Autonomous Learning of User's Preferences improved through User Feedback

Asier AZTIRIA [a] Juan Carlos AUGUSTO [b] and Alberto IZAGUIRRE [a]
[a] *University of Mondragon, Spain*
email:{aaztiria,aizagirre}@eps.mondragon.edu
[b] *University of Ulster, United Kingdom*
email:{jc.augusto}@ulster.ac.uk

Abstract. Ambient Intelligent (AmI) environments are supposed to act proactively anticipating the user's needs and preferences, therefore the capability of an AmI system to learn those elements out of daily life behaviour of those using the environment is very valuable. In this paper we present a system that discovers patterns related to user's actions and improves them through user feedback. The core of this system is an algorithm which taking as starting point information collected by sensor discovers these patterns. Coupled with the algorithm, a language to represent those patterns has been developed. This system allows the user experiencing the AmI environment to verbally interact with the system and give his/her feedback about patterns that have been discovered. The speech based interaction provides a natural communication for the user and the simple protocol established makes the system available to users without sophisticated training.

Keywords. Ambient Intelligence, Learning Behavioral Patterns, Temporal Relations, Human-Computer Interaction, Speech-based Interaction

1. Introduction

Ambient Intelligence (AmI) [4,11] refers to 'a digital environment that proactively, but sensibly, supports people in their daily lives' [3]. Other terms such as Ubiquitous Computing [23] or Smart Environments [8] are used with similar connotations. Supporting people in their daily lives means, for example, making an environment safer, more comfortable and more energy efficient. In order to achieve these objectives, the environment should learn patterns of user behavior in a unobtrusive and transparent way. The acquired patterns allow, as well as the understanding of behavior, the automation of devices and detection of hazardous or abnormal situations. Proactive and autonomous system behaviour in these situations can then help the user to have a more comfortable and safer life.

In this paper, we explain the Patterns of User Behavior System (PUBS) which is a system that allows AmI environments to learn such behavioral patterns. The

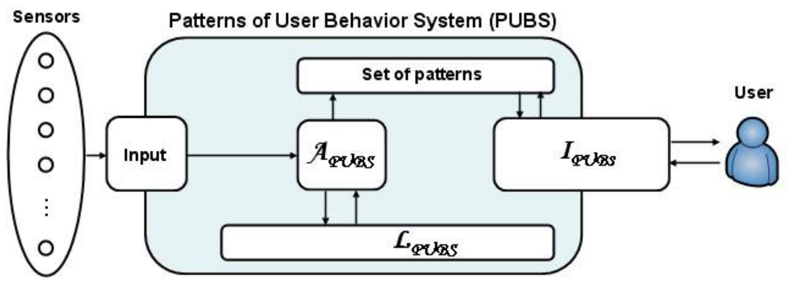

Figure 1. Global architecture of PUBS

core of this system is $\mathcal{A}_{PUBS}$, an algorithm that discovers patterns using the information collected by sensors. The language $\mathcal{L}_{PUBS}$ included within PUBS provides a standard framework to represent patterns with clear syntax and facilitates the definition of algorithms as well as the interaction with the user.

Finally, due to essential role of users in AmI environment, PUBS includes a speech recognition based interface, $\mathcal{I}_{PUBS}$, which allows the user to interact with PUBS either accepting, deleting or modifying the discovered patterns. The essential components of the PUBS architecture are shown in Figure 1.

The rest of the paper is organized as follows. Section 2 summarizes previous related work done in learning and Human-Computer Interaction (HCI) for AmI environments. In Section 3 we explain the nature of the data collected. Section 4 explains the learning algorithm $\mathcal{A}_{PUBS}$ and how patterns are represented through $\mathcal{L}_{PUBS}$. Section 5 illustrates how the interaction system $\mathcal{I}_{PUBS}$ can be used. Section 6 explains our plans for future work and finally we provide our conclusion in Section 7.

2. Related Work

Learning and Human-Computer Interaction have risen independently from AmI environments. Here we analyze each area separately and in later sections we show how they are amalgamated in our system to provide an essential feature of AmI systems.

