Noname manuscript No.
(will be inserted by the editor)

Discovering Frequent User-Environment Interactions in Intelligent Environments

Asier Aztiria · Juan Carlos Augusto · Rosa Basagoiti · Alberto Izaguirre · Diane J. Cook

Received: date / Accepted: date

Abstract Intelligent Environments are expected to act proactively, anticipating the user's needs and preferences. To do that, the environment must somehow obtain knowledge of those need and preferences, but unlike current computing systems, in Intelligent Environments the user ideally should be released from the burden of providing information or programming any device as much as possible. Therefore, automated learning of a user's most common behaviors becomes an important step towards allowing an environment to provide highly personalized services.

In this paper we present a system that takes information collected by sensors as a starting point, and then discovers frequent relationships between actions carried out by the user. The algorithm developed to discover such patterns is supported by a language to represent those patterns and a system of interaction which provides the user the option to fine tune their preferences in a natural way, just by speaking to the system.

Keywords Ambient Intelligence · intelligent environments · pattern learning · machine learning techniques

1 Introduction

Ubiquitous Computing, a term introduced by Mark Weiser (Weiser, 1991), refers to a paradigm in which a new type of relation between users and technology is established

Asier Aztiria
University of Mondragon. E-mail: aaztiria@eps.mondragon.edu

Juan Carlos Augusto
University of Ulster. E-mail: jc.augusto@ulster.ac.uk

Rosa Basagoiti
University of Mondragon. E-mail: rbasagoiti@eps.mondragon.edu

Alberto Izaguirre
University of Mondragon. E-mail: aizagirre@eps.mondragon.edu

Diane J. Cook
Washington State University. E-mail: cook@eecs.wsu.edu

such that technology is both widespread and transparent to the user. Now computing devices of various types are all around us, embedded in different objects we interact with and in that way they influence our lives. An important further development of this concept has resulted in concepts such as Ambient Intelligence (AmI) (Aarts, 2004; Ducatel et al, 2001; Cook et al, 2009; Nakashima et al, 2009) or Intelligent Environments (Callaghan et al, 2009). Other terms, such as Smart Environments (Cook and Das, 2005) or Pervasive Computing (Friedemann and Mahmoud, 2002), are used with similar connotations. All them refer to digital environments that proactively, but sensibly, support people in their daily lives (Augusto, 2007). As mentioned by Augusto (Augusto, 2009) 'In order to be sensible, a system has to be intelligent. That is how a trained assistant, e.g. a nurse, typically behaves. It will help when needed but will restrain to intervene unless is necessary. Being sensible demands recognizing the user, learning or knowing her/his preferences and the capability to exhibit empathy with the user's mood and current overall situation'. In Intelligent Environments learning means that the environment must gain knowledge about the preferences, needs and habits of the user in order be in a better position to assist the user adequately (Galushka et al, 2006).

Let us consider the following scenario, which exemplifies the common behavior of a user. *On weekdays Michael's alarm clock goes off ('Alarm on') few minutes after 08:00AM. Approximately 10-15 minutes after getting up he usually steps into the bathroom ('Bathroom on') and (2 seconds after) he turns on the lights of the bathroom ('BathLights on') if the bathroom is dark (bathroom light level <10). On Tuesdays, Thursdays and Fridays he usually takes a shower ('Shower on'); Michael prefers the temperature of the water to be around 24-26 degrees Celsius in the winter and around 21-23 degrees Celsius in the summer. When he finishes taking a shower and 10 seconds after he turns on the fan of the bathroom ('BathFan on'), only when the relative humidity level is >75%. Before he leaves the bathroom ('Bathroom off')he turns off the fan ('BathFan off') and the lights ('BathLights off').*

1.1 Acting based on predefined patterns vs Acting based on learned patterns

For an environment to be perceived as acting intelligently it has to offer non-obtrusive and personalized services which can effectively assist users in their daily lives. There already exist sensing systems which can act using predefined patterns, for example, turning on the lights when detecting someone's presence in a room or systems that turn on a fan in the bathroom when someone turns on the lights. Problem is that the systems operate with manually (and rigidly) predefined patterns. Thus, a system that turns on the lights of the bathroom when Michael goes into the bathroom may satisfy his needs but a system that turns on the fan every time the lights are turned on would not, because he only turns on the fan after taking a shower and only if the relative humidity level is greater than 75%.

In order to provide personalized and adapted services, it is necessary to know preferences and frequent habits of users. Knowing users' frequent behaviours allows the environment to act intelligently and proactively. In Michael's case, it could mean that the environment automatically turns on and off the lights and the fan, sets the temperature of the water and so on. Unlike previous systems, *automation* of actions and/or devices in intelligent environments is based on learned patterns, making sure they adapt to users' common behaviours. The knowledge extracted from these patterns

can also be used in order to *understand* his behaviour. For example, the analysis of frequent interaction with objects and devices in the house can facilitate the detection of unhealthy habits (e.g., the system detects that Michael does not brush his teeth in the mornings). Making the environment more efficient in terms of *saving energy* (e.g. by turning on the fan only when he takes a shower) or increasing *safety* (e.g. turning off the water or issuing alarms when detecting that Michael left it on and he will not return soon) are other dimensions of daily life that can be supported by the Intelligent Environment thanks to the knowledge it has collected.

In order to achieve these objectives, we have developed software which allows an Intelligent Environment to discover frequent behavioral patterns, and we have named it Patterns of User Behaviour System (PUBS). The system is composed of several modules; at its core there is an algorithm, $\mathcal{A}_{PUBS}$, that discover behavioral patterns. The language $\mathcal{L}_{PUBS}$, included within PUBS, provides a standard framework to represent patterns clearly; finally, PUBS includes an interaction system, $\mathcal{I}_{PUBS}$, which allows the user to interact with PUBS in a natural way. The version of PUBS that we describe in this article is focused on discovering relationships between two simple actions; in other words, it only discovers what we call "one-to-one relationships". In Michael's example it would first discover that 10-15 minutes after he gets up he goes into the bathroom and then, as another pattern, it would discover that he switches on the light just after he goes into the bathroom. Besides discovering frequent relationships, PUBS is able to identify the usual time relations between both actions (e.g. he goes into the bathroom *10-15 minutes* after getting up), as well as to define the conditions under which a pattern appears (e.g. *if the bathroom relative humidity level is* $>75\%$). The essential components of the PUBS architecture are shown in Figure 1.

