Learning frequent behaviours of the users in Intelligent Environments

A thesis submitted
in fulfillment of the requirements for the degree of
Doctor of Philosophy

Asier Aztiria Goenaga
(Mondragon Unibertsitatea)

Supervised by Alberto Izaguirre and Juan Carlos Augusto

September 16, 2010

Mondragon Goi Eskola Politeknikoa,
Department of Computer Science and Electronics,
Mondragon Unibertsitatea

Ama ta Attei.

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 humans 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 user, even anticipating his/her 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 clear the need of knowing preferences and frequent habits of users. Thus, the ability to learn patterns of behaviour becomes an essential aspect for the successful implementation of IEs. In that sense, a perfect learning system would gain knowledge about everything related to users that would help the environment act intelligently and proactively.

The efforts in this research work are focused on discovering frequent behaviours of the users. For that, it has been designed and developed the Learning Frequent Patterns of User Behaviour System (LFPUBS) that, taking into account all the particularities of IEs, learns frequent behaviours of the users.

The core of the LFPUBS is the Learning Layer that unlike some other components is independent of the particular environment in which the system is being applied. On the one hand, it includes a language that allows the representation of discovered behaviours in a clear and unambiguous way. On the other hand, coupled with the language, an algorithm that discovers frequent behaviours has been designed and implemented.

Finally, LFPUBS was validated using data collected from two real environments. Results obtained in such validation tests showed that LFPUBS was able to discover frequent behaviours of the users. Moreover, some improvements were identified for future versions of the system.

Resumen

Los Entornos Inteligentes (EIs) tratan de facilitar las actividades diarias a las personas que se encuentran en él. El concepto de Entornos Inteligentes supone un cambio radical en las relaciones entre los usuarios y la tecnología. El cambio consiste en que se pasa desde una perspectiva centrada en la tecnología a una perspectiva centrada en el usuario. A diferencia de los sistemas actuales donde el usuario se tiene que adaptar a la tecnología, ahora, esta es la que se adapta a las preferencias, costumbres o gustos del usuario.

Los entornos inteligentes deberán aprender cuáles son los comportamientos frecuentes de los usuarios para así adaptarse a los usuarios y proveer servicios personalizados. De este modo, la capacidad de aprendizaje se convierte en un requisito indispensable para dichos entornos.

El objetivo de este trabajo de investigación es desarrollar un sistema que aprenda de forma automática, los comportamientos frecuentes de los usuarios de entornos inteligentes. Para ello, se ha diseñado y desarrollado el Learning Frequent Patterns of User Behaviour System (LFPUBS), que teniendo en cuenta todas las particularidades de dichos entornos, descubre tales comportamientos.

El núcleo del LFPUBS es la capa de Aprendizaje que a diferencia de otros componentes es independiente del entorno donde está siendo aplicado el sistema. Dicha capa, por un lado incluye un lenguaje que permite representar los patrones de una forma clara y no ambigua y por otro, en concordancia con el lenguaje, incluye el algoritmo que descubre dichos patrones.

Finalmente, LFPUBS ha sido validado utilizando los datos recogidos en dos entornos inteligentes reales. Los resultados obtenidos durante esos experimentos permitieron comprobar que LFPUBS es capaz de descubrir los comportamientos frecuentes de los usuarios. Además, mejoras para futuras versiones del sistema fueron identificadas.

Laburpena

Ingurune Adimentsuek bertan bizi diren pertsonei beren egunerokotasunean laguntzea dute helburu. Kontzeptu berri honek pertsona eta teknologiaren arteko erlazioan aldaketa bat dakarki berarekin, teknologiari garrantzia ematetik pertsonengan oinarritzera. Horrela, orain arteko sistemekin alderatuz, non pertsonek teknologia nola erabili ikasi behar duten, orain teknologia (ingurunea) bera da pertsonengana egokitu behar dena. Horretarako, inguruneak bertan dauden pertsonen behar, ohitura, etab. ezagutu behar ditu.

Jakintza guzti hori ordea, pertsonak inondik inora gogaitu gabe lortu behar du inguruneak. Pertsona bakoitzari dagozkion edo nahi dituen zerbitzuak emateko inguruneak pertsona horren ohiko jokabideak jakitea behar du. Orduan, ingurune horrek ikasketa prozesu bat jarraitu behar du, non, pertsona horien ohiko jokabideak era automatiko eta garden batean lortuko dituen.

Ikerketa lan honen helburua horixe izan da, ingurune adimentsu bateko pertsonen ohiko jokabideak era automatiko eta garden batean deskubrituko dituen sistema bat diseñatu eta garatzea. Gainera, garatutako sistema horrek, Learning Frequent Patterns of User Behaviour System (LFPUBS), ingurune adimentsuen berezitasun guztiak hartzen ditu kontutan.

LFPUBS-en barruan berebiziko garrantzia du Ikasketako geruzak, zein ez dagoen sistema aplikatzen ari den ingurunearen menpe. Geruza horretan bi osagaik merezi dute aparteko aipamena, alde batetik, ikasitako jakintza era garbi batean adieraztea ahalbidetzen duen hizkuntzak, eta beste alde batetik, jakintza bera deskubritzen duen algoritmoak.

Azkenik, esan beharra dago LFPUBS balioztatua izan dela benetako bi ingurune adimentsuetan jasotako datuekin. Lortutako emaitzek garbi adierazten dute LFPUBS-ren gaitasuna ohiko jokabideak deskubritzeko garaian. Gainera, emaitzen analisi sakon batek LFPUBS-ri buruzko hobekuntzak argitaratzeko balio izan du.

Agradecimientos

Dokumentu honek azkenengo lau urteetan egindako lana laburbiltzen du, eta holako denbora tarte batek, eta batizpat holako lan batekin zabiltzanean, jende askoren laguntza dakar berarekin. Lagundua izateak eskertzea zor duenez:

Lo primero de todo, querría agradecer a mis dos directores de tésis: Alberto Izaguirre y Juan Carlos Augusto. Gracias Tito por darme esta oportunidad. A tí Juan, no se cómo agradecerte todo lo que has hecho por mí, por tu dedicación y por todo lo que me has enseñado sin pedir ni esperar nada a cambio. Y como no, por cómo me acogiste en mi estancia por tierras irlandesas.

I made an internship during this thesis at the School of Computing and Mathematics of University of Ulster, where I could meet wonderful researchers and make really good friends. Thanks Paul, Gearoid, Simon, Anthony, Bronagh, David, Anyela,...

I would really like to thank Diane J. Cook at Washington State University, for allowing us to use the data they collected in their Intelligent Environments to validate our system.

Unibertsitatean izandako lankideak, lagunak azken finean, ezin ba ahaztu. Behar izan dudanean, azkenengo momenturaino, laguntzeko prest agertu diren guzti horiei, Aitor, Iker eta departamentu guztiei.

Azkenik, zuei ze esan. Atte, ama, arreba, amama, Ainhoa ta kuadrilakuek. Zuentzako ez dago hitzik, zuei naizena zor dizuet, besteik gabe.

Contents

Contents **xiii**

1 Introduction **1**
1.1 Intelligent Environments (IEs) . . . 2
1.2 Motivation . . . 4
1.2.1 Different types of knowledge about the user . . . 5
1.2.2 Advantages/Disadvantages of Learning Frequent Behaviour . . . 5
1.2.3 Intelligent Environments' Special Features . . . 6
1.3 Hypothesis, Objectives and Limitations . . . 10
1.4 Methodology . . . 11
1.5 Thesis outline . . . 13

2 State of the Art **15**
2.1 Artificial Neural Network . . . 16
2.1.1 Applications . . . 16
2.1.2 Strengths and Weaknesses . . . 17
2.2 Classification techniques . . . 17
2.2.1 Applications . . . 17
2.2.2 Strengths and Weaknesses . . . 18
2.3 Fuzzy Logic rules . . . 18
2.3.1 Applications . . . 19
2.3.2 Strengths and Weaknesses . . . 20
2.4 Associated sequence discovery . . . 20
2.4.1 Applications . . . 20
2.4.2 Strengths and Weaknesses . . . 21
2.5 Instance-Based Learning . . . 21
2.5.1 Applications . . . 22
2.5.2 Strengths and Weaknesses . . . 23
2.6 Reinforcement Learning . . . 23
2.6.1 Applications . . . 23
2.6.2 Strengths and Weaknesses . . . 24
2.7 Summary . . . 25

3 General Architecture **27**
3.1 Transformation Layer . . . 28
3.1.1 Inference of simple actions . . . 28
3.1.2 Inference of complex actions . . . 29
3.1.3 Splitting actions into sequences . . . 30
3.2 Learning Layer . . . 31
3.3 Application Layer . . . 31
3.3.1 Applications of extracted knowledge . . . 31
3.3.2 Applications based on specific learning processes . . . 32
3.3.3 Interaction system . . . 33
3.4 Graphical User Interface . . . 35
3.5 Summary . . . 35

4 Learning Frequent Behaviours: the Pairwise Approach **37**
4.1 Introduction . . . 37
4.2 Architecture of the Learning Layer . . . 39
4.3 Representing patterns with $\mathcal{L}_{LFPUBS}$ . . . 39
4.3.1 Event Definition . . . 40
4.3.2 Condition Definition . . . 40
4.3.3 Action Definition . . . 41
4.4 Learning patterns with $\mathcal{A}_{LFPUBS}$ . . . 41
4.4.1 Identifying Frequent Relations . . . 42
4.4.2 Identifying Time Relations . . . 46
4.4.3 Identifying Conditions . . . 51
4.5 Summary . . . 54

5 Learning Frequent Behaviours: the Action Map Approach **55**
5.1 Introduction . . . 55
5.2 Representing patterns with $\mathcal{L}_{LFPUBS}$ . . . 56
5.2.1 Evolution of the $\mathcal{L}_{LFPUBS}$ . . . 56
5.3 Learning patterns with $\mathcal{A}_{LFPUBS}$ . . . 58
5.3.1 Identifying Frequent Sets of Actions . . . 58
5.3.2 Identifying Topology . . . 62
5.3.3 Identifying Time Relations . . . 71
5.3.4 Identifying Conditions . . . 73
5.4 Summary . . . 77

6 Validation **79**
6.1 Validation Environments and Collected Data . . . 79
6.1.1 MavPad Environment . . . 79
6.1.2 WSU Smart Apartment Environment . . . 80

6.2 Pairwise Approach . . . 83
6.2.1 Validating the Pairwise Approach with the MavPad dataset . . . 83
6.2.2 Validating the Pairwise Approach with the WSU Smart Apartment dataset . . . 88
6.3 Action Map Approach . . . 90
6.3.1 Validating the Action Map Approach with the MavPad dataset . . . 90
6.3.2 Validating the Action Map Approach with the WSU Smart Apartment dataset . . . 98
6.4 Comparing both Approaches: The final Discussion . . . 103
6.4.1 Modelling Frequent Behaviours: A comparison . . . 103
6.4.2 Identifying Time Relations: A comparison . . . 104
6.4.3 Identifying Conditions: A comparison . . . 106
6.4.4 Runtime of different steps . . . 106
6.5 Summary . . . 107

7 Conclusions and Further Research **109**
7.1 Conclusions . . . 109
7.2 Contributions . . . 112
7.3 Relevant Publications . . . 112
7.3.1 International Journals . . . 113
7.3.2 Book Chapters . . . 113
7.3.3 International Conferences . . . 113
7.4 Future Work . . . 114
7.4.1 Improving the State of the Art . . . 114
7.4.2 Improving the Architecture . . . 114
7.4.3 Improving the Action Map Approach . . . 114
7.4.4 Improving the Validation . . . 115
7.4.5 More General Improvements . . . 115
7.5 Final Remarks . . . 116

8 Appendix **125**

Bibliography **171**

List of Figures

1.1 Major trends in Computing 2
1.2 Michael's morning ritual represented in a sequence 9
1.3 Schematic view of the research process. 12

3.1 Three-layered global architecture. 27
3.2 Initial screen of the Graphical User Interface. 36

4.1 Selecting the Approach by means of the GUI. 38
4.2 Essential components of the Learning Layer 39
4.3 Set of steps to be performed by the learning algorithm. 42
4.4 Defining the minimum confidence and support levels. 43
4.5 Information about the discovered Frequent Relations. 45
4.6 Selecting the algorithm to use in order to identify quantitative Time Relations. 47
4.7 Defining the allowed deviation for the 'Basic Algorithm'. 48
4.8 Time Distances between occurrences of 'Shower Off' and 'BathroomFan On' . 49
4.9 Information about the discovered Time Relations. 50
4.10 *Covered* and *non-covered* tables with calendar and context information 53
4.11 Information about the discovered Conditions. 53

5.1 Differences between the outputs of both approaches. 56
5.2 Steps to be performed by the learning algorithm. 58
5.3 Defining the basic minimum confidence, extra actions' minimum confidence and minimum similarity levels. 59
5.4 Information about the discovered Frequent Sets. 62
5.5 Defining the parameters to identify repetitive actions. 64
5.6 Defining the parameters to identify unordered subsets of actions. 65
5.7 The basic representation of Michael's behaviour. 66
5.8 Michael's behaviour with repetitive actions. 68
5.9 Michael's behaviour with unordered subsets of actions. 68
5.10 Michael's behaviour without considering the Allowed Maximum Granularity parameter. 69
5.11 Michael's behaviour considering the Allowed Maximum Granularity parameter. 69
5.12 Graphical representation of the topology. 71

5.13 'Shower Off - BathroomFan On' and 'Shower Off - BathroomLights Off' tables with calendar and context information 75
5.14 Graphical representation of the topology. 77

6.1 MavPad sensors on objects, context and motion sensors [You05]. 80
6.2 WSU Smart Apartment motion sensors [Coo08]. 81
6.3 Percentage of the Patterns with Time Relations. 86
6.4 Percentage of the Patterns with Conditions. 87
6.5 Percentage of the Patterns with Time Relations. 97
6.6 Percentage of the Patterns with Specific Conditions. 97
6.7 Both approaches' runtimes for the task of modelling frequent behaviours. . . 104
6.8 Both approaches' runtime considering the task of identifying Time Relations. 105
6.9 The runtime of different steps of the Pairwise Approach. 106
6.10 The runtime of different steps of the Action Map Approach. 107

List of Tables

2.1 Strengths and weaknesses of Learning Techniques 26

6.1 Actions involved in each ADL. 82
6.2 Number of patterns obtained in different trials. 84
6.3 Number of patterns with Time Relations (out of total patterns and the percentage) obtained in different trials. 84
6.4 Number of patterns with Conditions (out of total patterns and the percentage) obtained in different trials. 85
6.5 The number of patterns that remain with different confidence levels. 87
6.6 The number of patterns obtained in different trials and the Experiments' runtimes (in milliseconds). 91
6.7 Experiments' runtimes (in milliseconds) when discovering the Topology of Frequent Sets. 93
6.8 The number of Action Patterns with Time Relations, the percentage of the total and the experiments' runtimes (in milliseconds) obtained in different trials. 94
6.9 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. 95
6.10 The runtime of different steps of the $\mathcal{A}_{LFPUBS}$. 102

Índice de acrónimos

AAL *Ambient Assisting Living*

ADL *Activity of Daily Living*

AI *Artificial Intelligence*

ALZ *ActiveLeZi*

AmI *Ambient Intelligence*

ANN *Artificial Neural Network*

associatedSeT *associated sensor triggering*

CBR *Case-Based Reasoning*

ECA *Event Condition Action*

EM *Expectation-Maximization*

GUI *Graphical User Interface*

HCI *Human-Computer Interaction*

IBL *Instance-Based Learning*

IBM *International Business Machines Corporation*

ID3 *Iterative Dichotomiser 3*

IE *Intelligent Environment*

k-NN *k Nearest Neighbour*

LFPUBS *Learning Frequent Patterns of User Behaviour System*

mainSeT *main sensor triggering*

MavHome *Managing an Adaptive Versatile Home*

ML *Machine Learning*

PC *Personal Computer*

PDA *Personal Digital Assistant*

WSU *Washington State University*

Chapter 1

Introduction

Throughout history, human beings have been characterized by their ability to develop new tools that have allowed their survival in adverse environments. Scientific and technological breakthroughs in the past decades have totally changed the lifestyle of the current generation from previous generations. Many of those developments were aimed at improving the quality of life, either by facilitating daily tasks or by increasing safety.

The current generation will not be an exception. In fact, a large number of those advances are occurring now, in a more or less unperceived ways, and they are influencing our daily life. Slowly and silently, technology is becoming interwoven in our lives in the form of a variety of devices that are beginning to be used by people of all ages [Aug09].

The miniaturization process of electronics has made a wide range of small computing devices available, thereby making it possible to embed sensing and computing capabilities in many different objects like home appliances (e.g., washing machines and microwave ovens), ordinary objects (e.g., mirrors and chairs) or electronic devices (e.g., mobile phones and PDAs). Mark Weiser [Wei91] predicted a trend (See Figure 1.1) in the use of computers, suggesting an evolution to a new paradigm, where computers would disappear by becoming embedded in different types of objects. In the first generation of computers, a machine (mainframes) was shared by many highly trained programmers. Later, it became possible for many people, not necessarily with a high levels of training, to have access to a Personal Computer (PC). Currently, many people have access to several computing devices such as PCs, mobile phones or PDAs. All indications point to the continuation of this trend, making an environment possible wherein the user is surrounded by many devices/objects with sensing and computing power.

This new paradigm is a source of new problems and opportunities that demand new solutions. One of the most attractive solutions is developing Intelligent Environments (IEs), which would be aimed at facilitating users' daily tasks as well as increasing their safety.

Before defining objectives, methodologies or techniques, it is important to clarify the meaning of some terms used within this work that might lead to misunderstandings. The possibility of confusion arises because of the large variation in the uses of terms contained in this work, such as intelligence or learning, which makes it difficult to provide a unique and clear definition. In order to avoid misunderstandings and clarify the meanings of such terms in this work, a definition is given first.

