Context-Aware Computing System for Heterogeneous Applications

Yoshinori Isoda, Shoji Kurakake, and Kazuo Imai

***Abstract*— It is a very important to develop context-aware systems that can handle, at the same time, multiple heterogeneous applications that require different contexts with different levels of abstraction. This paper proposes a framework for such systems. To handle the heterogeneity of the context required by the applications, we introduce a user activity context detection method based on the combination of a multi spatio-temporal description of measured sensor data, a description of detected context with multiple levels of abstraction, and an order-sensitive description of the context model required by an application. We also introduce an algorithm that implements the context detection method by reflecting the context detection capabilities of any given environment. We build a prototype system by embedding sensors into an experimental house; evaluations show it promise.**

***Index Terms*— context recognition, decision tree, spatio-temporal representation, ubiquitous**

I. Introduction

Context-aware computing systems are being studied, and many prototypes have been implemented over the years [1]. The popularity of context-aware computing research indicates that systems that can identify the user's context and that of the surrounding environment have the profound potential to provide services that are much more user-specific. Context awareness is especially relevant to mobile and ubiquitous applications because some of those applications are suitable only at a particular time and/or location for users in a specific situation. The ability to detect context in different environments is essential for a system that provides context-aware mobile ubiquitous applications over a wide area. Time, location, identity and activity have been proposed as the primary elements of context [2]. Because it is relatively easy to detect time and location, a lot of location-aware systems such as guides for city tours [3], [4] and guides for museums [5] have been designed. Activity is much more difficult to identify, but some aspects of activity can be detected by placing sensors in the environment. Advanced context-aware applications using activity context information have been realized for a specific smart environment [6]. Prior researches tend to focus on specific applications in specific environments. The goal is, however, to support different context-aware applications simultaneously in the same environment and across different environments. We believe that context-aware systems must become capable of handling multiple heterogeneous applications and environments if such systems are to be brought to market. We will use the following scenario to further illustrate the motivation of our work.

A. Motivation Scenario

User A is at home in the morning. He launches a home monitoring application, which records his physical activity context at home throughout the day and also alerts him if pre-registered events are detected within the house. While preparing breakfast for himself and his dog, he starts a cooking assist application. He carries an RFID tag. Motion detection sensors, touch detection sensors and other sensors are attached to the dishes, pans and other kitchenware. Cameras are sited around the kitchen. Once he selects a recipe for breakfast, the cooking assist application shows him cooking instructions step-by-step according to the selected recipe on a near-by screen. He can interact with the application by touching or moving the appropriate kitchenware. While general soup making instructions are displayed on the screen, he can request more detailed instructions by shaking a soup pan. He can skip to the next instruction by tapping the pan. If his body is too close to a pan of boiling water, the home monitoring application flashes an alert light in the kitchen. When he goes to the family room to feed his dog, the cooking assist application is automatically suspended. If the movement of some dish is detected while he is in the family room, the home monitoring application sends an alert message and visual information captured by the cameras in the kitchen to the screen closest to him in the family room. When he goes back to the kitchen, the cooking assist application is resumed. His dog wears an RFID tag. RFID tag readers are mounted on walls, and motion detection sensors are mounted on doors and some furniture. The home monitoring application can track the location of the dog and detect if the dog touches doors or furniture. The application passes messages to the screen nearest to him when his dog moves from one room to another. When the application notifies him that his dog is trying to get into the study room, he stops cooking, rushes to the study, stops the dog, and carries it back to the family room. In addition, a daily chores reminder application is always running. It reminds him of tasks to be done. Just after he wakes up, the application tells him to take his medicine. When he is about to leave home, the application indicates which windows are still open and confirms that he wants them to remain open. It also suggests that he take an umbrella since the weather forecast is for rain. While cooking, he found that the rice would run out within a week. He enters this into the daily chores application. He has registered with several context detection services at shopping malls that he often visits. This means that his daily chores application knows

where to buy rice; it will keep reminding him every time he is physically close to a shop with rice until he buys it. When he is out of his home, the home monitoring application automatically changes its behavior and notifies him, via his cellular phone, only when the dog leaves the house or unauthorized people try to enter it.

We make three observations from this scenario.

1) Different contexts will have significantly different spatial and temporal resolutions. The time period of the *touching kitchenware context* is a couple of seconds, and that context happens within the kitchen. The context of *leaving home* may occupy a couple of minutes. The context of *looking for rice* may extend over a week and the context should be detected by using sensors spread city-wide.
2) The same sensor data can be used to detect the different contexts required by different applications. For example, the data from the motion detection sensor attached to a dish is basically used by the cooking assist application. The same data, however, can be used by the home monitoring application to detect suspicious events. Note that even the same application may require different levels of context to be abstracted from the same sensor data.
3) Some of the context-aware applications that are triggered by the detection of the same pre-registered multiple contexts require an ordered sequence of context detection, other do not. The cooking assist application, for example, requires the appropriate contexts to be detected in a pre-defined order. The daily chores application does not care which open window was closed first.

B. Our Approach

To address the issues identified in the scenario above, we developed a user activity context detection method that can, at the same time, cope with multiple context-aware applications that may require different kinds of context with various levels of abstraction. Our method decouples the description of measured sensor data from the description of detected context, and the description of required context model associated with an application. Note that the terms “detected context description” and “required context model” refer to different concepts. User activity context descriptions have multiple abstraction levels for describing detected user context. The model of required context is instantiated by available abstraction levels of context depending on the detection capability of each environment and the level of context required by the applications. Our sensor data description is flexible in terms of temporal and spatial resolution. The duration of sensor activation is the most important feature to detect user context, and the sensor data description can adapt the time scale according to the duration of sensor activation of interest. We describe the required contexts to control an application based on regular expressions. This enables the description of ordered and semi-ordered contexts as condition elements for satisfying application requirements.