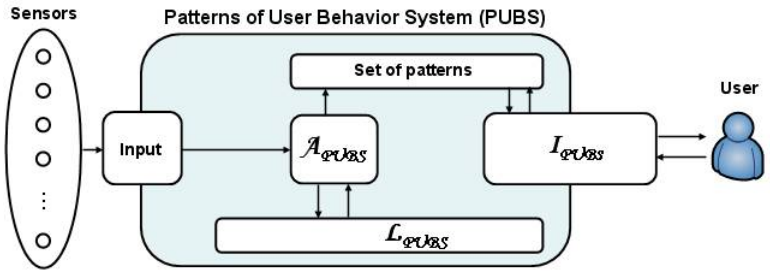

Fig. 1 Architecture of PUBS

The remainder of the paper is organized as follows. Section 2 describes the special features of Intelligent Environments to be considered when performing the learning process. Section 3 reviews the different approaches employed by different groups. In Section 4, we explain our approach (PUBS) for discovering frequent patterns, and Section 5 shows the results of our validation experiments. Finally, in Section 6 we provide some conclusions and plans for future work.

2 Intelligent Environments' Special Features

As mentioned above, unlike current computing systems, where the user has to learn how to use the technology, in Intelligent Environments the environment adapts its behavior to the user. This swapping of roles entails that Intelligent Environments have some special features that must be taken into account in the process of learning; we list these in the following sections.

2.1 Importance of the user

Users are the focus of any development in Intelligent Environments, and the fact that the environment is technologically rich must not translate into any extra effort on behalf of the users to obtain the benefits of an Intelligent Environment (Dooley et al, 2006; Muller, 2004). This implies that the data acquisition process and feedback obtaining process have to be carried out as unobtrusively as possible (Rutishauser et al, 2005); for example, by means of sensors installed in standard devices. The concept of an ideal proactive environment suggests an environment where the interaction with the user is carried out through standard devices such as switches or remote controls, but when this is not possible, friendly multimodal interfaces are considered (Turunen et al, 2007; Partala et al, 2006; Coen, 1998). In that sense, being aware of our system's need to interact with the user, we have developed a Human-Computer Interaction system ($\mathcal{I}_{PUBS}$) which facilitates interaction with the system without demanding that the user learn how to use PDAs, a computer mouse or keyboard or how to navigate through a complicated computer program.

2.2 Collected Data

The data collected from the sensors will influence the learning process globally; all patterns will depend upon the data captured. Therefore, the data collected from sensors demands a thorough analysis, covering many different factors that can influence the learning process.

Transformation of the collected data

Collecting data correctly is an important task in order to be able to extract meaningful information from that set of data.

The data will be collected in a continuous way from different information sources. Integrating data from different sources usually presents many challenges, because different sources will use different styles of record keeping, and different devices may also be of different types (e.g., digital vs analog). Moreover, as in other areas, noisy data, with missing or inaccurate values, will be common, so finding out how to deal appropriately with that is another obstacle.

It is also necessary to consider the meaning of the collected data. Sometimes the raw data will not be meaningful enough, and so a combination of different sources of raw data will be necessary in order to infer and recognize meaningful activities. For example, in order to infer "Michael has gone into the bathroom" we will have to combine the raw data of "There is a motion in the corridor", "RFID in the bathroom

door detects that Michael is passing through the door" and "There is motion inside the bathroom".

Spatio-temporal aspects

As we have mentioned above, in Intelligent Environments the interaction between the user and the environment must ideally be by means of standard devices. Thus, electronic items such as lights, television, kettle, etc., as well as furniture (sofa, bed, and so forth) are fitted with sensors. The location of those devices provide us valuable spatial information as to where a user's actions take place. Additionally, other sensors, such as motion sensors, which can be distributed in all rooms, can also provide valuable information that help us to identify broad regions (such as the kitchen) where the activities take place.

It is important to understand that handling temporal information is one of the most challenging aspects in Intelligent Environments. This is not only because the user's actions happen during a specific time, but because most often, the user's actions are themselves related in terms of time (for instance, Michael goes into the bathroom 10-15 minutes after he gets up).

Nature of the collected data

Finally, one of the most important factors to consider is the nature of the data once it is collected and transformed. Different types of sensors provide information of different types and that can be used for different purposes in the learning process. Thus, our system considers three main different groups of sensors.

- (type O) Sensors installed in objects (devices, furniture, domestic appliances, etc). These sensors provide direct information about the actions of the users. For example, a sensor installed in the bedroom lamp may indicate when that lamp was switched on and off.
- (type C) Context sensors. These sensors provide information about context, but not about user actions directly. Temperature, light and smoke sensors are examples of type C sensors.
- (type M) Motion sensors. These sensors can be used to infer where the user is (in the bedroom, outside the house, or elsewhere).

It is worth mentioning that we are aware there are, additionally, other types of sensors, such as those that indicate the health status of the user or alarm pendants; these will be included in future versions of the system.

2.3 Representation of patterns

Learned patterns can be used for many different purposes. In that sense, different Machine Learning techniques provide different types of outputs, of which some, such as artificial neural networks, produce output that is almost impossible to translate into humanly understandable language. If learned patterns are used in order to understand behaviors of the user (to detect bad or unhealthy habits, for instance) or if it is necessary for the system to explain its decisions to the user, it will be essential that the output is comprehensible.

Comprehensible representations can be achieved in different ways. So far, the most common approach relates the actions of the user (gathered by O-type sensors) with

the status of the environment (gathered by C-type sensors) (Gal et al, 2001) (Hagras et al, 2004). We consider, as have other authors (Duman et al, 2008) (Jakkula et al, 2007), that detecting relationships between the user's actions facilitates understanding of the user's behavior, using information about the environment to contextualize such a relationship. Besides facilitating the understanding of the user's behavior, finding these relationships also allows us to use relative time references instead of absolute times. Thus, in Michael's case, we relate the action of "finishing to take a shower" with the action of "turning on the fan", and we define a time relation of "10 seconds" if and when "the bathroom relative humidity level is >75%".

3 Related Work

Learning is an essential feature in any Intelligent Environment. However, given the diversity of elements that must converge in order to obtain the infrastructure needed for an Intelligent Environment, learning has not received as much attention in the literature as it may require. Some notable exceptions are mentioned below.

Mozer et al. (Mozer et al, 1995) and Chan et al. (Chan et al, 1995) were amongst the first reports on applications for AmI environments in which user patterns were considered. In the case of Mozer et al., the aim of their system (installed in the Adaptive House) was to design an adaptive control system that considers the lifestyle and energy consumption of the inhabitants. For that, they used a feed-forward neural network to develop an occupancy predictor and zone anticipator, which were used to predict where the user would be in the coming seconds and control lighting based on those predictions. Chan et al. developed a similar application in order to asses if a given situation was normal or abnormal. Other authors have kept using ANNs in order to provide personalized services. Thus, Campo et al. (Campo et al, 2006) developed a system that calculated the probability of occupation for each area of the house and systematically compared the probability with the current situation. A survey of those works can be found in (Begg and Hassan, 2006). The use of ANNs has a limitation related to their black box nature; therefore their internal structure is not comprehensible.

