor a sequence of patterns in user behavior as well as to manage the aggregated user profiles in one server. However, it is a great burden for the whole environment to attend to the individual desires of the users. Thus, we propose a user profile management framework in which the user's desires, needs, and preferences for services are managed through a wearable personal station, a terminal device worn by users. We exploit service-specific user preferences by describing them as context to provide personalized services. Also, we dynamically update the preferences by learning them. The details such as service properties or contents can be adaptively changed according to the learning results. The proposed framework can be widely exploited for ubiComp-enabling applications which require the interaction between a user and an environment. Moreover, it may support personalized services while ensuring user's privacy.**

***Index Terms*— context-aware, learning, personalized service, user profile**

I. INTRODUCTION

Various kinds of context information are flowing in ubiquitous or wearable computing environments with a wide range of networking, computing, and distributed contents and services. Over the last few years, research activities on context and context-aware have been reported [1]. In these researches, there have been mentioned several kinds of contexts such as environment contexts, user contexts, computing resource contexts. Among those kinds of contexts, user-related context information is required to provide personalized services. Recent studies have made a great attempt to develop frameworks describing and utilizing various kinds of user-related context information to support personalized services. ClixSmart Navigator [2] implemented a Navigation Server between a content store and a WAP gate-way. As a similar system, Intelligent Software Agents Group of computing science department in University of Aberdeen developed a recommendation system called RECO [3] which aims to facilitate the use of context-aware services in a dynamic environment where users can access a variety of services from different locations. Meanwhile, there also have been researches on the user profile data model to support context-aware service in the field of ubiquitous computing [4, 5, 6]. These are established through a Web interface. KDDI cooperation in Japan has been developing a framework for constructing common profile model and managing the personalized profile of user activities from mobile terminal devices [7]. In aggregating, updating and disseminating user profile, they implemented the framework as a client-server architecture in which Profile Aggregator called "Home Server" collects user profiles from a number of desktop PCs.

These approaches to researches on provision of context-aware services exploiting well-known user profiles have some limitations in that, they have the environment keep continuously monitoring a sequence of patterns in user behavior to provide users with context-aware services. They did not take user-centric context-aware service into consideration. In other words, in ubiquitous or wearable computing environments, users are likely to prefer spontaneously to disseminate their personal information within the desired situation rather than be monitored by the environment for enjoying context-aware services. Besides, those systems are restricted just to the level of recommending the expected user-desired service in sequence. That is, they do not support the personalized services in which the details such as service properties or contents are able to be adaptively changed according to a user.

To overcome limitations above, in this paper, we propose a framework in which wearable personal station (WPS) with a user and application services in an environment are the distributed independent entities, respectively, to interact with each other just when it requires exchanging a context between them. For this to result, we developed the mechanism to support providing context-aware services without a centralized home server. It can reduce a great burden for the whole environment to attend to the individual desires of the users by managing the user's desires, needs, and preferences for services through a WPS. Moreover, we exploit service-specific user preferences by describing them as context to provide personalized services. Then, we dynamically update the preferences by learning them. The details such as service properties or contents can be

Manuscript created June 17, 2005. This work was supported by SeonDo project of MIC, Korea

Youngjung Suh is with Gwangju Institute of Science and Technology, Gwangju, 500-712, S.Korea (+82-62-970-2279, e-mail: ysuh@gist.ac.kr)

Dongoh Kang is with Electronics and Telecommunication Research Institute, DaeJeon, 500-712, S.Korea (+82-42-860-5012, e-mail dongoh@etri.re.kr)

Woontack Woo is with Gwangju Institute of Science and Technology., Gwangju, 500-712, S.Korea (corresponding author to provide phone: 82-62-970-2226, fax:62-970-2249, e-mail: wwoo@gist.ac.kr)

adaptively changed according to the learning results. Especially, in the proposed framework, it is considered for a user to take care of his personal information protecting his right to privacy. That is, users do need to distribute their profile only when they indeed want to enjoy services.

This paper is organized as follows. In section 2, related works are explained. We describe User Profile Management Framework in section 3. And the method for user profile management is explained in section 4. Section 5 deals with experiments. Finally, we conclude our work and give a brief remaining work in Section 6.

II. RELATED WORKS

Over the last few years, research activities on context-aware services have been reported [2]. These earlier studies mainly concentrated on the development of application-specific systems. However, those researches are gradually putting their focus on the infrastructure for ubiquitous computing, according as context-aware applications require the well-organized environment in which sensing, processing, and networking techniques could be exploited in effect. Context toolkit [3] provided a toolkit for framework for managing sensor-based context information to support rapid prototyping of context-aware services. Recent studies have made a great attempt to develop frameworks describing and utilizing various kinds of user profiles to support personalized services. ClixSmart Navigator [4] implemented a Navigation Server between a content store and a WAP gate-way. It automatically learns about users' preferences as they browse and personalizes each browsing experience so that relevant options are more readily accessible. In addition, it has the ability to manually modify and customize the mobile portal through the web. Ultimately, the end-user receives the right information, at the right time, every time, putting an end to user frustration. As a similar system, Intelligent Software Agents Group of computing science department in University of Aberdeen developed a recommendation system called RECO [5] in which aims to facilitate the use of context-aware services in a dynamic environment where users can access a variety of services from different locations. In RECO, they replace the standard browser with an agent-based client which allows the dispatch of recommendation messages without an explicit request from the user. At the core of the RECO system is a multi-agent architecture which provides the agents that are needed in order to monitor user behavior and make recommendations. Sequence patterns in user behavior discovered by AprioriAll [6] and context is exploited in establishing a user model. Thus, by using user preference profiles, it creates and learns the rules to justify validity of recommendation at a certain situation which can be described by time and location. The traditional software agent's approaches to researches on provision of context-aware services exploiting well-known user profiles have some limitations in that, they have the environment keeping continuously monitoring a sequence of patterns in user behavior to provide users with context-aware services such as recommendation services of the expected user-desired services. They did not take user-centric context-aware service into consideration. In other words, in ubiquitous or wearable computing environments, users are likely to prefer spontaneously to disseminate their personal information within the desired situation rather than be monitored by the environment for enjoying context-aware services. Moreover, they did not apply service-specific user preferences to provide the personalized services in which the details such as service properties or contents can be adaptively changed according to a user.