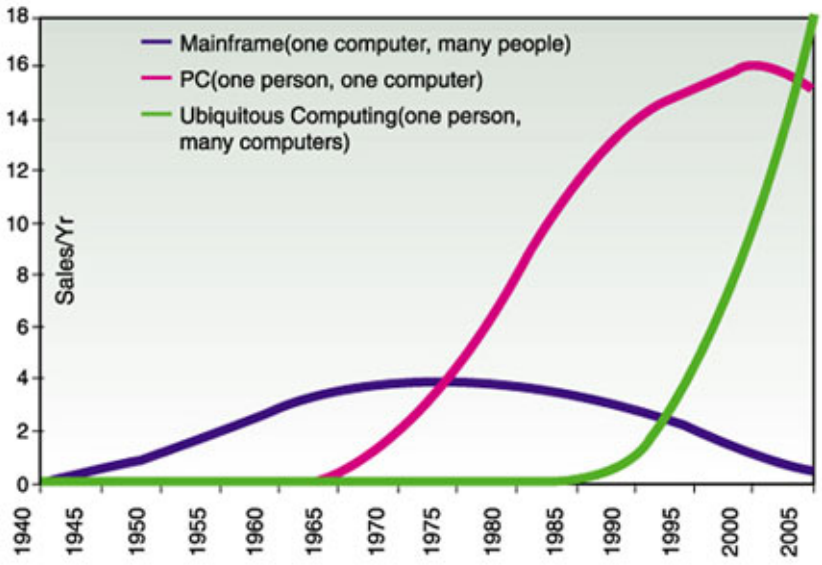

Figure 1.1: Major trends in Computing

1.1 Intelligent Environments (IEs)

Many different terms have been suggested in reference to environments that intelligently support people in their daily lives. Many authors have equated some of these terms without considering subtle details that make a significant difference.

The term *Disappearing computer*, introduced by Weiser [Wei91], was the first notion to refer to the possibility of embedding devices with sensing and computing power in the environment.

> The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

The notion of a disappearing computer is directly linked to the notion of *Ubiquitous Computing* [Wei93], or *Pervasive Computing* [Sah03] as IBM later called the phenomenon. These two terms mainly refer to environments where technology, i.e., embedded computers, is spread widely throughout the environment. As Augusto points out [Aug07b] Ubiquitous[1]/Pervasive[2] systems emphasize the physical presence and availability of resources, but they miss a key element: the explicit requirement of *intelligence*.

Being aware of the need for intelligence to achieve a real environment that sensibly supports people in their daily lives, new terms that aim to extend the idea of Ubiquitous Computing one step further have been suggested. In that sense, *Ambient Intelligence* (AmI)

[1] *Ubiquitous*: adj. present, appearing, or found everywhere (The Oxford Pocket Dictionary of Current English;2006)

[2] *Pervasive*: adj. (esp. of an unwelcome influence or physical effect) spreading widely throughout an area or a group of people (The Oxford Pocket Dictionary of Current English; 2006)

[Duc01; Aug07a] is one of the most well-known terms to suggest a multidisciplinary approach that covers many areas of research in order to achieve intelligent environments. Ambient Intelligence is mainly a term used in Europe; similar developments in the U.S.A and Canada are referred as *Smart Environments* or *Intelligent Environments*.

Many different types of environments can be enriched with Intelligent Environments systems. Sometimes, descriptions of these enriched environments are confused with terms like Intelligent Environments or Ambient Intelligence, but they should properly be described as some possible implementations. The most extensively explored class of examples of this type of environments are the *Smart Homes* [Aug06a]. Other environments that could be enriched by Intelligent Environments systems include *Smart Cars* and *Smart Classrooms*.

One of the most promising areas for potential enrichment by Intelligent Environments systems are health-related environments. These environments fall under the description *Ambient Assisting Living* (AAL) systems [3], and they look to enhance the quality of life of people through the applications of Intelligent Environments systems for healthcare and well being in general. For example it is well known that the majority of elderly people prefer to live in their own houses independently as long as possible [Fri05]. AAL systems aim to achieve this goal in the following ways:

- Extending the time people can live in their preferred environment;
- Maintaining the health and functional capability of the elderly individuals;
- Promoting better and healthier lifestyles for individuals at risk;
- Supporting carers, families and care organizations.

This research work considers the term Intelligent Environments (IEs) better defines the characteristics of this type of environments. Henceforth, this research work will use this term to refer to environments that proactively and sensibly support people in their daily lives.

Scenarios

Let us consider two scenarios that illustrate two different IEs that make the life of the users easier and safer.

Scenario 1: *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 a few minutes after 08:00 a.m.; approximately 10-15 minutes later, he usually steps into the bathroom. At that moment, 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 around 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%). Before he leaves the bathroom he turns off the fan and the lights.*

[3] The ambient assisted living joint programme, http://www.aal-europe.eu

When he goes into the kitchen the radio turns on so that he can listen to the news while he prepares his breakfast. When he is preparing his breakfast the system reminds him that he has medicine to take. He leaves the house 15-20 minutes after having breakfast. At that moment, all the lights are turned off, and safety checks are performed in order to detect potentially hazardous situations in his absence (e.g., checking if the stove is turned on), and if needed, the house acts accordingly (e.g., turning the stove off).

Scenario 2: *Sarah is a 75-year-old woman who is frail and therefore needs help to carry out daily tasks like getting dressed or having a shower. Fortunately, she lives in a modern building where people live in separate apartments and share the communal services of nursing-care, rehabilitation, entertainment etc. The staff members know (through previous reports generated by the environment) that Sarah usually likes having a shower just after getting up; so when Sarah's alarm goes off, nurses are ready to help her. Regarding her rehabilitation, Sarah has the freedom to choose what type of exercises she wants to do. Specialized staff members, after monitoring and detecting Sarah's preferences, design a personalized treatment program highly suited to her needs and preferences. On Tuesday and Thursday nights she likes watching her favourite sitcom, so she goes to bed around 11:00 p.m.; other nights, she goes to bed around 10:00 p.m..*

Staff members are concerned about Sarah's recent behaviour because the system has detected that although she takes pills every day, she takes them immediately before having lunch, which is not desirable for their mode of action. Finally, doctors are concerned because there are some indications that show that she could be in the first stage of Alzheimer's disease. In order to confirm or disprove these suspicions, they decide to check if she undertakes repetitive tasks in short periods of time or shows signs of disorientation (e.g., going back repetitively to places where she has been).

1.2 Motivation

One of the hidden and most important assumptions in IEs is that they propose a transition from techno-centered systems to human-centered systems. IEs suppose a change of roles in the relationships between human and technology. Unlike current computing systems where the user has to learn how to use the technology, an IE adapts its behaviour to the user, even anticipating his/her needs, preferences or habits.

For that shift to take place, an environment should learn how to react to the actions and needs of the user, and this goal should be achieved in an unobtrusive and transparent way. Due to the complexity of IEs (hardware, software and networks must cooperate in an efficient and effective way to provide a suitable result to the user), initial developments have been focused upon the needs associated with hardware and network as supporting infrastructure. This focus has resulted in a simple automation that implements a reactive environment, that does not taken into account the personalized and adaptive features of IEs. There exist sensing systems that are, wrongly considered to be *intelligent* because they act over the user using manually predefined patterns of behaviour.

Such missclassifications can be perfectly exemplified by considering the activation of the fan in the bathroom in Michael's example. When Michael has a shower, the fan can always be activated independently of the relative humidity level, it can be activated when the relative humidity level exceeds a certain level or it can 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 fulfil the requirements of adapting the environment to specific users and providing personalized services; whereas in the second case, such an adaptation is achieved only by disturbing the user or another person. The ideal environment adapts itself to Michael's preferences and habits without disturbing him.

In order to provide personalized and adapted services, the requirement of knowing the preferences and frequent habits of users is clear. Thus, the ability to learn patterns of behaviour becomes an essential aspect for the successful implementation of IEs, because knowing such patterns allows the environment to act intelligently and proactively. 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 [Gal06; Lea06].

1.2.1 Different types of knowledge about the user

As previously stated, a perfect learning system should gain knowledge about everything related to users that would help the environment act intelligently and proactively. Knowledge about users can cover types of information with very different natures. Users' preferences for devices defined by quantitative values (e.g., Michael prefers the temperature of the water to be around 24-26 degrees Celsius in the winter) or needs (e.g., Sarah has to take a pill) are useful pieces of knowledge about the user. However, the current work believes that, at first, knowledge of the users' frequent behaviours better defines users and allows IEs to act intelligently.

A frequent behaviour, initially, can be initially defined as a set of actions and/or activities that a user usually performs under certain conditions. Michael's morning habits are an example of it, where actions like 'Alarm On', 'Bathroom On', 'Shower On', ... 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).

1.2.2 Advantages/Disadvantages of Learning Frequent Behaviour

Assuming that human beings perform behaviours based on habits, it could be inferred that patterns describing past and present behaviours will define future behaviours as well. In that sense, discovered patterns can be used for many different purposes, depending on the objectives of each particular environment.

Advantages

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, turn the radio on and so 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.

In Sarah's case, patterns are not used to automate actions or devices, but they are used to *understand* her behaviour and act in accordance with it. From Sarah's perspective, the staff members are always at the correct place and time. On the other hand, the knowledge of the habits of different patients allows the staff members to organize their time in an efficient way and to provide more personalized services to patients. The understanding of ordinary patterns also allows the detection of unhealthy habits (e.g., taking pills immediately before having lunch).

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.

Disadvantages

As with any newly proposed methodology, the IE vision is not without criticisms. For example, there are concerns regarding a loss of privacy [Aug09] or fear of an increasingly individualized society. These criticisms have previously been applied to Computer Science as a whole. Clearly, the importance and complexity of users in IEs require that human aspects must be taken into account during the learning process.

Behavioural patterns (e.g., when the user usually leaves home and when he/she will return) are usually considered personal and sensitive. These patterns should therefore be stored securely to prevent public access and preclude potentially disastrous consequences (e.g., a criminal accessing the information).

1.2.3 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. In that sense, IEs have some features that serve to differentiate them from other environments. In what follows, the most important features that influence the process of learning are analysed.

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

[Mul04; Doo06]. In other words, the learning process must be accomplished as unobtrusively as possible while remaining transparent to the user. These requirements imply that:

- Data have to be collected by means of sensors installed either on standard devices or in the environment.

- System actions relating to the user must be performed to maximize the user's satisfaction.

- The user's feedback must be collected either through the normal operation of ordinary devices (e.g., the light switch) or through friendly interfaces such as multimodal user interfaces (e.g., voice and image processing technologies) [Coe98; Par06; Tur07].

Collected Data

The importance of sensing increases when considering the learning process. The data collected from the sensors will greatly influence the learning process, and all patterns will depend upon the data captured. This dependency is hindered by technical problems associated with the gathering and interpretation of the data collected by sensors.

First, data will be typically collected in a continuous way from different information sources. Integrating data from different sources usually presents many challenges, because different sources will use different recording styles and different devices will have different possible statuses [Wit05]. Finally, as in other areas of computing, finding out how to appropriately accommodate the common phenomena of 'noisy' data with missing or inaccurate values is another important challenge.

Moreover, it is also necessary to consider the nature of the collected data. The first aspect to be considered is the nature of raw data. Sometimes, raw data are not meaningful and a combination of different sources of raw data is necessary to recognize meaningful activities (for further details, see Section 3.1).

Another aspect to consider is that different types of sensors provide information of different natures that can be used for different purposes in the learning process. Some sensors provide direct information about the actions of the user (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 to be considered are those that provide information about the health and emotional status of the user (e.g., sensors that capture parameters like heart rate). Finally, externally gathered information can be included to enrich data collected from sensors. Externally gathered information will typically be domain-related knowledge, such as the medical background of patients, preferences of the user specified in advance by the user or calendar information (e.g., when the user goes on holiday). It is worth noting that in this work sensors that provide information about the health and emotional status and externally gathered information are not considered.

Let us consider a brief example of data collected in Michael's environment is:

Devices' activations	*Other sensors*
(date;device;status;value)	*(date;device;status;value)*
2008-10-20	*2008-10-20*
08:02:12;Alarm;on;100	*08:02:14;TempBathroom;on;18*
08:15:55;Bathroom;on;100	*08:05:19;HumBathroom;on;65*
08:15:57;BathroomLights;on;100	*08:13:42;TempBathroom;on;19*
08:17:10;Cabinet;on;100	*08:18:40;TempBathroom;on;20*
08:17:15;Mouthwash;on;100	*08:22:07;HumBathroom;on;66*
08:17:16;Cabinet;off;0	*08:27:19;TempBathroom;on;18*
08:19:23;Cabinet;on;100	*08:28:20;HumBathroom;on;70*
08:19:29;Towel;on;100	*08:29:27;TempBathroom;on;19*
.	.
.	.
.	.
2008-10-21	*2008-10-21*
08:10:50;Alarm;on;100	*08:11:41;TempBedroom;on;22*
08:23:18;Bathroom;on;100	*08:12:50;HumBathroom;on;50*
08:23:20;BathroomLights;on;100	*08:21:25;TempBathroom;on;19*
08:23:58;Cabinet;on;100	*08:22:49;TempBathroom;on;20*
08:24:02;Mouthwash;on;100	*08:24:21;HumBathroom;on;53*
08:24:03;Cabinet;off;0	*08:25:18;TempBathroom;on;22*
.	.
.	.
.	.

Spatio-temporal aspects of the collected data also play an important role in the learning process. Every act of a person situated in an environment has both spatial and temporal elements. The spatial information of an act is given by the location of either devices (e.g., the light switch) or users (e.g., motion in the bathroom), whereas temporal information is given by the timestamp of each action. Various authors have highlighted the importance of considering both spatial and temporal aspects of IEs [Aug04; Aug06b; Got06; Azt08].

Representation of the Discovered Knowledge

Depending on the objectives of each environment, different representations can be demanded. 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, most of the time the representation of the user's patterns is relevant. In these cases, a human-understandable,

i.e., comprehensible, representation of the patterns is an essential feature for the success of the system.

Moreover, it may be necessary for the system to explain the decisions it makes to the user. For instance, the system could explain to Michael that it has turned on the light in the bathroom because it has strong evidence of his habit to do so. At other times, the output of the learning process must be integrated into a bigger system or must be combined with other types of knowledge in order to make good high level decisions.

Representing frequent behaviours by means of sequences of actions (*Action Maps*) seems to be a promising approach. Figure 1.2 shows part of Michael's morning habits represented by means of an Action Map. 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'). Finally, conditions are necessary to further describe the occurrence of events in this sequences. 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%).

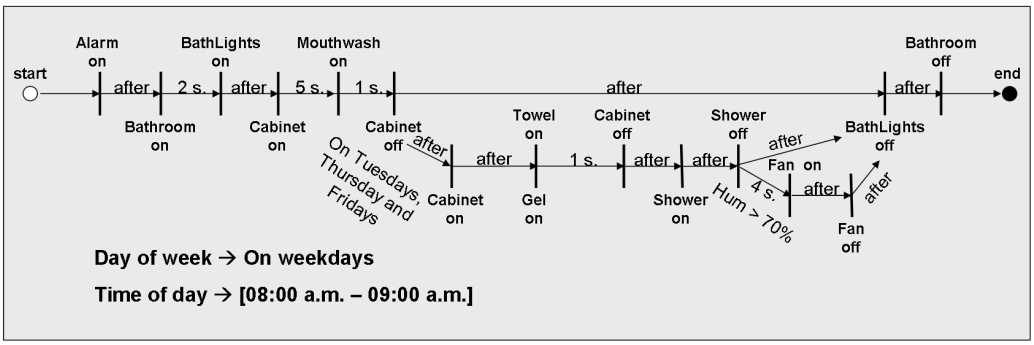

Figure 1.2: Michael's morning ritual represented in a sequence

Scheduling the Learning Process

As mentioned in Section 1.2, this work is focused on discovering frequent patterns of behaviour from data collected by sensors. Even so, the development of a complete IE demands other considerations.

On the one hand, 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.

On the other hand, once patterns have been discovered, it seems clear that those patterns must be continuously revised and updated because:

- The user can change his/her preferences or habits (e.g., Sarah now prefers other types of exercises for her rehabilitation),
- New patterns could appear (e.g., Sarah has started receiving visitors on the weekends),
- Previously learned patterns were incorrect (e.g., the system wrongly concluded that Sarah likes going to bed at 9:00 p.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.

It can be concluded that an ideal IE needs to consider, at least, three learning periods. During the first period, the IE looks to act as intelligently as possible without patterns while starting to gather data. During the second period, the IE must learn the users' frequent behaviours. Finally, while the system is acting in accordance with patterns previously learned, it must update those patterns continuously. As stated previously, this work focuses on the second period, when the IE learns the frequent behaviours of the users.

1.3 Hypothesis, Objectives and Limitations

Based on the changing role of IEs the central hypothesis of this work is:

> Users' frequent behaviours can be learned in an unobtrusive and transparent way in order to provide environments with the intelligence and the ability to adapt to users' habits.

In order to validate the hypothesis, the general goal of this research work is:

> To design and implement a system that learns users' frequent behaviours in an unobtrusive and transparent way.

This objective can be divided into several more focused sub-objectives:

- To design a general architecture for learning in IEs, making clear the dependency of each module on specific environments or applications.
- To define a language that represents users' frequent behaviours.
- To design and implement an algorithm, that learns frequent behaviours. It is necessary to identify the necessary steps and design an architecture for such an algorithm.
- To design and implement an interaction framework that allows the system to interact with end users in applications that involve learnt patterns.
- To validate the system using data collected from real environments.

Besides achieving the objectives mentioned above, the system should satisfy the following requirements:

- Environment-independent: The components that discover and represent the user-knowledge, i.e., the language and the algorithm, should be independent of any particular environment. Thus, the system should ensure that these elements are not designed and developed based on particular needs of certain environments. In order to adapt it to specific environments the system should provide additional means to transform both the collected data and the discovered knowledge.
- Efficiency: The system must verify its ability to satisfy the following requirements when discovering frequent behaviours:
  - Frequency: The definition of a frequent behaviour may vary. Thus, the system must guarantee that it discovers all those behaviours that occur more frequently than the demanded minimum frequency threshold.
  - Response Time: Depending on particular environments, how long the system takes to discover frequent behaviour may or may not be critical. Therefore, it is impossible to define a specific response time for the system. Even so, the system must provide mechanisms that allow a user of the system to prioritize response time over quality of knowledge.

The system to be developed does not learn all types of knowledge about users. Limitations of the designed system regarding the discovery of knowledge include the following:

- The system does not learn preferences of the users for a specific device. Thus, it would not learn that Michael likes to set the temperature of the water around 24-26º degrees Celsius in the winter and around 21-23º degrees Celsius in the summer.
- The system does not learn relationships between environmental conditions. In other words, it does not discover relationships between context sensors (e.g., temperature or humidity) that indicate the status of the environment. There could be interesting relationships (e.g., if the external temperature is >30ºC and the window is open, then the living room relative humidity level is >70%) that relate environmental conditions, but they are not considered in this initial work, because they do not provide any information about users.
- Regarding different periods in the system's lifecycle, it does not consider the processes of providing intelligence in the absence of learned patterns or the process of updating those patterns.