The group that has been working on the MavHome and Casas projects is one of the most active groups in this area. The first applications developed by this group were focused on building universal models, represented by means of Markov models, in order to predict either future locations or activities (Cook and Das, 2007). In this sense, they made notable improvements by developing applications to discover daily and weekly patterns (Heierman and Cook, 2002). Additionally, they constructed applications with the ability to infer abstract tasks automatically, identifying corresponding activities that were likely to be part of the same task (Rao and Cook, 2004). Jakkula and Cook (Jakkula and Cook, 2007) extended this work to predict actions using temporal relations, defined by means of Allen's temporal logic relations (Allen, 1984). They have also developed applications to recognize activities (Kim et al, 2010).

Researchers at Essexs iDorm lab have given prominence to the problem of learning and are one of the most active groups in this area. Their initial efforts (Hagras et al, 2004) (Doctor et al, 2005) were focused on developing an application that generated a set of fuzzy rules to represent users patterns. Recording the changes caused by the user in the environment, they generated membership functions as well as fuzzy rules that mapped data into fuzzy rules. Due to the excessive number of rules they generated, they introduced an improvement (Duman et al, 2008) where they identified relevant

and important associations between given actions, so that irrelevant aspects of the rules (and, by extension, some rules as well) were removed.

Other techniques have also been used. The group that works in the environment named SmartOffice (Gal et al, 2001) developed an algorithm that discovered conditions for situations in which examples indicate different reactions. Research conducted under the MyCampus project filtered messages based on the preferences of the user (Sadeh et al, 2005). Including a system, based on CBR, they significantly improved the quality of filtering (i.e., user satisfaction increased from 50% to 80%).

a survey of all these works can be found in (Aztiria et al, 2010). We can state that due to specific characteristics of Intelligent Environments, each problem calls for the use of certain techniques, but as Muller pointed out (Muller, 2004), "the overall dilemma remains: there does not seem to be a system that learns quickly, is highly accurate, is nearly domain independent, does this from few examples with literally no bias, and delivers a user model that is understandable and contains breaking news about the user's characteristics".

4 Patterns of User Behaviour System (PUBS)

PUBS is a system that discovers a user's common behaviors and habits from data recorded by sensors. In that sense, it is worth mentioning that PUBS only discovers one-to-one relationships, i.e. patterns where only two actions (no more) are involved, thus ruling out larger sets of actions.

As Figure 1 shows, PUBS is made up of three main modules($\mathcal{L}_{PUBS}$, $\mathcal{A}_{PUBS}$ and $\mathcal{I}_{PUBS}$) which will be explained below using on of the patterns associated with Michael's scenario:

"Michael turns bathroom fan on 10 seconds after he finishes to take a shower If the bathroom humidity level is >75%"

4.1 Representing patterns with $\mathcal{L}_{PUBS}$

Due to the complexity of Intelligent Environments, defining a language that allows us to represent learned patterns in a clear and non-ambiguous way is difficult but necessary. The language integrated within PUBS is based on ECA (Event-Condition-Action) rules (Augusto and Nugent, 2004). Besides providing a standard way of representing patterns, it makes sure those patterns are clearly specified and enables other technologies to check their integrity (Augusto and McCullagh, 2007).

As ECA rules, $\mathcal{L}_{PUBS}$ basically relates two situations (defined by ON and THEN clauses) and the specific conditions (defined by an IF clause) under which that relationship occurs. Finally, $\mathcal{L}_{PUBS}$ allows us to define the time relation between both actions. Considering the abovementioned pattern, using $\mathcal{L}_{PUBS}$ it would be represented as:

```
(Pattern 1)

 ON occurs (Shower, Off,t0)

 IF context (Bathroom humidity level (>,75%))

 THEN do (On, BathFan, t) when t=t0+10s
```

For the complete specification of $\mathcal{L}_{PUBS}$ see the Appendix.

Event Definition

The part of the pattern defined by the ON clause defines the event that occurred and triggered the relationship specified by the pattern. From this point on, the action that triggers the pattern will be called *associated sensor triggering (associatedSeT)*.

The components of the Event Definition are the *associatedSeT* (*'Shower'*), the nature of the action (*'off'*) and the timestamp of such an action ('t0'). As patterns relate user behaviors, the ON event must be the effect of a user's action. In that sense, there are two possibilities.

– User acts upon objects fitted with O-type sensors.

```
(Event 1)

 ON occurs (Shower, Off,t0)
```

– User's presence or movement (detected by M-type sensors) triggers the pattern.

```
(Event 2)

 ON occurs (Bathroom, On,t0)
```

Condition Definition

The IF clause defines the necessary conditions under which the action specified in the THEN clause is the appropriate reaction to the event listed in the ON clause. Due to the fact that is almost impossible that an Event-Action relation is true under any condition, appropriate conditions are necessary in order to represent accurate patterns. Below we provide some examples of conditions:

```
(Condition 1)

 IF context (Living room temperature (<,20 C))

(Condition 2)

 IF context (Time of day (>,20:30:00))

(Condition 3)

 IF context (Day of week (=, Tuesday))
```

Conditions are defined by means of attribute-value pairs. Whereas ON and THEN clauses define actions of the user, the conditions must specify the status of the environment at that moment, so that the information involved in that clause must be related to the context. Thus, we have considered two different possibilities to define the attribute.

– Information coming from C-type sensors (e.g. 'humidity level' (Pattern 1) or 'temperature' (Condition 1))
– Calendar information (e.g. 'time of day' (Condition 2) or 'day of week' (Condition 3))

As to the possible values for such attributes, they depend on the nature of each attribute. $\mathcal{L}_{PUBS}$ allows us to define two types of values.

– Qualitative values (e.g. 'Tuesday' (Condition 3)).
– Quantitative values (e.g. 20 C (Condition 1) or 20:30:00 (Condition 2)).

Action Definition

Finally, the THEN clause defines the action that the user usually carries out given the ON clause and given the IF conditions. It is made up of the triggered action, called *main sensor triggering (mainSeT)*, the nature of the action (on/off) and the time relation between the Event and Action situations.