Meanwhile, there also have been researches on user profile data model to support context-aware service provision in the field of ubiquitous computing. Ubisworld [7] proposes a basic semantics and RDF-based data structure for representing property attributes regarding physical objects (persons and objects), spatial and temporal information, and user activities through a web-based interface. The Haystack [8] is developed by the Haystack Project at the MIT Laboratory for Computer Science and Artificial Intelligence. Haystack is a Semantic Web browser that aggregates RDF from multiple arbitrary locations and presents it to the user in a human-readable fashion, with point and click semantics that let the user navigate from one piece of Semantic Web data to other, related pieces. SOUPA Project [9] defines the shared ontology, regarding beliefs, desires and intentions, time, space, events, user profiles, actions and policies for security and privacy. These are established through a Web interface. Not to be based on only web-based input data so that an adaptive update method for collecting user related information from diverse sources could be realized, studies regarding middleware and infrastructure support for context-aware systems (e.g. Context Fabric [10] and GAIA [11]) have also been attempted. CoBrA [12] is a broker-based agent framework for building smart spaces, e.g., smart meeting rooms, and intelligent homes via SOUPA ontology mentioned above. As an illustration of these, KDDI cooperation in Japan has been developing a framework for constructing common profile model and managing the personalized profile of user activities from mobile terminal devices [13]. In aggregating, updating and disseminating user profile, they implemented the framework as a client-server architecture in which Profile Aggregator called "Home Server" collects user profiles from a number of desktop PCs.

However, in ubiquitous/wearable computing environments, it should be reasonable that a user maintains their own personal information, i.e. user profile while application services has only to provide users with context-aware services just by exploiting user-related information distributed by the user. At the same time, wearable personal station (WPS) with a user and application services in an environment are the distributed independent entities, respectively, to interact with each other just when it requires exchanging a context between them. For this to result, there needs the mechanism to support providing context-aware services without a centralized home server. As mentioned above, frameworks for providing context-aware services in ubiquitous computing environments have been developed so far to allow the application services in the environment to keep monitoring a sequence of patterns in user behavior as well as manage the aggregated user profiles in one server. Especially, in wearable computing environments, it is reasonable for a user to take care of his personal information protecting his right to privacy. On the other hand, it is a great burden for the whole environment to attend to the individual desires of the users. Thus, it is necessary that the user's desires, needs, and preferences for services be managed through a wearable personal station which they carry on their hands. Also, users can enjoy personalized services by selectively disseminating their private information while ensuring their privacy. That is, users do need to distribute their profile only when they indeed want to enjoy services. In this paper, we propose the framework to effectively manage user profiles which contain user activity, demands and service-specific preferences in order to provide personalized services to users in ubiquitous or wearable computing environments.

III. USER PROFILE MANAGEMENT FRAMEWORK

A. *Context Model*

To aware context information from user's activities in a daily life, a unified context is needed to create context for triggering services users want by analyzing and integrating information obtained from various kinds of sensors. Context used in the proposed framework is defined as a flexible but unified context describing a user's situation to trigger application services. In other words, it describes user's situation as who, when, where, what, how, and why (5W1H) and is shared between sensors and services [14]. We define different kinds of context, i.e. preliminary, integrated, conditional, and final context. The preliminary context generated from sensors is not enough to trigger an appropriate service. In other words, the preliminary context from a sensor may not be accurate or even incomplete since a sensor may not fill up all 5W1H, in general. Thus, we define integrated context, conditional context, and thus final context. The integrated context is obtained by integrating preliminary contexts from a set of sensors. We determine the final context which contains a set of parameters and a service function to be used to trigger a user-centered service. As a result, an application developer may easily develop context-aware application by specifying the condition of service to be triggered as a 5W1H of conditional context.

B. *wear-UCAM*

ubi-UCAM [14],Unified Context-aware Application Model, for providing context-aware services in a ubiquitous computing environment have been developed so far to allow the ubiServices [15] in the environment to manage all user conditional contexts describing user's desires on a particular service. In addition, Interpreter in each ubiService provides graphical user interface to have users reflect their service-specific preferences by setting up user conditional context. To alleviate a great burden for all application services in the whole environment to care for the individual desires of the users for the application services, we came to propose wear-UCAM [16], Unified Context-aware Application Model for Wearable Computing.

Especially, in wear-UCAM, the point that we would like to emphasize is to allow users to take care of their personal information while protecting their privacy. Thus, we make it possible that the user's desires, needs, and preferences for services are managed through a WPS. wear-UCAM is a framework which offers personalized services to users according to service-specific user preferences by analyzing various kinds of contexts obtained from ubiSensors in an environment or wearSensors worn by users. The Fig. 1 shows the overall architecture of wear-UCAM.

The main features of wear-UCAM are as follows.

1) updating user profile based on User Conditional Context,

2) obtaining high-level context analyzed through primitive context extracted from biological sensors which users have in wear, and

3) protecting user's right to privacy from an environment. Although wear-UCAM leverages the communication between wearSensors and wearServices by utilizing unified context as same in ubi-UCAM, it is able to be distinguished from ubi-UCAM in that privacy protection can be supported by exploiting a user profile management technique in a WPS. Ultimately user profile management framework can be cooperated with ubi-UCAM.

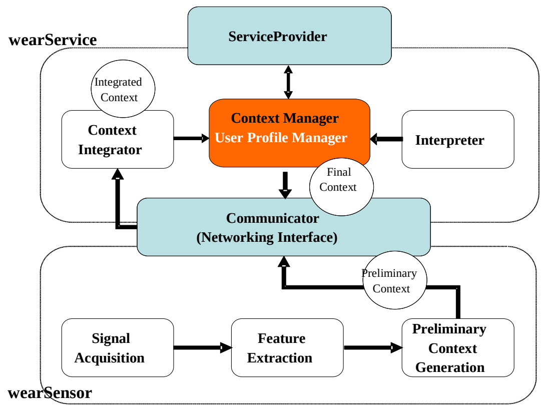


Fig. 1. wear-UCAM Architecture

C. *Service Scenarios*

We has implemented 'ubiHome'[15], a testbed for applying ubiComp enabling technologies to home environments. In ubiHome, a variety of sensors are equipped pervasively in order to generate preliminary contexts about residents, and personalized services are provided by exploiting the resident's context. To show the usefulness of the proposed framework, we show the sample service scenarios in ubiHome. The unified ubiHome application service incorporates ubiTV, cMP, cMail Checker, etc. which are already developed. Organized interaction between application services is ensured by intelligent context-awareness of ubi-UCAM and wear-UCAM.

For example, while sitting on a couch watching movie, a user can observe outsider scene through camera monitoring. Also, he can check his E-mail, and send reply using a PDA.

