User-Centric Situation Awareness in Ubiquitous Computing Environments

Gautham Pallapa[1] and Sajal K. Das[2]

[1] West Corporation, Omaha NE 68154, USA
gpallapa@west.com
[2] University of Texas at Arlington, Arlington TX 76019, USA

Abstract. The rising popularity of ubiquitous computing has resulted in a paradigm shift from generic to user-centric solutions, requiring seamless integration of heterogeneous devices and sensors in the environment to constantly monitor and perform tasks traditionally performed by the user. There is a considerable push, therefore, to develop systems capable of perceiving user behavior, and adapting to their idiosyncrasies. In this paper, we discuss limitations of the interpretations of context, and extend them for improved context awareness. We discuss a user-centric approach to perception of activity in the environment, and use the obtained knowledge in understanding user activities. We present a system for perceiving situations, and discuss an approach to empower the user to develop complex, yet intuitive, rules. We evaluate the performance of the system in a dynamic ubiquitous environment.

Keywords: Ubiquitous Computing, Situation, Context, User-centric.

1 Introduction

Ubiquitous computing represents the concept of seamless *"everywhere"* computing and aims at embedding technology unobtrusively within everyday devices [1], by providing appropriate services and information to users based on their current position (location), current applications running on user devices (state), or activities performed in the environment (situation awareness). A ubiquitous system should perceive appropriate user context, and develop dynamic rules and policies customized to user behavior, while reducing user interaction. Behavior can be described by a finite number of activities, characterized by entities playing particular roles and being in relation within the environment. Indeed, it is imperative to understand the potential relationship between computation and context, resulting in the need for effective situation perception.

In this paper, we present a user-centric system for capturing user behavior and activity in a ubiquitous computing environment. A new definition of context, focused on activity is proposed, and a new structure called a Situation Tree is developed to represent context, devices, and actions. Our system dynamically adapts to user behavior, and empowers users to customize the system according to their requirements, intuitively, yet efficiently. The performance of the system is analyzed with two complex user activities.

N. Meghanathan et al. (Eds.): CCSIT 2012, Part II, LNICST 85, pp. 543–552, 2012.

2 Motivation and Related Work

Human behavior is defined by a finite number of states called situations, characterized by entities playing particular roles within the environment. Perceptual information from different environmental sensors is associated to situations, roles, and relations, connected within a network, where a path describes behavior. Human behavior and needs evolve over time, requiring an adaptive context model.

Consider the following scenario: *John is a patient in an assisted environment, and his physician uploads the regimen to the system and monitors his progress remotely, with system alerts to remind John to take his medicine when needed. Based on his recovery, John's physician changes his regimen or medication, and the system now alerts John accordingly. The system also registers usage of medication, and informs John to fill his prescription in advance.* This scenario incorporates concepts of remote and local patient monitoring, handling sensitive information, and pro-active predictive actions performed by the system.

Consider another scenario: *Mary wants to try a recipe and accesses the ingredients from the refrigerator and pantry. Some of the items required are depleted during preparation and Mary makes a note to add them to the grocery list. After cooking, she finds that she wants to store the recipe for future reference.* Normally, Mary would file the recipe for future reference, and add the depleted items to her grocery list. At a future date, she would need to manually search for the recipe and check if necessary ingredients are available for preparation. It would benefit the user if the system could accomplish all these tasks with a minimal amount of intervention.

The system should perceive user situations, and perform most of the tasks, reducing user interaction. Recognizing daily activities is challenging, especially in a home or assisted environment [2, 3], since users can perform these activities in several ways. Underlying sensors must report features robustly across various sensing contexts [4]. Human activity recognition in the context of assisted environments using RFID tags has been investigated in [5, 6].Though this approach involves extensive tagging of objects and users, it demonstrates that hierarchical organization of probabilistic grammars provides sufficient inference power for recognizing human activity patterns from low level sensor measurements. A sensing toolkit for activity detection and recognition has been discussed in [7]. Systems deployed in ubiquitous environments are characterized by multiple smart cooperating entities and perform high-level inferencing from low-level sensor data reporting [8, 9]. Presence of such heterogeneous sensors and devices drives the need for appropriate perception of situations.