Such a time relation can be either quantitative (Action 1) or qualitative (Action 2), the usefulness of each type of relation being different. Both define, although in different ways, the behavior of the user, but whereas quantitative relations can be used to automate *mainSeT*, qualitative relations cannot be used for this purpose, for in addition to knowing that one action follows another one we need to know the specific time relation between them.

```
(Action 1)

 THEN do (On, BathFan, t) when t=t0+10s

(Action 2)

 THEN do (On, BathFan, t)
 when t is after t0
```

4.2 Learning patterns with $\mathcal{A}_{PUBS}$

As core of PUBS, in accordance with $\mathcal{L}_{PUBS}$, we have developed the $\mathcal{A}_{PUBS}$ algorithm to discover patterns in data collected by sensors. The steps of the algorithm are summarized in the following algorithm:

$\mathcal{A}_{PUBS}$ Algorithm (for learning patterns)

---

for each sensor of type O (consider it as *mainSeT*)

Identify the *associatedSeT* of type O or M (See below)

for each *associatedSeT*

Identify possible time relations (See below)

if there exists a time relation **then** make it more accurate using temporal and context information (type C) (See below)

Emphasizing O-type sensors as *mainSeT* follows from the fact that those sensors provide us direct information about users' actions, so that discovering patterns about them we will discover patterns about users' actions.

Let us consider that a part of Michael's morning activities and context data collected by sensors for several days are:

Data Collected from Sensors
(time;type of sensor;sensor;status;value)

Day 1; (20-10-2009)
08:24:02;bHumidity;on;68
08:24:53;Shower;off;–
08:24:55;bHumidity;on;76
08:25:05;BathFan;on;–

Day 2; (22-10-2009)
08:32:41;bTemp;on;26
08:33:29;bHumidity;on;80
08:34:08;Shower;off;–
08:34:17;BathFan;on;–

Day 3; (23-10-2009)
08:19:47;bTemp;on;26
08:22:21;Shower;off;–
08:22:26;bHumidity;on;72
08:29:38;BathLights;off;–

Day 4; (27-10-2009)
08:38:08;bHumidity;on;70
08:40:58;bHumidity;on;78
08:41:03;Shower;off;–
08:41:13;BathFan;on;–

Day 5; (29-10-2009)
08:20:20;bHumidity;on;70
08:21:09;Shower;off;–
08:22:36;bTemp;on;24
08:23:12;BathLights;off;–

Day 6; (30-10-2009)
08:28:27;bTemp;on;23
08:29:02;bHumidity;on;77
08:29:11;Shower;off;–
08:29:20;BathFan;on;–

where *Shower*, *BathFan* and *BathLights* are O-type sensors, whereas *bHumidity* and *bTemp* are C-type sensors.

Identifying associated sensor triggering

The aim of this first step is to get a list of possible related sensors (associated), in order to minimize the complexity of the learning process. For the purpose of discovering possible *associatedSeT*, we search for previous events from other sensors that happened before each event of *mainSeT*. If $\mathcal{A}_{PUBS}$ discovers that before instances of *mainSeT*, frequently there is an occurrence of an event at another sensor, then the latter will be considered as *associatedSeT*. Finding an *associatedSeT* does not mean definitively that there will be a pattern that describes a relation *associatedSeT* - *mainSeT*, but indicates there could potentially be one.

A list of possible *associatedSeT*'s is obtained through a similar approach to the Apriori method for mining association rules (Agrawal and Srikant, 1995), with only two differences:

- Limit possible associations with the object we are analyzing (*mainSeT*).
- The result does not consider a pair (*mainSeT*, *associatedSeT*) as a pattern, but only as sensors that can be potentially related in a meaningful way.

An Apriori algorithm modified by adding the aforementioned constraints has been used in this step. As in every association mining process, minimum coverage, support and window size values must be provided. Considering Michael's example, for the *mainSeT* 'BathFan on' there will be at least one relationship with 'Shower off', which will be considered as *associatedSeT*.

Identifying time relations

Once we know what other actions could be related to *mainSeT*'s, the next step is to discover if there are possible meaningful time relations between them. To this end, for each *associatedSeT* we collect the time distances between occurrences of *mainSeT* and

previous appearances of *associatedSeT*. Considering Pattern 1 and the data collected, the time distances between *mainSeT* 'BathFan on' and *associatedSeT* 'Shower off' are depicted by Figure 2.

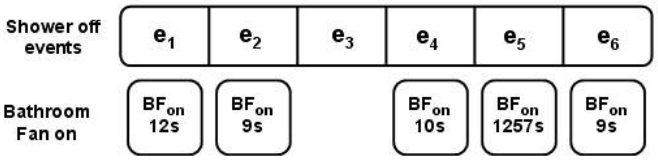

Fig. 2 Time Distances between both actions

Taking as starting point these time distances {{e1,12s} {e2,9s} {e3,-} {e4,10s} {e5,1257s} {e6,9s}}, the next step is to create groups, taking into account the similarities among them and checking to see if there is any time distance that groups enough instances to consider it as interesting. The technique for creating groups could be as complex as we can imagine. In this case the technique we have used is based on joining values that are within a range established by (1):

$$[min, max] = \overline{x} \pm (\overline{x} * tolerance) \; where \; \overline{x} = \frac{\sum_{i=1}^{n} a_i}{n} \tag{1}$$

with: tolerance = tolerated deviation from $\overline{x}$ (%); a_i = time distance of an element; and n = number of elements

Let us consider the time distances depicted in Figure 2 and a tolerance of 50%. Grouping those values creates two groups; the first group with mean value '10s', which covers 4 instances (e1,e2,e4,e6) and the second group with mean value '1257s' and 1 instance (e5). The group(s) that covers more instances than the minimum level required (defined manually, e.g. 25%) is considered as a pattern where the Event and Action are known. Considering the two groups generated in our example, only the first group (with a confidence level of 4/6) will be considered as a pattern, generating a pattern like:

```
ON occurs (Shower, Off,t0)

IF [...]

THEN do (On, Bathroom Fan, t) when t=t0+10s
```

Identifying appropriate conditions

In the previous step we generated patterns relating two situations (represented by ON and THEN clauses), but it is almost impossible to define patterns associated to a specific object based on only one relation. For instance, in our example the defined pattern has a 4/6 confidence level, so that it misclassifies 2/6 instances. Finding out (if possible) under what conditions a pattern will appear or not will be the last step in order to obtain accurate patterns. As has been mentioned before, calendar and context information given by C-type sensors will be used to define these possible conditions.