Context-based Music Player

The Context-based Music Player provides user-centered services based on the context such as user's identity (Who), user's location (Where), time (When), body condition (How), object for music player (What) and user's stress level (Why). For example, after a resident enters a living room, he sits on a sofa in front of the TV. Then, an ubiService menu automatically pops up on the monitor. If the resident selects music player from the menu, Context-based Music Player plays a list of music titles according to the service history of the user. Also, it can automatically play the music which relieves the user from a tension if it is identified that he is very stressful through the analyzed context on user's body condition obtained from biological sensors.

IV. USER PROFILE MANAGEMENT METHOD

The one of the important components for context-aware application model in a wearable computing environment is User Profile Manager (UPM). It is an extensible and featured version of Context Manager in ubi-UCAM, as shown in Fig. 2. The basic role of UPM is to generate Final Context comparing between Integrated Context and Conditional Context. Firstly, UPM gets Integrated Contexts from Context Integrator. Then, it compares Integrated Context with Conditional Context which consists of User Conditional Context and Service Conditional Context. At last, it generates Final Context and sends it to wearSP. The featured role of UPM is to learn the preferences for services of the user so that it can support the personalized service. Specifically, it generates UCCs and disseminates them into the environment while automatically updating them.

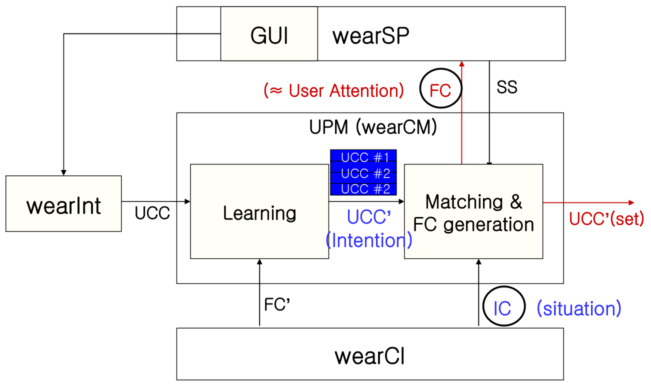


Fig. 2. User Profile Management in wearService

A. Context-based User Profile Description

User profile can be categorized into two according to its characteristics: one is static, the other is dynamic. Static user-related information is personal information such as name, age, address books, etc. Also, information which a user can initially set as his service-specific desires through Graphical User Interface (GUI) offered by Service Provider is considered as static. In case of the dynamic user profile, there can be the meaningful context information integrated from preliminary contexts describing user's biological conditions such as stress level, attention status, etc. These are obtained from biological sensors which users have in wear. Next, user's feedback information on services, service-specific user preferences, is likely to be changed dynamically. wear-UCAM shares unified context with ubi-UCAM. Thus, user-related information, user profile, is systematically and extensively described as each field of Context5W1H in the unified context [14]. Description of user profile with Context5W1H is as follows.

Basically, 'Who' comprises of user's identity, characteristics, and relationship. Identity can be used to authenticate him as unique in his group. Characteristics describes user's attributes or physical features. Relationship is important to contain priority information to access services in user's group as home and office.

'Where' consists of user's indoor or outdoor location. Especially, there exists a field to describe the body position where biological sensors are attached.

'What' is information of sensors which users attach on their body and services which users want. There are two subentries of 'ServiceWhat': one is service identity, the other is service attribute. A specific service that users want to enjoy is identified by the entry of service identity. The entry of service attribute includes useful information to provide a proper service to users: which type of service the users like, which contents the service is able to offer to the users, which object users are attending on, etc. 'SensorWhat' identifies a specific sensor among sensors which are attached on user's body.

'When' is time information generated when a user has an access to services. It represents the time biological signals are converted into preliminary context. It is also utilized to manage user's history on usage of services. In addition, it is an important context in that it can describe time information users get to have the attention on a specific object.

'How' represents body conditions or gestures of a user. The body condition information consists of EMG (electromyogram), BVP (blood volume pressure), heart rate, GSR (galvanic skin response), respiration, and temperature. These are able to be obtained by analyzing raw data extracted from biological sensors. The gestures contain body motion, position of hands and movement of legs.

'Why' consists of user's intention, and emotion. Intention is the user's will to access a service with explicit or implicit commands. In case of user's will with implicit command, it can be inferred by using the other context information. Emotion can describe a user's mental conditions but is difficult to do, in general.

B. *User Profile Handling*

1) *Context Aggregation*

As mentioned before, user profile can be categorized into two according to its characteristics: one is static, the other is dynamic. Static user-related information is personal information such as name, age, address books, etc. Also, information which a user can initially set as his service-specific desires through GUI offered by Service Provider can be considered as static. These static user profiles are described in 5W1H of User Conditional Context (UCC), 'Who' being as a focal factor. Then, the dynamic user profiles are described in Context5W1H of Final Context (FC) since Final Context contains the results of service execution which a user desires. Thus, we can acquire the service-specific user preferences which are dynamically changed by collecting Final Context from ubiServices and wearServices. Integrated Context (IC) is also one of meaningful user profile describing user's biological conditions such as stress level, attention status. Finally, after learning and matching contexts, we can update User Conditional Context which reflects user's indirect intention to enjoy a specific service, so that users can be provided the personalized services. Fig. 3 shows the user profile construction module.

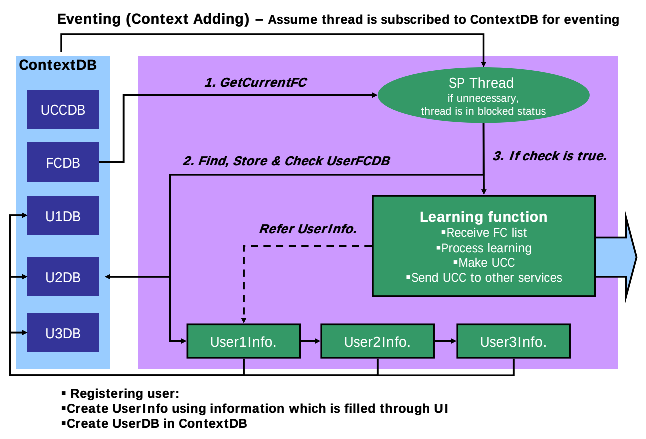

Eventing (Context Adding) – Assume thread is subscribed to ContextDB for eventing
ContextDB
UCCDB
FCDB
U1DB
U2DB
U3DB
1. GetCurrentFC
SP Thread
if unnecessary, thread is in blocked status
2. Find, Store & Check UserFCDB
3. If check is true.
Refer UserInfo.
Learning function
•Receive FC list
•Process learning
•Make UCC
•Send UCC to other services
User1Info.
User2Info.
User3Info.
▪ Registering user:
•Create UserInfo using information which is filled through UI
•Create UserDB in ContextDB

Fig. 3. User Profile Construction Module

Procedures of user profile construction in UPM are as follows. At first, UCC from Interpreter is stored in UCC Database which consists of a primary and secondary database. At an initial point, users can provide their personal specifics such as name, favorites, schedules through GUI in WPS just like PDA. It is descried as UCC and maintained in a secondary database. At an arbitrary point, UCCs for users to provide their explicit demands through GUI can be stored in a primary database with a higher priority than a secondary one. UPM aggregates FCs in FC Database at a regular interval (T). Every interval (T), it gets FCs from FC database and feeds them into input module of Neural Network as a training data set.