3 Context

Context is defined as *"any information relevant to an interaction that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves"* [10]. This implies

that context only exists if there is an interaction between the user and the application, limiting context to an occurrence of an event. If the user is sleeping and not interacting with any application, we would lose valuable information of the context (sleeping). In a pressure sensitive floor, when no one walks or sits, absence of a user is still information. Context therefore, should not be just an interaction between the user and application, but any information obtained from the user actions (or inactions) with respect to an application.

Context is more than just data or information - it is knowledge. We define these three terms in the following way: *Data* is just an informal piece of information without explicit structure or format. *Information* is interpretation of informal pieces of data which are associated with a particular context. When contextual information is interpreted and understood, we then have *knowledge.*

We define Context of an entity as "*a collection of measured and inferred information of a state or environment in which the entity interacts, either passive or active, to enable perception of the current state of the entity from the obtained knowledge, which influences inference of the future states*". Our definition stresses upon information collected from various sensors, and inherent information obtained from reasoning about the state or environment, which form knowledge. We use this knowledge to perceive the state of the entity and consider decision making as a function of prior context about an entity, allowing us to predict future states.

4 Perceiving Situation

Consider the environment shown in Figure 1(a). Let $S = \{s_1, s_2, \ldots, s_m\}$ sensors be distributed in this environment. Each sensor monitors a zone around it, calculated in the following manner: Draw a straight line connecting a sensor and its neighbor. The perpendicular bisector of this line forms the edge demarcating the zones of these adjacent sensors. If a wall is encountered within the zone, then that wall forms the edge of the zone for the sensor.

Definition 1. ***Context Element***

A context element c_i contains the information from sensor s_i. Therefore, $C = \{c_1, c_2, \ldots, c_m\}$ contains the data from m sensors distributed in the environment.

Let $D = \{d_1, d_2, \ldots, d_k\}$ be k devices present in the environment. We define a device as an object in the environment which the user accesses or interacts with. Sensors and devices are collectively called nodes, and assume n nodes in the environment, where $n = m + k$.

In the initial discovery phase, location of all nodes is obtained and a Minimum Spanning Tree (MST) is calculated, to enable tracking of user activity (Figure 1(b)). We represent a user entering and leaving a zone as an edge in the MST. If an edge is not found, then we log the activity, and upon repeated usage of that path, we append it to the MST and remove the existing path between the two nodes. When a user enters a node's zone, we assume that the node generates a context element and transmits it to the system.

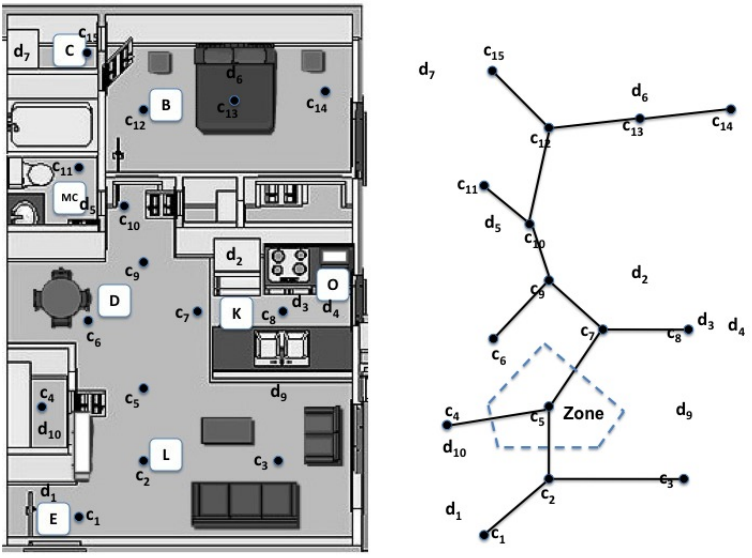

Fig. 1. Ubiquitous computing environment of our system showing (a) Floor plan and distribution of nodes, (b) Minimum Spanning Tree and calculation of a zone

Definition 2. ***Situation***

A situation $\gamma(t)$ *is a sequence of context elements* $c_1, c_2, \ldots$, *terminated by a device at time* t. *In other words,*
$\gamma(t) = c_1 c_2 \ldots, c_i d_j$, *where* $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$ *represents a situation at time* $t, \gamma(t) \in \Gamma$.

Context elements correspond to non-terminal, and devices correspond to terminal symbols respectively. As the user moves in the environment, context elements corresponding to zones in which the user traverses are obtained, with elements processed in an online manner.