For the purpose of discovering these conditions, two tables, covered and non-covered tables, are generated. In the covered table there will be instances classified correctly by the pattern together with the calendar and context information collected when they

happened, whereas the same information regarding instances where the patterns fails is registered in the non-covered table (See Figure 3).

covered

	e_1	e_2	e_4	e_6
time of day	18:50:12	17:15:30	19:05:10	17:02:27
day of week	monday	tuesday	wednesday	thursday
bTemp Sensor	25	26	22	23
bLight Sensor	5	6	5	7

non-covered

	e_3	e_5
time of day	17:30:28	16:05:37
day of week	monday	tuesday
bTemp Sensor	26	24
bLight Sensor	12	15

Fig. 3 Covered and Non-Covered tables

Using the information they contain, dividing both tables allows us to know when the pattern properly defines the relationship between *mainSeT* and *associatedSeT*. Considering our example, the easiest way to separate *covered* and *non-covered* tables (as the example contains few instances, it can be separated in many different ways) seems to be by using the sensor bHumidity which indicates the humidity level in the bathroom when the action happens.

Adding these conditions does not increase the number of instances the pattern includes (it still includes the same number of instances, 4/6), but we make it more accurate, making sure that it does not include instances that do not exhibit that pattern. Thus, in this step we will define the IF clause of the pattern, obtaining a pattern such as:

```
ON occurs (Shower, Off,t0)

IF context (Bathroom humidity level (>,75%))

THEN do (On, Bathroom Fan, t) when t=t0+10s
```

The task of separating both tables has been considered as a classification problem, using the JRip algorithm (Witten and Frank, 2005). Even so, a modification has to be made due to the fact that JRip provides rules only with the unique objective of separating the two classes (covered and non-covered), whereas in our case it is desirable to obtain rules about the covered class. In this way, we always get a set of conditions that indicates when a pattern defines the relation well, instead of a mix of conditions that indicates when the pattern is well-defined and when it is not.

4.3 Interacting with $\mathcal{I}_{PUBS}$

Once patterns in the user's common behavior have been learned, they can be used for different purposes. Patterns represent frequent user-environment interactions, so that

one exciting application is the automation of devices (e.g. turning on the bathroom fan as Pattern 1 shows), allowing an environment to act proactively for the benefit of its users. An ideal proactive environment would be an environment where interaction with the user (both data acquisition and obtaining feedback) is carried out through the normal operation of standard devices such as switches or remote controls, thus trying to avoid any "ad hoc" set up.

But apart from automating devices, the patterns discovered can be used for other purposes, such as understanding user behavior or detecting hazardous or abnormal situations. Let us consider a house where an occupant is affected by the onset of Alzheimer's and their actions are being monitored to better understand changes in behavior that can be harmful to that person. This system also requires a friendly and easy way of interacting with it (Aghajan et al, 2009), so that the patterns learned can be efficiently used and personalized to specific cases.

We have developed a simple HCI system based on speech, which, based on the $\mathcal{L}_{PUBS}$ representation, allows users to fine tune the patterns discovered by $\mathcal{A}_{PUBS}$. As explained in Section 4.1, all patterns are represented based on $\mathcal{L}_{PUBS}$. This makes the use of patterns easier, because every part of the pattern is well-defined. Our system can interact with the user by voice to gather feedback about the patterns that have been learned and to provide the user an opportunity to further refine them. Next we illustrate the different functionalities of the interaction module ($\mathcal{I}_{PUBS}$).

First of all, the system welcomes the user and then asks the user if he/she wants to interact with $\mathcal{I}_{PUBS}$. If the user confirms the desire to interact with $\mathcal{I}_{PUBS}$ then the system asks the user to choose a *mainSeT* (including the possibility of listening to all patterns of all sensors):

System: *Hello, welcome to the interaction system. Patterns have been discovered by the algorithm. Do you want to listen to them? (yes/no)*

User: *Yes*

System: *Please choose a main sensor. These are the possible main sensors: Bathroom Light, Bathroom Fan or All*

User: *Bathroom Fan*

Once a *mainSeT* is chosen, $\mathcal{I}_{PUBS}$ lists patterns related to that *mainSeT*. Every pattern is mentioned in order to obtain the user's feedback about it. The following steps are carried out for each pattern. Let us consider Pattern 1, shown in Section 4.1, as one of the patterns of the Bathroom Light.

System: *Pattern 1*

System: *Occurs Shower is off and If Bathroom Humidity Level is greater than 75% Then turn on the Bathroom Fan 10 seconds after*

System: *Do you want to accept, refine or delete it?*

User: *Accept*

By means of $\mathcal{I}_{PUBS}$ the user can accept, refine or delete a pattern. Accepting a pattern means user accepts a pattern as useful and therefore the environment will use it to act proactively in the future. If the user chooses to delete a pattern, it is removed from the set of patterns so that the environment will not use it. Finally, the user can choose to refine a pattern if he/she considers it is a useful pattern but some aspect needs tuning. In deleting or accepting operations, the action to be carried out

by $\mathcal{I}_{PUBS}$ is simple, removing or not removing the pattern from the set of patterns; but in refining patterns, $\mathcal{I}_{PUBS}$ guides the user through the different aspects of the pattern that the user may like to change.

So far the interaction system has been conceived to obtain feedback from the user about the patterns it has learned, but it is worth mentioning that the system can be evolved in many different ways to suit the needs of different users in different environments.

Technical aspects

$\mathcal{I}_{PUBS}$ has been developed using a speech synthesizer and a speech recognizer. In order to facilitate integration with $\mathcal{A}_{PUBS}$ (developed in Java), we have chosen a synthesizer and recognizer written entirely in Java. We chose FreeTTS 1.2 [1] for the speech synthesizer and Sphinx-4 [2] as the speech recognizer.

Both FreeTTS and Sphinx make interaction with the user easier by providing easy-to-use tools. Complications arise mainly due to the changing nature of Intelligent Environments. For example, $\mathcal{I}_{PUBS}$ cannot know beforehand what devices are in the environment; thus, grammars for the recognizer must be created and loaded dynamically to integrate the interaction module into a specific environment.

5 Validation of PUBS

In order to validate the system we applied it to artificial data generated at the University of Ulster and then to two different real datasets collected from Washington State University's (WSU's) Smart Apartment and MavPad.

5.1 Validating PUBS with the WSU Smart Apartment dataset

The data collected in the WSU smart apartment (Cook and Schmitter-Edgecombe, 2008) represents participants performing five Activities of Daily Living (ADLs) in the apartment: making a phone call, washing hands, cooking, taking medicine/eating the food and cleaning (See Table I for actions involved in each activity).

In all, the sensors installed in WSU smart apartment are:

- 14 sensors on objects such as phone, medicine container or cabinet.
- 27 motion sensors.

As the set of actions involved in these 5 ADLs and the order of such actions were known in advance, we knew what patterns should be discovered by PUBS. Thus, the validation of PUBS using this dataset was especially critical for the first step of $\mathcal{A}_{PUBS}$ ("Identifying associated sensor triggering") because we knew what relationships were hidden in this dataset.