We also introduce an algorithm that can implement the context detection method in a given environment. The algorithm learns the context detection capabilities of the sensors at each site. This enables the heterogeneity in environment with respect to sensing capability to be well handled. We have also developed a framework that allows a context-aware application to adapt to the different context detection capabilities. We assume that an ID is assigned to each class of context and that the IDs are shared among the applications and context detection environments. Thus the required context designated by an application is mapped to context detected by a local context detection system by using context ID.

We have built an experimental environment house into which we placed sensors such as floor-mounted weight sensors and RFID tags with touch sensor. Tests were conducted in this house and show that our methods and framework are promising.

The remainder of the paper is organized as follows: Section 2 presents the design of overall system, Section 3 describes a user activity context detection method, Section 4 presents an algorithm for implementing the context detection method, Section 5 describes the experimental house, the sensors, and the system. Section 6 describes the experiment and analysis of the results, and Section 7 draws our conclusions.

II. Design of the System

The design of the proposed system is based on decoupling the description of measured sensor data from the description of detected context, and the description of required context model associated with an application. The benefit of this approach is that users can employ context-aware applications in various ubiquitous environments (UE).

The framework of the system is shown in Fig. 1. The global network stores and manages application templates. When a user wants to use a specific application, the user downloads the corresponding application template to the user’s terminal (Fig. 1, (1)). Each application template consists of a pointer to the application and a description of the required context, which explains the necessary conditions in terms of the contexts that will trigger, suspend, and resume the application. We call the description of the required context model because context itself is referred to as the abstract context concept.

In each UE, sensor data is mapped to context by using a decision tree. At the each abstraction level of the decision tree, each leaf is assigned to a user state, and each node has a description of decision making conditions with respect to the value of sensor data to further split the user state of a node into more specific user states. Thus, the decision tree describes detectable user states with multiple abstraction levels by using sensors in the environment. Some of the user states in the decision tree are associated with context. In practice, some leaves in the decision tree are treated as context leaves. The decision tree is stored in a context server. The context server receives sensor data and maps it to the detected context descriptions. If a user wants to use a specific application and has already downloaded its application template to his mobile

terminal, the user compares the required context model and the decision tree stored in the context server at the site that he will visit. By using the shared context class ID, the user can determine which abstract level of context needs to be detected at the site for each context described in the required context model. The user can carry out this process over the net or at the site. When the user is in some UE, the user receives the detected context description from the context server (Fig. 1, (2)). Thereafter, the user allocates the detected context to the elements of the required context model in the application template (Fig. 1, (3)). For example, when the application needs to know only whether the user is indoors or not, the user selects a user state with high abstraction level in the decision tree. On the other hand, when the application requires some specific behavior in some specific place (e.g. cooking in the kitchen) as context, the user must select a state from a lower abstraction level. As a result of this procedure, the appropriate user states at appropriate abstraction levels are instantiated in all required context model elements. The application is then controlled according to the description in the application templates in the mobile terminal.

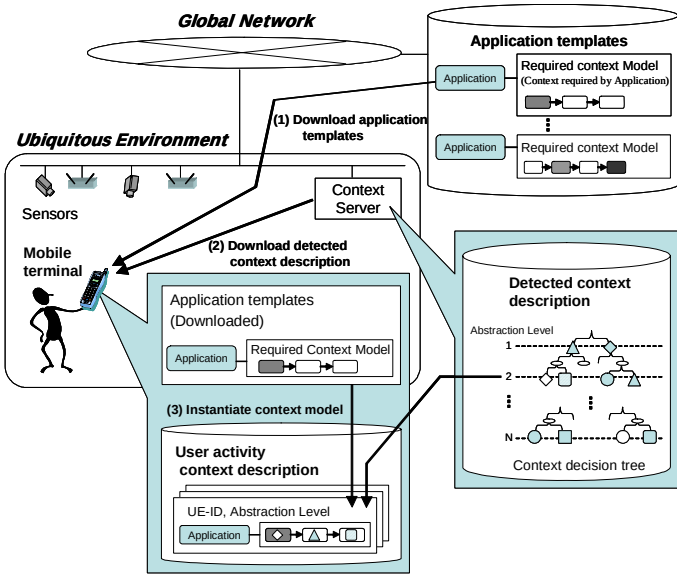

Fig. 1. Framework of the proposed system

III. User Activity Context Detection Method

A. Context Model

As we mentioned above, each context model is used to trigger a specific application and each model consists of context elements that describe specific user states. In the real world, users perform a wide variety of tasks. In this study, a task refers to work performed by a user; work here consists of performing specific state transitions, such as checking that doors are locked in multiple locations, or preparing a meal by following a series of procedures. When the user performs the tasks, the time ranges of user states included in the task change accordingly. Therefore, the context model deals with only transitions in user state. Context models take the form of regular expressions. Each state consists of context models partitioned by commas. For example, the ordered sequential state transition sequence $[S_1 \rightarrow S_2 \rightarrow S_3]$ is represented as "$\hat{}S_3, S_2, S_1$". However, when a user performs a specific task, it is not always necessary for the states associated with the task to occur consecutively. For example, user states S_1, S_2 are elements of one context model. The relationship between S_1, S_2 is ordered, but they don't always occur consecutively. In this case, the context model is represented as "$\hat{}S_2$, ([$\hat{}$,]*,) S_1". Moreover, sometimes a user performs different tasks in parallel. Therefore, the context models in the user activity model are managed separately, and checks are performed for multiple applications to determine which ones should be executed.