1.4 Methodology

Similar research goals can be sought in completely different ways depending on aspect of the context of research including the availability of infrastructures, the accessibility and proximity

of experts, synergies with ongoing research projects and so forth. Because of our research context, we designed a research strategy based on the following activities:

1. Update our knowledge by reviewing recent and state-of-the-art publications, and attending congresses.

2. Design and develop the different parts of the model and architecture enlarging the scope gradually in an iterative process.

3. Experiment on and evaluate the system.

4. Attend congresses and workshops to present partial results and to learn of existing state-of-the-art advancements.

5. Network with experts in congresses, in meetings, via email, and by visiting other research centres[4].

6. Redesign the system with the feedback obtained from all the above means.

7. Develop and deploy the final system for learning frequent behaviours in real world-like scenarios to gather results.

8. Disseminate the obtained knowledge and experiences to the research community.

Figure 1.3 illustrates graphically this research procedure, including the major activities, as well as the inputs and outputs that contributed to the final results.

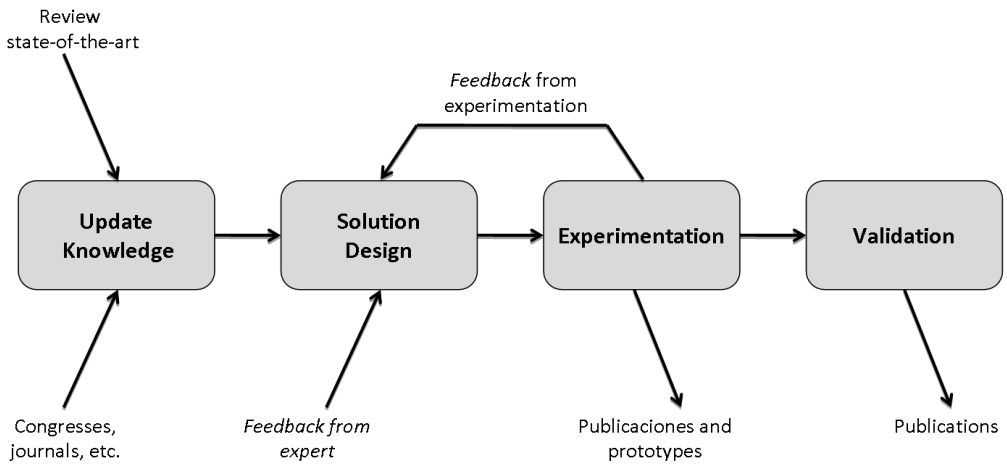

Figure 1.3: Schematic view of the research process.

[4]The author was a visiting Ph.D. student during one year at the School of Computing and Mathematics of University of Ulster

Underlying this research process is the action-research methodology composed of five different phases:

- Diagnosing: identifying the problem.
- Action planning: considering possible courses of action.
- Taking action: selecting a course of action.
- Evaluating: analysing the consequences of the course of action.
- Specifying results: identifying general findings.

These phases will be applied for all of the outlined research activities with the aim of providing rigor, reflexive critiques and continuous challenges.

1.5 Thesis outline

The thesis is divided into seven chapters and a number of appendices.

Chapter 1 (the current chapter) outlines the motivation, hypothesis and goals of the research, as well as the methodology.

Chapter 2 analyses different Machine Learning techniques and the work carried out by different research groups using such techniques. The strengths and weaknesses of each technique are discussed. A comparative table is provided at the end of the chapter.

The architecture designed for the learning system is introduced in Chapter 3, clearly separating those components that are dependent on particular environments from those that are environment-independent.

Chapter 4 provides the first approach developed to learn frequent behaviours of users. This first approach is focused on discovering frequent pairwise relations that describe his/her behaviour. For this approach, an environment-independent language and algorithm have been developed.

A further development of the first approach is explained in Chapter 5. This second approach discovers frequent behaviours without any limitation on the number of actions involved. This developments demands improvements to both the language and the algorithm.

Chapter 6 provides a description of different tests carried out to validate the system and the results. The validation is performed using different datasets collected from two real environments.

Finally, Chapter 7 draws the conclusions of this research work. The hypothesis and goals are revisited, and the major contributions are outlined and several issues are discussed. Future avenues of research and challenges are also sketched, and the chapter ends with some final remarks.

The thesis also includes some appendices that provide more information about technical aspects of the work such as the type of data used by the system, the specifications of the developed language or patterns discovered in different validations tests.

Chapter 2

State of the Art

IEs, as a technological paradigm, have the potential to make a significant impact upon daily human life by positively altering the relationship between humans and technology. The area of IEs has attracted a significant number of researchers, and some applications are already being deployed with different degrees of success. Taking into account the complexity of IEs systems (hardware, software and networks must cooperate in an efficient and effective way to provide a suitable result to the user), each project has focused upon different aspects of such complex architectures. In that sense, it is understandable - and even logical - that the first developments have been focused upon needs that require hardware and networking as supporting infrastructure. This focus has resulted in simple automations that implement reactive environments. However, it is necessary to give more importance to the intelligence component because of its relevance for achieving the core aspects of IE systems.

Although many researchers to date have noted the importance of the learning [Fri05; Aug06b; Das06; Ram08] little emphasis has been placed in general upon the subject. In that sense, the objective of the current work, i.e., learning users' frequent behaviours, is essential for providing intelligence to the environments. Although little emphasis has been placed in this particular subject, notable exceptions can be found.

Artificial Intelligence (AI) [Rus03] has been identified as the area that will provide IEs with intelligence [Aug07b]. AI, understood in its broad sense, encompasses areas like agent-based software, robotics and machine learning. Understood broadly and considering the objective of the current research, Machine Learning (ML) techniques have shown their ability to extract knowledge in many different areas.

On one hand, hundreds of papers and even books [Mit97; Wit05] relating to ML topics have been published. On the other hand, a survey relating learning skills to IEs is not available, even though some surveys on different IE topics have been published [Jia04; Pan06; Coo07]. Thus, this chapter reviews the research work carried out in IEs using ML techniques in order to gain knowledge about users' behaviours.

A brief analysis of different applications developed by different groups shows that current applications are very specific with focused goals. In addition to analysing the knowledge learned in each application, strong and weak aspects of each ML technique used in the applications is analysed. Such an analysis also considers the special features of IEs (see Section 1.2.3).

ML techniques have also been used for other necessary tasks in IEs, such as activity recognition or anomaly detection. However, the current work focuses on learning users' frequent behaviours. Therefore, the strengths and weaknesses of different ML techniques will

only be analysed in relation to that goal.

2.1 Artificial Neural Network

Artificial Neural Networks (ANNs) were inspired by the observation that biological learning systems are built of very complex webs of interconnected neurons [Mit97]. As a rough analogy, ANNs might be described as being built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs and produces a single real-valued output.

2.1.1 Applications

The aim of the system developed by Mozer et al.[Moz95] and installed in the Adaptive House was to design an adaptive control system that considers the lifestyle and energy consumption of the inhabitants. Such an environment was provided with different types of sensors (temperature, light status, illumination and so on) that reported the state of the environment. Moreover, the system had the ability to control the status of the lights, the water heater and the gas furnace. Based on this environment and using a feed-forward neural network, they developed two applications. The first application, an 'occupancy predictor', predicted the expected amount of time spent in the home by the inhabitants in the next 30, 60 or 90 minutes. The second one, a 'zone anticipator', predicted whether a particular was going to be occupied in the coming two seconds, so that the lights were turned on prior to a zone being entered.

Chan et al.[Cha95] developed an application in order to assess whether a situation was normal or abnormal. For this application, they assumed an elderly person had fairly repetitive and identifiable habits. Training ANNs with these regular habits, they were able to detect discrepancies to his/her usual behaviour. After validating this application in an institution for elderly and disabled people, they claimed that the system had 90% chance of providing correct predictions.

These works were amongst the first reports on applications for IEs in which user patterns were considered. Other authors have continued using ANNs to provide personalized services.

Campo et al. [Cam06] developed a system that calculated the probability of each area of the home being occupied at a given moment based on continuous observation of the users' habits.

Boisvert and Gonzalez Rubio [Boi99] also used ANNs to develop an intelligent thermostat. Learning about the behaviour of the occupants, the objective of this application was to reduce the number of interactions with the user and eliminate the need for users to learn how to program the device. Additionally, the thermostat reduced energy consumption by turning off whenever occupants were absent. Thus, people who have fairly foreseeable behavioural patterns significantly reduced (9-16%) their energy consumption by using a prototype of this thermostat.

Finally, see [Beg06] for a survey focused on ANNs for Smart Homes.

2.1.2 Strengths and Weaknesses

Most of the authors that have used ANNs for the learning process highlight their ability to generalize as well as their robustness when faced with complex data (e.g., noisy or missing values). In order to clarify the strengths and weaknesses of ANNs, Michael's scenarios will be used as an example.

Due to the capacity of ANNs to manage complex data and create complex models, a system based on ANNs will provide correct responses in situations such as turning on the lights when Michael goes into the bathroom or getting the shower ready on Tuesdays, Thursdays and Fridays. There are already systems (see applications mentioned above) that use ANNs to predict the presence of the user or the occurrence of an action. In that sense, ANNs are one of the techniques that better accommodate the complexity (type of data, data inconsistency etc.) of IEs.

However, ANNs have an important limitation related to their black box nature; their internal structure is not human-readable. Thus, the system would be able to turn on the light, but it would not be able to explain, in a comprehensible way, how it inferred such an output. If understanding users' frequent behaviours is considered as essential (e.g., Sarah's scenario) ANNs faces an insuperable difficulty. Even if the understanding is not the main objective, the central role that the user plays in IEs, makes the development of a complete learning system based only on ANNs quite difficult.

2.2 Classification techniques

Classification techniques are practical methods for inductive inference. The idea behind classification techniques is to infer a solution or consequent based on a set of conditions or antecedents. Decision trees and classifications rules are the most well-known classification techniques. Decision trees represent the inferred knowledge by means of a tree where each node defines a condition and each leaf a solution, whereas classification rules represent the knowledge by means of 'IF ... THEN ...' rules.

2.2.1 Applications

The group that works on the environment named 'SmartOffice' [Gal01] was the first to identify the use of rules in order to act proactively. SmartOffice was comprised of 50 sensors (cameras and microphones) and 3 actuators (a video projector and two speakers). Given these sensors and actuators, the researchers used a set of predefined rules to integrate different components into a coherent application. One of the main reasons rules were used in this application was because they allowed the addition, deletion or modification of rules without influencing other rules. Thus, they guaranteed scalability of the system.

The SmartOffice group continued to use classification techniques in IEs. In order to justify the use of classification techniques, they pointed out that 'a user is only willing to accept an intelligent environment offering services implicitly if he understands and foresees its decisions' [Brd05]. Taking as a starting point a pre-defined context model, they identified

situations where examples indicated different reactions for such situations. Thus, it was necessary to define under what conditions a reaction would or would not take place. With the knowledge that decision trees were able to perform classifications, they experimented with FIND-S, Candidate Elimination and ID3 methods, finding the last to be the best.

Stankovski and Trnkoczy [Sta06] also analysed the possibility of using Decision Trees in Smart Homes. The application they proposed was the detection of abnormal situations by means of Decision Trees. Based on the assumption that events that usually happened in a Smart Home may be considered normal events, they induced a decision tree. Then, each new situation was analysed and the decision tree determined whether it was abnormal or not.

2.2.2 Strengths and Weaknesses

Firstly, it is worth mentioning that both classification techniques considered in this section have the same characteristics with regards to IEs. Thus, their strengths and weaknesses will be considered together.

One of the main advantages of these classification techniques for IEs is the way they represent knowledge. Due to their human-readable representation, extracted knowledge can be used by a third party to understand a user's behaviour, as well as to explain to the user the decisions made by the system.

As mentioned in one of the applications, classification techniques can be very useful for discovering conditions where certain actions follow other specific actions. For example, in Michael's case, the environment would realize that following the action of 'Shower off', sometimes he turns on the fan in the bathroom and sometimes he does not. Using classification techniques, the environment would be able to discover that the action 'BathroomFan On' follows the action 'Shower Off' if the relative humidity level in the bathroom is >70%.

Although classification techniques have been proposed for discovering abnormal or hazardous situations (in Michael's case the environment would be able to detect when Michael leaves the stove on when leaving the house), this system would generate many *false positives*. These false positives occur because all new situations would always be identified as abnormal.

The advantages of representing a user's behaviour by means of rules are clear. Even so, a single rule does not give any sense of sequence to the actions, so something else is required to discover and represent a user's behaviours by means of sequences.

2.3 Fuzzy Logic rules

Fuzzy sets, a term introduced by Lofti A. Zadeh [Zad65], are those sets whose elements have degrees of membership. Unlike binary logic, where an element either belongs to a set or does not, fuzzy logic permits a graded assessment of the membership of elements in a set. This type of variability allows the use of linguistic variables (e.g., temperature, height or speed) that facilitate the definition of rules and facts.

Systems with fuzzy variables represent the learned knowledge by means of 'IF ... THEN ...' rules. The main difference when compared to classification techniques is that the rules defined

using fuzzy variables can better match IEs' inherent characteristics.

2.3.1 Applications

Researchers at Essex's iDorm lab focused on the problem of learning and were one of the most active groups in this area [Hag04] [Doc05]. Their objective was to develop learning and adaptation techniques for embedded agents. To that end, they developed a test bed, iDorm (later on iDorm2, iSpace and iSpace2), where seven input sensors were monitored (e.g., internal/external light level or bed pressure) and ten output actuators were controlled (e.g., desk and bed side lamps or window blinds).

Their initial efforts were focused on developing an unsupervised approach for extracting fuzzy rules and membership functions from data to develop a fuzzy controller that would model the user's behaviours. The data were collected by monitoring the user in the environment over a period of time. The learned controller provided an inference mechanism that produced output control responses based on the current state of the inputs. They defined a five phases approach to create a fuzzy controller.

- Monitoring the user and capturing input/output data.
- Extraction of the fuzzy membership functions from the data. To achieve this extraction, they used a double-clustering approach [Cas02], combining fuzzy-C-means and hierarchical clustering.
- Extraction of fuzzy rules from the recorded data. The extraction approach used was based on an enhanced version of the Mendel Wang method [Wan92] developed by L.X. Wang [Wan03].
- Control of the environment by the agent controller environment on behalf of the human according to his/her desires.
- Adaptation mechanism. Whenever the user was dissatisfied with the agent's actions, he/she could always override the agent's control responses by simply altering the manual control of the system. When this occurred, the agent adapted its rules online or added new rules based on the new user preferences.

They validated their fuzzy approach by performing experiments in which a user lived in the iDorm for a period of five consecutive days. They collected 408 instances of data from the user's interactions in the iDorm over the initial three-day period, and the agent initially learned 186 rules from that data. Over the next two days, the agent added 120 new rules during the adaptation process, the researchers compared the offline performance of their approach to three other soft-computing-based techniques: genetic programming, the adaptive-neuro fuzzy inference system [Jan93], and the multilayer perceptron neural network.

Analysing the results, the researchers realised their approach generated too many rules. This weaknesses was due to the fact that the agent related actions to the global situation of that moment, and the number of possible global situations was excessively large. Thus, they

made an improvement [Dum08] that identified relevant and important associations between actions, so that irrelevant aspects of the rules (and, by extension, some rules as well) could be removed. In an experiment carried out in the same environment, this change allowed for a 91% reduction in the number of rules.

Vainio et al. [Vai08] also used fuzzy rules to represent habits of a user. In contrast to the approach followed in the iDorm project, these authors manually constructed the membership functions and used reinforcement learning to replace old rules in order to prevent single overriding events from having too large an impact.

2.3.2 Strengths and Weaknesses

The nature of rules generated in this way will be similar to those rules obtained using the classification techniques described in the previous section. They are considered more robust when dealing with data of a continuous nature (e.g., temperature, humidity and time). In Michael's case, for those actions performed when the global situation was similar (e.g., by taking a shower on Tuesdays, Thursdays and Fridays), the controller would provide a correct output.

Due to the multiplicity of sensors and the number of different situations that can be generated when combining sensors, it seems clear that relating actions only to global conditions (without relating actions to other actions) will result in an excessive number of generated rules with very little meaning. In Michael's case, it is clear that the action of turning on the lights in the bathroom is typically associated with the action of going into the bathroom. Thus, it is essential to discover frequent relations between actions. Even so, the system does not relate actions in sequences of actions, but instead only discovers pairwise relations.

2.4 Associated sequence discovery

A sequence defines a set of actions/activities that are inter-related. Most of the time, that relation is interesting because it is frequent. Thus, association techniques, e.g., algorithms like Apriori [Agr95], have been developed to discover frequent relations between any attributes.

2.4.1 Applications

The group working on the MavHome and Casas projects was one of the most active groups in this field of research. The first applications developed by this group were focused on building universal models, represented by Markov models, to predict future locations or activities [Rao04] [Coo07]. The researchers made notable improvements by developing applications to discover daily and weekly patterns [Hei02]. Additionally, they constructed an application with the ability to infer abstract tasks automatically and identify corresponding activities that were likely to be part of the same task [Rao04].

However, the major contributions of this research group have been their research on discovering frequent relations between events [Jak07a] [Jak07b]. After collecting data, they first

identified temporal relations that occurred among events, and they then applied association rule mining techniques to focus on the event sequences and temporal relations that frequently occurred. They used the temporal relations between events as a basis for reasoning to perform anomaly detection and prediction of events. In order to define temporal relations, they used Allen's temporal logic [All84], which produced fairly intuitive sequences of actions.

Once their new approach was developed, they tested it using a dataset collected from the MavLab smart workplace [You05], which contained two months of data. Additionally, they generated a synthetic data set containing about 4000 events representing two months of activities. Then, using their previous approach, ActiveLeZi (ALZ)) [Gop04], they compared the prediction accuracy with and without temporal rules. There was 1.86% prediction performance improvement in the real data and 7.81% improvement in the synthetic data using the temporal rules.

2.4.2 Strengths and Weaknesses

The knowledge discovered by associating actions/activities can easily be represented in a comprehensible way. Moreover, relating such events temporally provides a sequential representation that is one of the most promising representations in IEs (see Section 1.2.3). In Michael's case, the system would be able to detect that he first gets up, then goes into the bathroom and then turns on the light. As stated previously, this representation produces intuitive sequences of actions, allowing the system to detect anomalies as well as to predict future events.