Learning is a essential feature in any AmI system. However, given the diversity of elements that need to converge in order to realize the infrastructure needed for an AmI system, learning has not been devoted as much attention in the literature as it may require. Some notable exceptions are listed next. The use of Artificial Neural Networks [15,17] was the first serious approach in order to infer rules for smart homes, and a survey of those works can be found in [5]. Attempts made within MavHome project were aimed at predicting the next smart home inhabitant action using pattern discovery and Markov model techniques [9]. Jakkula and Cook [13] extend this work to predict actions using temporal relations, defined by means of Allen's temporal logic relations [2]. Other techniques, such as Fuzzy-Logic in iDorm [9], Case-Based Reasoning in MyCampus [20] or Decision Trees in SmartOffice [12] have been used. We can state that due to specific characteristics of AmI environments, each problem favours the use of certain techniques, but as Muller pointed out [16] ‘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’.

Interaction systems have also been accepted as a essential feature of AmI environments. The possibilities given by HCI systems in order to understand human behavior has been widely analyzed [10,22]. One of the most active research groups in HCI involving AmI environments is the Tampere Unit for Computer-Human Interaction [14,21]. Focusing on speech interaction systems, there have been works that emphasize the importance of sounds in environments where computers will be “hidden" to the user making the system transparent and unobtrusive [18].

3. Collecting Data

The process of learning patterns will be carried out based on information coming from sensors in a way which is as unobtrusive as possible [19]. Due to the fact that different sensors provide different type of information, they will be used for different purpose in the process of learning. Taking into account the nature of the information provide by each sensor, three main different groups of sensors are considered (although we are aware there are more types of sensors, e.g. alarm pendants or RFID (Radio-frequency identification)):

- (type O) Sensors installed in objects; They provide direct information about user actions, e.g. a sensor installed in a light switch indicates when the user has switched that light on or off.
- (type C) Context sensors; They provide a continuous information about the environment, e.g. a temperature sensor installed in a room will measure room temperature continuously. Although user actions, e.g., changing the setting of the thermostat, influence their future measurements they do not provide direct information about user actions.
- (type M) Motion sensors; They indicates the location of the user at all time, e.g. a motion sensor installed in the bedroom can help to infer (for example in connection with an RFID sensor in the door) if the user is inside the bedroom.

Figure 2 illustrates a sequence of actions where different sensors provide different information. As the figure shows, type O sensor installed in the Bedroom Lamp indicates when user has turned on/off that lamp, type M sensor shows when user is within the bedroom, whereas type C sensor measures room light level continuously. This is a simple example to give an idea of each type of sensors, but in real environments there will be a large amount of sensors of each type.

4. Learning and representing patterns of user behavior

Once information coming from different sensors has been collected our learning algorithm ($\mathcal{A}_{PUBS}$) tries to discover possible relations among user actions ana-

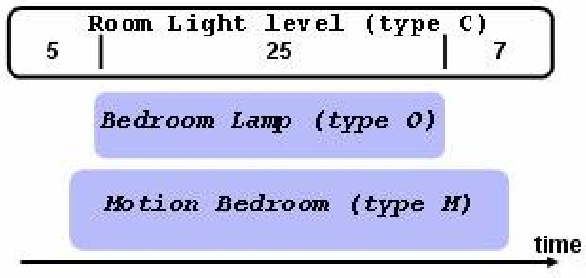

Figure 2. Example of a temporal pattern

lyzing relations among sensors. Let us consider the situation described in figure 2, assuming that usually:

'Motion Bedroom has been turned on and If Room Light Level is lower than 10 Then the Bedroom Lamp is turned on 2 seconds after' (Pattern 1)

$\mathcal{A}_{PUBS}$ tries to discover this type of pattern where the bedroom lamp is considered as *main sensor triggering (mainSeT)* and the Motion sensor that detects when user goes into the bedroom will be considered as *associated sensor triggering (associatedSeT)*. Before explaining the process carried out by $\mathcal{A}_{PUBS}$ to discover these patterns, it is necessary to explain the language ($\mathcal{L}_{PUBS}$) used to represent patterns as this language largely influence $\mathcal{A}_{PUBS}$ as well as $\mathcal{I}_{PUBS}$.