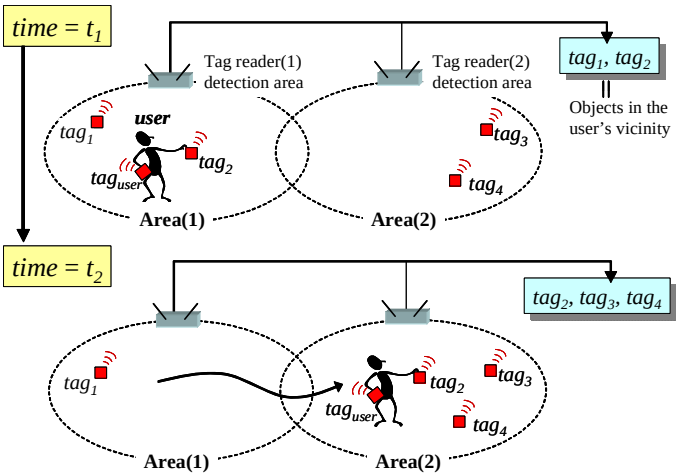

Fig. 2. Changes in RFID tags output associated with movement of the user. RFID tags are attached to objects and the user.

B. Description of User States

Each UE offers a different combination of sensors (type and number). The sensor data descriptions should support the use of many types of sensors. In this paper, sensor information is transformed into spatio-temporal attributes. Each attribute consists of temporally continuous sensor data, and the attributes are arranged to support each spatial resolution needed. Complex sensors, such as cameras have already been used for recognizing user state. Computer visions tracking [7], [8] and behavior recognition [9], [10], [11] often work in the laboratory but sometimes fail in real environments due to the lighting variations and occlusions that are frequent in natural environments. Given their relative immaturity, we restrict ourselves to simple sensors such as RFID tags and weight sensors to achieve robust user state detection.

As an example, we explain the description of user states when RFID tags and floor sensors are employed as sensors. RFID tags have recently attracted interest as a means of detecting objects in the vicinity of the user. In this case, the attributes describing the user's state are derived from information obtained from the RFID tags carried by the user and attached to objects. As shown in Fig. 2, the RFID tag information detected by each RFID tag reader is collected. This

yields attributes that relate the state of each user to the objects in the user's vicinity. In addition to information on the presence or absence of objects in each user's vicinity, the temporal continuity of the presence of these objects is also used as attributes of the user's state. In other words, the user's state is described by attributes that include information on how long each object has been present in the user's vicinity due to the user's movements and whether or not the objects in question are being carried around by the user. The overall set of attributes describing the state of user *u* in relation to these objects at time *t* is as follows:

$$Tag(u,t) = [\, tag_1(t), tag_2(t), \ldots, tag_i(t), \ldots, tag_n(t) \,] \quad (1)$$

Here, $tag_i(t)$ is a positive integer value indicating how long object *i* has been remained (continuously) in the vicinity of user *u* at time *t*. If object *i* does not exist in the user's vicinity, the value of $tag_i(t)$ is zero. Here, the user's vicinity corresponds to the detection range of the tag reader that detects the tag carried by the user, so all objects detected by the same tag reader are regarded as being in the user's vicinity. When plural users occupy the same detection range of a tag reader, it is difficult to distinguish which objects are in each user's vicinity. Therefore, the detection range of the tag reader is restricted from several tens centimeters to one meter by using an attenuator. In this way, the user's state is identified, in part, from the attributes expressing which objects are in the user's vicinity, and if so, for how long. For example, Fig. 3 shows the attributes of objects around the user at times t_1 and t_2 in Fig. 2.

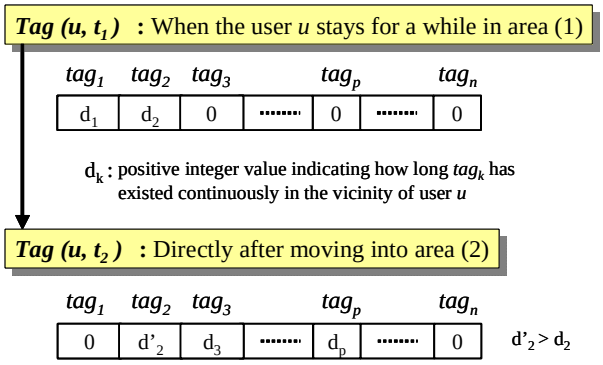

Fig. 3. Representation of attributes relating to objects.

RFID tags provide information on the spatial relationships between the user and objects. In some UE, sensors that detect information on the absolute position of the user will be employed. In this case, the multiple absolute user positions, needed to satisfy the multiple spatial resolutions, are used as attributes representing user state. As Fig. 4 shows, one floor can be partitioned into several cells. Several cells are grouped into larger cells. Additional attributes are used to express whether or not the user is present in each cell, and if so, for how long. The attributes related to the position of user *u* at time *t* are as follows.

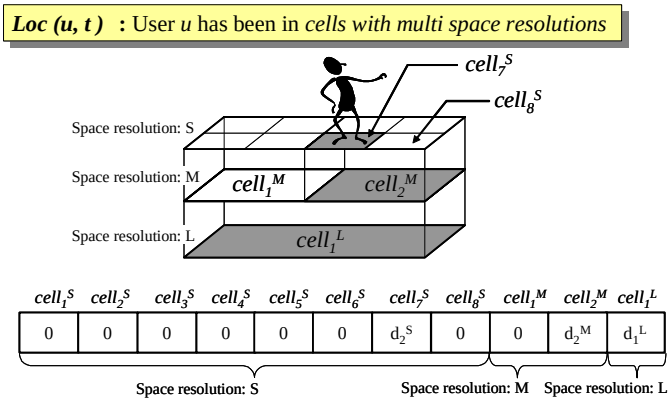

Fig. 4. Attribute representation of user position with multi spatial resolution.

$$Loc(u,t) = [\, cell_1^1(t), cell_2^1(t), cell_3^1(t)..., cell_1^R(t),..., cell_j^R(t), \ldots, cell_m^Q(t) \,] \quad (2)$$

Here, $cell_j^R(t)$ is also a positive integer value indicating whether or not the user has been continuously detected in cell *j* with space resolution *R* at time *t*.