4.1 Capturing Action

In order to capture user activity, we associate action words or "*verbs*" to each node. Let v_i be a verb associated with node i. Each type of node is assigned verbs according to their capability. For instance, a sensor capturing user motion would be assigned verbs "walk", "stand", and "sit". If we assume that a normal person walks 1 m/s, and average house size is 200 m^2, the user would enter an exit a zone once per second on an average. The nodes, therefore, would need to report with a very low frequency (about 1 Hz). When the user is within the zone of node i, we assign v_i to it, where the verb would correspond to "walk". If the user is still in the same zone after 2 reporting cycle, we upgrade the verb as $v_i^{'}$ = "*stand*$''$". Some

verbs associated with devices are "switch on", "switch off", "access", "replace", etc. In order to differentiate between verbs of the context element and verbs of the device, let $V = \{v_1, v_2, \ldots, v_p\}, p \leq m$ correspond to verbs associated with context elements and $A = \{a_1, a_2, \ldots, a_p\}, q \leq k$ correspond to the set of verbs associated with devices. A situation in our approach is interpreted as an activity in the environment, and can be represented as *user* (*subject*) → *verb* (*action*) → *environment* (*devices*, *context elements*).

4.2 Initial Configuration

Initially, the system has to be trained to perceive user activity. Consider an arbitrary user activity pattern where the system informs the user to take medication. The user, currently in the living room, gets up and moves to the bathroom, via the bedroom, and accesses the medicine cabinet (d_5). This activity is represented by sequence $c_2v_1c_5v_1c_9v_1c_{10}v_1c_{11}v_1d_5v_2$, with the user's initial position as a location in zone of c_2. The system obtains the first (context, verb) pair and compares it with the next (context, verb) pair in the sequence. Since verbs in both the pairs are similar, it uses the following rule:

Rule 1 ***Simple Contraction:*** *Sequence $xvyv, x, y \in N$ can be represented as $(x, y)v$. This rule is commutative, i.e., $vxvy, x, y \in N$ is represented as $v(x, y)$.*

The system contracts the first two (context, verb) pairs and compares this with the next (context, value) pair observed, Rule 2 is then used.

Rule 2 ***Compound Contraction:*** *Sequence $(c_1, c_2)vc_3v, \{c_1, c_2, c_3 \in C\}$ is contracted to $(c_1, c_3)v$. This rule is commutative, contracting the sequence to $v(c_1, c_3)$, eliminating redundant context when identical action is performed over multiple context elements.*

Rule 3 ***Device Listing:*** *Sequence $v(d_1, d_2)vd_3, \{d_1, d_2, d_3 \in D\}$ contracts to $v(d_1, d_2, d_3)$. This rule captures devices that the user has interacted with.*

The system continues contracting the sequence till we obtain $(c_2, c_{11})v$. The Situation Tree is constructed upon encountering terminal symbol d_5.

Definition 3. *An activity is considered complete when any situation s_i, terminating at device d_i with verb v is immediately followed by a situation s_{i+1} terminating at the same device, but with verb v'.*

Rule 4 ***Complement Rule:*** *Sequence terminating with $v(d_1, d_2, \ldots, d_i), \{d_i \in D\}$, is constructed until complement $v'(d_1, d_2, \ldots, d_i), \{d_i \in D\}$ is encountered. This rule ensures that any device accessed/switched-on is replaced/switched-off.*

Rule 5 ***Activity:*** *An activity is defined as usage of a device d_i if situation γ_i, terminating at device d_i with verb $v =$ "access/switch on" is immediately followed by a situation γ_{i+1} terminating at d_i with verb $v' =$ "replace/switch off".*

For situations with activities between accessing and replacing a device (for e.g., talking on a phone while cooking), the sequence is decomposed into: activity performed up to device access, intermediate activity, and device release.