Although this is one of the most promising approaches, a few aspects that need improvement can be noted. First, this system does not determine that a group of activities is part of the same sequence but rather detects relations separately (relating actions in a pairwise manner). Second, this system only considers Allen's temporal logic relations (which define relations qualitatively), thereby ruling out quantitative relations. Thus, the term 'after' means that Michael goes into the bathroom and then he turns on the lights; however, the likely delay between one action and the next cannot be measured. Defining relations by means of quantitative values allows the system to automate actions, which is impossible with purely qualitative values (e.g., the system knows that turning the lights on comes after a given event, but it does not know if is the time delay is 2 seconds, 5 minutes or 2 hours after the first event). Finally, it is worth mentioning that this method does not discover conditions; such a concept is, in fact, very useful if two activities are related in different ways. In Michael's case, the relation 'Shower Off, after, BathroomFan On' occurs only if the bathroom's relative humidity level is >70%, so that conditions should be considered in order to provide more accurate patterns of his behaviour.

2.5 Instance-Based Learning

In contrast to learning methods that construct a general, explicit description of learned knowledge, instance-based learning (IBL) methods simply store the data. Generalizing be-

yond these data is postponed until a solution to a new situation is required. Each time a new situation is encountered, its relationship to the previously stored examples is examined in order to decide the best solution for the new situation. IBL methods are sometimes referred to as 'lazy' learning methods.

IBL can be considered as a family of learning algorithms that includes methods like k-Nearest Neighbour (k-NN) learning or Case-Based Reasoning (CBR).

2.5.1 Applications

The MyCampus group at Carnegie Mellon University [Sad05] developed some interesting applications for IEs using CBR. Their main objective was to provide a set of services to enhance everyday campus life. Thus, applications for recommending services (e.g., where to eat or public transportation) or for reminding users about tasks were developed. One of the most interesting services was a message filtering service, which allowed a user to specify preferences as to when he/she wanted to see different types of messages based on the nature of the message (i.e., subject and sender). In addition, users could provide feedback to help the system refine the preferences they originally entered.

In the first iteration, users had to specify their message filtering preferences ('a priori preference') for different categories of messages. Based on this static information the system filtered the messages and then participants were asked to review each individual message they had received and indicate what the ideal filtering action for that message should have been ('a posteriori preference'). Analysis of the results collected during the experiment showed that users were only satisfied with the 50% of the messages they received.

Seeing the poor results obtained by using 'a priori' preferences, the group implemented a CBR module, which attempted to learn preferences for individual users based on their feedback. The CBR module considered each message as a new case, and the user expressed his/her a posteriori preference for processing that message. Whenever a new case came arose, they used Aha's Nearest Neighbourhood algorithm [Aha91], as adapted by Cercone and Chan [Cer99]. Experiments carried out with the CBR showed an accuracy of over 80%, a significant improvement in the quality of the filtering decisions.

Apart from the MyCampus project, some other researchers have also used CBR to acquire knowledge about users. Kushwaha et al. [Kus04] proposed an intelligent agent for ubiquitous computing environments (UT-AGENT), which had the objective of determining users' information requirements and helping them by providing a task of interest. They stored the user's behaviour as cases, and new queries were classified according to its similarity with previous recorded queries.

Recently San Martin et al. [San09] used the K-Nearest Neighbours (k-NN) algorithm in order to act intelligently in an IE. They monitored users' actions, saving the context (activity, location, sensor values etc.) each time the user set the value for a parameter. Thus, the first time that the user configured a parameter, it was taken as a new type of preference, i.e., a new case, so that these new parameters comprised the training set. They used the k-NN algorithm because, unlike other algorithms that need a large set of training data to obtain

sufficiently precise results, k-NN is able to offer accurate results if new cases are similar to the training examples. San Martin et al. evaluated their approach using synthetic data, and they considered the response time of the system as an important measurement to decide if the approach was feasible for IEs. Their evaluations showed that the response time of their initial approach was unacceptable in some situations.

Finally, see [Lea06] and [Gal06] for a survey and new possible opportunities regarding using CBR techniques for Smart Homes.

2.5.2 Strengths and Weaknesses

Considering the use of IBL techniques in Michael's scenario, their strengths and weaknesses will be clarified. Given a situation similar to one stored previously, the system would act properly because IBL techniques provide similar solutions to similar problems/situations without any initial model. Thus, when Michael goes into the bathroom, the system would compare that situation to previous ones and correctly turn on the lights. The same response would occur for other situations similar to previous occurrences (e.g., by having a shower on Tuesdays, Thursdays and Fridays).

However, the use of IBL techniques has some limitations. As this process infers a solution for each specific situation, it does not create a model that represents patterns. Therefore, it would not be possible to extract a general pattern indicating the behaviour of Michael to turn on the lights after going into the bathroom. Further, as each situation can be represented by means of a large number of parameters, the matching process could be very difficult because there are no clues regarding the importance of each parameter in each situation. Considering Michael's habit of having a shower, if we consider the parameter 'day of the week', it seems clear when he takes a shower and when he does not. However, other parameters (e.g., light level or temperature) that would shape the pattern differently could also be considered, making the process of matching difficult.

2.6 Reinforcement Learning

Each time the environment performs an action, reinforcement learning algorithms consider a reward or penalty provided by users or supervisors, which indicates the desirability of the action. The task of the environment is to modify its knowledge to produce the greatest cumulative reward.

2.6.1 Applications

Some of the groups mentioned in the previous sections, such as the Adaptive Home (See Section 2.1.1) and SmartOffice (See Section 2.2.1) groups, have added a module to provide the environment with the ability to adapt. All have employed reinforcement learning.

As seen in Section 2.1.1 Mozer et al. developed a system that predicted whether a zone in the house would be occupied. In addition to this system, these researchers developed

other methodologies, using the Q learning algorithm [Wat92] for lighting regulation. The system controlled the status of the lights (on/off) and their intensity. Starting with the assumption that the inhabitant had no preferences for the device setting, the system tried to minimize energy consumption as long as the inhabitant did not express discomfort. Once the system received feedback from the user, it tried to balance user's preferences with energy consumption.

The SmartOffice group has also used reinforcement learning in their research work [Zai08]. Their main objective was to construct automatically a context model by applying reinforcement techniques, where the user gave rewards by expressing his/her satisfaction with the system's actions. They employed this approach because systems developed so far have problems when considering the users and their needs. The problems they mention are listed below:

- The behaviour of the system was completely incoherent at the beginning and needed time to converge.
- The user did not want to wait the time required to train the system.
- The user's habits may change over time and the environment should integrate these changes quickly.

Thus, starting with a pre-defined set of actions, they adapted that knowledge progressively to its particular user using the Q learning algorithm. In doing so, the default behaviour made the system ready-to-use whereas learning was a life-long process. They validated it in a simulated environment, converging after 195 actions.

2.6.2 Strengths and Weaknesses

In Michael's example, if we consider that the system already has a model (either defined manually or learned by means of previously mentioned techniques), reinforcement learning techniques can be used in order to adapt such patterns. Let us hypothesize that learned patterns define that the shower must be ready every weekday. Every time Michael does not have a shower would be a penalty for the system, i.e., it would be considered as negative feedback. After collecting feedback, reinforcement learning would change the pattern and adapt it to Michael's new preferences, i.e., to have the shower ready only on Tuesdays, Thursdays and Fridays.

Still, the use of this technique demands a set of initial patterns that ideally should be learned automatically instead of from pre-defined models (which could annoy users and even make difficult the process of learning habits without any bias). Although other techniques have the same limitation, the inherent difficulty in reinforcement learning is interpreting user's feedbacks; this is particularly important for reinforcement learning because this system is based mainly on the interpretation of this feedback.

2.7 Summary

Analyses of different techniques clearly show that each one has specific strengths and weaknesses for the task of learning user's patterns in IEs. As Muller noted [Mul04] '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'.

The solution may rely on the combination of most of them, taking advantage of the strengths of each technique. In order to serve as a guide for that, Table 2.1 summarizes the strengths and weaknesses of each technique relative to IEs considering the different learning stages defined in Section 1.2.3.

Table 2.1: Strengths and weaknesses of Learning Techniques

	Strengths/ Weaknesses	**Artificial Neural Network**	**Classification techniques**	**Fuzzy logic rules**	**Sequence Discovery**	**Instance Based Learning**	**Reinforcement Learning**
Short Period Learning without patterns	Strengths	—	—	—	—	No need for training.	—
	Weaknesses	Requires data for training. No Human-readable output	Requires data for training	Requires data for training	Requires data for training	Sensitive to large attributes set and importance of each one	Requires a model
Pattern Extraction Process from Collected Data	Strengths	Capacity to generalize and model complex situations	Human-readable output. Discovers of conditions	Human-readable output. Robust to uncertainties	Generates rules relating events. Discovers qualitative time relations	Possible source of information for other techniques	—
	Weaknesses	No Human-readable output	Event-Situation relations only	Event-Situation relations only	Unable to define quantitative time relations	No creation of general patterns	Requires a model
Adaptation Process to Reflect Changes in Dynamic Environments	Strengths	Possibility of adding neurons dynamically	Easy to add, delete or change rules	Easy to add, delete or change rules	Easy to add, delete or change sequences	Possibility of providing a solution to instances out of patterns	Adapts patterns interacting with the user
	Weaknesses	Requires restructuring of the network	Requires restructuring to avoid conflicts	Requires restructuring to avoid conflicts	Possible conflicts when adding new sequences	Computational and temporal cost	Difficulties in understanding feedbacks

Chapter 3

General Architecture

One of the main characteristics of IEs is the key role that the user plays as the focus of the entire process, from the beginning to the end. In other words, 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. The variety of user types that can be involved in each IE, and the multitude of potential objectives of each particular environment demands an exhaustive analysis of all components to be included.

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 3.1 shows the global architecture of the LFPUBS.

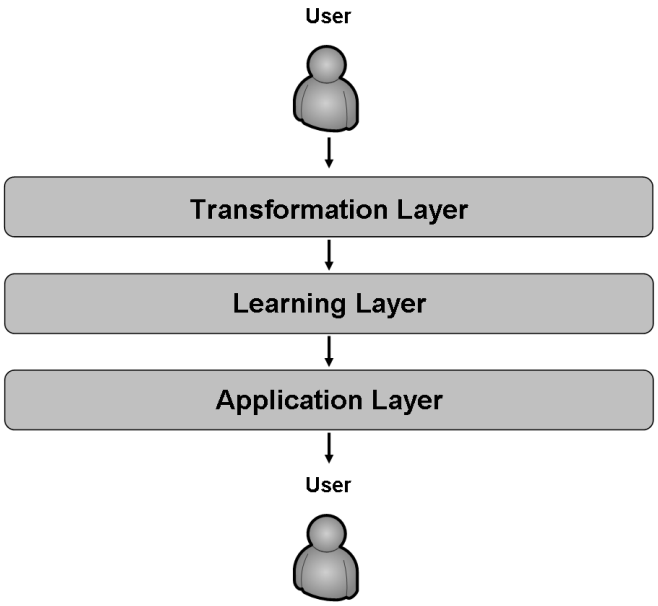

Figure 3.1: Three-layered global architecture.

3.1 Transformation Layer

The objective of the first layer is to transform raw data, i.e., data collected from sensors, into useful information for the learning layer. In addition to dealing with missing or noisy data, the features of IEs demand some specific transformations. Most of the transformations carried out to get useful information are environmentally dependent. Therefore, although some general transformations can be defined, different environments will demand different transformations. In the following, different types of transformations will be suggested based on the data shown in Section 1.2.3.

3.1.1 Inference of simple actions

Once data from sensors have been collected, an important task is to infer meaningful information from this raw data. Sometimes the information provided by sensors is already meaningful. An example is below:

from

2008-10-20T08:15:57, SwitchBathroomLights, on, 100

it is inferred

2008-10-20T08:15:57, BathroomLights, on, 100

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 quite difficult to infer from the simple activation of a sensor. For example, the inference of the simple action 'Go into the Bathroom' is not possible from the activation of a simple sensor, so it must be inferred by combining different actions. The following example 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. It can be inferred that Michael has entered the bathroom. Thus, the transformation of those three actions into only one meaningful action allows the addition of meaning to the sequence of raw data items.

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

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 labelled either as mandatory or as optional. As far as constraints are concerned, they can affect the order of the actions or the duration. The template for the action 'Go into bathroom' is defined as:

'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

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

The objective of these first transformations is to make all actions meaningful by the end of this first step. It is clear that the definition of templates depends on particular environments because they are defined in terms of particular sensors installed in the environment and by the set of actions to be identified. This initial step of inferring meaningful actions is important because once such actions are identified the rest of the learning process will depend upon them.

3.1.2 Inference of complex actions

After inferring from raw data to 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. The 'Make coffee' action's template could be defined as:

'Make Coffee (MakeCoffee, On, 100)'

Actions:

Put Kettle on (Optional)

Open Cupboard (Optional)

Get Coffee (Mandatory)

Take a cup (Mandatory)

Open fridge (Optional)

Get Milk (Optional)

Constraints:

Time

$T_{FirstAction} - T_{LastAction} < 5min.$

Combining different actions into only one action does not make it impossible to define its internal structure. For example, in retrieving all the cases labelled as 'Make coffee', a particular learning process can be carried out in order to detect if there is a pattern that defines how the user makes coffee.

3.1.3 Splitting actions into sequences

In addition to providing meaning, other aspects of the data must be considered. One of these aspects is that data will be collected in a continuous way from sensors, so that they will be represented as a string of actions with a temporal ordering but without any extra structure or organization. The aim of the third transformation is to structure the data collected from sensors according to the meaning of the actions.

In that sense, many different organizations can be suggested. The approach proposed here assumes that the user carries out actions in a sequenced way, and such actions are mainly influenced by prior and later actions. Thus, the string of actions is split into sequences, but instead of using a quantitative window-width, a more flexible criteria that determines the end of one meaningful sequence and the beginning of a new one is used. For instance, going to bed and staying for more than 2 hours or going out and staying out for more than 30 minutes are considered as 'landmarks' that demarcate sequences.

This task is environment-dependent because different environments will demand different criteria. For example, a criterion defined for a Smart Home will not make sense in a Smart Car.

3.2 Learning Layer

The objective of this layer is to transform the information coming from the Transformation Layer into knowledge to be used by the Application Layer. This layer is the core of the system, i.e., the layer that makes it possible to discover frequent behaviours of the users.

In addition to being the layer that allows the environment to discover users' frequent behaviour, the importance of this layer is enhanced because of its independence from particular environments. Unlike the Transformation and Application Layers, the Learning layer is not dependent on particular environments, so that it can be used in any environment without any modification in its design.

As mentioned in the introduction of this chapter, the architecture suggested in the current work tries to separate those aspects dependent on particular environments from those that are not. In that sense, the Transformation Layer is the responsible for making sense of the data collected from sensors, so that the Learning Layer works with meaningful data. Once the Learning Layer extracts knowledge from those data, the Application Layer is the responsible for interpreting and adequately using that knowledge in particular environments. Thus, the Learning Layer is free of any external influence.

The design and the implementation of the components of this layer are the core of this research. Throughout this work, two different approaches for learning users' frequent behaviours have been developed. The first, named the 'Pairwise Approach', discovered those pairs of actions that frequently occurred in the behaviour of the user. The second approach, named the 'Action Map Approach', discovered sets of actions (without any limitation in the number of actions) that frequently occurred in the behaviour of the user. It can be said that the latter approach extends the former by adding new functionalities that better fit the features of IEs. These two approaches are explained in Chapter 4 and Chapter 5, respectively.

3.3 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. In what follows, some of the most promising applications are proposed. Additionally, an interaction system that facilitates the use of that knowledge in different applications is proposed.

3.3.1 Applications of extracted knowledge

The scenarios described in Section 1.1 show some desired applications.

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. It is worth noting that standard and comprehensible representation

of patterns allows the translation of those patterns into any type of model, such as Markov models or finite state machines which can be used to automate actions.

Another interesting application for patterns of behaviour, as suggested by Sarah's scenario suggested, is helping provide *understanding* of users' frequent behaviours in order to detect unhealthy habits or provide help and support for his/her daily tasks. In Sarah's case, even though the house looks after her, the patterns will not be shown to her but to one of the carers. A representation of patterns and related actions must be comprehensible to make understanding them easier. Knowing when Sarah ordinarily has a shower or when she likes to go to bed allows staff members to act in unobtrusive and more efficient ways. Staff members can also detect potentially bad or unhealthy habits. Let us consider that one of the patterns shows that Sarah usually takes pills immediately before having lunch. Staff members know that it is better for her to take the pills just following lunch. Thus, they can persuade Sarah to change her habits.

3.3.2 Applications based on specific learning processes

Up to now, the learning process has been focused on discovering users' frequent behaviours without determining particular steps of the learning process. However, some applications can demand specific learning processes. Let us consider the situation where Sarah's carers are worried because they have detected some indications that could show that she is in the first stages of Alzheimer's disease. Repetitive tasks and disoriented behaviours could be interpreted as a signal of this condition, but these indicators are not discovered by the system because they occur infrequently. In these cases, the learning process should be modified in order to identify these particular patterns. For instance, in this case two modifications would be required:

- The threshold to consider a pattern as frequent must be lower.
- Because repetitive tasks and disoriented behaviours entail a repetition of tasks or visited places, patterns could be included or ruled out by checking for the existence of loops in the sequence. A loop can mean that the user carries out the same tasks or visits the same places many times.

Sarah's carers also want to know if she uses the computer to keep in touch with her son, who lives in another country. Her use of computers is not frequent enough to appear in any frequent pattern, but a specific learning process that focuses on the use of a computer can discovery it. Thus, other types of specific learning processes can be specialized for the performance of specific actions or use of specific devices. In Sarah's case, the specific action would be the use of a computer. To address this case, the following modifications to the learning system would be needed:

- The threshold to consider a pattern as frequent must be lower.
- Only patterns that include the use of a computer would be considered, ignoring all other patterns.

In general, these specific learning processes allow a focus on specific aspects that would not be discovered if the system only considers frequent patterns. In Sarah's case for instance, discovering these sorts of patterns helps staff members confirm or disprove their suspicions about Sarah's behaviour regarding the incidence of Alzheimer's disease or contact with relatives.

3.3.3 Interaction system

An important aspect of IEs has to do with their interaction with users [Aug07b], a key element in the process of efficiently applying the extracted knowledge. One pressure is to reduce the Human-Computer Interaction (HCI) because the system is supposed to use its intelligence to infer situations and user needs. On the other hand, a diversity of users may need or voluntarily seek direct interaction with the system.