2) *UCC generation and dissemination*

After a proper training, inference network is resulted in. Inference network plays a role in learning service-specific user preferences. Learning results are exploited as a service execution history of the users. Then, IC coming from Context Integrator is provided as input data for inference network. Inference network associates the inputs with specific service categories or service parameters, that is, categories the user's 4W1H context to several service categories ('what'). It produces a predictable data (prediction context) of 4W1H along with the induced 'what'. Then, UPM makes a comparison between the resulting prediction context and UCC in UCC database. In the first step, it compares between UCC (user's explicit demands) in a primary database and the current prediction context. If there is the consistent context, UPM sends the current prediction context to FinalContextGenerator. If there is no consistent context, it goes to the second step of comparison with UCC (learning results) in a secondary database and the current prediction context. If there is the consistent context in the second step, UPM sends the current prediction context to FinalContextGenerator. If there is also no consistent context in the step, UPM updates a secondary database with the current prediction context, send the current prediction context to FinalContextGenerator, and disseminate it through a network to services in the environment. Fig. 5 illustrates the process of generating UCCs and disseminating them into the environment through a network.

The brief mention on the procedure of UCC generation and dissemination is as follows. First of all, we construct a primary database in UCC Database from static user-related information provided through GUI. Then, Final Contexts are aggregated at a periodic interval and stored in FC Database. At a regular pace, we analyze and learn the dynamic user-related information from Final Contexts. At last, through context matching process, we make a new User Conditional Context, update a secondary database in UCC Database with it, and send it to other ubiServices and wearServices.

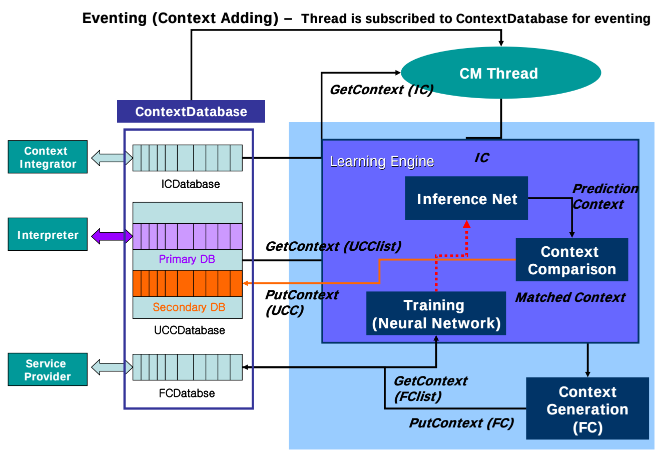


Fig. 4. User Profile Handling Module

UCCs disseminated through a network can be exploited for several ubiServices to match with ICs from the environment for triggering the corresponding service. Fig. 5 shows the interaction between UPM in wearService and ubiServices in the environment. UCCs come into usage for ubiServices to provide a user with their contents. Since UPM makes UCCs which reflect the user's preferences that suit him best, it can play a key role in providing personalized services in the environment.

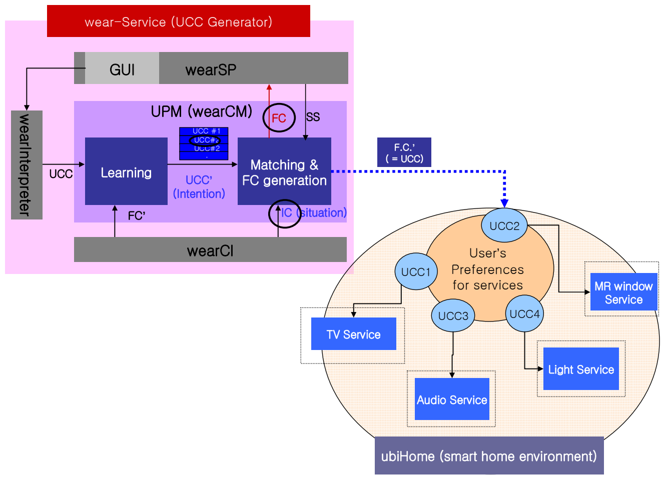


Fig. 5. Interaction between User Profile Manager and services in ubiHome environment

C. *User Profile Update using Neural Network*

Although users new in a home environment can provide their personal information and service-specific preferences through GUI offered by Service Provider in wear-UCAM, it is needed to automatically update user's service-specific preferences when user's command is not explicitly provided. And what if a user wants to enjoy a service which he has not initially set his preferences on it through GUI? In case of these situations, it is also needed to learn the user's behavior patterns to infer service-specific preferences, so that results of learning can be reflected into the dynamic update of user profile.

In updating user profile, learning engine expects the appropriate service for given input. The input represents the current situation of user's surroundings. We applied Neural Network to the learning engine. The context can be the input patterns. Users can determine the input patterns using GUI, for service which he wants to be provided. And they also can be generated from both Integrated Contexts and the collected Final Contexts, in the form of 5W1H. And then, learning engine outputs the proper result which represents service information. The history of service-specific user preferences is stored in the database for learning change of user's demands. Learning engine analyzes the history and determines new rules according to the learning result through a neural network. For learning user preferences, a Multiple-Layer Perceptron (MLP) neural network is used [17]. It is shown in Fig. 6.

