Learning frequent behaviours of the users in Intelligent Environments

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

***Abstract*— Intelligent Environments (IEs) are expected to support people in their daily lives. One of the hidden assumptions in IEs is that they propose a change of perspective in the relationships between human and technology, shifting from a techno-centered perspective to a human-centered one. Unlike current computing systems where the user has to learn how to use the technology, an IE adapts its behaviour to the users, even anticipating their needs, preferences or habits. For that, the environment should learn how to react to the actions and needs of the users, and this should be achieved in an unobtrusive and transparent way. In order to provide personalized and adapted services, it is necessary to know the preferences and habits of users. Thus, the ability to learn patterns of behaviour becomes an essential aspect for the successful implementation of IEs.**

This paper presents a system, Learning Frequent Patterns of User Behaviour System (LFPUBS), that discovers user's frequent behaviours taking into consideration the specific features of IEs. The core of LFPUBS is the Learning Layer, which, unlike some other components, is independent of the particular environment in which the system is being applied. On one hand, it includes a language that allows the representation of discovered behaviours in a clear and unambiguous way. On other hand, coupled with the language, an algorithm that discovers frequent behaviours has been designed and implemented. For that, it uses association, workflow mining, clustering and classification techniques.

LFPUBS was validated using data collected from two real environments. In MavPad environment, LFPUBS was tested with different confidence levels using data collected in three different trials, whereas in WSU Smart Apartment environment LFPUBS was able to discover a predefined behaviour.

***Index Terms*— Ambient Intelligence, intelligent environments, pattern learning, machine learning techniques.**

I. Introduction

Intelligent Environments (IEs) defined as *digital environments that proactively, but sensibly, support people in their daily lives* [1], present the opportunity to use technology for the benefit of society in a range of applications. Potential benefits include making our environment more comfortable, safer and more energy efficient. Consider the following scenario that illustrates an IE that makes the life of its user easier and safer.

Michael is a 60-year-old man who lives alone and enjoys an assistance system that makes his daily life easier. On weekdays, Michael's alarm goes off close to 08:00 a.m.; 10-15 minutes later, he usually steps into the bathroom and the lights are turned on automatically. On Tuesdays, Thursdays and Fridays, he usually takes a shower; Michael prefers the temperature of the water to be 24-26 degrees Celsius in the winter and around 21- 23 degrees Celsius in the summer. When he finishes taking a shower, the fan of the bathroom is turned on if the relative humidity level of the bathroom is high (in Michael's case >70%). When he leaves the bathroom the fan and the lights are turned off automatically.

When he leaves home, the lights are turned off. Safety checks assess potentially hazardous situations (e.g., checking if the stove is turned on), and if needed, the assistance system acts accordingly (e.g., turning the stove off).

A. Motivation

Environments developed so far, are reactive environments that based on predefined patterns provide actions that are considered 'intelligent'. IEs will be really intelligent and adaptive if they are able to provide personalized services instead of services based on rigid or predefined patterns. For that, it is necessary to discover user habits and patterns.

Considering the activation of the bathroom fan in the example. When Michael has a shower, identified using activity recognition technologies, the fan can always be activated independently of the relative humidity level. It can also be activated when the relative humidity level exceeds a certain level (predefined level) or it may never be activated. If the environment uses a pre-defined rule to activate a device, the general rule does not take into account the preferences of the user, or else the rule is defined by someone who knows the user's preferences. In the first case, the environment does not fulfill the requirements of providing personalized services; in the second case, such an adaptation is achieved only by human input. The ideal environment automatically adapts to Michael's preferences and habits. In order to provide personalized and adapted services, the requirement of knowing the preferences and frequent habits of users is clear. In IEs, learning means that the environment has to gain knowledge about the preferences, needs and habits of the user in order to better assist the user [2], [3].

A frequent behaviour, can be initially defined as a set of actions and/or activities that a user frequently performs under certain conditions. Michael's morning habits include actions like 'Alarm On', 'Bathroom On', 'Shower On', ... which are related in a specific way (e.g., 'Shower On' comes after 'Bathroom On'), and they occur under certain conditions (e.g., 'Shower On' occurs only on Tuesdays, Thursdays and Fridays).

B. Advantages of Learning Frequent Behaviours

Assuming that human beings perform behaviours based on habits, it could be inferred that patterns describing past and

Manuscript received ??; revised ??.

A.Aztiria, R.Basagoiti and A.Izaguirre are with the University of Mondragon (Spain), J.C.Augusto is with Middlesex University (U.K.), D.J.Cook is with Washington State University, (U.S.A.).

present behaviours will define future behaviours as well.

Michael's example shows how the environment, knowing his frequent behaviours, can act proactively to make his life easier and safer. In this case, acting proactively means the environment can automatically turn the lights and the fan on and off or turn the radio on. The automation of actions and/or devices can be considered as positive side effects that can be obtained once the environment has learned his frequent habits.

Apart from automating actions or devices, patterns can also be used to understand his behaviour and act in accordance with it. Thus, unhealthy habits can be detected (e.g., he does not take the pill) and act in response (e.g., reminding him that he has medicine to take).

Making the environment more efficient in terms of saving energy (e.g., turning the fan on only when the relative humidity level is $>70\%$) or increasing safety (e.g., turning off the stove or issuing an alarm whenever Michael leaves it on) are other dimensions of daily life that can be supported by an IE because of knowledge it has discovered.

C. Intelligent Environments' special features

Learning systems, i.e., systems that automatically discover new knowledge, are being developed or used in many different areas. However, each area has different objectives, needs and features that influence the learning process. IEs have some features that can be used to differentiate them from other environments. In what follows, the most important features that influence the process of learning are analysed.

1) Importance of the user: Users are the focus of any development in IEs, and the fact that the environment is technologically rich must not require any extra effort by the users to obtain benefits from the IEs [4], [5].

2) Collected data: The importance of sensing increases when considering the learning process. The data collected from the sensors will greatly influence the learning process. One should consider the nature of the raw data. Actually, within the IEs field there are specific areas, named 'Data Fusion' and 'Activity Recognition' that deal with this problem.

Different types of sensors provide information that can be used for different purposes in the learning process. Some sensors provide direct information about user actions (e.g., a sensor installed in the bathroom's light switch provides direct information about when someone switches on the light). Other sensors provide information about the environment itself (e.g., a temperature sensor installed in the bathroom). Other types of sensors provide information about the health and emotional status of the user (e.g., sensors that capture parameters like heart rate) or externally gathered information (e.g., agenda information).

3) Representation of the discovered knowledge: Depending on the objectives of each environment, different representations can be desirable. For example, if the only goal is to automate, i.e., to provide an output given certain inputs (e.g., switch on the light given the current situation), the environment does not require a representation of the patterns that can be understood by the user. However the representation of the user's patterns can be relevant. In these cases, a human-understandable, i.e., comprehensible, representation of the patterns is an essential feature for the success of the system.

Representing frequent behaviours by means of Action Maps seems to be a promising approach. Action Maps, based on workflow representation, represent the complexity of users' behaviours (see Figure 1 for an example). This type of representation allows inter-relations between actions (e.g., 'Bathroom On' and 'BathroomLights On'). At the same time, it allows the representation of time relations using relative time references instead of absolute times (e.g., 'Shower Off'; 4 seconds later; 'BathroomFan On'). General Conditions help to contextualize the whole sequence (e.g., 'On weekdays between 8 a.m. and 9 a.m.'), whereas Specific Conditions describe the conditions under which an action is performed or not (e.g., 'BathroomFan On' only if the relative humidity level is $>70\%$).

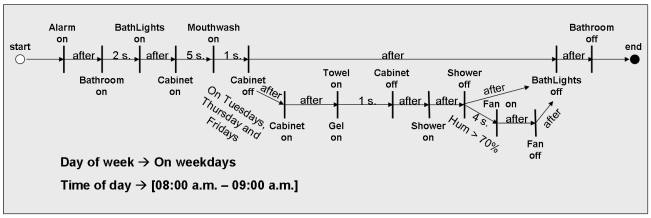


Fig. 1. Example morning ritual represented in an Action Map.

4) Scheduling the learning process: Although this work is focused on discovering frequent patterns of behaviour from data collected by sensors, the development of a complete IE demands other considerations.

It is desirable for the system to act as intelligently as possible from the very beginning, i.e., even when it is just beginning to collect data. Typically, actions performed at this point will not be as intelligent or efficient as those performed after learning the patterns of the user, and minimal services can be expected at this stage.