When the above two kinds of sensors are employed in a UE, the entire set of attributes $A(u,t)$ representing the state of user *u* at time *t* is as follows:

$$A(u,t) = [\, Tag(u,t), \; Loc(u,t) \,] \quad (3)$$

C. User State Decision

Several methods to discriminate user state have been proposed. Combinations of neural networks and lookup tables were used to predict the temporal mobility patterns of a user in the Neural Network House [12], [13]. However, since neural nets do not provide information about the underlying model of the prediction process, it is difficult to extend them to incorporate prior knowledge. Specific types of Dynamic Bayesian networks [14] have been used to track the daily activities of users from sensor information such as active badges [15] and cricket-devices [16] embedded in a house [17]. The algorithm can differentiate the activities according to duration and user location. Dynamic Bayesian networks have proven to be one of the most powerful and efficient ways of representing temporal events and fusing information from multiple sensors [18]. However, the complexity of networks and learning algorithms for Hidden Markov models make it difficult to apply them to cases involving many sensors.

Each application requires a different level of user state abstraction as context. Some applications simply need user position with coarse resolution; others may need more detailed user states such as user behavior. Therefore, user states should be detected with various levels of abstraction. With the goal of detecting user state hierarchically, we use a decision tree that discriminates user states with multiple levels of abstraction. The benefit of using a decision tree is that it generates understandable rules. Therefore, if high-level knowledge about

discriminating user states exists, the decision tree can be modified manually. In each UE, the detected context description from sensor data is described as a decision tree that detects user states hierarchically with different abstraction levels. If more than one person is in a UE, the decision tree is managed to detect each user context separately.

IV. Context Detection Method

It is difficult to create the rules that form a decision tree manually because each UE will have different sensors and the user would have to consider the meaning and significance of sensor information and the relationships between sensors. Moreover, given the large number of sensors in one UE, some sensors will be prone to error. This means that the user would have also considered the robustness of sensor data. Our solution is to employ the learning-based approach to construct the decision tree semi-automatically.

All sensor data is recorded while a user performs actions such as using the toilet, bathing, grooming and so on. These actions typically consist of specific user states. For example, the action of bathing consists of states such as standing in front of a bathroom, standing inside the bathroom, using the bath, and so on. However, identifying these states depends on the sensors in the UE. Our context detection method extracts hierarchical states of each action as leaves of the decision tree.

Sensor data recorded in each time interval is transformed to spatio-temporal attributes according to (1)-(3) and the data for each time interval is used as one training data set for constructing a decision tree. At first, the user allocates action labels to the specific time regions of the training data by hand. These time regions correspond to specific actions such as bathing. Next, our system finds user states identified by the actions by using the C4.5 algorithm [19]. The C4.5 algorithm constructs a decision tree by calculating the information gain ratio of each attribute from a set of training data, and successively employing the attributes with the highest values (e.g., $tag_i(t)$) as the nodes of the decision tree. Moreover, the C4.5 algorithm can treat various kinds of sensor information as attributes. This makes it easier to integrate different kinds of sensors and to use them to discriminate user states. In the real world, sensor error can occur at any time. If the training data contains instances of sensor error, the C4.5 algorithm selects the attributes of sensors that are able to discriminate user state robustly.

Let D denote a set of cases of training data. Each case is labeled with its state class c_j $(1 \le j \le N)$. Some attribute A with mutually exclusive outcomes $A_1, A_2,..., A_k$ is used to partition D into subsets $D_1, D_2,..., D_k$, where D_i contains those cases that have outcome A_i. $p(D, c_j)$ is the proportion of cases in D that belong to the jth class. The residual uncertainty about the class to which a case in D belongs can be expressed as

$$Info(D) = -\sum_{j=1}^{N} p(D, c_j) \times \log_2\left(p(D, c_j)\right) \quad (4)$$

and the corresponding information gained by an attribute A with k outcomes as

$$Gain(D, A) = \text{info}(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times Info(D_i) \quad (5)$$

The information gained by an attribute is strongly impacted by the number of outcomes and is maximal when there is one case in each subset D_i.. On the other hand, the potential information obtained by partitioning a set of cases is based on knowing the subset D_i into which a case falls; this *split information*

$$Split(D, A) = -\sum_{i=1}^{k} \frac{|D_i|}{|D|} \times \log_2\left(\frac{|D_i|}{|D|}\right) \quad (6)$$

tends to increase with the number of outcomes of an attribute. The gain ratio is expressed as

$$GainRatio(D, A) = Gain(D, A) / Split(D, A) \quad (7)$$

The gain ratio criterion assesses the desirability of an attribute as the ratio of its information gain to its split information. The gain ratio of every possible attribute is determined and, among those with at least average gain, the split with maximum gain ratio is selected.

In a decision tree constructed in this way, the attributes that are valid for discriminating the user states identified by the actions are learned as nodes of a decision tree. Leaves of the decision tree are the user states to be discriminated. Applying the C4.5 algorithm to each state recursively, our system extracts the hierarchical states of each action. In the following, we detail the procedure by using an example. As shown in Fig. 5, the user allocates label ${}^1\mathrm{A}$ to the specific time regions that correspond to the specific action in the training data. Time regions other than those labeled ${}^1\mathrm{A}$ are assigned label ${}^1\mathrm{E}$. This is the first level of the decision tree. By using this set of training data, the C4.5 algorithm constructs a decision tree that decides valid attributes of sensor data as nodes and states of the action ${}^1\mathrm{A}$ as leaves.