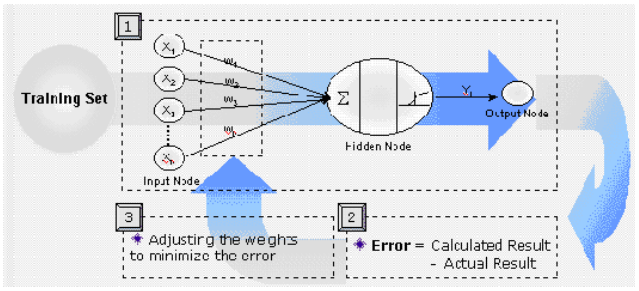

Training Set
1
Input Node
Hidden Node
Output Node
3
Adjusting the weights to minimize the error
2
Error = Calculated Result - Actual Result

Fig. 6. MLP with Backpropagation

It has long been regarded that neural networks might provide the established basis for approximating any (linear or nonlinear) function. In particular, MLP might provide the most valid way to map any nonlinear relation, given sufficient neurons in the hidden layers. Neural network is composed of simple neurons operating in parallel. These neurons are inspired by biological nervous systems. The network function is determined by the connections between neurons. We can train a neural network to perform a particular mapping function by adjusting the values of the connections (weights) between neurons. By training neural networks, we can enable a specific input to lead a specific target output. There, the network is adjusted, based on a comparison of the actual output and the target output, until the actual output from a network is consistent with the target one. Typically lots of input and target pairs are used, in this supervised learning, to train a network. The learning algorithm used in this system is the popular backpropagation of which the direction of adjusting weights is the reverse direction of processing data as shown in Fig. 6. If it is carefully trained, neural network may induce generalization beyond the training data to produce approximately correct results for new cases that were not exploited for training. Neural network is useful for solving non-linear problems and it takes less time and cost than other algorithms.

V. EXPERIMENTS

We have implemented 'ubiHome [18]', a testbed for applying ubiComp enabling technologies to home environments. In ubiHome, various sensors and services are equipped being pervasive. Sensors generate preliminary contexts describing residents and their surroundings, and personalized services are offered by exploiting the resident's context. Each sensor is regarded as a smart sensor as it is individually connected to a PC and does work with inherent processing, networking, and sensing abilities. We performed the experiments by using resources in ubiHome.

For a dynamic update of user profile, we mentioned that we applied a neural network to the learning engine of UPM. The neural network architecture in Fig. 7 consists of three layers; input layer, hidden layer, and output layer. We use a set of contexts as input because contexts express user's behavior pattern. If the number of elements of each context is E_N and the number of context is C_N, then the total number of input neurons is equal to $E_N \times C_N$. The number of neurons in the hidden layer is selected by experiments and the number of neurons in the output layer is the number of categories in services to be provided to the users.

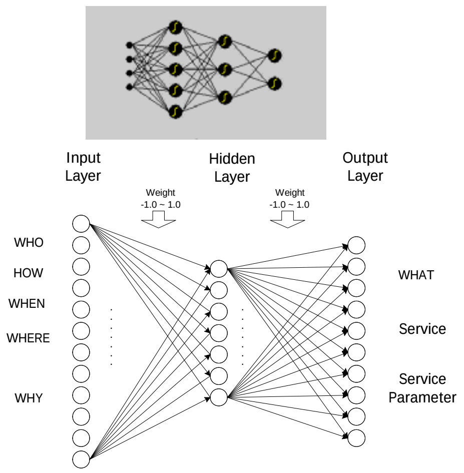


Fig. 7. Organization of MLP

The input and output pairs of training pattern are fed to the neural network because the supervised learning is exploited. The adopted MLP is performed in two steps, i.e. learning and then mapping. First, MLP learns the nonlinear relationship between the context information (3W1H) and the user's intention ('what') through examples, rather than complicate behavior pattern analysis and mapping. The Fig. 8 shows the procedures of network training, context inferring, and context matching.

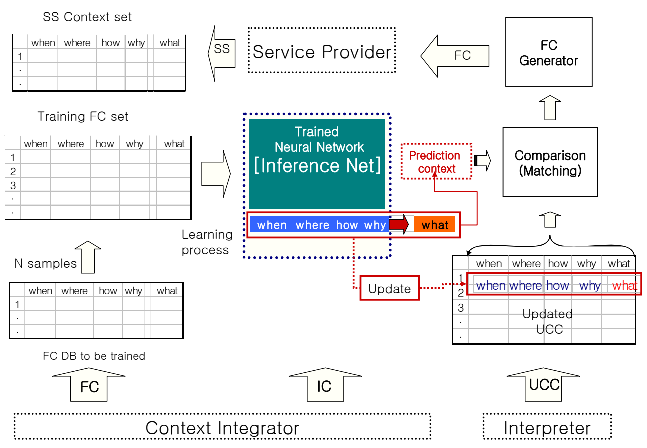


Fig. 8. Context Inferring and Matching

The procedure of training the network is as follows: Firstly, the input data are obtained with the form of 3W1H from FC database. After getting the context (3W1H), we apply the neural networks (NN) to induce an intention of a user from input contexts. In the neural network, each input vector is converted into the numerical value. These numerical input values are the input to the hidden layer for processing. In this processing, weights and biases in networks are adjusted. It compares the trained results and the given target data. If they are identical, it puts an end to training networks. If they are not, it adjusts networks again until the trained result is consistent with the target data. After proper training, the MLP associates the input vectors with specific service/service parameters category, i.e. categorizes the user's contexts to several service categories. In this framework, a set of user's context (i.e. when, where, how, and why) are mapped directly to a set of symbolic representation of services i.e. {TV.On/Off, Audio.On/Off, TV.On.Comedy, etc.}, as input and output vectors of the MLP. Fig. 9 shows one mapping sets of each layer in MLP. In the input layer, as shown in this figure, only one neuron is assigned to each element of 3W1H context, i.e. 'when', 'where', 'how', 'why'. Also, in the output layer, only one neuron is assigned to 'what' context.