Once patterns are discovered, they must be continuously revised and updated:

- The user can change his/her preferences or habits (e.g., Michael takes a shower every weekday).
- New patterns could appear (e.g., Michael has started receiving visitors on weekends).
- Previously learned patterns were incorrect (e.g., the system wrongly concluded that Michael gets up around 07:00 a.m.).

This adaptation process could mean the modification of parameters in a pattern discovered previously, adding a new pattern or even deleting a pattern. This sustained process will last throughout the lifetime of the environment. To achieve this constant revision effectively, the user's feedback is essential. This feedback is obtained using an interaction system (See Section III-D) for more details.

II. State of the art

Mozer et al. [6] and Chan et al. [7] 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. Chan et al. developed a similar application in order to assess if a given situation was normal or abnormal. Other authors have kept using Artificial Neural Networks (ANNs) [8]. The use of ANNs has a limitation related to their internal structure not being easily comprehensible.

Cook and colleagues initially focused on building universal models, represented by means of Markov models, in order to predict either future locations or activities [9]. They made improvements by developing applications to discover daily and weekly patterns [10]. Jakkula and Cook [11] extend this work to predict actions using temporal relations, defined by means of Allen's temporal logic relations [12].

The efforts of researchers at Essex's iDorm [13] [14] focused on developing an application that generated a set of fuzzy rules to represent users' patterns.

Probabilistic methods such as Bayesian logic networks and Markov logic networks have been used to model activities [15] [16].

Other techniques have also been used. The 'SmartOffice' group [17] 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 [18]. Including a system, based on Case-Based Reasoning (CBR), they significantly improved the quality of filtering (i.e., user satisfaction increased from 50% to 80%).

[19] analyses the strengths and weaknesses of these techniques for learning frequent user behaviours. 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 [4], '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'.

III. General Architecture

One of the main characteristics of IEs is the key role that the user plays as the focus of the entire process. The process starts by collecting data about the user and the environment in which the user is situated, and it finishes by acting intelligently for the user.

This research suggests an architecture for learning users' frequent behaviours that distinguishes those aspects of the learning process related to particular environments in which each particular environment requires a different treatment (environment-dependent), from those aspects that can be generalized for all types of environments (environment-independent). The system proposed in the current work, Learning Frequent Patterns of User Behaviour System (LFPUBS), is based on a three-layered architecture that takes into account all aspects related to the learning process (Figure 2).

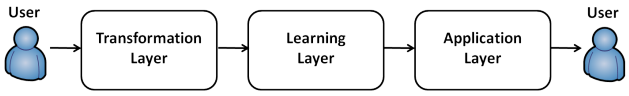


Fig. 2. Three-layered global architecture of LFPUBS.

A. Transformation Layer

The Transformation Layer transforms raw data, i.e., data collected from sensors, into useful information for the learning layer. An important transformation includes inferring users' actions from raw data. See Figure 3 for an example where the information provided by sensors is already meaningful.

from
2008-10-20T08:15:57, SwitchBathroomLights, on, 100
it is inferred
2008-10-20T08:15:57, BathroomLights, on, 100

Fig. 3. Example where Transformation Layer can easily transform sensor information.

In this case, the action itself is meaningful because the action of the user can be directly inferred from it. However, there are other actions that are difficult to infer from the sensor activation. For example, the inference of the simple action 'Go into the Bathroom' requires combining data coming from different sources to infer such an action. Figure 4 shows that there is a motion in the corridor, followed by the RFID tag installed in the door of the bathroom detecting the presence of Michael and finally there is motion in the bathroom.

from
2008-10-20T08:15:54, Motion Corridor, on, 100
2008-10-20T08:15:55, Bathroom RFID, on, Michael
2008-10-20T08:15:55, Motion Bathroom, on, 100
it is inferred
2008-10-20T08:15:55, Bathroom, on, 100

Fig. 4. Example where inferring a meaningful action demands to combine different actions.

The most basic way of inferring these actions is by means of templates. Templates define which actions must be combined as premises as well as which constraints must be considered. The importance of each action in the template is different, so that actions can be labeled either as mandatory or as optional. Constraints can affect the order of the actions or the duration. See Figure 5 for an example.

After transforming raw data into simple actions, all actions considered later are meaningful. Once simple actions have been inferred, a similar process can be carried out in order to infer complex actions such as 'Make coffee' or 'Take a pill'. This inference might be necessary because simple actions do not always represent the type of actions to be analysed. As in inferring simple actions, the most basic method for inferring complex actions is the use of templates, with one difference. Whereas the former transformation combines raw data, inferring complex actions combines simple actions.

'Go into Bathroom (Bathroom, On, 100)'

Actions:
- Motion Corridor (Mandatory)
- RFID Detection (Mandatory)
- Open Door (Optional if already open)
- Motion Bathrooom (Mandatory)

Constraints:

Order
- Motion Corridor <RFID Detection <Open door <Motion Bathroom

Time
- $T_{MotionBathroom} - T_{MotionCorridor} < 3seg.$

Fig. 5. The template for the action 'Go into bathroom'.

B. Learning Layer

The Learning Layer transforms the information from the Transformation Layer into knowledge to be used by the Application Layer. This layer discover users' frequent behavior in an environment-independent manner so that it can be used in any environment without any modification in its design. The Learning Layer is described in detail in Section IV.

C. Application Layer

Once pieces of knowledge about users' frequent behaviours have been learned, they can be used for different purposes. This use will be mainly influenced by the objectives of particular environments.

Michael's scenario suggested that discovered frequent behaviours could be used in order to automate the activation/deactivation of devices. For instance, if the environment knows that on weekdays between 8 a.m. and 9 a.m., Michael turns on the lights of the bathroom 2 seconds after he goes into the bathroom, it can act to proactively anticipate Michael's actions in those cases.

Another interesting application for patterns of behaviour is that they allow us to understand users' frequent behaviours in order to detect unhealthy habits or provide help and support for his daily tasks. A representation of patterns and related actions must be comprehensible to make understanding them easier. Here it detects when Michael does not take his medicine and the environment generates a reminder to take medicine with breakfast.

D. Interacting with LFPUBS

An important aspect of IEs has to do with their interaction with users, a key element in the process of efficiently applying the extracted knowledge. Given the importance of users for the success of an IE, it is essential that there is a friendly and easy way for the user to interact with the environment. The ability of HCI systems to understand and react to human behaviours has been widely analyzed [20], [21].

Here, a speech-based interaction system provides IEs with a more natural way of interacting with all types of users. As a first approach, a speech-based HCI system has been developed. The goal of this system was to allow users to give their feedback about discovered behaviours.

Different environments and different objectives require the development of different interaction systems. The methodology to develop an HCI system where users' frequent behaviours are involved is the same in all applications. Depending on the nature and the objectives of each environment, it will be necessary to modify the possible questions as well as the options given to the user, but the technology will remain untouched. In this case, the chosen speech synthesizer is FreeTTS 1.2 and the speech recognizer is Sphinx- 4 (See [22]).

A Graphical User Interface (GUI) allows the user of LFPUBS to design the learning process that will be carried out. Some of the components of LFPUBS allow the user of LFPUBS to design the learning process based on the requisites (runtime, complexity of the knowledge to discover etc.) of different applications.

IV. Learning Frequent Patterns of User Behaviour System (LFPUBS)

Two different approaches for LFPUBS have been developed. The general approach, named 'Action Map Approach', attempts to discover patterns of users' behaviours and represent them in a comprehensible way (as depicted in Figure 1). The second approach, named 'Pairwise Approach', is a particular instantiation of the Action Map approach because it is focused on discovering pairwise relations between the actions of the user. In this work we are going to focus on the Action Map Approach.

The main objective of this approach is to discover users' frequent behaviours and represent them in a comprehensible way. As mentioned in Section I, different sensors provide different types of data that define different aspects of users' behaviours. Therefore, the nature of different pieces of information must be taken into account in the learning process. Thus, LFPUBS considers two main different groups of information:

- (type A) Information about the actions of the users. This information can be directly provided by sensors installed in objects (devices, furniture, domestic appliances etc.) or inferred by combining different pieces of information in the Transformation Layer. This set of actions is denoted by $A = \{a_i\}$.
- (type C) Context information. Some sensors provide information about context, but not about actions of the user. Temperature, light and smoke sensors are examples of type C sensors. The context information is denoted by $C = \{c_i\}$.

The Learning Layer is made up of two modules, the representation module and the discovering module. The core of the representation module is a language ($\mathcal{L}_{LFPUBS}$) that provides a standard conceptualization of the patterns so that the environment is able to represent all type of patterns that can occur in the environment [22]. The process of discovering is based on an algorithm ($\mathcal{A}_{LFPUBS}$) that takes into account for characteristics of the IEs' attempts to discover frequent patterns.