In Fig. 6, action ${}^1\mathrm{A}$ is divided into n leaves in the decision tree at the second level. These leaves are taken as user states at abstraction level 2 of action ${}^1\mathrm{A}$, and assigned the label of ${}^2\mathrm{S}_i$ $(1 \le i \le n)$. In the training data, each time region of state ${}^2\mathrm{S}_i$ is a region in which sensor attributes that discriminate ${}^2\mathrm{S}_i$ hold (Fig. 6). At the third level of the decision tree, new decision tree branches are created from each state ${}^2\mathrm{S}_i$. In this level, the time regions corresponding to ${}^1\mathrm{A}$ in the first level are used as new training data (Fig. 7). Sensor attributes that were used to discriminate ${}^2\mathrm{S}_i$ are deleted from the new training data. That is, the sensor attributes used in the second level aren't used in the third level. The time regions other than those labeled ${}^2\mathrm{S}_i$ are re-labeled ${}^2\mathrm{E}_i$. The decision tree at the third level is constructed from this new training data set. For example, ${}^2\mathrm{S}_i$ is divided into m leaves. Each new leaf is assigned the label ${}^3\mathrm{S}^i_j$ $(1 \le j \le m)$, and together they are considered as the user states derived from

^2S_i at abstraction level 3. These procedures are executed recursively to construct sub-trees. For example, the k^{th} node at the fourth level of decision tree is labeled $^4S_k^{i,j}$. The result is one decision tree that can discriminate user states at multiple levels. Fig. 8 shows an example of one such decision tree.

In the context detection phase, the resulting decision tree and the current information from sensors are used to discriminate the current user state in each abstraction level.

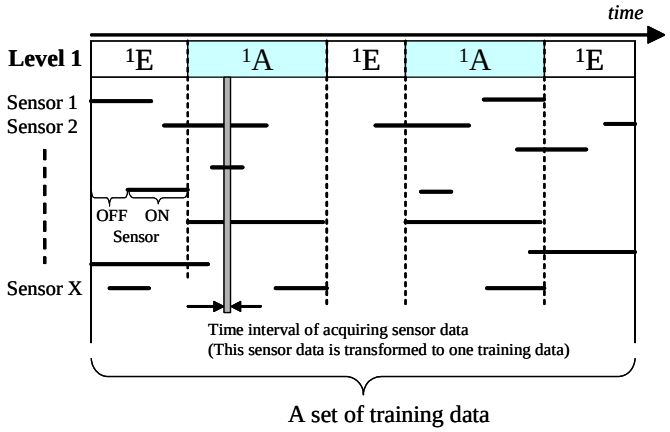

Fig. 5. User state allocation in specific time intervals; manually performed at the first level.

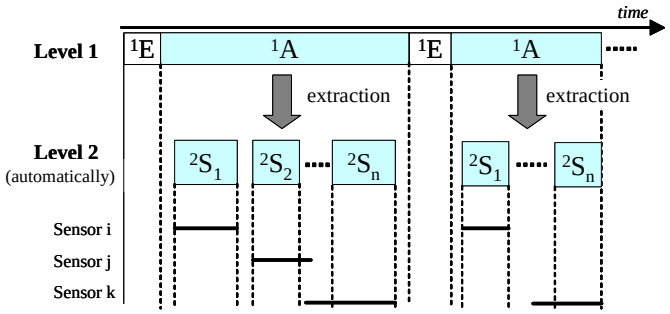

Fig. 6. User states are extracted at second level of decision tree.

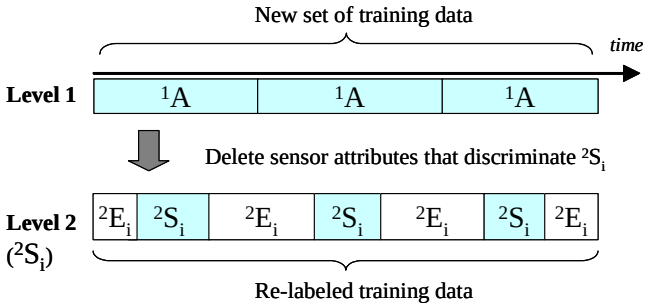

Fig. 7. New training data for extracting use states at level 3 for ^2S_i.

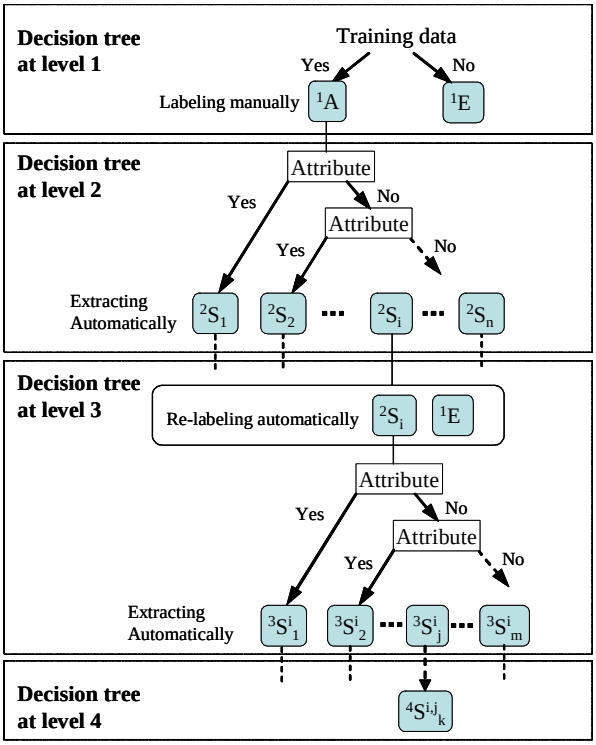

Fig. 8. Example of a decision tree that discriminates user states at multiple levels.

V. User Activity Sensing and Detection Environment in Ubiquitous Experimental House

We have developed an experimental house in which multiple types of information about human activities can be acquired [20]. It contains various embedded sensors and output devices. Fig. 9 shows its floor plan. To support a wide variety of experiments, this house has workspaces set into the floor, attic, and inside walls (striped areas in Fig. 9). These workspaces make it easy to set up and wire many sensors and devices. Fig. 10 shows the configuration of the prototype system constructed inside the experimental house. The next subsection details system elements.