Given the importance of users for the success of an IE, it is essential that there be 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 [Sha07; Agh09]. Many types of interactions are feasible. For instance, there are HCI systems based on keyboards, PDAs or touch screens. These types of HCI systems can be very useful in some environments, but they can discriminate against many users of IEs. For example, the elderly and those with mobility restrictions may not be able to use some of these systems adeptly. Thus, in this research work, a speech-based interaction system has been chosen because it 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 before using these patterns of behaviour to automate the activation/deactivation of devices. Thus, beginning with the knowledge discovered by the Learning Layer, the speech-based HCI system allows users to accept, delete or refine the patterns for automation.

Technical Aspects

In the process of developing a speech-based HCI system, a speech synthesizer and a speech recognizer are necessary. The chosen speech synthesizer has been FreeTTS 1.2 [1] while Sphinx-4 [2] has been chosen as the speech recognizer. Both FreeTTS and Sphinx-4 make the interactions with the user easier by providing easy to use tools. Complications arise mainly due to changing nature of IEs. For example, the interaction system cannot know beforehand which devices are in the environment, so that grammars for the recognizer must be created and loaded dynamically to connect the interaction module with a specific environment.

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

[2] http://cmusphinx.sourceforge.net/sphinx4/

HCI system for automating devices

The HCI system developed for automating devices shows the user the knowledge discovered by the Learning Layer. Such knowledge is represented based on a standard language defined in the Learning Layer (for further details see Section 4.3 and Section 5.2), which ensures a standard representation of all frequent behaviours.

The aim of this first approach is to allow the user, for as long as he/she wants, to decide whether he/she wants the environment to use the discovered patterns to automate the activation/deactivation of devices.

Once the user confirms that he/she wants to interact, the environment displays all the patterns discovered in the Learning Layer. Let us consider Michael's behaviour of turning on the fan when he finishes taking his shower if relative humidity level in the bathroom is higher than 70%.

System: *Occurs Shower is off Then turn on the Bathroom fan 4 seconds after If Bathroom relative humidity level is >70%*

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

Accepting a pattern means that the user accepts it as useful, and therefore the environment will use it to automate devices (the fan of the bathroom in this case). If the user chooses to delete the pattern, it will not be used to automate the action. Finally, the user can choose to refine a pattern if he/she considers it to be useful pattern but that it requires some modification. In this case, the interaction system must guide the user through the pattern to find out exactly what needs to be changed. Let us consider the situation where Michael wants to turn the fan on even when relative humidity level is lower than 70% (let us consider when the relative humidity level is higher than 50%). The interaction could proceed along the following lines:

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

User: *Condition*

System: *The Condition part was: Bathroom relative humidity level is >70%*

System: *The attribute was 'Bathroom relative humidity level'. Do you want to modify it? (yes/no)*

User: *No*

System: *The value was 70%. Do you want to change it? (yes/no)*

User: *Yes*

System: *Please, say the new value as a percent*

User: *50*

System: *Do you want to confirm 50 as the new value? (yes/no)*

User: *Yes*

This HCI system has been conceived to get the user's feedback and provide the environ-

ment with a set of patterns that can be safely used to act proactively. Thus, accepted and modified patterns will be stored while deleted patterns will not.

Other applications of speech-based HCI systems

Different environments and different objectives require the development of different interaction systems. In Sarah's scenario, allowing the user to decide what patterns the environment may use to automate devices does not make sense, because the main objective of the environment is to allow staff members to understand her behaviour. In this case, an interesting HCI system would be one that allows staff members to label Sarah's behaviours as 'normal', 'hazardous' or 'unhealthy', so that when new situations match behaviours labelled as 'hazardous' or 'unhealthy', the system can warn staff members about these situations. For instance, if staff members label the behaviour of taking a pill before having lunch as 'unhealthy', the next time Sarah shows that behaviour the system would warn staff members.

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 all the applications. In other words, the developed HCI system can be easily adapted in many different ways to suit the needs of different users and different environments.

3.4 Graphical User Interface

In order to facilitate the use of the LFPUBS, in parallel to the architecture, a Graphical User Interface (GUI) has been developed. Its main objective is to allow the user of the LFPUBS to design the learning process that will be carried out. Some of the components of the LFPUBS allow the user of the LFPUBS to design the learning process based on the requisites (runtime, complexity of the knowledge to discover etc.) of different applications. Figure 3.2 shows the first screen of the GUI in which the user can load the data to be analysed. In the following chapters, how the GUI allows one to modify different parameters will be shown when such concepts are explained.

3.5 Summary

The first objective of the architecture designed for the LFPUBS is to cover all aspects of the system from data collection to applying the knowledge discovered by the system. Because LFPUBS will be applied in different environments (Smart Homes, Smart Cars, Smart Classrooms etc.) and for different types of users (elderly people, students, people with disabilities etc.), it is clear that the LFPUBS should somehow consider the particularities of each of them.

The main objective of the designed architecture was to separate the environment-dependent aspects that will require a different treatment in each particular environment,

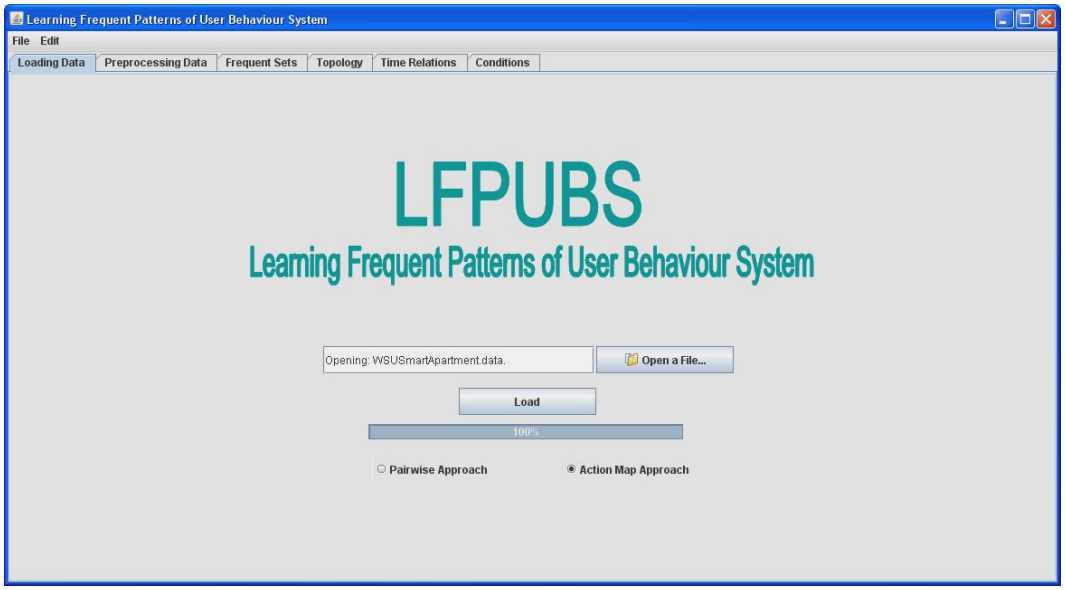

Figure 3.2: Initial screen of the Graphical User Interface.

from those environment-independent aspects that can be generalized for all types of environments. To that end, the main effort was focused on designing an architecture that allows the development of an environment-independent Learning Layer.

Finally, a 3-layered architecture was designed. The objective of the first layer, the Transformation Layer, is to transform raw data into meaningful data, i.e., to infer actions/activities of the user from raw data collected from sensors. This layer is clearly environment-dependent because it is highly influenced by the types of sensors installed in the environment and the actions/activities to be identified.

The core of the system is the Learning Layer, which has the objective of discovering the frequent behaviours of the user starting from data output from the Transformation Layer. This layer is environment-independent because the Transformation Layer fills the gap between the particular environment and the LFPUBS. Once knowledge has been discovered the Application Layer is the responsible for applying it to the particular environment.

Thus, the aim of the Application Layer is to employ the knowledge discovered by the Learning Layer in particular environments. The specific applications will mainly depend on the particular purposes of the particular environments. Therefore this layer is also environment-dependent.

Finally, in order to allow the user of the LFPUBS to define the parameters of different components of the architecture, a GUI has been developed.

Chapter 4

Learning Frequent Behaviours: the Pairwise Approach

A three-layered general architecture for the LFPUBS has been defined in Chapter 3. Together with the general architecture, the objectives and functionalities of Transformation and Application Layers have also been explained. It can be said that the Transformation Layer fills the gap from the real world to the LFPUBS, whereas the Application Layer fills the opposite gap, from the LFPUBS to the real world. However, the core of the LFPUBS is the Learning Layer, which makes it possible to transform data into knowledge, i.e., it is the layer that discovers the frequent behaviours of the user.

Aside from the key role that the Learning Layer plays in the process of providing the environments with intelligence, its importance is increased because of its independence of particular environments. Therefore, the architecture of the layer, as well as all of the components developed within this layer, must consider all types of possibilities that can come up when dealing with users' behaviours.

As mentioned in Section 3.2, two different approaches for the LFPUBS have been developed in this research work. LFPUBS's GUI allows the selection of any of the approaches (see Figure 4.1). The first approach, which is the basis for the second approach, is focused on only learning about the pairwise relations between the actions of the user. In this chapter, different aspects of the Learning Layer for this *Pairwise Approach* are explained in more detail.

4.1 Introduction

This first approach is focused on discovering frequent relations between simple actions. This implies that such relations do not involve more than two actions (that is why they are called pairwise relations).

To make this idea clear, let us consider Michael's scenario. His morning habits involve a set of actions that he frequently performs together. In the process of discovering frequent relations between the actions of the user, this first approach would discover a frequent relation between the actions 'Alarm On' and 'Bathroom On'. Then, as another frequent relation it would discover that the action 'Bathroom On' is related to the action 'BathroomLights On'. In this sense, it should be mentioned that once frequent relations have been discovered, this first approach also discovers the time relation between both actions as well as the conditions

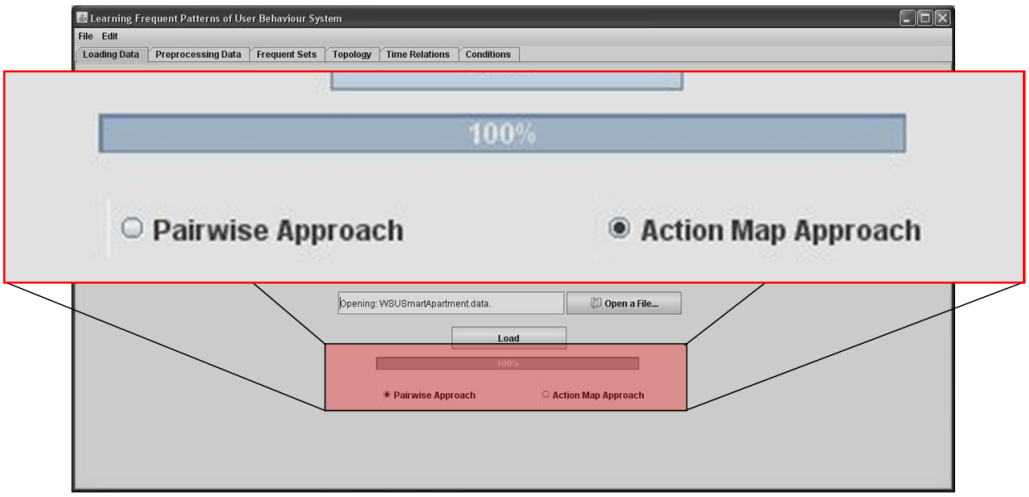

Figure 4.1: Selecting the Approach by means of the GUI.

under which such a relation is true.

As mentioned in Section 1.2.3, 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, the LFPUBS considers three main different groups of information in this first approach:

- (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 (for further details see Section 3.1).

- (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.

- (type M) Motion information. This information can be used to infer where the user is (in the bedroom, outside the house or elsewhere). This type of information is mainly used in the Transformation Layer to infer the actions of the user.

It is clear that other types of information already exist, such as those that indicate the health status of the user or alarm pendants, which could be interesting in IEs. The inclusion of other types of sensors is being considered for future versions of the system.

In the following sections, let us assume that a small example of the data collected for Michael's case is shown in Appendix A.

4.2 Architecture of the Learning Layer

Being an environment-independent layer, the architecture as well as different modules must be designed and developed taking into account all of the characteristics of the different environments. The architecture proposed for this layer is depicted in Figure 4.2.

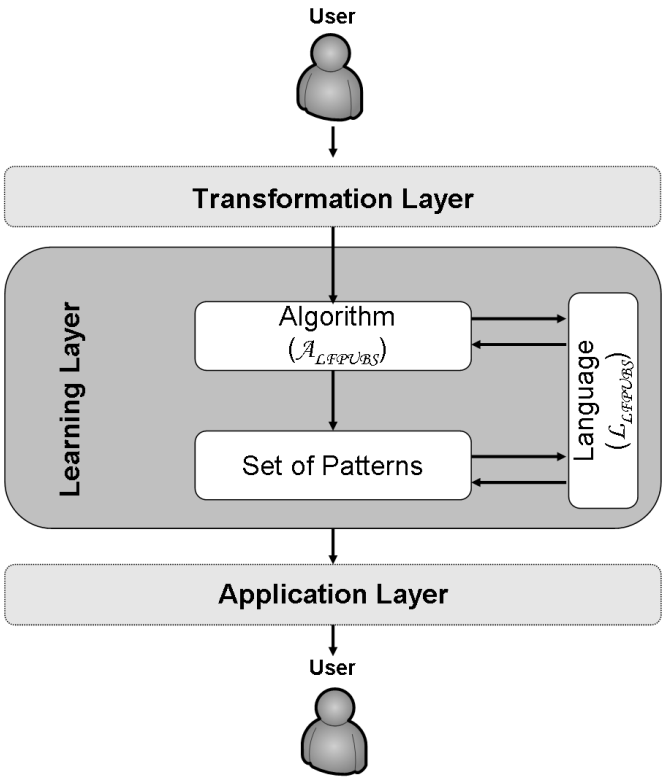

Figure 4.2: Essential components of the Learning Layer

The underlying idea is the separation of the representation of the discovered patterns from the process of discovering per se. The core of the representation module is a language ($\mathcal{L}_{LFPUBS}$) that provides a standard conceptualisation of the patterns so that the environment is able to represent all type of patterns that can occur in the environment. On the other hand, the process of discovering is based on an algorithm ($\mathcal{A}_{LFPUBS}$) that taking into account all of the characteristics of the IEs attempts to discover frequent patterns.

In the following sections, these two modules of the Learning Layer will be explained in more details.

4.3 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 the LFPUBS is based on ECA (Event-Condition-Action) rules [Aug04]. Besides providing a standard way of representing patterns, it makes sure those patterns are clearly specified and enables other technologies to check their integrity [Aug07c].

In the same way as ECA rules, $\mathcal{L}_{LFPUBS}$ basically relates two actions (defined by the ON and THEN clauses) and the specific conditions (defined by the IF clause) under which that relation occurs. Finally, unlike basic ECA rules, $\mathcal{L}_{LFPUBS}$ allows the environment to define the time relation between both actions. Considering Michael's behaviour of turning on the fan in the bathroom; using $\mathcal{L}_{LFPUBS}$ it would be represented as follows:

```
(Pattern 0)
   ON occurs (Shower, Off,t0)
   IF context (Bathroom relative humidity level (>,70%))
   THEN do (On, BathroomFan, t) when t = t0 + 4s
```

For the complete specification of $\mathcal{L}_{LFPUBS}$, see the Appendix B.

4.3.1 Event Definition

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

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. In this case, such actions are collected by means of A-type sensors. In Michael's case, the Event Definition is defined as

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

4.3.2 Condition Definition

The IF clause defines the necessary conditions under which the action specified in the THEN clause is the appropriate reaction to the event listed in the ON clause. Because it is almost impossible for an Event-Action relation to be true under any condition, appropriate conditions are necessary to represent accurate patterns. Below, some examples of conditions are provided:

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

(Condition 2)
   IF context (TimeOfDay (>,20:30:00))

(Condition 3)
   IF context (DayOfWeek (=, Tuesday))
```

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. Thus, $\mathcal{L}_{LFPUBS}$ has two possible ways to define the conditions:

- Information coming from C-type sensors (e.g., 'Relative humidity level' (Pattern 0) or 'Temperature' (Condition 1))
- Calendar information (e.g., 'Time of Day' (Condition 2) or 'Day of Week' (Condition 3))

The possible values of such attributes depend on the nature of each attribute. In that sense, $\mathcal{L}_{LFPUBS}$ considers two types of values:

- Qualitative values (e.g., 'Tuesday' (Condition 3))
- Quantitative values (e.g., '20ºC' (Condition 1) or '20:30:00' (Condition 2))

4.3.3 Action Definition

Finally, 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 is made up of the triggered action, called the *main sensor triggering (mainSeT)* and the Time Relation between the Event and Action situations. The *mainSeT* contains the device implied in the action ('BathroomFan') and the nature of the action ('On').

The Time Relation can be either *quantitative* (Action 1) or *qualitative* (Action 2), with the usefulness of each type of relation being different.

```
(Action 1)
   THEN do (On, BathroomFan, t) when t = t0 + 4s


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

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 have known 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.

4.4 Learning patterns with $\mathcal{A}_{LFPUBS}$

Coupled with $\mathcal{L}_{LFPUBS}$, an essential component in the Learning Layer is the algorithm ($\mathcal{A}_{LFPUBS}$) that discovers the frequent behaviours of the users. In order to coordinate with

$\mathcal{L}_{LFPUBS}$ and discover complete and unambiguous patterns, $\mathcal{A}_{LFPUBS}$ must consider all of the different aspects defined by the language.

Therefore, the $\mathcal{A}_{LFPUBS}$, will somehow have to be able to discover pairs of actions that frequently occur together as well as identifying if possible, a quantitative Time Relation between them. Finally, $\mathcal{A}_{LFPUBS}$ must also be able to discover the set of conditions (defined by the IF clause) for which the pattern is true.

The different steps to be performed by $\mathcal{A}_{LFPUBS}$ in order to discover the frequent behaviours of the users are depicted in Figure 4.3.