A. Representing patterns with $\mathcal{L}_{LFPUBS}$

Because of the complexity of IEs, defining a language that allows the environment to represent discovered patterns in a

clear and unambiguous way is difficult but necessary. The language integrated within LFPUBS is based on ECA (Event-Condition-Action) rules [23]. Besides providing a standard way of representing patterns, it makes sure those patterns are clearly specified and enables other technologies to check their integrity [24]. A frequent behaviour is defined by means of an Action Map, which contains all of the specific relations between actions. An Action Map is created relating actions in pairs, and each of those relations is an Action Pattern, which is defined by means of ECA rules. The complete Action Map is then created by linking different Action Patterns.

In the same way as ECA rules, each Action Pattern relates two actions (defined by the ON and THEN clauses) and the specific conditions (defined by the IF clause) under which that relation occurs. Unlike basic ECA rules, $\mathcal{L}_{LFPUBS}$ allows the environment to define the time relation between both actions. The behaviour of turning on the fan in the bathroom is represented using $\mathcal{L}_{LFPUBS}$ in Figure 6:

```
(Action Pattern)
  ON occurs (simple,(Shower,Off),t0)
  IF context (Bathroom humidity level(>,70%))
  THEN do (simple,(On,BathroomFan),t) when t = t0 + 4s
```

Fig. 6. Action Pattern for turning on fan.

The ON clause defines the event that occurs and triggers the relationship specified by the pattern. The components of the Event Definition are the device ('Shower') implied in the action, the nature of the action ('Off') and the timestamp of such an action ('t0'). As patterns relate users' behaviours, the ON event must be the effect of a user's action. Such actions are collected by A-type sensors.

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. Because it is almost impossible for an Event-Action relation to be true under any condition, appropriate conditions are necessary to represent accurate patterns. Conditions are defined by means of attribute-value pairs. Whereas the ON and THEN clauses define the actions of the user, the conditions must specify the status of the environment at that moment, such that the information involved in that clause must be related to the context. $\mathcal{L}_{LFPUBS}$ has two ways to define conditions:

- Information coming from C-type sensors (e.g., 'Relative humidity level' or 'Temperature').
- Calendar information (e.g., 'Time of Day' or 'Day of Week').

Possible attribute values depend on the nature of each attribute. $\mathcal{L}_{LFPUBS}$ considers two types of values:

- Qualitative values (e.g., 'Tuesday').
- Quantitative values (e.g., '20C' or '20:30:00').

The THEN clause defines the action that the user usually carries out given the ON clause and given the conditions defined in the IF clause. It must be a user action, so that is it made up of the device ('BathroomFan'), the nature of the action ('On') and the Time Relation between the Event and Action situations. The Time Relation can be either quantitative (t = t0 + 4s.) or qualitative (t is after t0), with the usefulness of each type of relation being different. Compared to qualitative relations, quantitative relations provide higher quality information because it is possible to use them for other purposes. One of those additional purposes is the automation of devices, which is possible with quantitative relations. Consider Michael's behaviour of turning on the fan 4 seconds after having a shower. If such a relation was defined by means of a qualitative term like 'after', the system would not be able to infer when it had to turn on the fan because it would not know whether the time delay was 4 seconds, 5 minutes or 2 hours. However, using quantitative relations (4 seconds in Michael's case) allows the system to turn on the fan at the right time.

B. Learning patterns with $\mathcal{A}_{LFPUBS}$

The algorithm ($\mathcal{A}_{LFPUBS}$) discovers the frequent behaviours of the users (See Figure 7). In order to coordinate with $\mathcal{L}_{LFPUBS}$ and discover complete and unambiguous patterns, $\mathcal{A}_{LFPUBS}$ must consider different aspects defined by the language. Therefore, the $\mathcal{A}_{LFPUBS}$ has to be able to discover Action Maps that represent users' frequent behaviours.

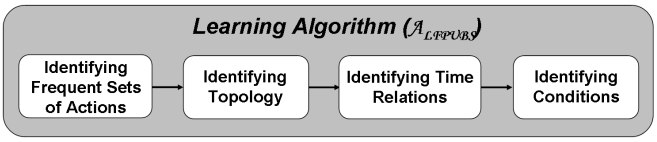


Fig. 7. Steps to be performed by the learning algorithm.

The pseudocode of the algorithm can be seen in Figure 8:

$\mathcal{A}_{LFPUBS}$ Algorithm

Input: $D = \{d1, d2...dn\}$ (Dataset) // for example,
(d_1 = 2008-10-20T08:15:55, Bathroom, on, 100
d_2 = 2008-10-20T08:15:57, BathroomLights, on, 100)
$Conf.$ (Demanded minimum confidence level)
Output: F (Frequent Behaviours)

$F \leftarrow IdentifyingFrequentSetsofActions(D; Conf.)$
for $f_i \in F$
  $p_i \leftarrow IdentifyingTopology(D, f_i)$
  for $p_{k,j} \in p_i$
    $T(f_{i_j}, f_{i_k}) \leftarrow IdentifyingTimeRelations(p_{k,j}, D)$
    $e_{j,k}(f_{i_k}, f_{i_j}) \leftarrow IdentifyingConditions(p_{k,j}, D)$
return F

Fig. 8. $\mathcal{A}_{LFPUBS}$ algorithm.

The parameters for the learning process can be defined by means of the GUI.

1) Identifying Frequent Sets of Actions: This step discovers the sets of actions that frequently occur together (Frequent Sets). Defining a demanded minimum level (minimum confidence level), it discovers all those sets of actions that occur more times than the minimum level (See Figure 9). These sets

of actions are treated as Frequent Sets. Let F denotes the set of Frequent Sets of a user. Then $f_i \in F$ is a set of actions $\{f_{i_k} : f_{i_k} \in A\}$. The set of Sequences in which the Frequent Set is present is also identified. To discover Basic Frequent Sets in large amounts of data, the Apriori algorithm [25] was used. Starting from 1-item sets (Step 1), the algorithm attempts to find subsets (Step 2) which are common at least a minimum number of times (Step 3).

Identifying Frequent Sets of Action

Input: $D = \{d1, d2...dn\}$ (Dataset)
$Conf.$ (Demanded minimum confidence level)
Output: F (Frequent Behaviours with Frequent Sets of Actions' information)

$F_1 \leftarrow \{1\text{-itemsets}\}$ // Step 1
$k \leftarrow 2$
while $F_{k-1} \neq \{\}$
  $c_k \leftarrow \{c | c \in a \cup \{b\} \wedge a \in F_{k-1} \wedge b \in F_{k-1} \wedge b \notin a\}$
  // Step 2
  $count(c_k) \leftarrow count(c_k, D)$
  $F_k \leftarrow \{c | c \in c_k \wedge count(c_k \geq Conf.)\}$// Step 3
  $k \leftarrow k + 1$
return $\cup F_k$

Fig. 9. Identifying Frequent Sets of Action algorithm.

In Michael's case, this first step would discover that the following actions frequently occur together.

Actions: *'Alarm On'; 'Bathroom On'; 'Bathroom Off'; 'BathroomLights On'; 'BathroomLights Off'; 'Cabinet On'; 'Cabinet Off'; 'Mouthwash On'; 'Towel On'; 'Gel On'; 'Shower On'; 'Shower Off'; 'BathroomFan On'; 'Bathroom-Fan Off';*

2) Identifying Topology: In order to properly model the user's behaviours, it is necessary to define the order of included actions. The goal of this step is to discover the frequent order (defined as Topology) of the actions in the behaviour of the user. In this context, representing the user's behaviours by means of Action Maps makes them easier to understand and makes it possible to use them in tasks such as prediction or automation of future actions. As stated previously, few groups have dealt with this problem in IEs. Because of this, other meaningful domains in which user's actions have been used to extract models of behaviour have been analysed. One of the closest domains is the area of Workflow Mining [26]–[28] in which process models are discovered from event logs. Instead of event logs, $\mathcal{A}_{LFPUBS}$ considers the actions of the user. Because of the nature of IEs, some particularities must be taken into account.

First, considering Frequent Set's actions, all relations defined by the data are represented in an Action Map. The objective is to define the initial probabilities, $P_{0,j} = Pr(f_{i_j}|f_{i_{start}})$ (Step 1) and the transition probabilities for each Action Pattern $P = [P_{k,j}]$ where $P_{k,j} = Pr(f_{i_j}|f_{i_k})$ (Step 2) (See Figure 10).