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

Defining a language that allows us to represent patterns of user behavior in AmI environments is necessary to have a clear and non ambiguous representation (See Appendix A and [7] for more details). The language integrated into our system, $\mathcal{L}_{PUBS}$, is based on ECA (Event-Condition-Action) rules [6]. ECA rules allow one to define what action has to be carried out when a event occurs under relevant conditions. Considering the pattern 1, it will be represented using $\mathcal{L}_{PUBS}$ as:

```
ON occurs (Motion Bedroom, On,t0)

IF context (Room light level (<,10))

THEN do (On, Bedroom Lamp, t) when t=t0+2s
```

As well as providing a way of representing patterns in order to create a clear and non ambiguous representation of patterns to be used by different modules, it makes sure patterns are clearly specified and enables other technologies that can check their integrity [5].

4.1.1. Event Definition

The event part defined by the ON clause defines the event(s) that occurred and triggered the relation specified by the pattern. In the event definition, following the terms used in $\mathcal{A}_{PUBS}$, the *associatedSeT* together with the action (on/off) is defined. As patterns relate user behaviors, the ON event(s) will be the effect of a user action over objects fitted with O-type sensors or the user's presence being detected by M-type sensors.

4.1.2. Condition Definition

The IF clause defines the necessary conditions under which the actions specified in the THEN clause is the appropriate reaction to the occurrence of events listed in the ON clause. Due to the fact that is almost impossible that 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:

```
IF context (Living room temperature (<,20 C)) (Condition 1)

IF context (Time of day (>,20:30:00)) (Condition 2)

IF context (Time of day (>,13:00:00)) & context (Time of day (<,14:15:00))
(Condition 3)

IF context (Day of week (=, Tuesday or Thursday)) (Condition 4)
```

Conditions are defined using an attribute and a value. Attributes can be either information coming from C-type sensors (light level in (Pattern 1); temperature in (Condition 1)) or calendar information (time of day in (Condition 2)(Condition 3); day of week in (Condition 4)). Values can be either qualitative ('Tuesday' and 'Thursday' in (Condition 4)) or quantitative (20 °C in (Condition 1); 20:30:00 in (Condition 2)). It is possible to define a range of values ([13:00:00-14:15:00] in (Condition 3)) when using quantitative values.

4.1.3. Action Definition

Finally, the THEN clause defines the action that user usually carries out given the ON clause and given the IF conditions. As well as defining the $mainSeT$ together with the action (on/off), it defines the time relation between Event and Action situations, being that relation either quantitative (Action 1) or qualitative (Action 2). The usefulness of each type of relation is different. Both define, although in different ways, user behaviour, but whereas quantitative relations can be used to automate $mainSeT$ actions, qualitative relations cannot be used for this purpose as besides knowing that one action follows another we need to know the specific time relation.

```
THEN do (On, Bedroom Lamp, t) when t=t0+2s (Action 1)

THEN do (On, Bedroom Lamp, t) when t is after t0 (Action 2)
```

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

In accordance with $\mathcal{L}_{PUBS}$, we have developed an algorithm ($\mathcal{A}_{PUBS}$) to discover patterns in data collected by sensors. The algorithm is detailed below:

$\mathcal{A}_{PUBS}$ Algorithm (for learning patterns)
for each sensor of type O (consider it as *mainSeT*)
Identify the *associatedSeT* of type O or M (See Section 4.2.1)
for each *associatedSeT*
Identify possible time relations (See Section 4.2.2)
if there exists a time relation **then** make it more accurate using context information, i.e., by using sensors of type C (See Section 4.2.3)

Emphasising O-type sensors as *mainSeT* is due to the fact that those sensors provide us direct information about users' action so that discovering patterns about them we will discover patterns about users' actions. Next, the main three steps applied for *mainSeT* are explained.

4.2.1. Identifying associated sensor triggering

Once information has been collected from sensors, the first step is to analyze the possible related sensors (called *associatedSeT*) to *mainSeT* in order to minimize the complexity of next steps. For the purpose of discovering possible *associatedSeT* we consider each event of *mainSeT* and collect the previous events that occurred within the time period specified by window-width (defined manually).