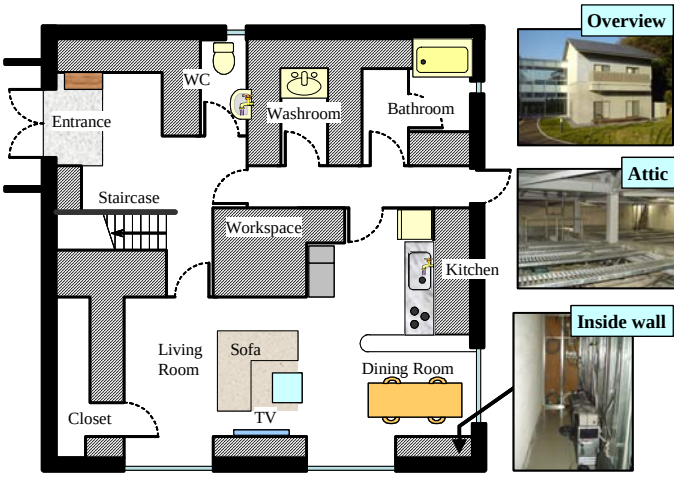

Fig. 9. Overview of Ubiquitous Experimental House.

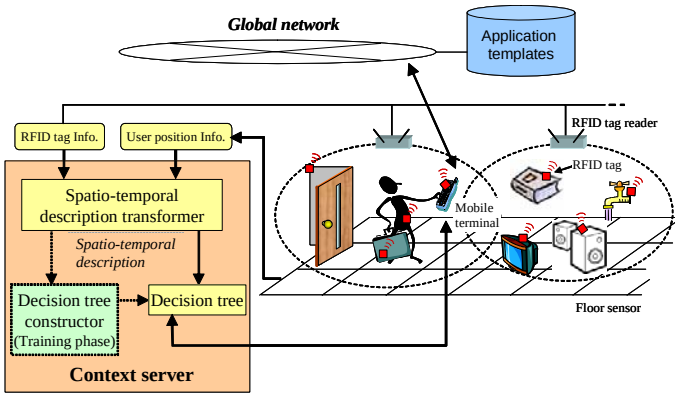

Fig. 10. Prototype system configuration.

A. RFID Tags

As shown in Fig. 11, the house has an active type RFID tag system consisting of RFID tag readers and RFID tags attached to the users and to various objects. We extended some of the RFID tags to suit the object. For instance, we attached a magnetic reed switch to an RFID tag to yield a position sensor. These RFID tags can detect human behavior when using objects such as turning a faucet, opening a door, using a towel and so on. Other RFID tags were modified with touch sensors. When the user touches electrodes on the object, the touch sensor detects the action and the RFID tag is activated. We can detect human behavior that involves touching objects such as holding a cup, holding a receiver, sitting on the toilet, and brushing teeth and so on. Moreover, other RFID tags were modified with small weight-sensitive sensors to detect user behavior such as sitting on a chair, lying on a sofa and so on. This tag system has low directionality in the tag detection ranges, and by attaching attenuators we were able to set the tag detection radius from a few tens of centimeters to a few meters. Each reader communicates with a PC by TCP/IP via a protocol converter. The RFID tags transmit at intervals of 0.4 seconds.

B. Floor Sensor

To recognize human activity, information of the user's position is useful. Therefore, the entire floor is covered with a large number of pressure sensors. Each pressure sensor covers 18 x 18 cm (a unit cell), and the binary value output by each cell is read into a PC via a serial port at 0.4 second intervals. These detector cells were grouped into blocks (e.g. 10×10, 5×5, 3×3), so user position is detected at multiple resolution levels.

C. Context Server

The context server operates in one of 2 phases. One is the phase of constructing a decision tree that discriminates user states with multiple levels of abstraction. Information from RFID tags and floor-mounted weight sensors are acquired and used as training data.

In the other phase, it users this decision tree to perform state decisions based on the current information provided by the RFID tags and floor-mounted weight sensors. When plural users are in the house, each user is distinguished by the RFID

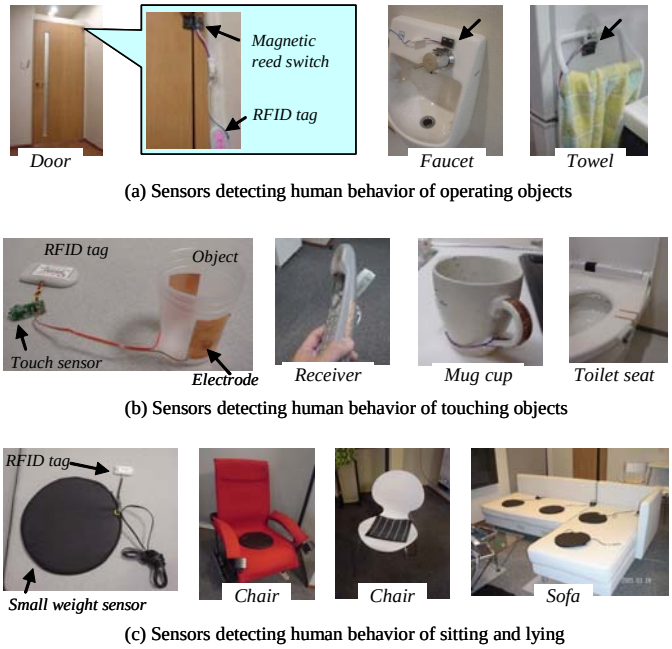

Fig. 11. Sensors detecting human behavior associated with objects.

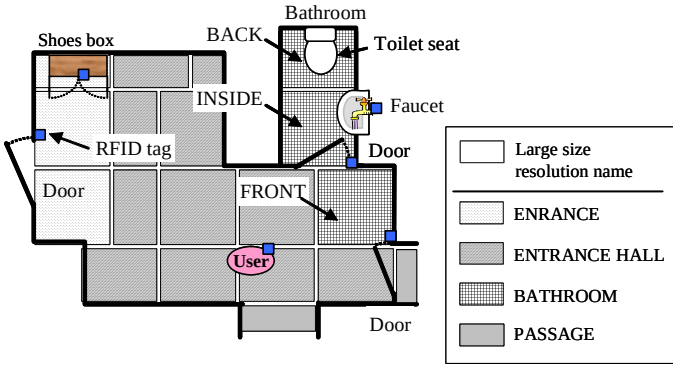

Fig. 12. Experimental environment.

tags carried by the users, and the decision tree is used for each user. The states of each user thus detected are sent to the user's mobile terminal. On the mobile terminal, user states are received and applications are executed by matching the user states and context elements.