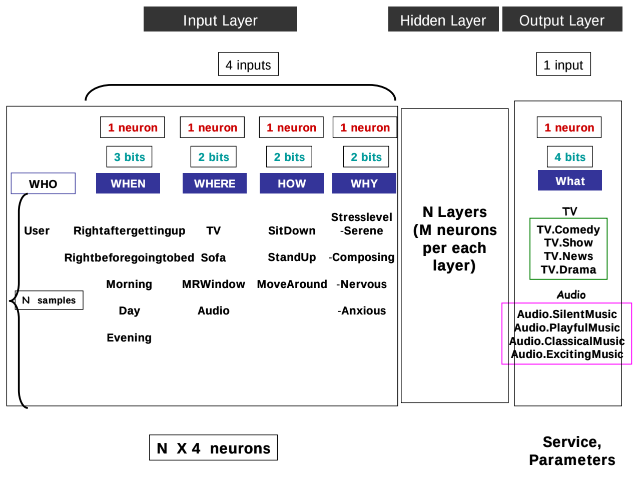
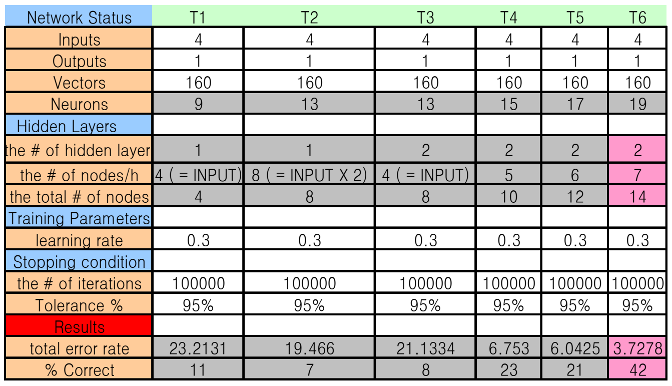


Fig. 9. 1st Data sets of each layer in MLP

Note that alternatively, the proposed UPM can be used to make the analysis procedure even simpler. For example, without extracting directly user's status and desire from physical sensors, context information {when, where, how, why} obtained from context-aware application model can be directly connected to the input layer of NN. Each context is denoted by numerical values, which stand for where the user is, when he is observed, which gesture he takes, and what his stress level is. Thus, these contexts can be trained to be mapped to the proper output categories. We described the experimental result in more detail as follows.

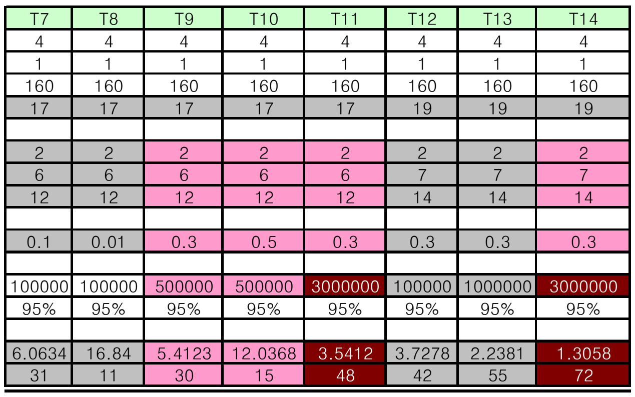
We indicate inputs, outputs, vectors, and neurons as Network Status in Fig. 10. In addition, hidden layer topology, training parameters, stopping condition and results are presented above. There is learning rate as training parameter. The number of iterations and error tolerance are included in stopping conditions. We can find that as the number of hidden layers is increased, the total error rate is decreased, other things being equal, as shown in Figure 10-a. Also, as the number of nodes within the hidden layer is increased, the total error rate is decreased. When the total number of nodes is fourteen in two hidden layers, the accuracy is at around 42%, which shows best performance with same experimental condition.



Network Status	T1	T2	T3	T4	T5	T6
Inputs	4	4	4	4	4	4
Outputs	1	1	1	1	1	1
Vectors	160	160	160	160	160	160
Neurons	9	13	13	15	17	19
Hidden Layers						
the # of hidden layer	1	1	2	2	2	2
the # of nodes/h	4 (= INPUT)	8 (= INPUT X 2)	4 (= INPUT)	5	6	7
the total # of nodes	4	8	8	10	12	14
Training Parameters						
learning rate	0.3	0.3	0.3	0.3	0.3	0.3
Stopping condition						
the # of iterations	100000	100000	100000	100000	100000	100000
Tolerance %	95%	95%	95%	95%	95%	95%
Results						
total error rate	23.2131	19.466	21.1334	6.753	6.0425	3.7278
% Correct	11	7	8	23	21	42

Fig. 10 – a . Learning Results with Network status (adjusting the topology of hidden layer)

In Figure 10 – b, it is shown that as we increase the number of iterations, others being equal, the total performance is improved, gradually. Comparing between 12 nodes and 14 nodes in two hidden layers with 3,000,000 iterations, latter is better than former. T14 is found as the best one of results which we got from this experiment.



T7	T8	T9	T10	T11	T12	T13	T14
4	4	4	4	4	4	4	4
1	1	1	1	1	1	1	1
160	160	160	160	160	160	160	160
17	17	17	17	17	19	19	19
2	2	2	2	2	2	2	2
6	6	6	6	6	7	7	7
12	12	12	12	12	14	14	14
0.1	0.01	0.3	0.5	0.3	0.3	0.3	0.3
100000	100000	500000	500000	3000000	100000	1000000	3000000
95%	95%	95%	95%	95%	95%	95%	95%
6.0634	16.84	5.4123	12.0368	3.5412	3.7278	2.2381	1.3058
31	11	30	15	48	42	55	72

Fig. 10 – b. Learning Results with Network status (adjusting learning rate and iteration number)

Fig. 11 shows another mapping sets of each layer in MLP. In the input layer, as shown in the figure, the different number of neurons is assigned to each element of 3W1H context, i.e. 'when'(5), 'where'(4), 'how'(3), 'why'(4). In the output layer, 17 neurons are assigned to 'what' context. That is, the number of neurons in each input data is the number of possible samples of each element in 3W1H context. The number of neurons in the output layer is the number of target samples of elements in 'what' context.

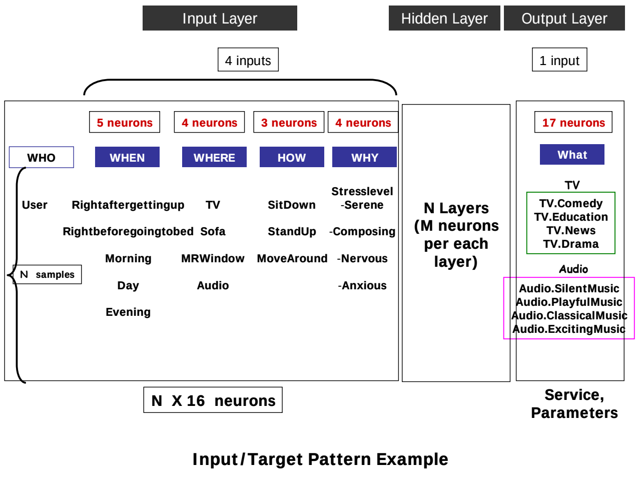
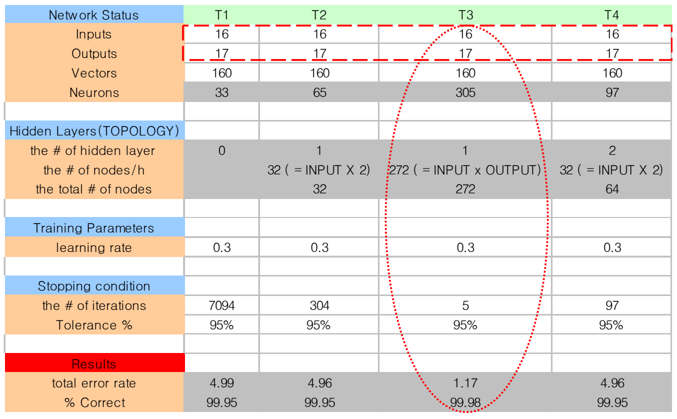