In order to get the list of possible *associatedSeT*s we use a similar approach to the Apriori algorithm [1] for mining association rules. The Apriori algorithm tries to discover frequent sequences. Unlike the Apriori algorithm, in our case:

- The possible associations are limited to *mainSeT*.
- The result does not have a sense of sequence, but the pair (*mainSeT*,*associatedSeT*) is considered as sensors that can be potentially related in a meaningful way. Besides, defining an *associatedSeT* does not mean there will be a pattern that describes a relation *mainSeT*-*associatedSeT*, but indicates there could potentially be one.

As in every association mining process, minimum coverage and support values must be provided manually.

4.2.2. Identifying time relations

The aim of this second step is to identify (if possible) the time relations among *mainSeT* and the *associatedSeT*s discovered in the first step. Thus, for each *associatedSeT* we collect the time distances between occurrences of *mainSeT* and previous appearances of *associatedSeT*. Considering again the pattern (1), let us imagine that the time distances between *mainSeT* (Bedroom Lamp) and *associatedSeT* (Motion sensor that detects user goes into the bedroom) are depicted by Figure 3.

Taking as starting point these time distances {{e1,2s} {e2,1s} {e3,-} {e4,3s} {e5,125s} {e6,2s}}, the next step is to make groups taking into account the similarities among them and check if there is any time distance that groups enough instances to consider it as interesting. The technique to make groups could be as

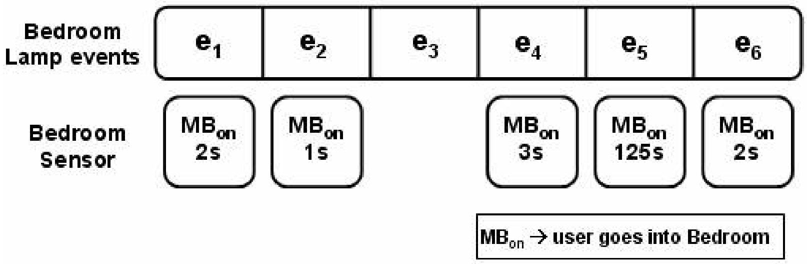

Figure 3. Time distances between *mainSeT* and *associatedSeT*

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):

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

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

Let us consider the time distances depicted in Figure 3 and a tolerance of 50%. Grouping those values two groups are created, the first group with mean value '2s', which covers 4 instances (e1,e2,e4,e6) and the second group with mean value '125s' and 1 instance (e5). The group(s) that covers more instances than minimum level demanded (defined manually, e.g. 25%) is considered as a pattern where Event and Action parts 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 pattern, generating a pattern like:

```
ON occurs (Motion Bedroom, On,t0)

IF [...]

THEN do (On, Bedroom Lamp, t) when t=t0+2s
```

4.2.3. Identifying appropriate conditions

In the previous step we have generated patterns relating two situations (represented in 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 appears or not will be the last step in order to get 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 the conditions, two tables, covered and non-covered tables, are generated. In the covered table there will be instances classified well by the pattern together with the calendar and context information collected when they happened, whereas the same information of instances where the patterns fails is registered in the non-covered table (See Figure 4).

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

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

Figure 4. *non-covered* and *covered* tables

Dividing both tables, using the information they contain, allows us to know when the pattern defines properly the relation 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 bLight which indicates the light level in the bedroom when action happens.

Adding these conditions do 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 have that pattern. Thus, in this step we will define the IF clause of the pattern, getting a pattern like:

```
ON occurs (Motion Bedroom, On,t0)

IF context (Room light level (<,10))

THEN do (On, Bedroom Lamp, t) when t=t0+2s
```

The task of separating both tables has been considered as a classification problem using the JRip algorithm [24] in order to do that. Even so, a modification has to be made due to the fact that JRip provides rules with the only unique objective of separating both 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 well the relation, instead of a mix that indicates when it defines well and when it does not.

4.3. Results

In order to validate the algorithm we have applied it to artificial data generated at the University of Ulster and then to a real dataset collected from MavPad, a smart apartment created within the MavHome project [25]. 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 1. Number of patterns and accurate patterns obtained in different trials.