Although this step does not discover anything, at this point,

Identifying Topology

Input: $D = \{d1, d2...dn\}$ (Dataset)
f_i (Frequent Set of Actions)
Output: p_i (Frequent Sets of Actions with Topology)

$p_i \leftarrow \{p_0, p_{k,j}\}$
for $f_{i_j} \in f_i$
  $p_{0,j} \leftarrow Pr(f_{i_j}|f_{i_{start}})$ // Step 1
  for $f_{i_k} \in f_i$
    $p_{k,j} = Pr(f_{i_j}|f_{i_k})$ // Step 2
return p_i

Fig. 10. Identifying Topology algorithm.

it provides the first formal representation of the behaviour based on the $\mathcal{L}_{LFPUBS}$ in which the ON and THEN clauses of different Action Patterns are defined. In Michael's case, his behaviour would be represented by the Action Map depicted in Figure 11.

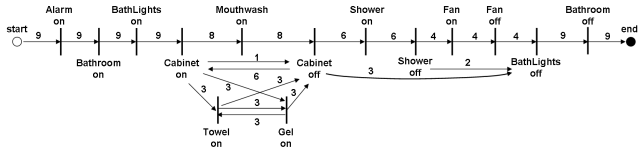


Fig. 11. The basic representation of Michael's behaviour.

Repetitive Actions

Unlike other domains in which an action is unique and there is no more than one occurrence of each action per Sequence, in IEs, there could be different occurrences of the same action. In fact, the nature of repetitive occurrences will probably be different because the user can do the same action with different purposes. Thus, a methodology to identify repetitive actions and create different instantiations of them is created based on the idea that the meaning, and by extension, the nature of an action is mainly defined by the previous and next actions. Thus, the nature of different actions is defined by creating groups of actions that take into account the similarities among previous and next actions. LFPUBS includes two different techniques to create groups:

- Manually define the number of groups or clusters to create. The system helps the user considering a.) the average number of occurrences of an action in a Sequence and b.) the maximum number of occurrences of an action in a Sequence.
- Automatically define the number of groups or clusters using the Expectation-Maximization (EM) algorithm [29].

In Michael's case, it would discover that the actions 'Cabinet On' and 'Cabinet Off' occur more than once with different objective. Thus, it would create as many instantiation as necessary (two in this case) for each one of those actions. In Michael's case, the representation of his behaviour, taking into account repetitive actions, can be seen in Figure 12.

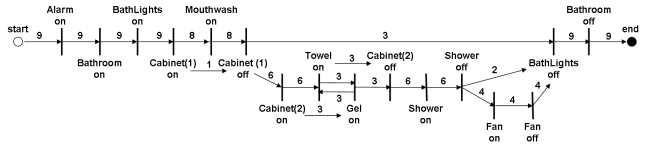


Fig. 12. Michael's behaviour with repetitive actions.

Unordered Subsets of Actions

An unordered subset of actions represents a set of actions in which it has not been possible to define an order for such actions. Michael's scenario offers a typical example of this. When he opens the cabinet, he sometimes gets the towel first and then the gel, and other times the order is reversed. As in the parallel actions of Workflow mining cases, the representation of unordered set of actions shows bidirectional relationships between such actions. To decide whether a bidirectional relationship (for example between a_1 and a_2) must be considered as an unordered set of actions, LFPUBS includes a set of parameters:

- *Minimum Level for Origin (MLO)* (%). The percentage of occurrences of a_1 followed by a_2 out of the total occurrences of a_1 must be higher than the demanded minimum level.

$$a_1 \rightarrow a_2 / Occurrences(a_1) \geq MLO \tag{1}$$

- *Minimum Level for Destiny (MLD)* (%). The percentage of occurrences of a_2 followed by a_1 out of the total occurrences of a_2 must be higher than the demanded minimum level.

$$a_2 \rightarrow a_1 / Occurrences(a_2) \geq MLD \tag{2}$$

- *Minimum Balanced Level (MBL)* (%). The percentage of occurrences of a_1 followed by a_2 out of the occurrences of a_2 followed by a_1 (and vice versa) must be higher than the demanded minimum level.

$$((a_1 \rightarrow a_1)/(a_2 \rightarrow a_1)) \& ((a_2 \rightarrow a_1)/(a_1 \rightarrow a_2)) \geq MBL \tag{3}$$

In Michael's case, the only bidirectional relationship involves the actions 'Towel On' and 'Gel On'. Depending on the parameters we define, those actions may be identified as part of an unordered subset of actions. In that case, Michael's morning behaviour could be represented as shown in Figure 13:

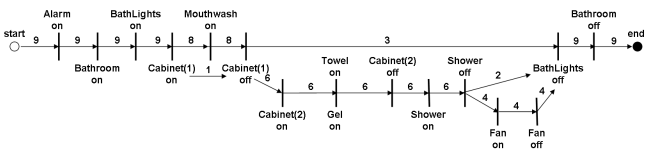


Fig. 13. Michael's behaviour with unordered subsets of actions.

Granularity and Allowed Maximum Granularity

Once the behaviours of the user have been discovered and represented, LFPUBS allows the user of the system to select the complexity (granularity) of the Action Map. The selected granularity indicates the threshold for the relationships. Thus, relationships with a lower frequency than the threshold will not be represented. Therefore, a high granularity will provide a more general representation of an Action Map, whereas a low granularity will provide a more complex representation of it. Selecting granularities without any limit can lead to illogical representations. To avoid illogical representations, LFPUBS includes the heuristic 'Uniform-cost search' [30], which calculates the maximum granularity (Allowed Maximum Granularity) that guarantees at least one path from start to end. Thus, the system does not allow the user to select a higher granularity level than this parameter, making sure that the Action Maps maintains a minimum logic in all cases.

Assume that in Michael's case LFPUBS allows the user of the system to select any granularity and he/she chooses '5', such that all relationships with a frequency under '5' would be removed. As can be seen in Figure 14, the representation of Michael's behaviour would lose its sense, making it difficult to understand it logically.

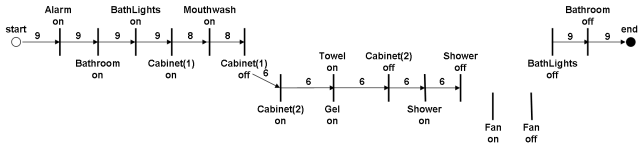


Fig. 14. Michael's behaviour without considering the Allowed Maximum Granularity parameter.

In Michael's case, using the 'Uniform-cost search' heuristic, LFPUBS would discover that the Allowed Maximum Granularity is '4'. Thus, the user of LFPUBS will not be able to select a granularity level higher than '4'.

3) Identifying Time Relations: The Topology defines a first temporal representation of the frequent behaviour by means of qualitative relations (using the term 'after') and their probabilities. The objective of this step is to discover frequent quantitative Time Relations between the actions defined by each one of the Action Patterns.

As stated above, qualitative relations allow one to understand the logical order of the actions. Such patterns would provide higher quality information if the relations were defined, if possible, by means of a quantitative relations. Therefore, the objective of this step is to discover quantitative Time Relations, $T(f_{i_j}, f_{i_k}), \forall f_i \in F$, in order to better define users' behaviours. LFPUBS includes two different algorithms.

Before applying any algorithm, the first task is to collect the necessary data. The relations to study are already defined by the patterns discovered in the previous step. Thus, for each pattern the system collects the Time Distances of all occurrences.

Once the Time Distances are collected, the next step is to identify possible quantitative time relations. LFPUBS includes two algorithms - the 'Basic Algorithm' and the 'EM Algorithm' - so that the user of the system may choose either of

them to identify such quantitative relations. Both algorithms are based on the same idea of grouping occurrences by taking into account their similarity and deciding whether a group represents a quantitative Time Relation. Figure 15 shows the pseudo code of the 'Basic Algorithm'. For each group, taking as starting point any of the time relations from the dataset (Step 1), the idea is to calculate the distance from each occurrence to all groups (Step 2), and assign it to that group with the minimum value (Step 3), until all groups are steady (Step 4). For more details about these algorithms see [31].

Identifying Time Relations: 'Basic Algorithm'

Input: $D = \{d1, d2...dn\}$ (Dataset)
p_i (Frequent relations defined by the topology)
Output: T (Time Relations for such topology)

for $t_i \in T$
  $t_i \leftarrow Ch(d_i) \in D$ // $Ch(d_i)$ extracts time of d_i // Step 1
$changed \leftarrow false$
repeat
  for $t_i \in T$
    for $d_i \in D$
      $minDist \leftarrow EuclidianDist(Ch(d_i), t_j) j \in \{1...n\}$
      // Step 2
      if $minDist \neq l(Ch(d_i))$
        $l(t_i) \leftarrow minDist$ // Step 3
        $changed \leftarrow true$
      else $changed \leftarrow false$
until $changed = true$ // Step 4
return T

Fig. 15. Identifying Time Relations algorithm.