As expected, it discovered the 17 relationships hidden in the dataset. In order to avoid misunderstandings, it is worth mentioning that some relationships defined in Table I are repeated in different activities, e.g., Water (ON) –> Water (OFF). Besides these relationships it discovered 6 irrelevant relationships, so that it discovered 23 patterns in all. Following some of the patterns that have been found are shown:

[1] http://freetts.sourceforge.net/docs/index.php
[2] http://cmusphinx.sourceforge.net/sphinx4/

Table 1 Actions involved in each ADL.

Activity	Involved Actions
Making a phone call	Phone Book (ON) –>Phone (ON) –>Phone (OFF)
Washing hands	Water (ON) –>Water (OFF)
Cooking	Cabinet (ON) –>Raisins (ON) –>Oatmeal (ON) –>Measuring spoon (ON) –>Bowl (ON) –>Sugar (ON) –>Cabinet (OFF) –>Water (ON) –>Water (OFF) –>Pot (ON) –>Burner (ON) –>Burner (OFF)
Taking medicine and Eating	Cabinet (ON) –>Medicine (ON) –>Cabinet (OFF) –>Water (ON) –>Water (OFF) –>Cabinet (ON) –>Medicine (OFF) –>Cabinet (OFF)
Cleaning	Water (ON) –>Water (OFF)

```
(Pattern 0)
  ON occurs (Phone Book, On,t0)
  IF (by default = true)
  THEN do (On, Phone, t) when t is after t0

 ...

(Pattern 2)
  ON occurs (Phone, On,t0)
  IF (by default = true)
  THEN do (Off, Phone, t) when t=t0+50s

 ...

(Pattern 9)
  ON occurs (Cabinet, On,t0)
  IF (by default = true)
  THEN do (On, Medicine, t) when t=t0+3s

(Pattern 10)
  ON occurs (Medicine, On,t0)
  IF (by default = true)
  THEN do (Off, Cabinet, t) when t=t0+2s

(Pattern 11)
  ON occurs (Cabinet, Off,t0)
  IF (by default = true)
  THEN do (On, Water, t) when t=t0+16s

(Pattern 12)
  ON occurs (Water, On,t0)
  IF (by default = true)
  THEN do (Off, Water, t) when t is after t0
```

```
(Pattern 13)
  ON occurs (Water, Off,t0)
  IF (by default = true)
  THEN do (On, Cabinet, t) when t is after t0

(Pattern 14)
  ON occurs (Cabinet, On,t0)
  IF (by default = true)
  THEN do (Off, Medicine, t) when t=t0+2s

(Pattern 15)
  ON occurs (Medicine, Off,t0)
  IF (by default = true)
  THEN do (Off, Cabinet, t) when t=t0+3s

 ...

(Pattern 20)
  ON occurs (Raising, On,t0)
  IF (by default = true)
  THEN do (On, Oatmeal, t) when t=t0+3s

 ...
```

Most of the patterns were logical relationships that we were expecting, such as *"after speaking on the phone he hangs up the phone"* (See Pattern 2). Apart from those trivial relationships some other interesting relationships were also discovered, for example the set of patterns (See Pattern 9-15) that shows how Michael takes his medicine or the pattern (See Pattern 20) that indicates in what order he took the necessary ingredients for cooking.

Although the main objective of this experiment was to validate the first step of $\mathcal{A}_{PUBS}$ and make sure that it discovers the relationships that are hidden in a dataset, it brought up new knowledge about those relationships that we were not considered. This new knowledge was mainly related to time relations. As expected, it discovered that after speaking on the phone the occupant will hang up the phone, but the algorithm also discovered the time relation between both actions, so that we know how long he usually speaks by phone - in our case, for example, there was a group with the average value of 50 seconds. In that sense, it was possible to define a reliable time relation in 11 out of 17 valid relationships.

There were no context sensors; hence it was not possible to discover conditions for the patterns obtained. In this case, we have considered *"true"* as default value.

5.2 Validating PUBS with MavPad dataset

MavPad is a smart apartment created within the MavHome project (Youngblood et al, 2005) that consists of a living/dining room, a kitchen, a bathroom and a bedroom, all fitted with sensors. The sensors installed in MavPad are:

- 26 sensors on objects such as lamps, lights or outlets.
- 53 context sensors such as light, temperature or humidity.
- 37 motion sensors distributed in all the rooms.

Table 2 Number of total and accurate patterns obtained in different trials.

		Confidence Level			
		25%	**50%**	**75%**	**100%**
Trial 1	**Total Patterns**	16	5	1	0
	Accurate Patterns	12	3	1	0
	Ratio of Accuracy (%)	75%	60%	100%	—
Trial 2	**Total Patterns**	40	18	5	0
	Accurate Patterns	33	14	3	0
	Ratio of Accuracy (%)	82.5%	78%	60%	—
Trial 3	**Total Patterns**	20	10	6	0
	Accurate Patterns	15	4	2	0
	Ratio of Accuracy (%)	75%	40%	33%	—

The dataset used to validate PUBS was collected in three different time periods: Trial 1 (spanning 15 days), Trial 2 (spanning almost 2 months) and Trial 3 (spanning 3 months). Unlike the WSU smart apartment, the patterns hidden in the MavPad dataset were not known in advance, so that we decide to carry out different experiments considering different minimum confidence levels (25%, 50%, 75% and 100%). Table II summarizes the number of patterns discovered in each trial. As well as the number of discovered patterns, it shows the number of accurate patterns (in this case information was available about the context, so that it was possible to discover conditions of occurrence).

It is also interesting to analyse the nature, in terms of frequency, of the patterns. Because the data collected in each trial is totally independent of the data collected in other trials, the number of patterns discovered in each trial is totally independent. In Trial 1, 5 out of 16 (31% ratio) discovered with Confidence Level 25% appear when Confidence Level is 50%. Out of those 5 patterns, only 1 pattern (20% ratio) remained as frequent with a Confidence Level of 75%. Table III summarizes the percentage of total and accurate patterns that remain to be frequent with different Confidence Levels.

Following, some accurate patterns, discovered in different trials, are shown. Conditions are related to either calendar information (see Pattern 1 or Pattern 3) or C-type sensors (see Pattern 2), the later as defined in Section IV.

```
(Pattern 1 --> Trial 1, Confidence level: 25%)
  ON occurs (Room Light, Off,t0)
  IF context ((time (> , 1:06:44)) &
              (time (<,2:46:32)))
  THEN do (On, Luxo lamp, t) when t=t0+0s
```

Table 3 Frequency of different patterns.