	Trial 1		Trial 2		Trial 3	
Confidence Level	Total Patterns	Accurate Patterns	Total Patterns	Accurate Patterns	Total Patterns	Accurate Patterns
25%	16	12	40	33	20	15
50%	5	3	18	14	10	2
75%	1	1	5	3	6	4
100%	0	0	0	0	0	0

The dataset used to validate $\mathcal{A}_{PUBS}$ was collected in three different time periods and different experiments using different minimum confidence levels (25%, 50%, 75% and 100%) have been carried out. Table 1 summarizes the number of patterns discovered in each trial, modifying the minimum confidence level. As well as the number of discovered patterns, it shows the number of accurate patterns (patterns where it has been possible to define conditions of occurrence).

The results show us how difficult it is to discover patterns with 100% confidence level, hence the importance of defining the right conditions. The results show that it has been possible to define conditions in most of the patterns (76% of the cases).

5. Interactive system ($\mathcal{I}_{PUBS}$)

Once patterns about user common behavior have been learned, they can be used for different purposes. One exciting application is on to automation of devices (e.g. turning on the bedroom light as pattern 1 shows), allowing environment to act proactively. An ideal proactive environment suggests an environment where the interaction (both process of data acquisition and process of getting feedback) with the user is carried out through the normal operation of standard devices such as switches or remote controls, trying to avoid any 'ad hoc' means.

But apart from automating devices, discovered patterns can be used for other purposes such as understanding user behavior or detecting hazardous or no normal situations. Let us consider an old people's home where their actions are monitored and usual patterns are learned. Those patterns can be used by staff members to understand the behavior of each patient or even to detect bad habits. It is necessary a Human-Computer Interaction system that allows a friendly and easy way of interaction, so that the learned patterns can be used efficiently and also take maximum advantage of them.

Even, considering patterns to automate devices and going beyond, a Human-Computer Interaction system which involves patterns can be very useful in a proactive environment in order to explain to the user (if required) why the environment has acted in the way it has acted.

Being aware of necessity of an interface in order to interact with PUBS, we have developed a HCI system based on speech which based on $\mathcal{L}_{PUBS}$ representation allows user to interact with the patterns discovered by $\mathcal{A}_{PUBS}$.

5.1. Interaction system's functionalities

As explained in Section 4, 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 and to gather feedback about the patterns that have been learnt and provide the user an opportunity to further refine them. Next we illustrate the different functionalities of $\mathcal{I}_{PUBS}$, the interaction module, through a few examples based on sessions collected through the testing of our system.

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 to choose a *mainSeT* (it includes 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: Bedroom Lamp, Bedroom Luxo Lamp, Motion Bedroom or All*

User: *Bedroom Lamp*

Once a *mainSeT* is chosen $\mathcal{I}_{PUBS}$ lists patterns related to that *mainSeT*. Every pattern is mentioned in order to get user's feedback about it. The following steps are carried out for each pattern. Let us consider pattern 1, shown in Section 4, as one of the patterns associated to Bedroom Lamp.

System: *Pattern 1*

System: *Occurs Motion Bedroom is turned on and If Room Light Level is lower than 10 Then turn on the Bedroom Lamp 2 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, that pattern is removed from the set of patterns so that the environment will not use it. Finally, a user can choose to refine a pattern if he/she considers it is a useful pattern but some aspect needs tuning. In case of deleting or accepting operations, the action to be carried out by $\mathcal{I}_{PUBS}$ is simple, removing or not the pattern from the set of patterns, but in case of refining $\mathcal{I}_{PUBS}$ must guide the user through the pattern in order to find out exactly what needs modification.

System: *Choose the part of the pattern you want to refine: Event, Condition or Action*

Following different samples of refinement are shown:

Case 1 : Refining an Event part

User: *Event*

System: *The Event part was: Occurs Motion Bedroom is turned on*

System: *The device was 'Motion Bedroom'. Do you want to modify it? (yes/no)*

User: *no*

System: *The action was 'on'. Do you want to modify it? (yes/no)*

User: *yes*

System: *Possible action values are: 'on' or 'off'*

User: *off*

System: *Do you want to confirm 'off' as new action value? (yes/no)*

User: *yes*

Case 2 : Refining a Condition part

User: *Condition*

System: *The Condition part was: If Room Light Level is lower than 10*

System: *The attribute was 'Room Light Level'. Do you want to modify it? (yes/no)*