Fig. 11. 2nd Data sets of each layer in MLP

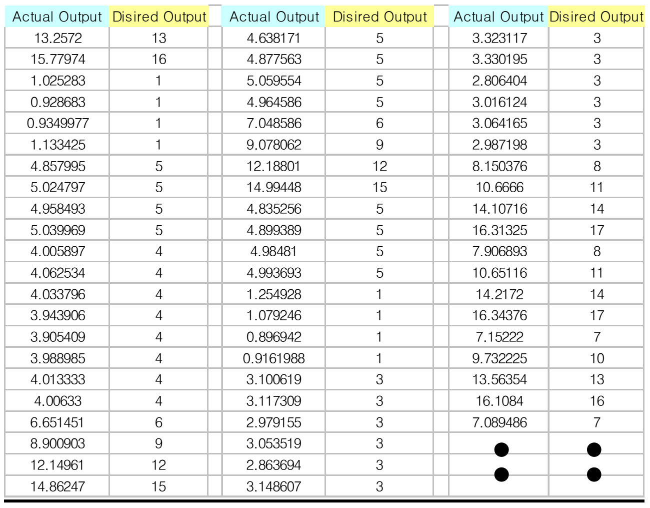
Fig. 12 represents the learning results with network status when 2nd data sets of each layer in MLP are exploited. In this experiment, we tried to observe how fast the network is well trained in reaching around 99% of accuracy. As shown in Fig. 12, the smallest number of iterations is 5 under T3 network. It means that the network is the most tolerant of errors under 1 hidden layer topology with 272 neurons (the number of INPUT nodes times the number of OUTPUT nodes).



Network Status	T1	T2	T3	T4
Inputs	16	16	16	16
Outputs	17	17	17	17
Vectors	160	160	160	160
Neurons	33	65	305	97
Hidden Layers(TOPOLOGY)				
the # of hidden layer	0	1	1	2
the # of nodes/h		32 (= INPUT X 2)	272 (= INPUT x OUTPUT)	32 (= INPUT X 2)
the total # of nodes		32	272	64
Training Parameters				
learning rate	0.3	0.3	0.3	0.3
Stopping condition				
the # of iterations	7094	304	5	97
Tolerance %	95%	95%	95%	95%
Results				
total error rate	4.99	4.96	1.17	4.96
% Correct	99.95	99.95	99.98	99.95

Fig. 12 . Learning Results with Network status (adjusting learning rate and iteration number)

Figure 13 shows the results of comparing between actual outputs of prediction results and desired outputs of training sets. We can realize that the actual outputs seem to be approximated to desired outputs, as we expected.



Actual Output	Disired Output	Actual Output	Disired Output	Actual Output	Disired Output
13.2572	13	4.638171	5	3.323117	3
15.77974	16	4.877563	5	3.330195	3
1.025283	1	5.059554	5	2.806404	3
0.928683	1	4.964586	5	3.016124	3
0.9349977	1	7.048586	6	3.064165	3
1.133425	1	9.078062	9	2.987198	3
4.857995	5	12.18801	12	8.150376	8
5.024797	5	14.99448	15	10.6666	11
4.958493	5	4.835256	5	14.10716	14
5.039969	5	4.899389	5	16.31325	17
4.005897	4	4.98481	5	7.906893	8
4.062534	4	4.993693	5	10.65116	11
4.033796	4	1.254928	1	14.2172	14
3.943906	4	1.079246	1	16.34376	17
3.905409	4	0.896942	1	7.15222	7
3.988985	4	0.9161988	1	9.732225	10
4.013333	4	3.100619	3	13.56354	13
4.00633	4	3.117309	3	16.1084	16
6.651451	6	2.979155	3	7.089486	7
8.900903	9	3.053519	3	●	●
12.14961	12	2.863694	3	●	●
14.86247	15	3.148607	3		

Fig. 13. Comparison btw actual outputs and desired outputs (Total Error Rate 1.3058, iteration # 3000000)

Fig. 14 shows the Error Distribution Chart of an output variable. Through this chart, we can analyze which sample data of training sets affects to result in errors.

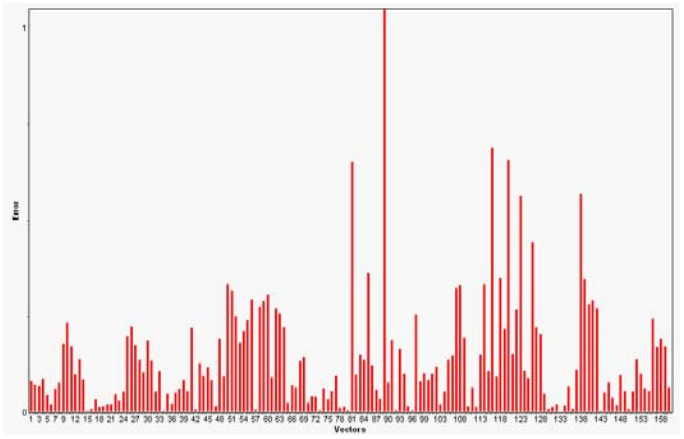


Fig. 14. Error Distribution Chart (Total Error Rate 1.3058, iteration # 3000000)

VI. CONCLUSION AND FUTURE WORKS

In this paper, we designed and tested UPM which employs learning mechanism of Neural Network. Without directly extracting user's status and desire from physical sensors, context information {when, where, how, why} obtained from context-aware application model can be connected to the input layer of NN. Each context is denoted by numerical values, which represent where he is, when he is observed, which gesture he makes, and what his stress level is. Thus, these contexts can be trained for being mapped to appropriate output categories. The final result after training MLP is enough good to recognize user's behavior pattern which can be used to infer the preferred service of the user. Remaining work is to consider the method which can reflect users' feedback in real-time. Also, when any one of the input contexts is not provided to the system, the system should still provide appropriate services to the user. We need to evaluate learning mechanism of Neural Network and consider a bayesian approach to human activity recognition [19]. Furthermore, it is important that users be able to enjoy context-aware personalized services by selectively disseminating their private information without compromising their privacy. That is, users only have to distribute their profile the time when they indeed want to enjoy services. Privacy control mechanism should be elaborated from now on.