Considering Michael's case, and particularly the Action Pattern that relates the actions 'Shower Off' and 'BathroomFan On', the objective of this step is to define a quantitative Time Relations for such a pattern. Consider that the collected Time Distances between both actions are depicted in Figure 16.

	O1 (Sequence 1)	O2 (Sequence 5)	O3 (Sequence 6)	O4 (Sequence 8)
Shower Off				
BathroomFan On	00:00:04 (4s)	00:00:03 (3s)	00:05:34 (334s)	00:00:05 (5s)

Fig. 16. Time Distances between occurrences of 'Shower Off' and 'BathroomFan On'.

After using any of the algorithms, two groups ('group 0' and 'group 1') are created, which cover three and one occurrences, respectively. In order to extract quantitative Time Relations, LFPUBS checks whether the confidence level of different groups is over the demanded one. 'group 0' represents a quantitative Time Relation because it covers 3 out of 4 occurrences (75%), whereas 'group 1' does not. Thus, the mean value of 'group 0' defines a quantitative Time Relation, creating a pattern shown in Figure 17:

4) Identifying Conditions: A final step that identifies the Specific and General Conditions for each Action Map is necessary in order to create accurate representations of the behaviours of the user. Next, both processes of identifying Specific and General Conditions are explained in more detail. Let E denotes a set of conditions. Then $e_i \in E$ is a set of contextual and temporal information, $\{e_{i_k} : e_{i_k} \in C\}$.

```
(Action Pattern)
 ON occurs (simple,(Shower,Off),t0)
 IF context ()
 THEN do (simple,(On,BathroomFan),t) when t = t0 + 4s
```

Fig. 17. Action Pattern for turning on fan with Time Relation.

Identifying specific conditions

All of the relations represented in an Action Map are supported by a number of occurrences. One situation is the case in which an action is followed by two (or more) different actions. In an Action Map, these situations are easily identified because these situations are represented as splitting points from which more than one relation are created. In those cases, it is necessary to identify under what conditions each of those relations is true.

Once such situations are identified, the process of identifying conditions is as follows. For each possible relation a table is created. In each table the occurrences covered by that relation are collected, together with the calendar and context information collected when such occurrences happened. Once the tables are created, separating both tables by using the information they contain allows one to discover conditions. The task of separating can be solved by treating it as a classification problem [32]. This is a recursive algorithm (See Figure 18) that starting from a tree (Step 1), calculates if that tree classifies well all occurrences (Step 2). For that, function $allPositive(e, D)$ and $allNegative(e, D)$ return if all occurrence of D are classified positively or negatively respectively by the e Node. If not, a new attribute is selected (Step 3) and added to the tree, creating a new branch for each possible value (Step 4) and checking if it is final (Step 5).

In Michael's case, the action 'Shower Off' is followed by either 'BathroomFan On' or 'BathroomLights Off' actions. As there are two possible relations two tables (See Figure 19) are created. Context and calendar information collected when occurrences of 'Shower Off' were followed by the action 'BathroomFan On' are recorded in the table 'ShowerOff-BathroomFanOn'. Information collected when occurrences of 'Shower Off' were followed by the action 'BathroomLights Off' are recorded in the table 'ShowerOff-BathroomLightsOff'.

There are different ways of separating both tables because of the small number of Sequences considered in Michael's case. The condition that better separates both tables is the context information Bathroom relative humidity level ('Hum. Bathroom'). For the relation 'Shower Off'-'BathroomFan On', the Specific Condition defined by the $\mathcal{A}_{LFPUBS}$ can be seen in Figure 20.

Identifying general conditions

It is necessary to define the general context in which an

Identifying Conditions

Input: $D = \{d1, d2...dn\}$ (Dataset)
p_i (Frequent relations defined by the topology)
Output: E (Set of conditions)

$e \leftarrow$ create Root Node // Step 1
if $allPositive(e, D)$ Return e // Step 2
if $allNegative(e, D)$ Return e // Step 2
$G \leftarrow calculateGain(e, D)$ // Step 3
$e \leftarrow G$
for $v_i \in G$
  $e \leftarrow D_{v_i} \subset D$ where $G = v_i$ //Step 4
  if $D_{v_i} = \{\}$ // Step 5
    $e \leftarrow leaf(e) = mostCommonValue(G)$
  else
    $IdentifyingConditions(e, D_{v_i})$
Return e

Fig. 18. Identifying Conditions algorithm.

Shower Off – BathroomFan On

	Sequence 1	Sequence 5	Sequence 6	Sequence 8
time of day	08:29:37	08:29:28	08:30:39	08:23:29
day of week	Monday	Friday	Monday	Wednesday
Temp. Bathroom	19	22	21	17
Hum. Bathroom	72	75	71	78

Shower Off – BathroomLights On

	Sequence 3	Sequence 10
time of day	08:28:18	08:29:07
day of week	Wednesday	Friday
Temp. Bathroom	22	18
Hum. Bathroom	65	69

Fig. 19. 'Shower Off - BathroomFan On' and 'Shower Off - BathroomLights Off' tables with calendar and context information

Action Map occurs. General Conditions refer to calendar and context information that allows the user of the system to understand under what conditions the whole Action Map occurs. Here, only calendar information ('Time of Day' and 'Day of Week') has been considered, although $\mathcal{L}_{LFPUBS}$ also allows one to represent General Conditions using context information.

In order to identify General Conditions, defined by the terms 'Time of Day' and 'Day of Week', we have adopted a very basic strategy. Such a strategy is based on covering all the occurrences. Thus, all occurrences are covered by the range defined by 'Time of Day' and 'Day of Week' terms.

In Michael's case, it discovers that the particular behaviour we are considering happens on weekdays and between 08:00 a.m. and 09:00 a.m.. This first version provides quite simple and 'excessive' conditions.

IF context (Bathroom humidity level(>,70%))

Fig. 20. Specific Condition for the Action Pattern

V. Validation

Part of the validation process involved the use of datasets collected from two real environments. Some of the validations were general, whereas others were focused on validating specific steps of the $\mathcal{A}_{LFPUBS}$. The validation process was carried out using different datasets collected in MavPad and WSU Smart Apartment environments.

A. Validating LFPUBS with the MavPad dataset

The MavPad is an on-campus student apartment located at University Village on the University of Texas at Arlington's campus [33]. Figure 21 shows the sensors installed in MavPad:

- A1...A16: Sensors for living room electrical outlets, lamps and fan.
- B1...B13: Sensors for Bathroom and Kitchen lights, electrical outlets and fan.
- C1...C10: Sensors for Bedroom lamps and electrical outlets.
- S1...S144: Temperature, humidity, smoke and gas sensors.
- V1...V40: Motion sensors.

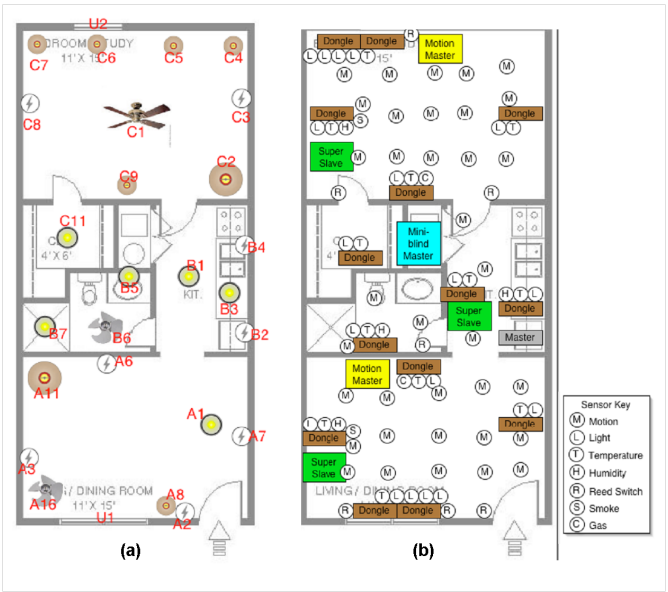


Fig. 21. a.) Sensors installed in different items. b.) Distribution of context and motion sensors. [33]

The dataset used to validate LFPUBS was collected in three different time periods: Trial 1 (spanning 15 days), Trial 2 (spanning almost 2 months) and Trial 3 (spanning 3 months). The frequent behaviours were not known in advance. The data collected in each trial was independent from the other trials' data.