VI. Experimental Evaluation

To confirm the automatic construction of a decision tree, we performed an experiment that focused on bathroom activities as the first step.

A. Experimental Conditions

Fig. 12 shows the experimental environment in the bathroom. A user performed a series of actions such as entering the bathroom, using the bathroom, washing hands, leaving the bathroom and walking around the house. We acquired these series of actions five times as training data. One RFID tag was continuously carried by the user. Other RFID tags were

attached to objects. Five RFID tags were extended with a magnetic reed switch, and attached to doors and faucet in the bathroom.

The information acquired by the floor-mounted weight sensors was represented with multiple levels of spatial resolution. At the largest resolution level, the experimental area was divided into four spaces: ENTRANCE, ENTRANCE HALL, BATHROOM, and PASSAGE. The medium level of resolution corresponded to the areas consisting of 5×5 blocks. The smallest level of resolution corresponded to areas of 3×3 blocks.

B. *Experimental Result*

Fig. 13 shows the decision tree that was constructed automatically, and Fig. 14 shows the decision results for each time interval at each level of the decision tree.

At the first level of the decision tree, two leaves were extracted as user states. Investigating the attributes of decision tree, one considers that the user is near the bathroom. The other considers that the user is not near the bathroom. They are named TOILET and OTHER, respectively. The largest space resolution, which corresponds to the area of the bathroom, was selected as the attribute to discriminate the two states at this level.

At the second level of the decision tree, four leaves were extracted as user states. Investigating the attributes of the decision tree, we see that the first considers that the user is in front of the bathroom, the second considers that the user is inside the bathroom, the third considers that the user is at back of the bathroom, and the fourth considers that the user is in some other state. They are named FRONT, INSIDE, BACK, and OTHER respectively. At this level, three areas in the bathroom with medium level of resolution were selected as attributes.

At the third level of the decision tree, no new leaves were extracted from the user states of FRONT and BACK. However, six leaves were extracted from the user state of INSIDE. The first leaf from INSIDE considers that the user is standing on both the cell of BACK and that of INSIDE simultaneously. This leaf is named INSIDE&BACK. The second leaf considers that the user is washing hands. This leaf is named WASH HANDS. The third leaf is detected from the attributes of the floor sensor at the area in front of the bathroom and the area inside the bathroom. This leaf is detected only when the user is just entering the bathroom (Fig. 14). Investigating the training data, the floor sensors corresponding to FRONT and INSIDE were not active simultaneously when the user left the bathroom. It is considered that the position relationship between the door, the faucet, and toilet seat influence human behavior. It can be said that this extracted leaf well matches real human behavior in the environment. Therefore, this leaf is named JUST ENTER. The fourth leaf from INSIDE considers that the user is just leaving the bathroom. When the user is entering the bathroom, the same attributes can be used. However in the case of entering the bathroom, the decision tree detects it as JUST ENTER. So, this leaf is named OPEN DOOR.

From these results, reasonable user states at different

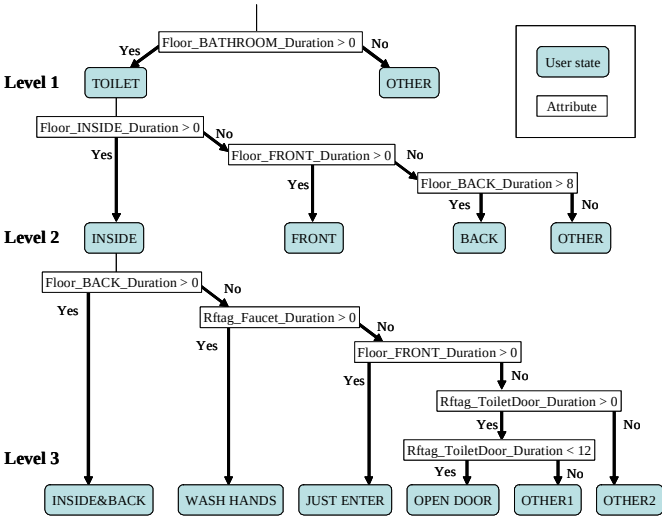

Fig. 13. Decision tree constructed in the experiment.

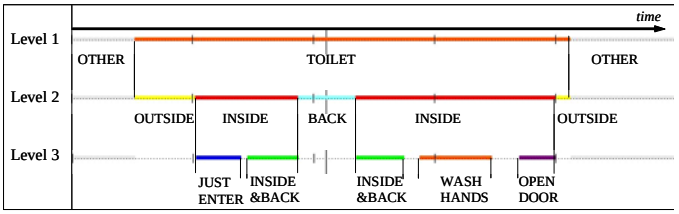

Fig. 14. User state decision for each time interval.

abstraction levels are extracted by using the C4.5 algorithm. These user states, extracted as leaves of decision tree, are detectable by using sensors in the experimental environment. According to the decision tree and application requirements, the user instantiates the appropriate user states at the appropriate abstraction levels in all required context model elements. However, this preliminary experiment considers only simple activity. As the next step, we will confirm the validity of decision trees constructed from more complex activities.

VII. Conclusion

In this paper, we proposed a framework for context-aware systems that allows context-aware applications adapt to the difference in context detection capability present in each environment. A user activity context description, which has multiple abstraction levels in terms of user context, is used. It is instantiated according to the level of context detection capability in each environment and application requirements. We also presented a user activity context detection method that can detect user context by using a multi spatio-temporal description of measured sensor data, a description of detected context with multiple levels of abstraction, and an order sensitive description of required context model associated with an application. The description of detected context takes the form of a decision tree. It is constructed automatically by the C4.5 algorithm.