User: *no*

System: *The value was 10. Do you want to modify it? (yes/no)*

User: *yes*

System: *Please, say the new value*

User: *one five*

System: *Do you want to confirm 15 as new value? (yes/no)*

User: *yes*

Case 3 : Refining an Action part

User: *Action*

System: *The Action part was: Then turn on the Bedroom Lamp 2 seconds after*

System: *The device was 'Bedroom Lamp'. Do you want to change it?*

User: *yes*

System: *These are the possible devices: Bedroom Lamp, Bedroom Luxo Lamp or Motion Bedroom*

User: *Bedroom Luxo Lamp*

System: *Do you want to confirm 'Bedroom Luxo Lamp' as new device? (yes/no)*

User: *yes*

System: *The action was 'on'. Do you want to change it? (yes/no)*

User: *no*

System: *The time relation was '2 seconds after'. Do you want to change it? (yes/no)*

User: *no*

This Interaction system has been conceived to get user feedback and provide the environment with a set of patterns that it can be safely used to act proactively. Thus, accepted and modified patterns will be stored whereas deleted patterns will not. It is worth mentioning that different environments and different aims make necessary the development of different interaction systems. Let us imagine an old people's home where nurses must decide if patterns discovered on a monitored elderly person are normal or are not. In this case the obtained feedback will not be used to automate the environment, but it could be use to warn nurses if an abnormal behavior is detected (for example, trying to go out of the house between 7pm and 7am).

The speech-based interaction module of PUBS can be evolved in many different ways to suit the needs of different users in different environments. Here we have presented one application but we consider it can be adapted to other cases.

5.2. Technical aspects

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

Both FreeTTS and Sphinx make the interaction with the user easier providing easy to use tools. Complications come mainly due to changing nature of AmI environments. For example $\mathcal{I}_{PUBS}$ cannot know beforehand what devices are in the environment, so that grammars for the recognizer must be created and loaded dynamically to tie the interaction module with a specific environment.

6. Future Work

We are currently improving different modules in PUBS. Thus, on the one hand, our efforts will be aimed at improving the $\mathcal{A}_{PUBS}$ so that it can cope well with patterns $\mathcal{L}_{PUBS}$ that have a qualitative component. Up to now $\mathcal{A}_{PUBS}$ discovers quantitative relations, so that our short-term efforts will be aimed at discovering patterns with qualitative relations. Another level of complexity will be to target the discovery of patterns involving the combination of more than two activated sensors. Further future work will also include the possibility of incorporating more complex information coming from devices such as PDAs.

In relation to $\mathcal{I}_{PUBS}$ our short-term efforts will be aimed at making it more flexible, adding functionalities in order to allow user to add new patterns or to in-

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

teract using more natural (flexible and complex) expressions. Further future work could also include a development of a general interaction system that, depending on the environment and the type of user, interacts with him/her in different ways.

7. Conclusions

Ambient Intelligent environments need to know the user's preferred and expected behavior in order to meaningfully assist (for example by automating devices, detecting hazardous situations, etc.). We have developed a system called Patterns of User Behavior System (PUBS) which aims precisely at supporting an AmI system in the task to acquire a notion of what is frequently the case in given environment. This supports decision-making to help the user and also flexible and continuous adaptation to the different behaviors we humans exhibit at different times of the day, different days of the week, different seasons, etc.

The fact that the environment is technologically rich must not translate into any extra effort for the users to obtain the benefits of an AmI system [16]. This means the acquisition process must be done as unobtrusively as possible. PUBS's input information is gathered through sensors installed in the environment. The algorithm ($\mathcal{A}_{PUBS}$) discovers patterns defining temporal relations between situations (detected through sensors) caused by the user. These patterns relate two situations in terms of time and specify the necessary conditions where that relation makes sense. Patterns discovered by $\mathcal{A}_{PUBS}$ are stored using $\mathcal{L}_{PUBS}$, which is a language we have defined to represent patterns in a univocal way.

Discovering patterns is a essential part of a system to act intelligently but taking into account the user is the focus of AmI environments, the interaction module in between the user and PUBS is essential in order to ensure user satisfaction with the patterns to be used by the AmI system. Thus, integrated within PUBS there is a interaction system ($\mathcal{I}_{PUBS}$) based on speech recognition by means of which user can fine tune the discoveries of $\mathcal{A}_{PUBS}$.