Regarding the 'Identifying Frequent Sets of Actions' step, the number and the nature of the Frequent Sets to be discovered were unknown. At this point, the tests were only able to confirm that LFPUBS was able to discover Frequent Sets of Actions in unknown datasets.

One of the Frequent Sets discovered in Trial 1 with a minimum confidence level of 50% shows that the user performed the actions of 'BedroomLight On', 'Bedroom- Light

Off', 'BedroomLuxo1 On' and 'BedroomLuxo1 Off' together. The number of Frequent Sets does not provide interesting information by itself. For that, it was necessary to discover the topology of each Frequent Set. Identifying the topology of different Frequent Sets implied the discovery of repetitive actions and unordered subsets of actions as well as identifying the Allowed Maximum Granularity for each Frequent Set.

Considering the Frequent Set that involved the actions of 'BedroomLight On', 'BedroomLight Off', 'BedroomLuxo1 On' and 'BedroomLuxo1 Off', the topology discovered for this set of actions shows that the user first turned on the room light. Then sometimes he/she first turned off the room's light before turning on the luxo lamp, and other times he/she first turned on the luxo lamp and then turned off the room's light. This behaviour is represented by means of an unordered subset that groups the actions of 'BedroomLight Off' and 'BedroomLuxo1 On'. The user turned off the luxo lamp. Once the topology was defined, it was possible to represent such a behaviour by means of the $\mathcal{L}_{LFPUBS}$. Table I shows the number of patterns discovered in each trial and the convergence time for the discovering of patterns and their topology.

TABLE I
THE NUMBER OF ACTION MAPS AND THE EXPERIMENTS' RUNTIMES (IN MILLISECONDS) OBTAINED IN DIFFERENT TRIALS.

	Trial 1	Trial 2	Trial 3
Confidence Level	**Total Patterns**	**Total Patterns**	**Total Patterns**
25%	8 (328 ms)	3 (5969 ms)	1 (200ms)
50%	4 (141 ms)	1 (5422 ms)	1 (115 ms)
75%	1 (109 ms)	1 (718 ms)	1 (103 ms)
100%	0 (35 ms)	0 (35 ms)	0 (35 ms)

Time Relations were calculated using the 'Basic Algorithm'. In this case, it was also impossible to foresee the quantitative Time Relations to be discovered. However, it was possible to extract interesting information. Table II shows the number of Time Relations (and the percentage they represent) discovered in each trial. The runtime of each experiment is directly related to the number of relations to be analysed and the number of particular Time Distances to consider in each one of them. For example, this last aspect is essential to understand the runtimes of Trial 1 and Trial 2, because in both situations the number of relations to analyse were equal (56), but the number of particular Time Distances to consider was higher in Trial 2.

In some situations there was not any relation to analyse. There could be two reasons for that, on one hand, because it was not discovered any Frequent Set (e.g., Trial 1, Trial 2 and Trial 3 with confidence level of 100%). It could be because all the actions of the Frequent Set were included in an unordered subset of actions (e.g., Trial 1 with confidence level of 75% and Trial 3 with confidence level of 25%, 50% and 75%).

Specific and General Conditions were discovered. In some cases, for example with 'Action Map 1', it was not necessary to discover Specific Conditions because there was no situation in which an action was followed by two different actions. In contrast, in other Action Maps, it was necessary to define Specific Conditions. Table III shows the percentage of times

TABLE II
THE NUMBER OF ACTION PATTERNS WITH TIME RELATIONS, THE PERCENTAGE OF THE TOTAL AND THE EXPERIMENTS' RUNTIMES (IN MILLISECONDS) OBTAINED IN DIFFERENT TRIALS.

	Trial 1	Trial 2	Trial 3
Confidence Level	**Action Patterns with Time Relations**	**Action Patterns with Time Relations**	**Action Patterns with Time Relations**
25%	48 (48/56 (86%)) (844 ms)	43 (43/56 (77%)) (2071 ms)	No Time Relations needed
50%	18 (18/22 (82%)) (437 ms)	5 (5/7 (71%)) (1651 ms)	No Time Relations needed
75%	No Time Relations needed	5 (5/7 (71%)) (578 ms)	No Time Relations needed
100%	No Time Relations needed	No Time Relations needed	No Time Relations needed

in which it was possible to discover Specific Conditions when situations required this. The runtime of these experiments does not directly depend on the number of situations in which conditions were needed, but it was more influenced by the amount of data to be considered in each one of them.

TABLE III
THE NUMBER OF ACTION PATTERNS WITH SPECIFIC CONDITIONS, THE PERCENTAGE OF THE TOTAL SITUATIONS THAT REQUIRED THIS AND THE EXPERIMENTS' RUNTIMES OBTAINED IN DIFFERENT TRIALS.

	Trial 1	Trial 2	Trial 3
Confidence Level	**Action Patterns with Conditions**	**Action Patterns with Conditions**	**Action Patterns with Conditions**
25%	7 (7/10 (70%)) (500 ms)	6 (6/9 (67%)) (1015 ms)	No Conditions needed
50%	2 (2/3 (67%)) (188 ms)	1 (1/2 (50%)) (1062 ms)	No Conditions needed
75%	No Conditions needed	1 (1/2 (50%)) (156 ms)	No Conditions needed
100%	No Conditions needed	No Conditions needed	No Conditions needed

Because of the ability of the algorithm to generalise, it was possible to discover General Conditions for all of the Action Maps. For example, LFPUBS discovered that the Action Map described above occurred between 08:00 a.m. and 04:15 a.m. of the next day. Once General Conditions were discovered, 'Action Map 1' was represented as shown in Figure 22:

B. Validating LFPUBS with the WSU Smart Apartment dataset

The testbed WSU Smart Apartment is an environment created at Washington State University [34]. Figure 23 shows the installed sensors:

- M01...M26: motion sensors.
- I01...I08: item sensors for oatmeal, raisins, brown sugar,bowl, measuring spoon, medicine container, pot and phone book.
- D01: cabinet sensor
- AD1-A, AD1-B, AD1-C: water and burner sensors.
- AD1-C: burner sensor.
- asterisk: phone usage.

```
(Action Map 1)
(General Condition)
context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<:04:15:00)) |
context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<:23:59:59))

(Action Pattern 0)
ON occurs (start,--,t0)
IF context ()
THEN do (simple,(On,BedroomLight),t) when –

(Action Pattern 1)
ON occurs (simple,(BedroomLight,On),t0)
IF context ()
THEN do (unordered,((Off,BedroomLight) & (On,BedroomLuxo1)),t)
          when t = t + 346s

(Action Pattern 2)
ON occurs (unordered,((BedroomLight,Off) & (BedroomLuxo1,On)),t0)
IF context ()
THEN do (simple,(Off,BedroomLuxo1),t) when t = t + 227s

(Action Pattern 3)
ON occurs (simple,(BedroomLuxo1,Off),t0)
IF context ()
THEN do (--,end,t) when --
```

Fig. 22. Action Map after general conditions were discovered.

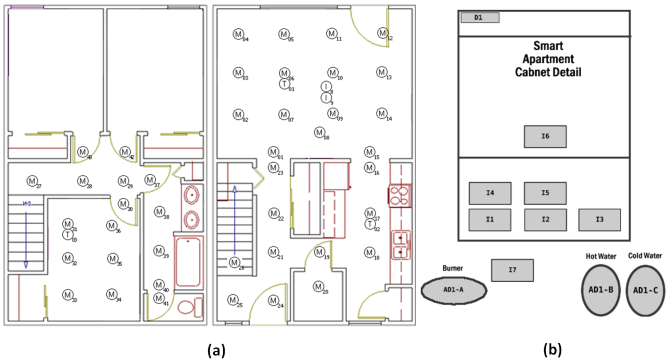


Fig. 23. a.) Distribution of motion sensors. b.) Cabinet and other sensors' distribution. [34]

Data collected in the WSU Smart Apartment represented participants performing the same five ADLs (Activities of Daily Living) in the apartment, so the frequent behaviours that LFPUBS should discover were known in advance. The five tasks were; make a phone call, wash hands, cook, eat and clean (see actions in Table IV).

As the behaviours were known in advance, this test evaluated the steps of 'Identifying Frequent Sets of Actions' and 'Identifying Topology'.

The objective of the first step, 'Identifying Frequent Sets of Actions', was to discover the set of actions involved in the different ADLs. Each particular Sequence showed a participant performing the five ADLs, so the same actions were involved in most of the particular Sequences. Thus, even for a high confidence level (for example, 60%), all of the actions involved in the ADLs, shown in Table IV, were identified as frequent.