Rule 6 ***Decomposition:*** *If a sequence terminates with a device or a set of devices, with the subsequent verb not a complement of the prior verb, construct a new sequence for the current activity, until the complement is encountered.*

Using these rules, sequences are represented using a structure called a *Situation Tree*. The Situation Tree ($S-Tree$) is a binary tree constructed bottom-up, from the sequence of context elements, verbs and devices, and possesses the following properties:

Property 1. *The root of any subtree of a S-Tree is always a verb.*

Property 2. *The left child of any verb is non-terminal(context element or verb).*

Property 3. *The right child of the root is either terminal (device) or a subtree of terminals.*

Property 4. *The right child of any intermediate verb, whose parent is not its complement, is a context element.*

Property 5. *The right child of any intermediate verb, whose parent is its complement, is a terminal or a subtree of terminals.*

Property 6. *The left subtree of any intermediate verb represents prior activity.*

4.3 Designing Complex Rules

In order to resolve perceiving current state of user activity, many systems incorporate a form of "$Event-Condition-Action$"(ECA) rules to perform actions based on event triggers. An example of an ECA rule is given below:

```
rule: "Cooking_Rule":
Event: (location == "Kitchen")
Condition
    (device == "Oven") &&
    (status == "On")
Action:
  assign activity == "Cooking"
```

ECA rules, however, tend to become complex and require manual decomposition and chaining of logical operations to encompass multiple events, reducing their user-friendliness. Additionally, since events trigger an action, conditions might not consider prior user activity (history). For instance, in the activity discussed in Section 4.2, developers might choose to discard user movement from c_2 to c_{10}, and focus on context information obtained from c_{11} onwards, resulting in loss of information, essential in understanding user behavior for situation prediction.

Our system improves user interaction and allows the user to specify custom rules naturally. Assume that the user needs to create a rule to turn the television on when she walks from the bedroom to the living room. Using ECA rules would involve initial location as "Bedroom", final location as "Living room", and a series of operations to include the activity of walking. Our system handles this in a graceful manner. The user enters the rule without decomposition as "If user walks from Bedroom to Living room, turn on the television". The system identifies the subject is the user and the rest of the rule, "walks from Bedroom to Living Room, turn on the television" is the activity. It then parses the rule sequentially. The first word is a verb "walk" v_1 followed by keyword "from". From Rule 2, it obtains the next two elements c_{12}, and c_3, and constructs the sequence $(c_{12}, c_3)v_1$. It then looks up the MST (Figure 1(b)) and expands the sequence to $c_{12}c_{10}c_9c_7c_5c_2c_3v_1$. The segment "turn on ($v_2$) the television" is then translated to v_2d_9 and appended to the initial sequence. After parsing the rule, therefore, we obtain the situation $\gamma(t) = c_{12}c_{10}c_9c_7c_5c_2c_3v_1v_2d_9$.

Suppose the user now moves from the bedroom to the living room along a different path $c_{12}c_{10}c_9c_5c_3$. The sequence obtained would be $c_{12}v_1c_{10}v_1c_9v_1c_5v_1c_3v_1$. Using Rules 1 and 2, the sequence still reduces to $(c_{12}, c_3)v_1$, and the system turns the television on. It also registers the new path taken by the user, and upon frequent usage, the system perceives that this is a preferred path, and updates its spanning tree. The system also observes user behavior, and develops dynamic rules based on user history. The advantages of this approach are two-fold: (1) The system allows the user to create user-friendly rules (2) The system can be dynamically customized to user behavior and idiosyncrasies.

5 Analysis

In this section, we analyze our system's performance using the cooking scenario described in Section 2. Mary has customized a rule in the system as "Store a new recipe". Our system is made to understand the meaning of "recipe" by defining a recipe with the following steps:

1. **Access** ingredients from the refrigerator and/or pantry.
2. **Spend** time at the kitchen counter preparing the ingredients for cooking
3. **Cook** the ingredients by **spend**ing time at the stove with the stove on
4. **Switch** off the stove and **move** dish to the dining table.

We assume that all users are honest. A honest user is one who accesses any device or item with the intent of using the device or item. A single usage is equal to one unit of the item consumed or one instance of the device being used. We have also limited recipe generation to entering the sequence of steps observed by the system, and additional nuances such as *stirring*, *sautéing*, *etc.*, along with quantities of the ingredients are additional user input.