		% of patterns remain from 25% to 50% Confidence Level	% of patterns remain from 50% to 75% Confidence Level
Trial 1	**Total Patterns**	31%	20%
	Accurate Patterns	25%	33%
Trial 2	**Total Patterns**	45%	27%
	Accurate Patterns	42%	21%
Trial 3	**Total Patterns**	50%	60%
	Accurate Patterns	27%	50%

```
(Pattern 2 --> Trial 2, Confidence level: 50%)
  ON occurs (Room Light, On,t0)
  IF context ((time (> , 19:19:49)) &
              (Bedroom light level (>,52)) &
              (Bedroom light level (<,143)))
  THEN do (Off, Luxo lamp, t) when t=t0+30s

(Pattern 3 --> Trial 3, Confidence level: 75%)
  ON occurs (Ceiling light, On,t0)
  IF context ((time (> , 22:10:27)))
  THEN do (Off, Ceiling light, t)
       when t=t0+115s
```

5.3 Discussion of the results

Interesting conclusions can be extracted from these results. Analysing the results obtained with MavPad dataset, first illustrates what we all intuitively expect: it is almost impossible in any realistic daily life scenario to define patterns associated with a specific object based on only one relation (in fact, there is no pattern with a 100% confidence level). We simply do not do activities with such a monotonous regularity hence defining in which specific circumstances (conditions) these patterns apply, is crucially important. Moreover, the figures show that in most of the patterns (87 out of 121) it was possible to define conditions of occurrence in order to obtain patterns with relatively high levels of confidence. The system can be run with different levels of confidence to find a useful pattern with a reasonable level of confidence.

Other conclusion that can be extracted in relation to the confidence level of patterns, is that in most trials less than the half of the patterns discovered with Confidence Level of 25% are considered frequent with a Confidence Level of 50% and that percentage drecreases when the Confidence Level increases up to 75%. Trial 3 is the exception of such a trend because 60% and 50% of total and accurate patterns discovered with Confidence Level of 50% remain as frequent with Confidence Level of 75%.

The results obtained in the validation process show the possibility of discovering comprehensible patterns that represent user's frequent behaviours. Thus, unlike approaches (Begg and Hassan, 2006) that use black box nature techniques (e.g., Artificial Neural Networks), patterns discovered by our algorithm allow us to understand such behaviours as well as to allow the automation of the house.

The patterns obtained when applying the algorithm to the WSU Smart Apartment dataset show users' behaviours are better described relating users' actions among them instead of relating users' actions to global situations (Hagras et al, 2004) (Doctor et al, 2005). In Michael's case, it is able to detect that the action of turning on the fan of the bathroom is typically associated with the end of the action of taking a shower.

As pointed out in Section III, approaches relating users' actions have already been developed (Jakkula and Cook, 2007). In addition, they relate those actions in terms of time using Allen's temporal logic relations (qualitative relationships). Unlike this approach, PUBS first tries to discover quantitative relationships in order to better define such relationships. Thus, in Michael's case the system knows that he usually turns on the fan 10 seconds after he finishes the action of taking a shower. For example, in the WSU dataset it was possible to define reliable time relations in 11 out of 17 patterns.

Classification techniques had already been used to discover conditions (Gal et al, 2001) for situations in which there were different reactions. Trials carried out using the MavPad dataset allowed us to define conditions in 87 out of 121 patterns, showing this approach is suitable to define accurate relationships.

The patterns obtained in both cases prove that it is possible to include all these steps in an algorithm (PUBS) in order to discover complete and comprehensible patterns that describe users' frequent behaviours.

6 Conclusions and Future Work

Intelligent Environments need to know the common behaviors and preferences of their users in order to meaningfully assist them. We have developed a system called PUBS which aims precisely at supporting an Intelligent Environment in the important task of understanding what the frequent behaviors of the occupant are in a given environment. This supports decision-making that can help the user, and it is also flexible enough to track the different behaviors we humans exhibit at different times.

PUBS was developed taking into account all of the specific characteristics of Intelligent Environments. Thus, both the acquisition process and the feedback process are carried out as unobtrusively as possible. In that sense, PUBS integrates an interaction system ($\mathcal{I}_{PUBS}$) based on speech which facilitates interaction with the user.

The experiments carried out to validate PUBS have shown the capacity of PUBS to discover relationships, time relations and the conditions of such relationships. These experiments (especially the one carried out using the WSU dataset) allowed us to realize that common behaviors of the user would be better represented by means of sequences of actions instead of one-to-one relationships. Such representation facilitates understanding behaviors, and they may be used for other purposes, such as the automation of actions. Thus, our current work is focused on discovering complete, frequent sequences of actions and the associated temporal and environmental conditions attached to each part of each sequence. Besides, once frequent behaviours are known, they can

be used to identify deviations which can help us to detect diseases such as depression or Alzheimer's in their early stages.

Acknowledgements Craig Wootton and Michael McTear provided initial guidance on available technologies for voice processing. This work was partially supported by Basque Government grant PC2008-28B.

References

Aarts E (2004) Ambient intelligence: A multimedia perspective. IEEE Multimedia pp 12–19

Aghajan H, Delgado RLC, JCAugusto (2009) Human-Centric Interfaces for Ambient Intelligence. Academis Press - Elsevier

Agrawal R, Srikant R (1995) Mining sequential patterns. In: Pro. 11th International Conference on Data Engineering, pp 3–14

Allen J (1984) Towards a general theory of action and time. In: Artificial Intelligence, vol 23, pp 123–154

Augusto JC (2007) Ambient Intelligence: the Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence, Springer London, pp 213–234. Intelligent Computing Everywhere

Augusto JC (2009) Past, present and future of ambient intelligence and smart environments. In: 1st International Conference on Agents and Artificial Intelligence (ICAART)

Augusto JC, McCullagh P (2007) Ambient intelligence: Concepts and applications. In: Computer Science and Information Systems, ComSIS Consortium, vol 4, pp 1–28

Augusto JC, Nugent CD (2004) The use of temporal reasoning and management of complex events in smart homes. In: Proccedings of European Conference on AI (ECAI 2004), IO Press, pp 778–782

Aztiria A, Izaguirre A, Augusto J (2010) Learning patterns in ambient intelligence environments: A survey. Artificial Intelligence Review 34:1–31

Begg R, Hassan R (2006) Artificial neural networks in smart homes, Springer-Verlag, pp 146–164. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D.