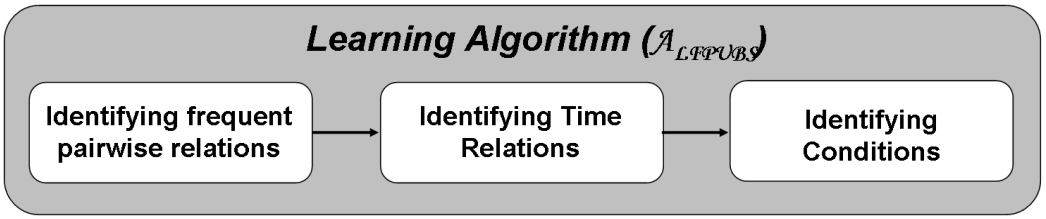

Figure 4.3: Set of steps to be performed by the learning algorithm.

The same idea represented by means of a pseudo algorithm is as follows:

$\mathcal{A}_{LFPUBS}$ **Algorithm**
For each action (type A), consider it as *mainSeT*
Identify the *associatedSeT*s of type A (See Section 4.4.1)
For each frequent relation (*mainSeT*-*associatedSeT*)
Identify possible quantitative Time Relations (See Section 4.4.1)
Identify Conditions using context/calendar information (See Section 4.4.3)

Emphasizing A-type sensors as *mainSeT* follows from the fact that they represent the actions of the users, so discovering frequent relations between them will define the frequent behaviours of the users. Considering the data shown in Appendix A, information provided by 'Alarm', 'Bathroom', 'BathroomLights', 'Shower', 'BathroomFan', 'Cabinet' and 'Tap' would be type A information, whereas 'TempBathroom' and 'HumBathroom' would be type C information. Using those data to clarify different concepts, the following section will explain the main three steps of the $\mathcal{A}_{LFPUBS}$ in more detail.

4.4.1 Identifying Frequent Relations

The aim of this first step is to discover those pairs of actions that frequently occur together, i.e., consecutively. In that sense, if the specific action of a user frequently precedes another specific action, it defines a frequent behaviour of such a user.

For this process, all of the different actions (A-type information) that occurred in the environment are considered. For each one of them, the system considers all of the occurrences of such an action and it collects the action that precedes each one of its occurrences. The

aim is to discover whether the action considered by the system is frequently preceded by a specific action. If the system discovers that there is an action that frequently precedes the action in question, it defines a frequent relation where the two actions are considered as the *associatedSeT* and the *mainSeT*, respectively.

The list of possible *associatedSeT*s for each *mainSeT* is obtained through a similar approach to the Apriori method for mining association rules [Agr95], with only two differences:

- The possible associations are limited to the *mainSeT*.
- The result do have a sense of sequence because it is known in advance that the *mainSeT* follows the *associatedSeT*.

As in every association mining process, the *minimum support* and the *confidence level* must be provided manually. Developed GUI allows the definition of such parameters (see Figure 4.4). The *support* is defined as the number of times that the *mainSeT* occurs, whereas the *confidence* level is a percentage value that shows how frequently the relation occurs among all of the occurrences of the *mainSeT*. The *confidence* level is defined by:

$$conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X) \tag{4.1}$$

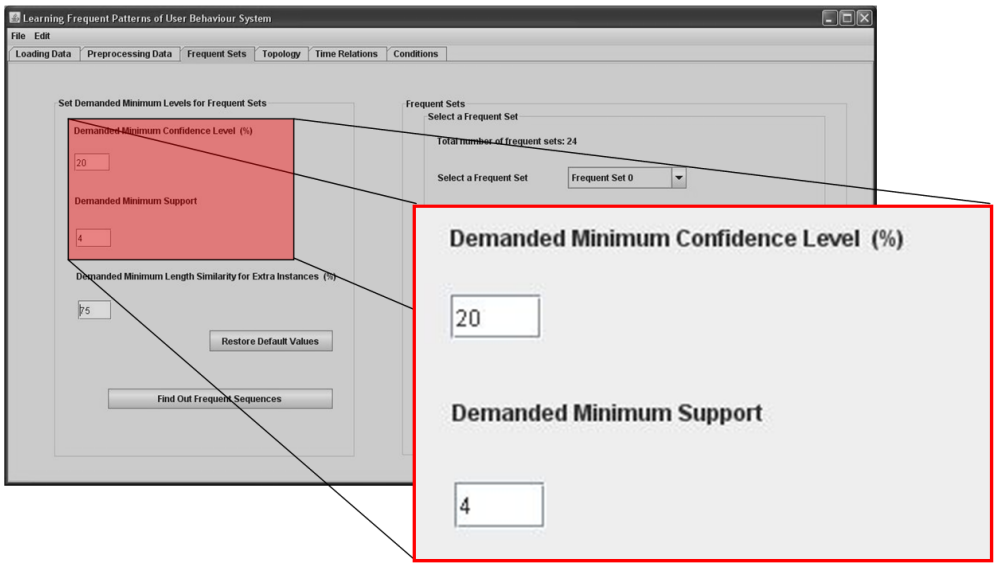

Figure 4.4: Defining the minimum confidence and support levels.

Next, all of these concepts will be clarified using data collected from Michael's scenario (see the data in Appendix A), and the process of identifying frequent relations will be explained step by step.

Identifying Frequent Relations in Michael's scenario

Considering Michael's data, the aim of this first step is to discover whether his behaviour shows some frequent relations between the actions he performs. For this example, let us consider a minimum support of four occurrences and a confidence level of 20%.

In this first step, $\mathcal{A}_{LFPUBS}$ considers all of the different actions (A-type). For each occurrence of these actions, it collects the previous action. After that, the support and the confidence level for each relation are calculated. For instance, the action 'BathroomFan On' occurs four times, so it fulfils the demanded minimum support. The previous action in those four occurrences was 'Shower Off', so its confidence level is of 100%. The relation of the action 'BathroomFan On' would be represented as follows:

'BathroomFan On' (support: 4) *is preceded by* 'Shower Off' (4 times; Conf: 100%)

Thus, in Michael's case, this first step would yield as a result the following list of actions with their previous actions.

'Alarm On' (sup:10) *is preceded by* —

'Bathroom On' (sup:10) *is preceded by* 'Alarm On' (10; Conf: 100%)

'Bathroom Off' (sup:10) *is preceded by* 'BathroomLights Off' (10; Conf: 100%)

'BathroomLights On' (sup:10) *is preceded by* 'Bathroom On' (10; Conf: 100%)

'BathroomLights Off' (sup:10) *is preceded by* 'BathroomFan Off' (4; Conf: 40%)
'Tap Off' (1; Conf: 10%)
'Shower Off' (2; Conf: 20%)
'Cabinet Off' (2; Conf: 20%)
'BathroomLights On' (1; Conf: 10%)

'Cabinet On' (sup:15) *is preceded by* 'BathroomLights On' (9; Conf: 60%)
'Cabinet Off' (6; Conf: 40%)

'Cabinet Off' (sup:14) *is preceded by* 'Mouthwash On' (7; Conf: 50%)
'Cabinet On' (1; Conf: 7%)
'Gel On' (3; Conf: 21.5%)
'Towel On' (3; Conf: 21.5%)

'Mouthwash On' (sup:8) *is preceded by* 'Cabinet On' (8; Conf: 100%)

'Towel On' (sup:6) *is preceded by* 'Cabinet On' (3; Conf: 50%)
'Gel On' (3; Conf: 50%)

'Gel On' (sup:6) *is preceded by* 'Cabinet On' (3; Conf: 50%)
'Towel On' (3; Conf: 50%)

'Shower On' (sup:6) *is preceded by* 'Cabinet Off' (6; Conf: 100%)

'Shower Off' (sup:6) *is preceded by* 'Shower On' (6; Conf: 100%)

'BathroomFan On' (sup:4) *is preceded by* 'Shower Off' (4; Conf: 100%)

'BathroomFan Off' (sup:4) *is preceded by* 'BathroomFan On' (4; Conf: 100%)

'Tap On' (sup:1) *is preceded by* 'Cabinet Off' (1; Conf: 100%)

'Tap Off' (sup:1) *is preceded by* 'Tap On' (1; Conf: 100%)

Once support and confidence levels are calculated, only those relations whose support and confidence levels are greater than the demanded minimum levels are considered as Frequent Relations. After this first step, the two actions involved in the relation define the ON and THEN clauses of a pattern. In Michael's case, when assuming a minimum support of four and a minimum confidence level of 20%, there are 20 patterns. Besides defining the parameters, the GUI allows the user of the LFPUBS to see the identified Frequent Relations and the information of each one of them (see Figure 4.5).

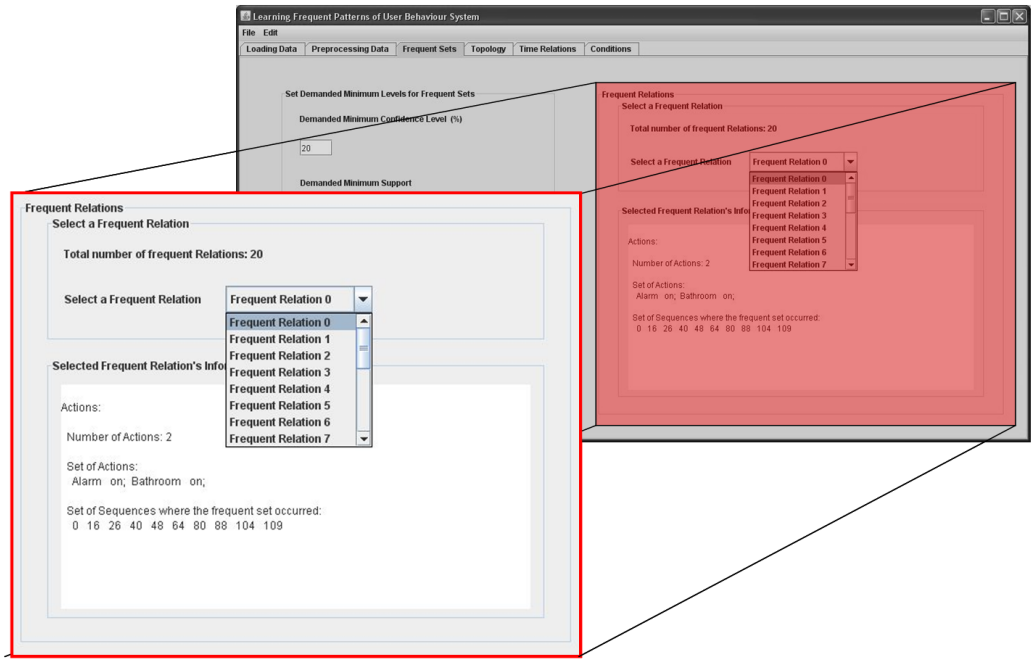

Figure 4.5: Information about the discovered Frequent Relations.

Some of them, represented using the $\mathcal{L}_{LFPUBS}$, are shown below as examples (all patterns are defined in Appendix C).

```
(Pattern 1)
 ON occurs (Alarm, On,t0)
 IF ---
 THEN do (On, Bathroom, t)
      when ---
```

```
(Pattern 2)
 ON occurs (BathroomLights, Off,t0)
 IF ---
 THEN do (Off, Bathroom, t)
      when ---
```

```
(Pattern 3)
 ON occurs (Bathroom, On,t0)
 IF ---
 THEN do (On, BathroomLights, t)
      when ---
```

```
(Pattern 4)
 ON occurs (BathroomFan, Off,t0)
 IF ---
 THEN do (Off, BathroomLights, t)
      when ---
```

```
(Pattern 5)                              (Pattern 6)
 ON occurs (Shower, Off,t0)               ON occurs (Cabinet, Off,t0)
 IF ---                                   IF ---
 THEN do (Off, BathroomLights, t)         THEN do (Off, BathroomLights, t)
      when ---                                 when ---

(...)

(Pattern 19)                             (Pattern 20)
 ON occurs (Shower, Off,t0)               ON occurs (BathroomFan, On,t0)
 IF ---                                   IF ---
 THEN do (On, BathroomFan, t)             THEN do (Off, BathroomFan, t)
      when ---                                 when ---
```

Pattern representation clearly shows that only ON and THEN clauses are defined after this first step, and even in the THEN clause, the Time Relation is not yet defined. Therefore, other steps are necessary to discover patterns that completely define the frequent behaviours of the users.

4.4.2 Identifying Time Relations

The step of 'Identifying Frequent Relations' discovers those frequent relations hidden in the data, and it provides a first representation of them. This first representation implies a first definition of a temporal relation because the ON and THEN clauses define a temporal order of the actions, as the action defined by the THEN clause is preceded by the action defined by the ON clause. In this sense, patterns implicitly use a qualitative relation defined by the term 'after'.

As stated in Section 4.3.3, qualitative relations allow one to understand the logical order of the actions. Even so, 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 in order to better define users' behaviours. For that purpose, the 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 in which the $associatedSeT$ is followed by the $mainSeT$.

Once the Time Distances are collected, the next step is to identify possible quantitative time relations. For that purpose, the 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. Such parameters can also be defined using the GUI (see Figure 4.6).

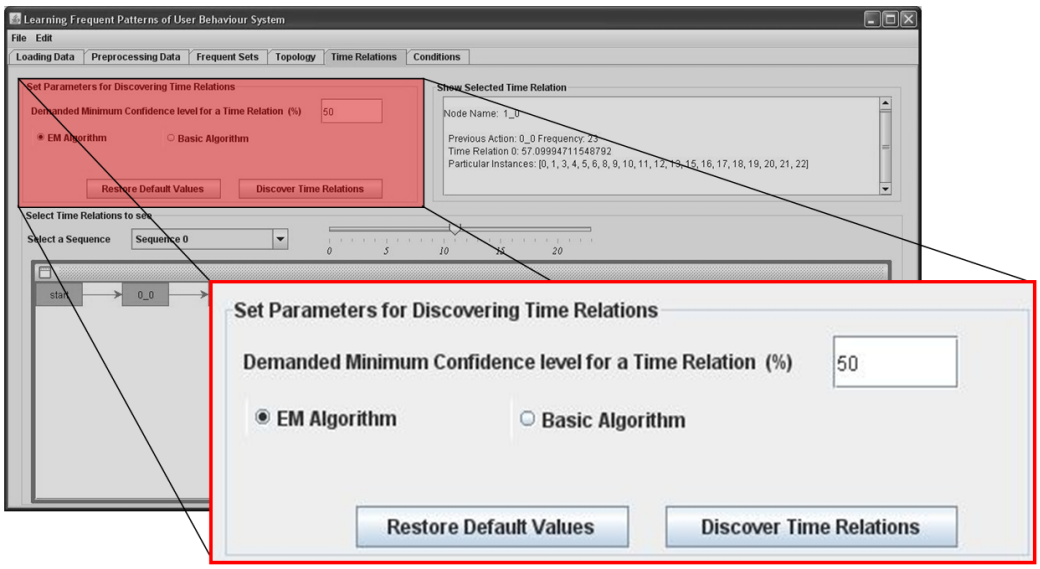

Figure 4.6: Selecting the algorithm to use in order to identify quantitative Time Relations.

'Basic Algorithm' for Identifying Time Relations

Many different algorithms can be proposed as a basic algorithm for discovering quantitative Time Relations. The LFPUBS proposes a very basic algorithm, which groups collected Time Distances by taking into account the similarity between them. The process of creating groups is defined by the following algorithm.

Basic Algorithm for Identifying Time Relations
Randomly select an initial Time Distance
Create a new group for that initial Time Distance
For each Time Distance
Calculate if it is within any existing group (See below for further details)
If it is, **Then** Include it in the group and recalculate the parameters of the group
If it is not, **Then** Create a new group and calculate the parameters of the group

To determine whether a Time Distance falls within an existing group, each group has a range that defines what Time Distances it covers. Such a range is established by the following:

$$[min, max] = \overline{x} \pm (\overline{x} * tolerance) \qquad where \qquad \overline{x} = \frac{\sum_{i=1}^{n} a_i}{n} \tag{4.2}$$

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

If a Time Distance does not fulfil the requirements to join any group, a new group is created using that value as the group's mean value. Every time a new value is added to a group, the mean value and the range of that group are recalculated.

Once the groups are created, it is clear that not all al the groups define interesting quantitative Time Relations. For this reason, a *minimum confidence level* must be established in this case too, so that only those groups that cover more instances than the minimum established by the confidence level define a quantitative Time Relation. In such cases, the mean value of the group is treated as the quantitative Time Relation. In 'Basic Algorithm' case, the GUI, apart from allowing the definition of the minimum confidence level, allows the definition of the deviation from the mean value (see Figure 4.7).

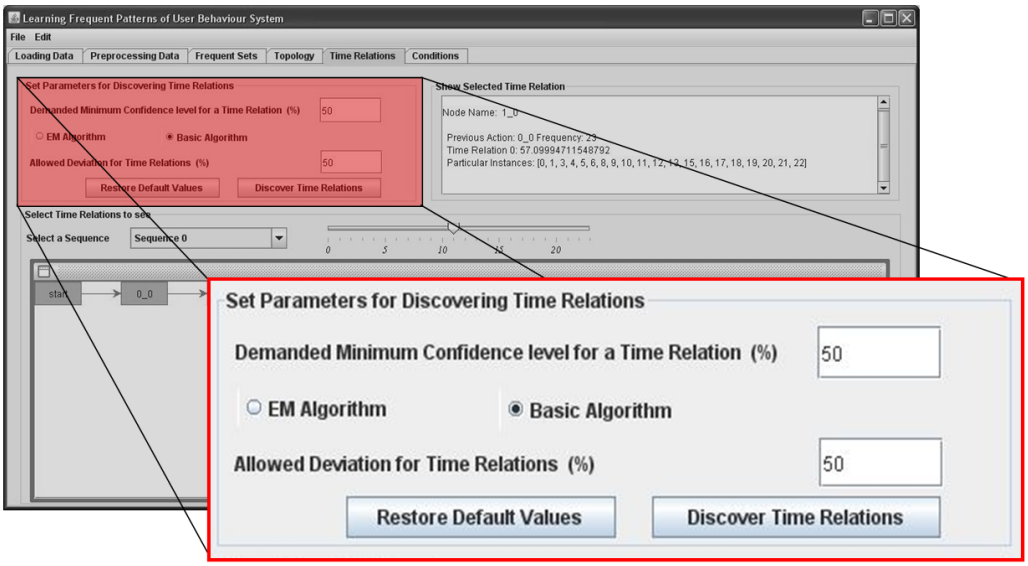

Figure 4.7: Defining the allowed deviation for the 'Basic Algorithm'.