Acknowledgements

Validation of the techniques presented in this paper were conducted over artificial data generated at University of Ulster and environment data from MavPad provided by Diane Cook from the University of Washington. 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

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Pro. 11th International Conference on Data Engineering*, pages 3–14, 1995.

[2] J. Allen. Towards a general theory of action and time. In *Artificial Intelligence*, volume 23, pages 123–154, 1984.

[3] J. C. Augusto. *Ambient Intelligence: the Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence*, pages 213–234. Intelligent Computing Everywhere. Springer London, 2007.

[4] J. C. Augusto and D. J. Cook. *Ambient Intelligence: applications in society and opportunities for AI.* 20th International Joint Conference on Artificial Intelligence (IJCAI-07). 2007.

[5] J. C. Augusto and P. McCullagh. Ambient intelligence: Concepts and applications. In *Computer Science and Information Systems*, volume 4, pages 1–28. ComSIS Consortium, 2007.

[6] J. C. Augusto and C. D. Nugent. The use of temporal reasoning and management of complex events in smart homes. In *Proccedings of European Conference on AI (ECAI 2004)*, pages 778–782. IO Press, 2004.

[7] A. Aztiria, J. C. Augusto, and A. Izaguirre. Spatial and temporal aspects for pattern representation and discovery in intelligent environments. In *Workshop on Spatial and Temporal Reasoning at 18th European Conference on Artificial Intelligence (ECAI 2008) (to be published)*, 2008.

[8] D. J. Cook and S. K. Das. *Smart Environments: Technology, Protocols and Applications.* Wiley-Interscience, 2005.

[9] D. J. Cook, M. Huber, K. Gopalratnam, and M. Youngblood. Learning to control a smart home environment. In *Innovative Applications of Artificial Intelligence*, 2003.

[10] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human Computer Interaction.* Prentice Hall, 3rd edition, 2003.

[11] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J. C. Burgelman. Scenarios for ambient intelligence in 2010. Technical report, 2001.

[12] C. Le Gal, J. Martin, A. Lux, and J. L. Crowley. Smartoffice: Design of an intelligent environment. *IEEE Intelligent Systems*, 16(4):60–66, 2001.

[13] V. R. Jakkula and D. J. Cook. Using temporal relations in smart environment data for activity prediction. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[14] A. Kainulainen, M. Turunen, J. Hakulinen, E. P. Salonen, P. Prusi, and L. Helin. A speechbased and auditory ubiquitous office environment. In *10th International Conference on Speech and Computer (SPECOM)*, pages 231–234, 2005.

[15] M. C. Mozer, R. H. Dodier, M. Anderson, L. Vidmar, R. F. Cruickshank, and D. Miller. *The neural network house: an overview*, pages 371–380. Current trends in connectionism. Erlbaum, 1995.

[16] M. E. Muller. *Can user models be learned at all? Inherent problems in machine learning for user modelling*, pages 61–88. Knowledge Engineering Review. 2004.

[17] F. Rivera-Illingworth, V. Callaghan, and H. Hagras. *A neural network agent based approach to activity detection in AmI environments*, pages 92–99. IEEE International Workshop on Intelligent Environments. 2005.

[18] D. Rocchesso and R. Bresin. *Emerging Sounds for Disapearing Computers*, pages 233–255. The Disapearing Computer. Springer-Verlag, 2007.

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

[20] N. M. Sadeh, F. L. Gandom, and O. B. Kwon. Ambient intelligence: The mycampus experience. Technical Report CMU-ISRI-05-123, ISRI, 2005.

[21] E. P. Salonen, M. Turunen, J. Hakulinen, L. Helin, P. Prusi, and A. Kainulainen. Distributed dialogue management for smart terminal devices. In *Interspeech 2005*, pages 849–852, 2005.

[22] Helen Sharp, Yvonne Rogers, and Jenny Preece. *Interaction Design: Beyond Human Computer Interaction.* John Wiley and Sons Ltd., 2nd edition, 2007.

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

[24] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed.* Elsevier, 2005.

[25] G. M. Youngblood, D. J. Cook, and L. B. Holder. Managing adaptive versatile environ-

ments. In *IEEE International Conference on Pervasive Computing and Communications*, 2005.

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
```