Let us assume that Mary was initially in the zone of c_6 (Figure 1(a)), moves to the kitchen, and accesses ingredients in the refrigerator and pantry. Let d_2 and d_3 correspond to the refrigerator and pantry respectively, Let $\{d_{2-1}, d_{2-2}, d_{2-3}, d_{2-4}\}$

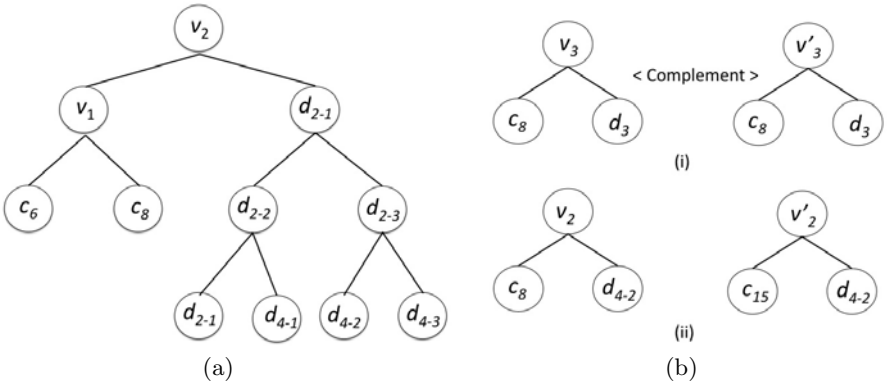

Fig. 2. Structure of S-Trees (a) After accessing ingredients, (b)(i) After switching off the stove, (b)(ii) When d_{4-2} is misplaced

and $\{d_{4-1}, d_{4-2}, d_{4-3}\}$ be the ingredients accessed, and verbs v_1 = "$walk''$, v_2 = "$access''$. Initially, the system monitors Mary's activity and observes her movement from c_6 to c_8, and terminates the situation when she accesses d_{2-1}. The initial sequence is therefore, represented as

$$\gamma_1 = c_6 v_1 c_7 v_1 c_8 v_1 c_8 v_2 d_{2-1} \tag{1}$$

Since the situation terminated with a device, the system, now observes the next (context, verb) pair, to satisfy the Complement Rule (Rule 4). Instead, when it encounters another device (d_{2-2}), it perceives that multiple devices are being accessed, and therefore, follows Rule 3. It simultaneously continues monitoring for a (context, verb) pair satisfying Rule 4, until:

$$\gamma_2 = c_8 v_2 d_{2-1} c_8 v_2 d_{2-2} c_8 v_2 d_{2-3} c_8 v_2 d_{2-4} c_8 v_2 d_{4-1} c_8 v_2 d_{4-2} c_8 v_2 d_{4-3} c_8 v_3 d_3 \tag{2}$$

The situation at this stage is given by $\gamma(t) = \gamma_1 + \gamma_2$. When the system encounters $c_8 v_3 d_3$, it perceives that the user has started a different activity. Therefore, from Rule 6, it creates a new S-Tree, and continues monitoring. The S-Tree for $\gamma(t)$ is constructed after applying Rules 1, 2, 3 is represented in Figure 2(a).

When Mary finishes cooking and switches off the oven, the sequence $c_8 v_3' d_3$ is encountered. The system then uses Rules 4, and 5 to ascertain that the activity of cooking has been completed (Figure 2(b)(i)). When Mary places the dish on the dining table, the sequence $c_8 v_1 c_7 v_1 c_6 d_8$ signals the system that the recipe is complete and can be filed in the system. When Mary starts replacing the ingredients, the system deletes the corresponding node in the S-Tree until no right child exists, and the S-Tree is cleared, as the system perceived that all tasks are completed.

Occasionally, the user might misplace an ingredient, and the system informs the user accordingly. Let us assume that Mary misplaced ingredient d_{4-2} in the closet(c_{15}) instead of the pantry. The system now has a situation where "access"

of d_{4-2} was at c_8 but "replace" was at c_{15}, as depicted in Figure 2(b)(ii). The system would then generate the complement of the expected replace as $c_8 v_2' d_{4-2}$, and prompts Mary to "replace d_{4-2} in d_4". Mary then accesses d_{4-2}, thereby deleting $c_{15} v_2 d_{4-2}$ (using Rule 4), and when the ingredient is replaced in the pantry, all S-Trees related to the activity are cleared.

6 Evaluation

We simulated the environment shown in Figure 1(a) with up to 100 sensors and 200 devices. We initially trained the system with 10 scenarios representing average daily user activity. We considered a total of 25 verbs, over 10 types of sensors to describe possible actions. We then simulated user movement, randomizing the path to introduce perturbations in generated sequences. Additionally, we altered user path after every 500 runs, to observe our system's adaptation to new user behavior. We conducted 10000 simulated runs and present an average of the results obtained.