'EM Algorithm' for Identifying Time Relations

Besides the 'Basic Algorithm', the LFPUBS includes another algorithm that creates clusters of Time Distances based on the Expectation-Maximization (EM) algorithm [Hog05]. The basic idea of the EM algorithm is to estimate the maximum likelihood between parameters. An important advantage of this algorithm is that it automatically calculates the necessary number of groups and includes each occurrence in its corresponding group.

Once groups are created by the EM algorithm, the process of deciding what groups defines a quantitative Time Relation is he same as the 'Basic Algorithm'.

Identifying Time Relations in Michael's scenario

Consider again Michael's data and the Frequent Relations found in Section 4.4.1. The objective of this step is to define possible quantitative Time Relations for such patterns. Let

us consider the 'Pattern 19', which shows how Michael usually turns on the fan after taking a shower. The collected Time Distances between both actions are depicted in Figure 4.8.

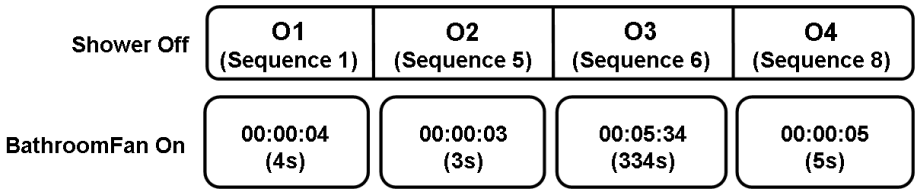

Figure 4.8: Time Distances between occurrences of 'Shower Off' and 'BathroomFan On'

Considering these Time Distances, the process of making groups using the 'Basic Algorithm' is as follows (let us assume 50% as the tolerance percentage):

(Sequence 1, 4s); There is no group; create(group0, $\overline{x}$ (4s), range: [2,6])

Tolerance = 4 ± (4 * 0.5)

(Sequence 5, 3s); 3=[2,6]; join(group0, $\overline{x}$ (3.5s), range: [1.75,5.25])

Tolerance = 3.5 ± (3.5 * 0.5)

(Sequence 6, 334s); 334≠[1.75,5.25]; create(group1, $\overline{x}$ (334s), range: [167,501])

Tolerance = 334 ± (334 * 0.5)

(Sequence 8, 5s); 5=[1.75,5.25]; join(group0, $\overline{x}$ (4s), range: [2,6])

Tolerance = 4 ± (4 * 0.5)

At the end of this task, two groups ('group 0' and 'group 1') are created, which cover three and one occurrences, respectively. In order to extract quantitative Time Relations, the LFPUBS checks whether the confidence level of different groups is over the demanded one (let us say 50% in this case). It is clear that 'group 0' represents a quantitative Time Relation because it covers 3 out of 4 occurrences (75%), whereas 'group 1' does not. Thus, in this case the mean value of 'group 0' defines a quantitative Time Relation (see Figure 4.9 for its representation in the GUI), creating a pattern like the following:

```
(Pattern 19)
  ON occurs (Shower, Off,t0)
  IF ---
  THEN do (On, BathroomFan, t)
       when t = t0 + 4 s.
```

If no quantitative relation exists that covers enough occurrences, the pattern will keep being defined using the qualitative term 'after'. Using Michael's data and the 'Basic Algorithm', at the end of this step, Michael's frequent behaviours are defined as follows:

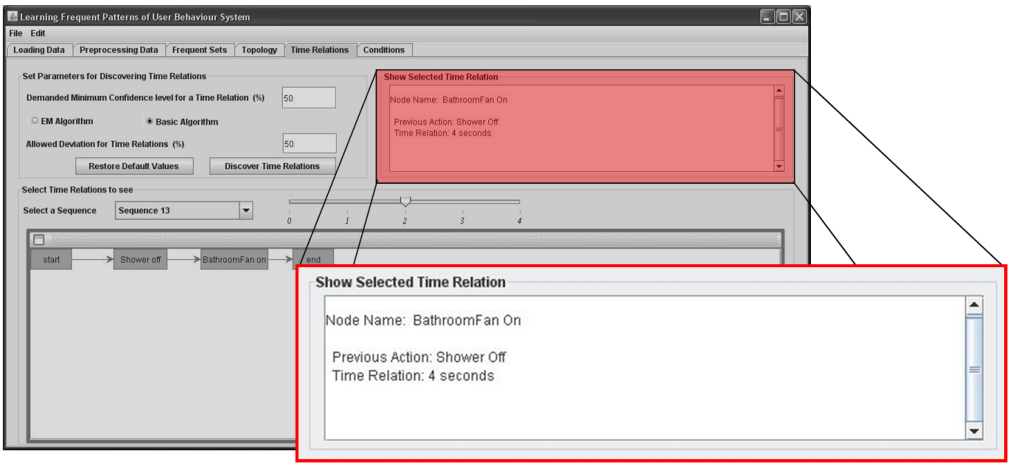

Figure 4.9: Information about the discovered Time Relations.

```
(Pattern 1)
 ON occurs (Alarm, On,t0)
 IF ---
 THEN do (On, Bathroom, t)
      when t is after t0
```

```
(Pattern 2)
 ON occurs (BathroomLights, Off,t0)
 IF ---
 THEN do (Off, Bathroom, t)
      when t = t0 + 1s
```

```
(Pattern 3)
 ON occurs (Bathroom, On,t0)
 IF ---
 THEN do (On, BathroomLights, t)
      when t = t0 + 2s
```

```
(Pattern 4)
 ON occurs (BathroomFan, Off,t0)
 IF ---
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(Pattern 5)
 ON occurs (Shower, Off,t0)
 IF ---
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(Pattern 6)
 ON occurs (Cabinet, Off,t0)
 IF ---
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(...)
```

```
(Pattern 19)
 ON occurs (Shower, Off,t0)
 IF ---
 THEN do (On, BathroomFan, t)
      when t = t0 + 4s
```

```
(Pattern 20)
 ON occurs (BathroomFan, On,t0)
 IF ---
 THEN do (Off, BathroomFan, t)
      when t is after t0
```

So far, only exact quantitative values have been considered, but other types of quantitative relations such as ranges (e.g., '10-15 minutes') are being considered for future versions of the system.

4.4.3 Identifying Conditions

Once the Frequent Relations and the Time Relations are discovered, it is clear that those patterns represent the frequent behaviours of the users in a comprehensible way. Even so, although such patterns are supported by the occurrences of such relations, it is possible that they do not represent all of the occurrences of the actions implied in the patterns, such that conditions are necessary in those cases. There are primarily two reasons to confirm the need to include conditions to obtain accurate patterns:

- The discovered Time Relations do not cover the Time Distances of all occurrences. For instance, the Time Relation discovered for 'Pattern 19' covered 3 out of 4 occurrences, so there is an occurrence (O_3 in that case) of such a relation that is not represented correctly by that Time Relation and extensively by the pattern.

- There is a high probability that occurrences of $associatedSeT$ exist that are not covered by such a relation. For instance, considering again 'Pattern 19', the action 'Shower Off' occurred six times, whereas the action 'BathroomFan On' followed this action only four out of those six occurrences. Therefore, there were two occurrences of 'Shower Off' (Sequence 3 and Sequence 10) that were not followed by the action 'BathroomFan On'.

The importance of discovering conditions is enhanced if the patterns are going to be used to automate future activation/deactivation of devices. If a device is going to be activated based on the relation defined by the pattern, there could be occurrences of $associatedSeT$ that are not covered by the pattern. For instance, if every time there is a 'Shower Off' action the environment turns on the fan 4 seconds afterwards, there is a probability that Michael will not like it.

Using Context and Calendar Information for Identifying Conditions

As mentioned above, there are at least two reasons for discovering conditions. By having two different sources for conditions, different approaches can be considered.

- Discovering conditions by considering all different sources together. In other words, this approach combines all occurrences that are not covered by the pattern and discovers conditions for all those cases.

- Discovering conditions by considering different sources as independent. First, it discovers conditions for those occurrences of $associatedSeT$ that are not covered. Then, it discovers conditions for those Time Distances that are not covered. Finally, it combines both sets of conditions to get the final conditions.

The LFPUBS uses the second approach; it discovers two types of conditions and then it combines them to define the final conditions.

In both cases, for the purpose of discovering conditions, two tables - *covered* and *non-covered* - are created. In the *covered* table, those occurrences that are correctly classified by the pattern appear, together with the calendar and context information collected when such occurrences happened. The same information for occurrences in which the patterns fails is registered in the *non-covered* table.

Once the tables are created, separating both tables by using the information they contain allows one to discover conditions. In that sense, the task of separating can be solved by treating it as a classification problem. The JRip Algorithm [Wit05] was used to accomplish this task. A small modification was needed because JRip provides rules only for the objective of separating both classes (*covered* and *non-covered*), whereas for users' patterns, it is desirable to obtain conditions that define the occurrences of the *covered* class. Thus, the LFPUBS always obtains a set of rules that indicates under what conditions the pattern is true, instead of a mix of rules that indicate when it is true and when it is not.

Identifying Conditions in Michael's scenario

In Michael's case, previous steps discovered a set of patterns that represented his frequent behaviours. Even so, the need for conditions in order to make these patterns more accurate is clear. The behaviour represented by 'Pattern 19' is a clear example of this. In this case, there are different sources of conditions, but let us consider the occurrences of 'Shower Off' ($associatedSeT$) that are not covered by 'Pattern 19' as an example to show how the LFPUBS discovers conditions.

The relation described by 'Pattern 19' is supported by four occurrences (Sequence 1, Sequence 5, Sequence 6 and Sequence 8) in Michael's data. Even so, there are two occurrences of the action 'Shower Off' (Sequence 3 and Sequence 10) that are not followed by the action 'BathroomFan On'. Thus, covered and non-covered tables are created using calendar and context information collected when the corresponding occurrences of 'Shower Off' occurred. The *covered* and *non-covered* tables created for this case are depicted in Figure 4.10.

Because these tables contain few instances, the covered and non-covered classes can be separated in many different manners. Even so, the most efficient manner is using the context information *Bathroom relative humidity level* ('Hum. Bathroom'). Thus, the condition obtained in this case is as follows (see Figure 4.11 for its graphical representation):

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

The Time Relation discovered for 'Pattern 19' also needs to define other conditions because it does not cover all of the Time Distances that occurred between the actions 'Shower Off' and 'BathroomFan On'. In this case, the condition discovered for the relation between $associatedSeT$ and $mainSeT$ also separates the *covered* and *non-covered* tables created for the Time Distances. Therefore, it is not necessary to add any new condition. If this was the case, the conditions obtained for those cases should be added to the existing ones.

covered

	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

non-covered

	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

Figure 4.10: *Covered* and *non-covered* tables with calendar and context information

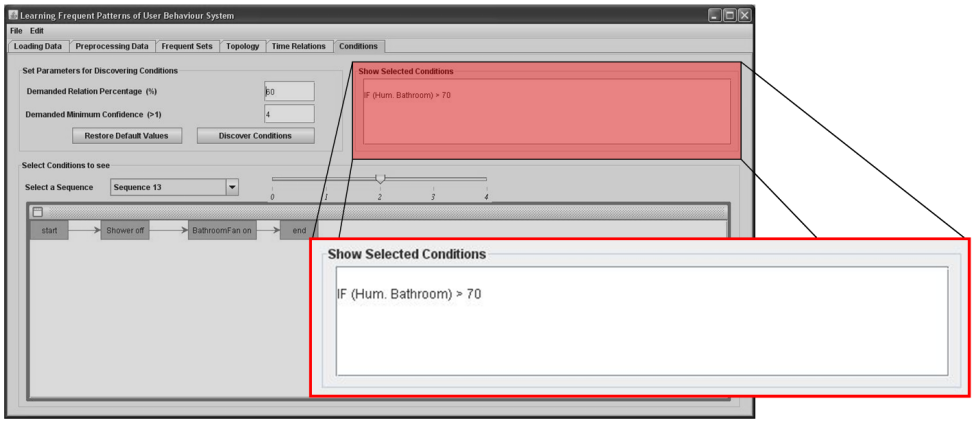

Figure 4.11: Information about the discovered Conditions.

Once conditions are discovered, Michael's frequent behaviours are represented by considering all the clauses. After this last step, Michael's habits are represented by the following patterns:

```
(Pattern 1)
 ON occurs (Alarm, On,t0)
 IF context ()
 THEN do (On, Bathroom, t)
      when t is after t0
```

```
(Pattern 2)
 ON occurs (BathroomLights, Off,t0)
 IF context ()
 THEN do (Off, Bathroom, t)
      when t = t0 + 1s
```

```
(Pattern 3)
 ON occurs (Bathroom, On,t0)
 IF context ()
 THEN do (On, BathroomLights, t)
      when t = t0 + 2s
```

```
(Pattern 4)
 ON occurs (BathroomFan, Off,t0)
 IF context ()
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(Pattern 5)
 ON occurs (Shower, Off,t0)
 IF context (Bathroom humidity
             level (<,70%))
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(Pattern 6)
 ON occurs (Cabinet, Off,t0)
 IF context (DayOfWeek (<>,Tuesday
         Thursday, Friday))
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(...)
```

```
(Pattern 19)
 ON occurs (Shower, Off,t0)
 IF context (Bathroom humidity
             level (>,70%))
 THEN do (On, BathroomFan, t)
      when t = t0 + 4s
```

```
(Pattern 20)
 ON occurs (BathroomFan, On,t0)
 IF context ()

 THEN do (Off, BathroomFan, t)
      when t is after t0
```

4.5 Summary

The first approach (Pairwise Approach) developed within the LFPUBS focuses on discovering pairwise relations that define the frequent behaviours of the users.

On the one hand, a language ($\mathcal{L}_{LFPUBS}$), based on ECA rules that allows the system to represent patterns in a clear and unambiguous way has been developed. On the other hand, coupled with $\mathcal{L}_{LFPUBS}$, an algorithm ($\mathcal{A}_{LFPUBS}$) that discovers the patterns has been developed. $\mathcal{A}_{LFPUBS}$ consists of three steps that are used to discover frequent relations that relate users' frequent behaviours with their corresponding Time Relations and Conditions.

In the process of developing both modules, because the Learning Layer was an environment-independent layer, all of the components developed within this layer had to allow one to define and discover all of the possibilities that can come up in an IE.

Chapter 5

Learning Frequent Behaviours: the Action Map Approach

Patterns discovered by the Pairwise Approach represent the frequent behaviours of the users. They represent such behaviours by means of small and independent pieces of knowledge: small because each pattern only relates two actions and independent because the knowledge represented by different patterns is not related in any way.

Although pairwise relations provide interesting knowledge, it is clear that the behaviours of the user are better defined by considering all of the actions that the user frequently performs together. Thus, the second approach (*Action Map Approach*) developed in this research work attempts to discover the frequent behaviours of the users without any limitation in the number of actions involved in the pattern. Unlike the first approach, the Action Map Approach represents all of the actions involved in the behaviour together. The differences between the two approaches are clearly shown in Figure 5.1.

5.1 Introduction

This second approach can be thought of as an evolution of the Pairwise Approach. Its advantages are clear because it provides patterns (*Action Map*s) that represent complete behaviours of the users. Thus, it facilitates the understanding of such behaviours. It shares some aspects with the Pairwise Approach, although other aspects needed to be modified. Those common aspects will be outlined next, while those aspects that needed to be modified will be explained in more details in the subsequent sections.

Regarding the data coming from the Transformation Layer, this second approach is also focused on the actions performed by the user. Therefore, the actions to be considered will be those defined by A-type information, whereas C-type information will continue to be used to define conditions.

A similarity between both approaches is the general architecture of the Learning Layer. The architecture depicted in Figure 4.2 remains the same. This second approach for the LFPUBS is also based on a language ($\mathcal{L}_{LFPUBS}$) and an algorithm ($\mathcal{A}_{LFPUBS}$), although in both components the internal architecture varies when compared with the first approach.

In the next sections, the new versions of $\mathcal{L}_{LFPUBS}$ and $\mathcal{A}_{LFPUBS}$ are explained in more details.

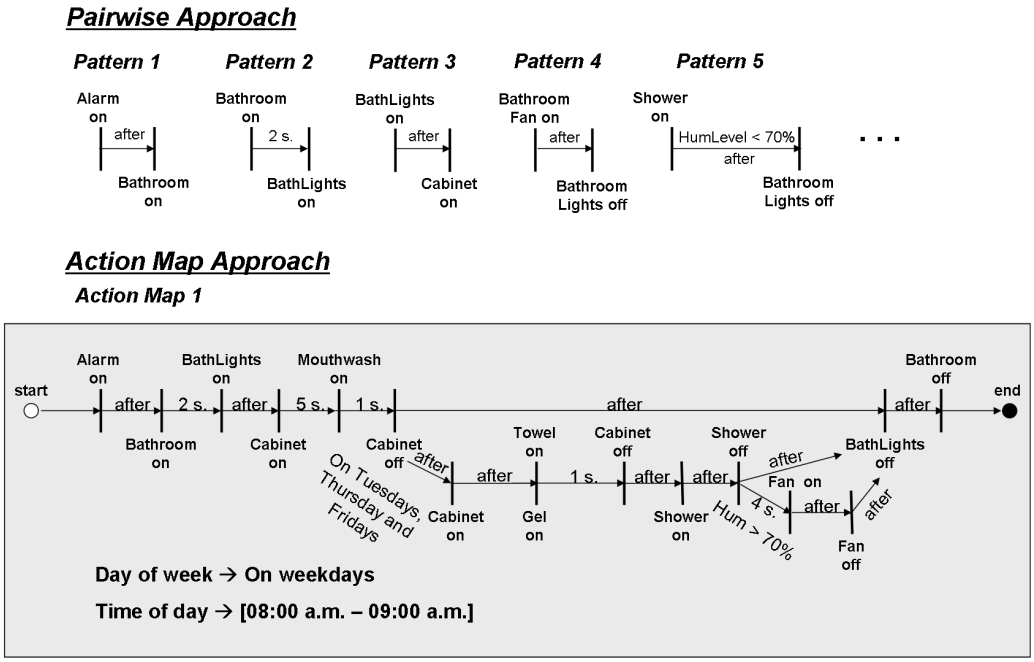

Figure 5.1: Differences between the outputs of both approaches.

5.2 Representing patterns with $\mathcal{L}_{LFPUBS}$

The goal of the new version of $\mathcal{L}_{LFPUBS}$ is to allow the system to represent Action Maps that describe the frequent behaviours of the users. Unlike the first approach in which a pattern related only two actions, in this new version a pattern represents a whole behaviour as well as the general conditions under which such a pattern occurs.