We introduced a ubiquitous experimental house in which various human activities can be acquired from the data of embedded sensors. We performed a preliminary experiment in this house in which a decision tree that detects context with multiple abstraction levels was automatically constructed from actual human activity data; the results were shown to be reasonable.

Issues for further study include evaluating the performance of automatic decision tree construction with respect to more complex situations, and modeling the required context descriptions of more complex tasks including unordered state series. It would also be worthwhile to evaluate the feasibility of our prototype system by adapting it to support real applications.

REFERENCES

[1] D. Saha, A. Mukherjee, "Pervasive Computing: A Paradigm for the 21st Century," IEEE Computer, IEEE Computer Society Press, pp. 25-31, March 2003.

[2] Dey, A. K., Abowd , G. D. , "Towards a Better Understanding of Context and Context-Awareness," Proc. of the CHI 2000 Workshop on The What, Who, Where, and How of Context-Awareness, The Hague, Netherlands, April 2000.

[3] Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M.: Cyberguide, "A mobile context-aware tour guide," Wireless Networks: special issue on mobile computing and networking: selected papers from MobiCom .96, Vol. 3, No 5. (1997) 421-433

[4] Cheverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C., "Developing a Context-Aware Electronic Tourist Guide: some issues and experiences, " Proc. CHI, The Hague, Netherlands, April 2000

[5] Broadbent, J., Marti, P., "Location-Aware Mobile Interactive Guides: Usability Issues," Proc. Inter. Cultural Heritage Informatics Meeting, Paris, France, 1997

[6] Abowd, G. A. Bobick, I. Essa, E. Mynatt, and W. Rogers, "The Aware Home: Developing Technologies for Successful Aging," Workshop held in conjunction with American Association of Artificial Intelligence (AAAI) Conference 2002, Alberta, Canada, July 2002.

[7] S. Stillman, R. Tanawongsuwan, and I. Essa, "A system for tracking and recognizing multiple people with multiple cameras," in Proc. 2nd International Conference on Audio-Vision-based Person Authentication. 1999.

[8] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Who, When, Where, What: A real time system for detecting and tracking people," In Third International Conference on Automatic Face and Gesture. Nara, 1998.

[9] H. Ishiguro, T. Nishimura, T. Sogo and R. Oka, "VAMBAM: View and Motion-based Aspect Models for Distributed Omnidirectional Vision Systems," Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01), No.651, 2001.

[10] J.W. Davis and A.F. Bobick, "The representation and recognition of action using temporal templates," In Proc. the IEEE Conference on Computer Vision and Pattern Recognition, pages 928–934. IEEE Press, 1997.

[11] X. Sun and C.W. Chen, "Probabilistic motion parameter models for human activity recognition," IEEE International Conference on Pattern Recognition (ICPR02), 1, 2002.

[12] M. Mozer, "The neural network house: an environment that adapts to its inhabitants," In Proceedings of the AAAI Spring Symposium on Intelligent Environments, Technical Report SS-98-02, pp.110–114, AAAI Press, Menlo Park, CA, 1998.

[13] M. Mozer, "Lessons from an adaptive house," In D. Cook & R. Das (Eds.), *Smart environments: Technologies, protocols, and applications*. J. Wiley & Sons, 2004.

[14] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," PhD thesis, University of California, Berkeley, 2002.

[15] H. Kautz, O. Etziono, D. Fox, and D. Weld, "Foundations of assisted cognition systems," Technical report cse-02-ac-01, University of Washington, Department of Computer Science and Engineering, 2003.

[16] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan, "The cricket location-support system," In *Proceedings of MOBICOM 2000*, pages 32–43, Boston, MA, August 2000. ACM, ACM Press.

[17] R.Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems*, 10(1), 1992.

[18] A. Garg, V. Pavlovic, and J.M. Rehg, "Audio-visual speaker detection using dynamic bayesian networks," IEEE International Conference in Automatic Face and Gesture Recognition, 2000.

[19] R. Quinlan, "C4.5: PROGRAMS FOR MACHINE LEARNING," Morgan Kaufmann Publishers Inc., 1993.

[20] Y. ISODA, S.KURAKAKE, and K. Imai, "Ubiquitous Sensor based Human Behavior Recognition using a Spatio-temporal Representation of User States," International Journal of Wireless and Mobile Computing (IJWMC), Special Issue on Applications, Services, and Infrastructures for Wireless and Mobile Computing (accepted in Dec 2004)

Yoshinori ISODA Senior Research Engineer, Network Laboratories, NTT DoCoMo, Inc. He received the master's degree from the Department of Systems Engineering, the University of Osaka in 1993. Since joining NTT in 1993, he has been researching sensor processing systems and ubiquitous computing systems. He is a member of the Information Processing Society of Japan (IPSJ) and the Robotics Society of Japan.

Shoji KURAKAKE (M'00) Executive Research Engineer, Network Laboratories, NTT DoCoMo, Inc. He received the master's degree from the Department of Mathematical Engineering and Information Physics, the University of Tokyo. Since joining NTT in 1985, he has been researching character recognition, image analysis, video media handling, seamless applications, and ubiquitous services. He is a member of IEICE, IEEE, and the Association for Computing Machinery (ACM).

Kazuo IMAI (M'84) Executive Director of Network Laboratories, NTT DoCoMo, Inc. He received the B.Sc. and M.Sc. degrees from Kyoto University in 1974 and 1976, respectively. Since he joined NTT Laboratories in 1976, he has been engaged in various R&D activities including digital data exchange systems design, ISDN user-network interface definition, and broadband switching and access systems research. Most recent activities at NTT were to lead the research into and implementation of a network services platform for NTT's information sharing services. Since April 2000, he has been with NTT DoCoMo where he leads the next generation networking research for mobile communications.