The effect of the number of verbs on false positives/negatives is shown in Figure 3. We observed that our system is not affected by false positives, though a significant increase in false negatives was observed with increase in the number of verbs. This could be attributed to the number of verbs assigned to the types of sensors. For instance, it is difficult for our system to differentiate between a user standing, or sitting in a location. Resolution of ambiguity required additional input from surrounding sensors.

We then varied the number of context elements with a fixed set of 10 verbs. Figure 4 depicts the average of 10000 test runs. While occurrence of false positives and negatives were comparable, increasing the verbs to 25 resulted in a significant increase in false negatives. Supplementing context information from surrounding sensors resolved ambiguity and improved situation perception.

Number of verbs	Runs = 5000		Runs = 10000	
	False Positives	False Negatives	False Positives	False Negatives
5	1	2	1	1
10	1	4	2	9
15	2	11	3	28
20	4	26	7	68
25	7	47	11	122

Fig. 3. Effect of verbs on False Positives and Negatives

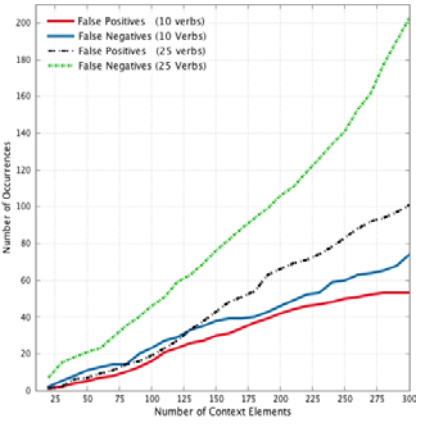

Fig. 4. Effect of context elements and verbs over false positives and negatives

7 Conclusion

In this paper, we presented a user-centric system for capturing user behavior and activity in a ubiquitous computing environment. We discussed limitations of current definitions of context, and proposed a definition of context focused on activity. Situation Trees were developed to represent context, devices, and actions. We investigated rules required for perceiving situations and evaluated our system with two complex user activities. We showed how our system dynamically adapts to user behavior, and empowers users to customize the system according to their requirements, intuitively, yet efficiently. Situation Trees are envisioned to construct dynamic situation grammar customized to user behavior and history of user activity, and facilitate developers and users to create complex context-aware rules effortlessly to handle diverse scenarios in a ubiquitous computing environment.

References

1. Weiser, M.: The future of ubiquitous computing on campus. Comm. of the ACM 41(1), 41–42 (1998)
2. Bell, G., Dourish, P.: Yesterdays tomorrows: notes on ubiquitous computings dominant vision. Personal Ubiquitous Computing 11(2), 133–143 (2007)
3. Wang, S., Pentney, W., Popescu, A., Choudhury, T., Philipose, M.: Commonsense-based Joint Training of Human Activity Recognizers. In: Proc. 21st Intl. Joint Conf. on AI (2007)
4. Ermes, M., Parkka, J., Mantyjarvi, J., Korhonen, I.: Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions. IEEE Trans. on Information Technology in Biomedicine 12(1), 20–26 (2008)
5. Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H., Hahnel, D.: Inferring activities from interactions with objects. IEEE Pervasive Computing 3(4), 50–57 (2004)
6. Tapia, E.M., Intille, S.S., Larson, K.: Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 158–175. Springer, Heidelberg (2004)
7. Wimmer, R., Kranz, M., Boring, S., Schmidt, A.: A Capacitive Sensing Toolkit for Pervasive Activity Detection and Recognition. In: Fifth Annual IEEE Intl. Conf. on Pervasive Computing and Communications, pp. 171–180 (2007)
8. Naeem, U., Bigham, J.: Activity recognition using a hierarchical framework. In: Second Intl. Conf. on Pervasive Computing Technologies for Healthcare, Pervasive Health 2008, pp. 24–27 (2008)
9. Osmani, V., Balasubramaniam, S., Botvich, D.: Human activity recognition in pervasive healthcare: Supporting efficient remote collaboration. J. Netw. Comput. Appl. 31(4), 628–655 (2008)
10. Dey, A.K.: Understanding and using context. Personal Ubiquitous Computing 5(1), 4–7 (2001)