'PhoneBook On', 'Phone On', 'Phone Off', 'Water On', 'Water Off', 'Cabinet On', 'Cabinet Off', 'Raisins On', 'Oatmeal On', 'MeasuringSpoon On', 'Bowl On', 'Sugar On', 'Pot On', 'Burner On', 'Burner Off', 'Medicine On', 'Medicine Off'.

TABLE IV
ACTIONS INVOLVED IN EACH ADL.

Activity	Involved Actions
Make a phone call	'PhoneBook On' –>'Phone On' –>'Phone Off'
Wash hands	'Water On' –>'Water Off'
Cook	'Cabinet On' –>'Raisins On' –>'Oatmeal On' –>'MeasuringSpoon On' –>'Bowl On' –>'Sugar On' –>'Cabinet Off' –>'Water On' –>'Water Off' –>'Pot On' –>'Burner On' –>'Burner Off'
Eat	'Cabinet On' –>'Medicine On' –>'Cabinet Off' –>'Water On' –>'Water Off' –>'Cabinet On' –>'Medicine Off' –>'Cabinet Off'
Clean	'Water On' –>'Water Off'

Once the set of actions involved in the Action Map was identified, the next step was to discover the frequent order of such actions. The particularity of this dataset was that, although the order of all of the actions was not clearly defined, the order of activities defined it in some way. The first difficulty faced was to identify repetitive actions because the same action could be performed as part of different activities. For example, the actions 'Water On' and 'Water Off' were involved in activities such as 'Wash hands', 'Cook', 'Eat' and 'Clean'. In the case of the actions 'Water On' and 'Water Off' LFPUBS was able to define that four different 'Water On' and 'Water Off' actions were needed.

Different participants performed the same activities in the same order, but this does not imply that they all performed all of the actions in the same order. For example, when it came to cooking, some of them took out the raisins first and then the oatmeal, and others did the opposite. This is proof that although the order of activities was defined in advance, unordered subsets of actions could exist and have to be identified. Considering the following parameters (*Minimum Level for Origin: 25%; Minimum Level for Destiny: 25%; Minimum Balanced Level: 50%*), only three unordered subsets were discovered. The first one included the actions 'Cabinet On' and 'Cabinet Off', the second one included the actions 'Oatmeal On' and 'Raisins On', whereas the last one included the actions 'Cabinet On', 'Burner Off' and 'Water Off'.

Once repetitive actions and unordered subsets of actions were identified, it was possible to define the topology that modelled participants' behaviour. As in any Action Map, the topology itself defined the qualitative Time Relations. To discover quantitative Time Relations, the 'Basic Algorithm' was

used. Considering all the relations defined by the Topology, the 'Basic Algorithm' was able to identify quantitative Time Relations in 25 out of 29 (86%) cases. The representation of the Action Map once Topology and Time Relations were defined as shown in Figure 24:

```
(Action Map 1)

(Action Pattern 0)
ON occurs (start,--,t0)
IF context ()
THEN do (simple,(On,PhoneBook),t) when –

(Action Pattern 1)
ON occurs (simple,(PhoneBook,On),t0)
IF context ()
THEN do (simple,(On,Phone),t) when t = t0 + 57s

(Action Pattern 2)
ON occurs (simple,(Phone,On),t0)
IF context ()
THEN do (simple,(Off,Phone),t) when t = t0 + 50s

(Action Pattern 3)
ON occurs (simple,(Phone,Off),t0)
IF context ()
THEN do (simple,(On,Water),t) when t is after t0

(Action Pattern 4)
ON occurs (simple,(Water, On),t0)
IF context ()
THEN do (simple,(Off,Water),t) when t = t0 + 23s
(...)
```

Fig. 24. Action Map after Topology and Time Relations were discovered.

Specific and General Conditions were identified. Regarding the Specific Conditions, very few situations demanded Specific Conditions (only three). Besides, the lack of context information meant Specific Conditions could only be identified using calendar information. Using only calendar information, it was possible to identify conditions in two out of three (67%) cases. The identified General Conditions indicated when the participants performed such actions. Thus, it was discovered that all of the actions were carried out on weekdays between 10:45 a.m. and 18:15 p.m.

C. Comparing LFPUBS with other learning systems

As it can be seen in Section II, there have been many different approaches to discover knowledge about users' behaviours. The main difference between LFPUBS and the rest of the systems is that LFPUBS approaches the problem holistically, i.e. it discovers all aspects of user behaviours (frequent actions, order (topology), time relations and conditions), whereas the other systems focus on one single aspect.

The first two steps (frequent actions and topology) can be compared to works such as [6], [9], [13], [15]. The main advantage of LFPUBS is that the output produced as a set of rules are much easier to understand than other forms of representation like Artificial Neural Networks [6], [7], Markov Models and Bayesian Networks. Obtaining an output which can be easily interpreted is important in this domain of application, for example, it can facilitate the detection of unhealthy habits and the deterioration of health or hygiene.

Jakkula and Cook [11] identified qualitative time relations between actions in order to predict actions with higher accuracy. Our LFPUBS's step of 'Identifying Time Relations' provides more accurate relations because it prioritizes quantitative relations over qualitative ones.

Classification techniques have been used by other groups [17] in order to discover contextual conditions. We used a similar approach in LFPUBS. The difference is that LFPUBS discovers conditions when the topology of the frequent behaviour demands it, whereas other approaches discover conditions in all the situations where there are different options, without considering the topology of frequent behaviours.

Table V represents the comparison among LFPUBS and other available approaches, considering their representation and execution characteristics for intelligent environments. By representation it is considered the format and comprehensibility of the patterns, and by execution what such patterns could be used in intelligent environments for.

TABLE V
REPRESENTATION AND EXECUTION CHARACTERISTICS.

	REPRESENTATION	EXECUTION
Frequent Actions Approach [6], [9], [13], [15]	Difficult to understand	Automation of actions
Time Relation Approach [11]	Qualitative relations between actions	Comprehensible relations between actions (no automation)
Contextual Condition Approach [17]	Conditions under a pattern becomes true	Automation of actions under such conditions (without considering the sequence of actions)
Ontologies [35], [36]	Ontology based representation (it needs a learning technique)	Automation of actions. Comprehend frequent behaviours. (it needs a learning technique)
LFPUBS	Frequent behaviours of users with easier to understand and use output.	Automation of actions informed by frequent behaviours.

Time performance is also an important aspect on these systems. Table V shows our system provides, through different means, services equivalent to a group of systems. As such, LFPUBS cannot be fairly compared with them in time performance. The validation section included time performance for the several steps of LFPUBS in different problems. It can be seen some steps are computationally much more demanding than others. LFPUBS was created to be used 'offline'. Initially the sensor system is used for a period of time for data accumulation only. LFPUBS will be then run to obtain an initial set of rules. From time to time (depending on the application), LFPUBS is run again over the newest and expanded data sets. This produces an updated set of rules to drive automation.

The system has produced good results efficiently even with several months of data. This is consistent, or in many cases better, than the state of the art.

VI. Conclusions and further research

Intelligent Environments (IEs) are real environments (Smart Homes, Smart Classrooms etc.) that sensibly support people in their daily lives. This pre-supposes a change of perspective in the relationships between human and technology, shifting from a techno-centered perspective to a human-centered one, where the technology adapts its behaviour to users' needs, preferences and habits. Therefore, an environment should learn how to react to the actions and needs of the user, and this should be achieved in an unobtrusive and transparent way. Past behaviour can be used to predict future behaviour, especially in terms of habits and preferences, so that IEs could provide personalised and adapted services. Thus, the ability to discover users' frequent behaviours becomes an essential aspect for the successful implementation of IEs, allowing them to act proactively.

Taking into account the need for an IE to be capable of learning, the special features of IEs and the current state of the art, a system that discovered users' frequent behaviours was designed and developed. The system, Learning Frequent Patterns of User Behaviour System (LFPUBS), is based on a three-layered architecture whose main objective is to separate those aspects that are dependent on particular environments in which the system is being used from those aspects that are environment-independent. Thus, both the Transformation Layer, which fills the gap from the real environment to LFPUBS, and the Application Layer, which fills the gap from LFPUBS to the real environment, are environment-dependent because their performance depends on specific aspects (e.g., types of sensors/actuators) of particular environments. However, the Learning Layer, which implements all of the algorithms that discover users' frequent behaviours, is free of any influence of particular environments.