REFERENCES

[1] Anind K. Dey and Gregory D. Abowd, .Towards a Better Understanding of Context and Context-Awareness., Proceedings of the CHI 2000 Workshop on .The What, Who, Where, When, and How of Context-Awareness., The Hague, Netherlands, April 1-6, 2000.

[2] R. Want, K. Fishkin, B. Harrison and A. Gujar, "Bridging Physical and Virtual Worlds with Electronic Tags," In *Proc of ACM SIGCHI 99*, May 1999.

[3] A. K. Dey, D. Salber, G. D. Abowd, " A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Computing," Human-Computer Interaction (HCI) Journal, Vol. 16, pp.97-166, 2001.

[4] Barry Smyth and Paul Cotter, "Personalized adaptive navigation for mobile portals", *ECAI 2002*, (2002).

[5] Context-Aware Personalised Service Delivery. In R Lopez de Mantaras & L Saitta (ed), Proceedings of the Sixteenth European Conference on Artificial Intelligence - ECAI-2004 (Valencia, Spain), IOS Press, Amsterdam, pages 1077-1078, 2004

[6] C. Muldoon, G.M.P. OHare, D. Phelan, R. Strahan, and R.W. Collier, 'Access: An agent architecture for ubiquitous service delivery', in *Proceedings Seventh International Workshop on Cooperative Information Agents (CIA)*, (2003).

[7] Ubisworld, http://www.u2m.org/

[8] D. Huynh, D. R. Karger and D. Quan, "Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF," In *Proc. of Semantic Web Workshop*, May 2002.

[9] SOUPA, http://pervasive.semanticweb.org/

[10] Jason I. Hong and James A. Landy, "An Infrastructure Approach to Context-Aware Computing," Human Computer Interaction, Vol. 16, 2001.

[11] A. Ranganathan and R. H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," In *Proc. of International Middleware Conference*, June 2003.

[12] H. Chen, T. Finin and A. Joshi, "Semantic Web in the Context Broker Architecture," In *Proc. of Percom 2004*, March 2004.

[13] Daisuke Morikawa, Masaru Honjo, Akira Yamaguchi, Masayoshi Ohashi, KDDI Corporation, "A Proposal of User Profile Management Framework for Context-Aware Service," In SAINT-W'05, January 31 - February 04.

[14] S. Jang, W. Woo, "ubi-UCAM: A Unified Context-Aware Application Model," LNAI (Context03), pp. 178-189, 2003.

[15] Y.Oh, W.Woo, "A unified Application Service Model for ubiHome by Exploiting Intelligent Context-awareness," UCS'04, pp. 117-122, 2004.

[16] D.Hong, W.Woo, "wear-UCAM: A Toolkit for Wearable Computing," ubiCNS05, accepted, 2005.

[17] Richard P. Lippmann, "An introduction to computing with neural nets", IEEE Acoustics, Speech and Signal Processing Magazine, 4(2):4--22, April 1987.

[18] Seiie Jang, Choonsung Shin, Yoosoo Oh, and Woontack Woo, "Introduction of "UbiHome" Testbed," *The first Korea/Japan Joint Workshop on Ubiquitous Computing & Networking Systems 2005(ubiCNS2005)*, pp. 000-000, 2005.

[19] A. Madabhushi and J. K. Aggarwal, "A bayesian approach to human activity recognition", In *Second IEEE InternationalWorkshop on Visual Surveillance (CVPR Workshop)*, pages 25–30, Fort Collins,CO, June 1999.

[20] (Handbook style) *Transmission Systems for Communications*, 3rd ed., Western Electric Co., Winston-Salem, NC, 1985, pp. 44–60.

[21] *Motorola Semiconductor Data Manual,* Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.
[22] (Basic Book/Monograph Online Sources) J. K. Author. (year, month, day). *Title* (edition) [Type of medium]. Volume(issue). Available: http://www.(URL)
[23] J. Jones. (1991, May 10). Networks (2nd ed.) [Online]. Available: http://www.atm.com
[24] (Journal Online Sources style) K. Author. (year, month). Title. *Journal* [Type of medium]. Volume(issue), paging if given. Available: http://www.(URL)
[25] R. J. Vidmar. (1992, August). On the use of atmospheric plasmas as electromagnetic reflectors. *IEEE Trans. Plasma Sci.* [Online]. *21(3).* pp. 876—880.Available:http://www.halcyon.com/pub/journals/21ps03-vidmar

Youngjung Suh received the B.S. degree in Computer Engineering from Chonnam National University in 2001 and M.S. degree in Information & Communication Engineering from Gwangju Institute of Science and Technology (GIST) in 2003. Research Interest: Mixed Reality, Vision Based Human Computer Interaction, Context-aware technology, Ubiquitous computing, Wearable computing

Dong-Oh Kang received his B.S. degree in electronic engineering from Yonsei University, Korea, in 1994. And, he received his M.S. and Ph.D. degrees in electronic engineering from Korea Advanced Institute of Science and Technology, Korea, in 1996 and 2001 respectively. Since 2001, he has been working at Electronics and Telecommunications Research Institute, and now, he is a senior member of engineering staff of the Post-PC Platform Research Team where he is developing a Post-PC platform. His research interests include context-aware middleware, home network middleware, distributed control, and Post-PC platform.

Woontack Woo received his B.S. degree in EE from Kyungpook National University, Daegu, Korea, in 1989 and M.S. degree in EE from POSTECH, Pohang, Korea, in 1991. He received his Ph. D. in EE-Systems from University of Southern California, Los Angeles, USA. During 1999-2001, as an invited researcher, he worked for ATR, Kyoto, Japan. In 2001, as an Assistant Professor, he joined Gwangju Institute of Science and Technology (GIST), Gwangju, Korea and now at GIST he is leading U-VR Lab. Research Interest: 3D computer vision and its applications including attentive AR and mediated reality, HCI, affective sensing and context-aware for ubiquitous computing, etc.