Callaghan V, Kameas A, Reyes D (eds) (2009) Proceedings of the 5th International Conference on Intelligent Environments. IOSPress

Campo E, Bonhomme S, Chan M, Esteve D (2006) Learning life habits and practices: an issue to the smart home. In: International Conference on Smart Homes and health Telematic, pp 355–358

Chan M, Hariton C, Ringeard P, Campo E (1995) Smart house automation system for the elderly and the disabled. In: Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics, pp 1586–1589

Coen MH (1998) Design principles for intelligent environments. In: Proceedings of the 1998 15th National Conference on Artificial Intelligence, AAAI, AAAI Press, pp 547–554

Cook D, Schmitter-Edgecombe M (2008) Activity profiling using pervasive sensing in smart homes. IEEE Transactions on Information Technology for Biomedicine

Cook D, Augusto J, Jakkula V (2009) Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5(4):277–298

Cook DJ, Das SK (2005) Smart Environments: Technology, Protocols and Applications. Wiley-Interscience

Cook DJ, Das SK (2007) How smart are our environments? an updated look at the state of the art. In: Pervasive and Mobile Computing, Elsevier Science, vol 3, pp 53–73

Doctor F, Hagras H, Callaghan V (2005) A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. In: IEEE Transactions on systems, man and cybernetics, vol 35, pp 55–65

Dooley J, Callaghan V, Hagras H, Bull P, Rohlfing D (2006) Ambient intelligence - knowledge representation, processing and distribution in intelligent inhabited environments. In: 2nd IET International Conference on Intelligent Environments, IE 06, pp 51–59

Ducatel K, Bogdanowicz M, Scapolo F, Leijten J, Burgelman JC (2001) Scenarios for ambient intelligence in 2010. Tech. rep., URL http://cordis.europa.eu/ist/istag-reports.htm

Duman H, Hagras H, Callaghan V (2008) Intelligent association exploration and exploitation of fuzzy agents in ambient intelligent environments. Journal of Uncertain Systems 2(2):133–143

Friedemann M, Mahmoud N (2002) Pervasive Computing, First International Conference. Springer-Verlag

Gal CL, Martin J, Lux A, Crowley JL (2001) Smartoffice: Design of an intelligent environment. IEEE Intelligent Systems 16(4):60–66

Galushka M, Patterson D, Rooney N (2006) Temporal data mining for smart homes, Springer-Verlag, pp 85–108. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D.

Hagras H, Callaghan V, Colley M, Clarke G, Pounds-Cornish A, Duman H (2004) Creating an ambient-intelligence environment using embedded agents. IEEE Intelligent Systems 19(6):12–20

Heierman EO, Cook DJ (2002) Improving home automation by discovering regularly occurring device usage patterns. In: Third IEEE International Conference on Data Mining, pp 537–540

Jakkula VR, Cook DJ (2007) Using temporal relations in smart environment data for activity prediction. In: Proceedings of the 24th International Conference on Machine Learning

Jakkula VR, Crandall AS, Cook DJ (2007) Knowledge discovery in entity based smart environment resident data using temporal relation based data mining. In: 7th IEEE International Conference on DataMining, pp 625–630

Kim E, Helal S, , Cook D (2010) Human activity recognition and pattern discovery. IEEE Pervasive Computing 9:48–53

Mozer MC, Dodier RH, Anderson M, Vidmar L, Cruickshank RF, Miller D (1995) The neural network house: an overview, Erlbaum, pp 371–380. Current trends in connectionism

Muller ME (2004) Can user models be learned at all? inherent problems in machine learning for user modelling. In: Knowledge Engineering Review, Cambridge University Press, vol 19, pp 61–88

Nakashima H, Aghajan H, JCAugusto (2009) Handbook on Ambient Intelligence and Smart Environments. Springer Verlag

Partala T, Surakka V, Vanhala T (2006) Real-time estimation of emotional experiences from facial expressions. Interacting with Computers 18(2):208–226

Rao SP, Cook DJ (2004) Predicting inhabitant action using action and task models with application to smart homes. International Journal on Artificial Intelligence Tools (Architectures, Languages, Algorithms) 13(1):81–99

Rutishauser U, Joller J, Douglas R (2005) Control and learning of ambience by an intelligent building. In: IEEE on Systems, man and cybernetics: a special issue on ambient intelligence, IEEE Systems, Man, and Cybernetics Society, pp 121–132

Sadeh NM, Gandom FL, Kwon OB (2005) Ambient intelligence: The mycampus experience. Tech. Rep. CMU-ISRI-05-123, ISRI

Turunen M, Hakulinen J, Kainulainen A, Melto A, Hurtig T (2007) Design of a rich multimodal interface for mobile spoken route guidance. In: Proceedings of Interspeech 2007 - Eurospeech

Weiser M (1991) The computer for the 21st century. Scientific American 265(3):94–104

Witten IH, Frank E (2005) Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed. Elsevier

Youngblood GM, Cook DJ, Holder LB (2005) Managing adaptive versatile environments. In: IEEE International Conference on Pervasive Computing and Communications

A Language Specification

```
Pattern::= ON (Event_Definition)
           IF (Condition_Definition)
           THEN (Action_Definition)

Event_Definition::= Primitive_Event | Composite_Event
  Primitive_Event::= User_Presence|User_Action
    User_Presence::= user_is_at(Location)
      Location::= home|bedroom|living room|...
    User_Action::= occurs(Device, Device_Action, time)
      Device::=  device_1|device_2|...|device_n
      Device_Action::= on|off
  Composite_Event::= Primitive_Event &...& Primitive_Event

Condition Definition::= Primitive_Condition|Composite_Condition
  Primitive_Condition::= Context_Condition
    Context_Condition::= context(Attribute,Quantitative_Condition|
                                        Qualitative_Condition)
      Attribute::= Calendar|Sensor
        Calendar::= time of day|day of week|...
        Sensor::= sensor_1|sensor_2|...| sensor_n
      Quantitative_Condition::= (Symbol,Quantitative_Value)
        Symbol::= =|<|>|=>|=<
        Quantitative_Value::= real_number
      Qualitative_Condition::= qualitative_value
  Composite_Condition::= Primitive_Condition &...& Primitive_Condition

Action Definition::= Primitive_Action|Composite_Action
  Primitive Action::= do(Device_Action, Device,time) when Relation
    Device_Action::= on|off
    Device::= device_1|device_2|...|device_n
    Relation::= Qualitative_Relation|Quantitative_Relation
      Quantitative_Relation::= (Symbol,Quantitative_Value)
        Symbol::= =|<|>|=>|=<
        Quantitative_Value::= real_number
```

```
    Qualitative_Relation::= Qualitative_Value
      Qualitative_Value::= after|while|...|equal
Composite_Action::= Primitive_Action &...& Primitive_Action
```