The main focus of this paper has been the design and the development of the necessary components of the Learning Layer. The Learning Layer is made up of two modules: the language module ($\mathcal{L}_{LFPUBS}$), which provides a standard conceptualisation of the patterns; and the algorithm module ($\mathcal{A}_{LFPUBS}$), which discovers the patterns. The system was validated using data collected from two real environments: MavPad and WSU Smart Apartment with very good results in terms of the set of patterns discovered and its efficiency. Still, as with any advanced system of this type we know several features can be improved or extended.

In future works, it is also necessary to consider other types of information such as health and emotional status of the user, agenda information etc. Besides, the current Action Map Approach should be adapted in order to consider other types of Time Relations (e.g. ranges), online adaptation of the patterns or multiuser environments. Finally, LFPUBS can be used for new applications, for example analyzing deviations from frequent behaviours in order to detect initial stages of some diseases (e.g. Alzheimer's disease).

References

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

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

[3] D. Leake, A. Maguitman, and T. Reichherzer, *Cases, context, and comfort: opportunities for case-based reasoning in smart homes*, ser. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006, pp. 109–131.

[4] M. E. Muller, "Can user models be learned at all? inherent problems in machine learning for user modelling," in *Knowledge Engineering Review*, vol. 19. Cambridge University Press, 2004, pp. 61–88.

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

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

[7] M. Chan, C. Hariton, P. Ringeard, and E. Campo, "Smart house automation system for the elderly and the disabled," in *Proc. of the 1995 IEEE Int. Conf. on Systems, Man and Cybernetics*, 1995, pp. 1586–1589.

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

[9] D. J. Cook and S. K. Das, "How smart are our environments? an updated look at the state of the art," in *Pervasive and Mobile Computing*, vol. 3. Elsevier Science, 2007, pp. 53–73.

[10] E. O. Heierman and D. J. Cook, "Improving home automation by discovering regularly occurring device usage patterns," in *Third IEEE International Conference on Data Mining*, 2002, pp. 537–540.

[11] 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.

[12] J. Allen, "Towards a general theory of action and time," in *Artificial Intelligence*, vol. 23, 1984, pp. 123–154.

[13] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman, "Creating an ambient-intelligence environment using embedded agents," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 12–20, 2004.

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

[15] M. Beetz, M. Tenorth, D. Jain, and J. Bandouch, "Towards automated models of activity of daily life." *Technology and disability*, vol. 22, pp. 1–11, 2010.

[16] N. Cislo, "Undernutrition prevention for disabled and elderly people in smart home with bayesian networks and rfid sensors," in *Proc. 8th int. conf. on Smart homes and health telematics (ICOST'10)*, 2010.

[17] C. L. Gal, J. Martin, A. Lux, and J. L. Crowley, "Smartoffice: Design of an intelligent environment," *IEEE Intelligent Systems*, vol. 16, no. 4, pp. 60–66, 2001.

[18] N. M. Sadeh, F. L. Gandom, and O. B. Kwon, "Ambient intelligence: The mycampus experience," ISRI, Tech. Rep. CMU-ISRI-05-123, 2005.

[19] A. Aztiria, A. Izaguirre, and J. Augusto, "Learning patterns in ambient intelligence environments: A survey." *Artificial Intelligence Review*, vol. 34, pp. 1–31, 2010.

[20] H. Sharp, Y. Rogers, and J. Preece, *Interaction Design: Beyond Human-Computer Interaction*. John Wiley and Sons Ltd., 2007.

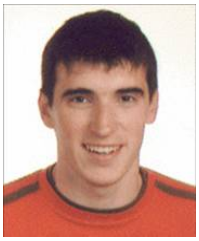
[21] H. Aghajan, J. Augusto, and R. L. C. Delgado, *Human-centric interfaces for ambient intelligence*. Academic Press, 2009.

[22] A. Aztiria, A. Izaguirre, R. Basagoiti, and J. Augusto, *Learning about preferences and common behaviours of the user in an intelligent environment*, ser. Behaviour Monitoring and Interpretation-BMI-Smart environments, Ambient Intelligence and Smart Environments. IOS Press, 2009, pp. 289–315.

[23] 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)*. IO Press, 2004, pp. 778–782.

[24] J. C. Augusto and P. McCullagh, "Ambient intelligence: Concepts and applications," in *Computer Science and Information Systems*, vol. 4. ComSIS Consortium, 2007, pp. 1–28.

[25] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Pro. 11th International Conference on Data Engineering*, 1995, pp. 3–14.
[26] A. Weijters and W. van der Aalst, "Process mining discovering workflow models from event-based data," in *Proc. of the 13th Belgium-Netherlands Conf. on Artificial Intelligence (BNAIC 2001)*, 2001, pp. 283–290.
[27] W. van der Aalst, A. Weitjers, and L. Maruster, "Workflow mining discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 9, pp. 1128–1142, 2004.
[28] L. Wen, W. van der Aalst, J.Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 145–180, 2007.
[29] R. Hogg, J. McKean, and A. Craig, *Introduction to Mathematical Statistics*. Pearson Prentice Hall, 2005, pp. 359–364.
[30] S. Russell and P. Norvig, *Artificial Intelligence: A modern approach, 2nd edition*. Prentice Hall, 2003.
[31] A. Aztiria, J. Augusto, R. Basagoiti, and I. Izaguirre, "Accurate temporal relationships in sequences of user behaviours in intelligent environments," in *Int. Symposium on Ambient Intelligence (ISAMI'10)*, 2010.
[32] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed.* Elsevier, 2005.
[33] G. M. Youngblood, D. J. Cook, and L. B. Holder, "Managing adaptive versatile environments," in *IEEE International Conference on Pervasive Computing and Communications*, 2005.
[34] D. Cook and M. Schmitter-Edgecombe, "Activity profiling using pervasive sensing in smart homes," *IEEE Transactions on Information Technology for Biomedicine*, 2008.
[35] D. Riboni and C. Bettini, "Owl 2 modeling and reasoning with complex human activities," *Pervasive and Mobile Computing*, vol. 7, pp. 379–395, 2011.
[36] R. Tan, J. Gu, Z. Zhong, and P. Chen, "Socom: Multi-sensor oriented context model based on ontologies," in *Proc. 8th int. conf. on Intelligent Environments(IE'12)*, 2012.



Asier Aztiria is a lecturer at the University of Mondragon's Polytechnical Engineering School. His research interests include machine learning techniques and knowledge discovering applied to intelligent environments. He received his PhD degree from University of Mondragon in 2010. He can be contacted at aaztiria@mondragon.edu

Juan Carlos Augusto is a lecturer at Middlesex University. His specific research interests are on the use of temporal reasoning, argumentation and verification to develop more intelligent and reliable Intelligent Environments. Amongst his latest editorial work he co-edited the 'Handbook on Ambient Intelligence and Smart Environments' (Springer Verlag) and 'Human-Centric Interfaces for Ambient Intelligence' (Academic Press). He is also the Editor in Chief of the Book Series Ambient Intelligence and Smart Environments (IOS Press) and the Co-Editor in Chief for the Journal on Ambient Intelligence and Smart Environments (IOS Press). He received his PhD in computer science from Universidad Nacional del Sur (Argentina) in 1998. He is a member of AAAI, ACM, BCS and SMC-IEEE. He can be contacted at j.augusto@mdx.ac.uk.

Rosa Basagoiti is a lecturer at the University of Mondragon's Polytechnical Engineering School. She received a B.S. degree from the University of the Basque Country in 1989 and a Ph. degree from the University of Mondragon in 2007. Her research interest include artificial intelligence, machine learning, vertical transportation and learning contents. She can be reached at rbasagoiti@mondragon.edu.

Alberto Izaguirre is a lecturer at the University of Mondragon's Polytechnical Engineering School. His specific research interests are on Computer Vision, Robotics and Artificial Intelligence. He received his PhD in Robotics and Control Theory from the University Paul Sabatier and the LAAS of CNRS (Toulouse, France). He undertook Post-Graduate Research work at the GRASP Laboratory of the University of Pennsylvania under the direction of Dr. Richard P. Paul, and was assistant professor at New Jersey Institute of Technology until 1996. He can be reached at aizagirre@mondragon.edu.

Diane J. Cook is currently a Huie-Rogers Chair Professor in the School of Electrical Engineering and Computer Science at Washington State University. She received a B.S. degree from Wheaton College in 1985, a M.S. degree from the University of Illinois in 1987, and a Ph.D. degree from the University of Illinois in 1990. Her research interests include artificial intelligence, machine learning, graph-based relational data mining, smart environments, and robotics. She can be reached at cook@eecs.wsu.edu.