In that sense, a frequent behaviour is defined by means of an Action Map, which contains all of the specific relations between actions. In other words, an Action Map is created relating actions in pairs, and each of those relations is called an *Action Pattern*, which are defined by means of ECA rules. The complete Action Map is then created by linking different Action Patterns. However, some differences exist that must be considered.

5.2.1 Evolution of the $\mathcal{L}_{LFPUBS}$

In the process of creating Action Maps, it is necessary to define a *start* and an *end* point. For that, $\mathcal{L}_{LFPUBS}$ allows one to define Action Patterns in which the ON clause contains the action *start* and the THEN clause contains the action *end*. Considering the Figure 5.1, these starting and ending Action Patterns would be defined as follows:

```
(Action Pattern 1)
 ON occurs (start,--,t0)
 IF context ()
 THEN do (simple,(On,Alarm),t)
      when ---
```

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

Regarding the nature of the actions, some behaviours of the users indicate that they carry out the same actions but in an unordered way. Going back to Michael's scenario, the data shown in Appendix A indicate that he sometimes gets a towel first and then the gel, and other times he first gets the gel and then the towel. In order to facilitate an understanding of those behaviours, $\mathcal{L}_{LFPUBS}$ allows the system to define *unordered subsets of actions*. In Michael's case, the Action Pattern that involves such an unordered subset would be defined as follows:

```
(Action Pattern n)
 ON occurs (unordered,((Towel,On)&(Gel,On)),t0)
 IF context ()
 THEN do (simple,(Off, Cabinet),t)
      when t = t0 + 1s
```

The last difference to be considered is related to the conditions. In the Pairwise Approach, it was clear that the IF clause defined under what conditions the relation defined by the pattern was true. In the Action Map Approach, because a pattern represents much more information, different types of conditions are necessary.

On one hand, there are Specific Conditions for each Action Pattern that define when the relation defined by such an Action Pattern is true. Such conditions can be considered as equivalent to the conditions defined in the Pairwise Approach, and they are represented in the IF clause.

On the other hand, apart from the conditions for specific relations defined by Action Patterns, the complete behaviour represented by the Action Map must also be contextualised too, using either context or calendar information. In that sense, $\mathcal{L}_{LFPUBS}$ allows one to use both types of information; using the syntax to define General Conditions equal to the syntax used to define Specific Conditions. Consider Michael's morning habit. It occurs on weekdays between 8 a.m. and 9 a.m., so the General Condition would be defined as follows:

```
(General Condition)

   context (DayOfWeek (=,Monday,Tuesday,Wednesday,Thursday,Friday)) &
   context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<:09:00:00))
```

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

5.3 Learning patterns with $\mathcal{A}_{LFPUBS}$

It is clear that the Action Map Approach has some advantages with respect to the Pairwise Approach. It discovers and represents the behaviours of the users as a unique map of action. In that sense, $\mathcal{L}_{LFPUBS}$ has already been modified to allow the representation of those new patterns (See Section 5.2). The component that needs more modifications is $\mathcal{A}_{LFPUBS}$ because discovering complete sets of frequent behaviours demands that new steps be added and existing ones be modified.

The architecture of $\mathcal{A}_{LFPUBS}$ for the Action Map Approach is depicted in Figure 5.2.

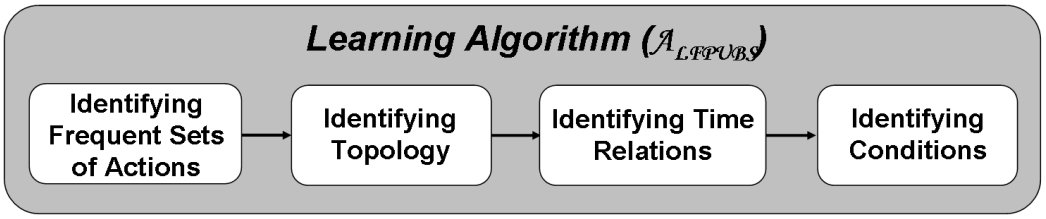

Figure 5.2: Steps to be performed by the learning algorithm.

The same idea represented by means of a pseudo-algorithm is given below:

$\mathcal{A}_{LFPUBS}$ **Algorithm**
Identify Frequent Sets of Actions (See Section 5.3.1)
For each Frequent Set
Identify the Topology (See Section 5.3.2)
For each Action Pattern
Identify possible quantitative Time Relations (See Section 5.3.3)
Identify Specific Conditions using context/calendar information (See Section 5.3.4)
Identify General Conditions (See Section 5.3.4)

Below, each one of the steps is explained in more detail. Some of the steps are new, whereas others are similar to those explained in the Pairwise Approach (Chapter 4).

5.3.1 Identifying Frequent Sets of Actions

The objective of this step is to discover the sets of actions that frequently occur together (Frequent Sets). The idea is the same as the step labelled 'Identifying Frequent Relations' of the Pairwise Approach, with only two differences:

- The number of actions involved in a Frequent Set is not limited.
- Frequent Sets of actions discovered at this stage do not define any order of the actions within the Action Map.

This task of identifying Frequent Sets is actually divided into three sub-tasks. As shown in Figure 5.3, LFPUBS allows the definition of the parameters considered in such sub-tasks.

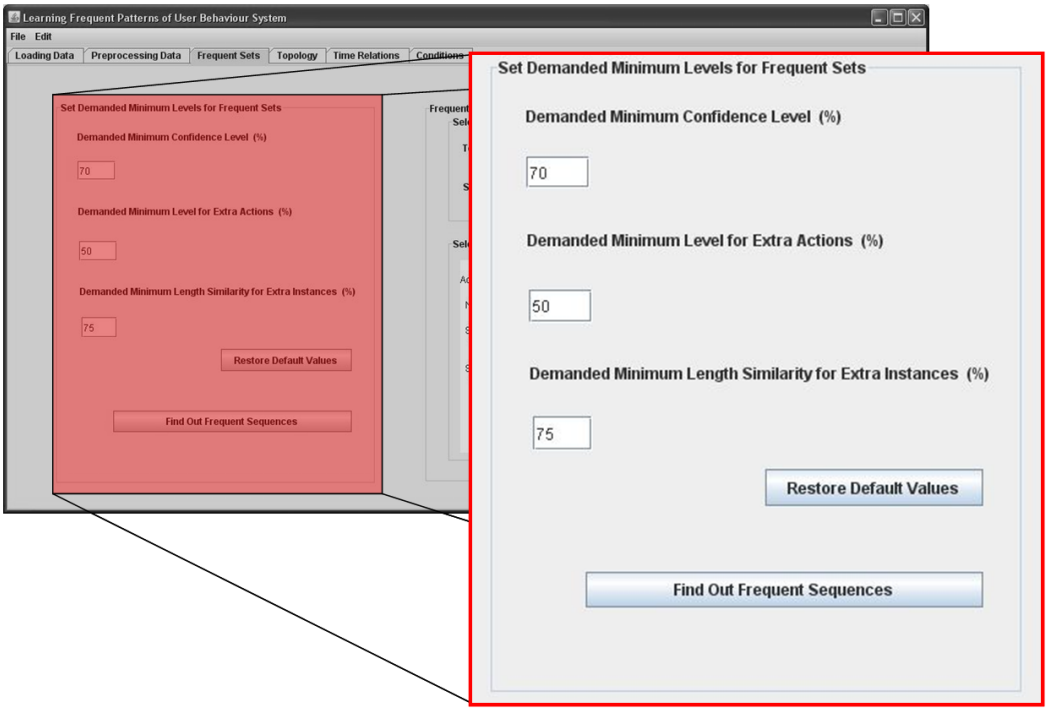

Figure 5.3: Defining the basic minimum confidence, extra actions' minimum confidence and minimum similarity levels.

Identifying Basic Frequent Sets of Actions

The underlying idea of this first sub-task is both simple and efficient. Defining a demanded minimum level (*minimum confidence level*), it discovers all those sets of actions that occur more times than the minimum level. These sets of actions are treated as *Basic Frequent Sets*. The set of Sequences in which the Basic Frequent Set is present is also identified. To discover Basic Frequent Sets in large amounts of data, the Apriori algorithm [Agr95] was used.

The sets discovered in such a way are frequent, but because of IEs' complexity, considering only those Basic Frequent Sets does not take into account all of the particularities that occurs. Because of the large amount of data and the defined minimum confidence level, some actions and some Sequences that are not included in a Basic Frequent Set could add interesting information.

Thus, taking the Basic Frequent Sets discovered in this sub-task as a starting point, it is necessary to go a step further and extend them to get sets of actions that better define the frequent behaviours of users.

Identifying extra actions for Frequent Sets

Once the Basic Frequent Sets have been discovered, an aspect to consider for each Basic Frequent Set is whether there is any action that was not discovered as frequent taking into

account all the Sequences, but it is frequent enough when considering only those Sequences where the Basic Frequent Set occurs. In other words, the goal of this task is to discover whether there is any *Extra Action* that frequently happens in those particular Sequences.

As in the previous sub-task, a minimum confidence level must be defined, and the process to discover such Extra Actions is the same. The only difference is that instead of all of the Sequences, only those Sequences in which the Basic Frequent Set is present are considered.

The Frequent Sets are created by adding these Extra Actions to the respective Basic Frequent Sets.

Identifying occurrences that support Frequent Sets

For the following steps of $\mathcal{A}_{LFPUBS}$ (e.g., 'Identifying Topology'), it is essential to know in which Sequences a Frequent Set is present. Although initially considering only those Sequences identified in the task of 'Identifying Basic Frequent Sets' seems like enough, other Sequences could also provide interesting information. There could be Sequences in which most of the actions of a Basic Frequent Set are present, but as they do not contain all of the actions, they are not considered.

Being aware that small deviations could mean a loss of information, a strategy to reduce the impact of such deviations has been designed. It is based on a parameter (*similarity level*) that indicates (as a percentage) the minimum number of actions of a Basic Frequent Set to be included in a Sequence in order to consider such a Sequence interesting. Thus, not only those Sequences that have all of the actions (similarity level = 100%) of the Basic Frequent Set would be considered interesting, but also those Sequences in which the similarity level is over the demanded level.

Identifying Frequent Sets of actions in Michael's scenario

Let us consider again the data depicted in Appendix A. The goal of this first step is to discover which sets of actions Michael frequently carries out together.

The first step is to identify the Basic Frequent Sets. To do this, it is necessary to define a minimum confidence level (let us consider 70%). Demanding a confidence level of 70% indicates that a set of actions must occur in at least 70% of the total Sequences. To discover such sets of actions, the Apriori algorithm is used. It is based on the idea of discovering Frequent Sets through evolving frequent subsets. In Michael's case, for a confidence level of 70%, the only set of actions that is frequent is made up of the following actions:

'Alarm On'; 'Bathroom On'; 'Bathroom Off'; 'BathroomLights On'; 'BathroomLights Off'; 'Cabinet On'; 'Cabinet Off'; 'Mouthwash On'

This set of actions is present in eight out of ten Sequences (confidence level: 80%); all but Sequence 9 and Sequence 10.

Once the Basic Frequent Sets are discovered, Extra Actions and occurrences that support the Basic Frequent Sets must be identified. As stated above, the process of discovering Extra Actions the process is the same as identifying Basic Frequent Sets but only those Sequence in

which the Basic Frequent Set is present are considered. Thus, in Michael's case, only those eight Sequences (Sequence 1, Sequence 2, Sequence 3, Sequence 4, Sequence 5, Sequence 6, Sequence 7 and Sequence 8) are considered. The confidence level of the rest of the actions is:

'Towel On' (Conf: 5/8 (62.5%))

'Gel On' (Conf: 5/8 (62.5%))

'Shower On' (Conf: 5/8 (62.5%))

'Shower Off' (Conf: 5/8 (62.5%))

'BathroomFan On' (Conf: 4/8 (50%))

'BathroomFan Off' (Conf: 4/8 (50%))

'Tap On' (Conf: 1/8 (12.5%))

'Tap Off' (Conf: 1/8 (12.5%))

In this case, let us consider a minimum confidence level of 50%. The set of actions that are frequent and therefore considered Extra Actions are:

'Towel On'; 'Gel On'; 'Shower On'; 'Shower Off'; 'BathroomFan On', 'BathroomFan Off'

Adding the Extra Actions to those actions involved in the Basic Frequent Set, the Frequent Set is created.

Frequent Set 1

'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', 'BathroomFan Off'

Once Frequent Sets have been discovered, it is necessary to identify those Sequences in which the Frequent Sets are present. This is essential because the information (e.g., the order of the actions or the Time Distances) contained in such Sequences is essential for the next steps of the $\mathcal{A}_{LFPUBS}$. It is clear that the eight Sequences in which the Basic Frequent Set is present must be considered, but apart from those Sequences, there could be other Sequences (e.g., Sequence 9 or Sequence 10) that do not contain *all* of the actions of the Basic Frequent Set but do contain most of them and therefore could contain interesting information. In order to discover such extra Sequences a similarity level is defined. Let us consider a similarity level of 75% in this case. Thus, considering that the Basic Frequent Set involves eight actions, all of those Sequences that involve at least six out of those eight actions are considered interesting.

In Michael's case, Sequence 10 involves all of the actions of the Basic Frequent Set except the action 'Mouthwash On', so its similarity level is of 87.5% (7 of 8). The information contained in such a Sequence will also be used in later steps. In comparison, Sequence 9 only involves 5 actions of the Basic Frequent Set (62.5%); therefore the information it contains will not be considered.

Thus, the final output of this first step in Michael's scenario is (the same Frequent Set represented by the GUI can be seen in Figure 5.4):

Frequent Set 1

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'; 'BathroomFan Off';

Sequences: Sequence 1; Sequence 2; Sequence 3; Sequence 4; Sequence 5; Sequence 6; Sequence 7; Sequence 8; Sequence 10;

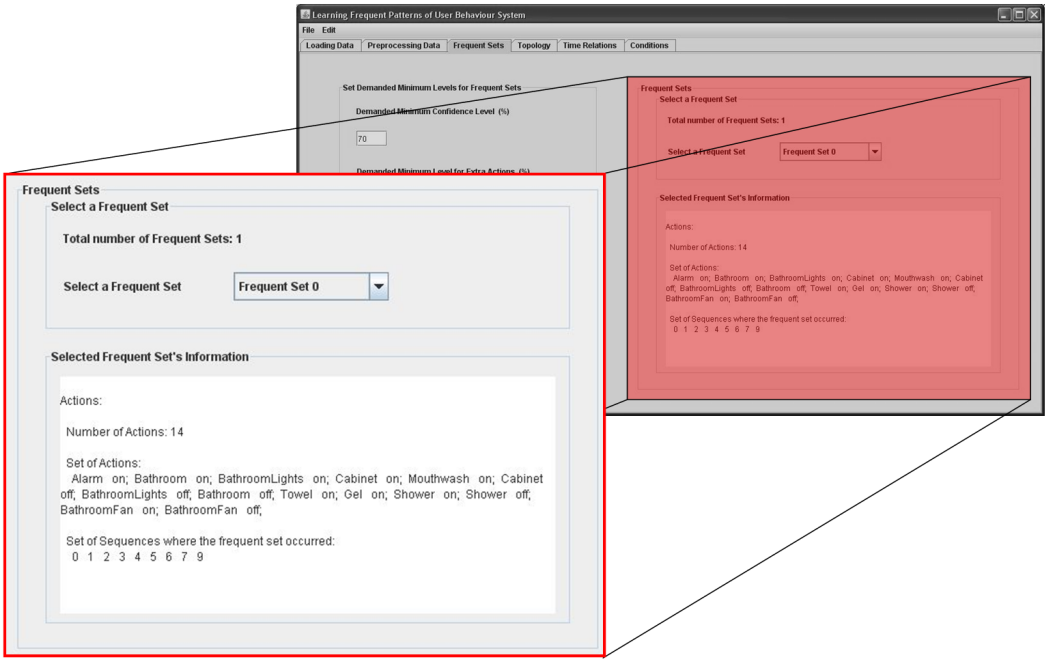

Figure 5.4: Information about the discovered Frequent Sets.

5.3.2 Identifying Topology

The step 'Identifying Frequent Sets' discovers which sets of actions frequently occur together. In order to properly model the user's behaviours defined by such sets of actions, it is necessary to define the order of such actions. That is the goal of this step, to discover the frequent order (defined as *Topology*) of the actions in the behaviour of the user. The actions involved in the behaviour and the Sequences that contain the information are provided by the previous step, 'Identifying Frequent Sets'.

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. Even so, as stated in Chapter 2, 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. In that sense, one of the closest domains is the area of Workflow Mining [Wei01; Aal04; Wen07] in which process models

are discovered from event logs. Both domains are equal, with the only difference being that instead of event logs, $\mathcal{A}_{LFPUBS}$ considers the actions of the user.

Even so, because of the nature of IEs, some modifications to the Workflow Mining algorithm must be considered. Next, the different aspects that need to be considered are explained.

Basic Methodology

The only goal of this first step is to represent the input data in an Action Map, keeping all of the actions as well as all of the relationships. Although this step does not discover anything, at this point, it is important to highlight that 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.

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.

Initially, identifying repetitive actions and creating different instantiations of them seems like it could make it difficult to understand the behaviour. Quite the contrary, creating different instantiations of the same action facilitates the understanding of the behaviour because they simplify the complexity of the Topology.

Considering the possible existence of more than one instance of the same action, a methodology to automatically discover such situations has been developed. Its objective is, on the one hand, to decide how many instances of each action are necessary, and on the other hand, to define the nature of particular actions in different Sequences.

This methodology is based on the idea that the meaning, and by extension, the nature of an action is mainly defined by the previous and next actions. In other words, the occurrence of an action is related to the previous and next actions because the set of those actions will probably follow a specific objective. Thus, the nature of different actions is defined by creating groups of actions that take into account the similarities among the previous and next actions of their occurrences.

In that sense, the first decision to make is to choose the number of previous and next actions to be considered. The LFPUBS allows the user of the system to select up to four actions to be considered. The greater the number of actions to be considered, the more accurate the definition of the nature will be. Of course, it can have a negative impact in the runtime of the process.

Once the number of previous and next actions to consider has been decided, the LFPUBS includes two different techniques to create groups:

- Manually define the number of groups or clusters to create, 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 EM algorithm [Hog05].

From this point on, once different instances of the same action are created, they are treated as different actions, and each particular occurrence is labelled by taking into account the group it belongs to.

All the parameters mentioned above, such as the number of previous/next actions to consider or technique to create groups, can be defined using the GUI (see Figure 5.5).

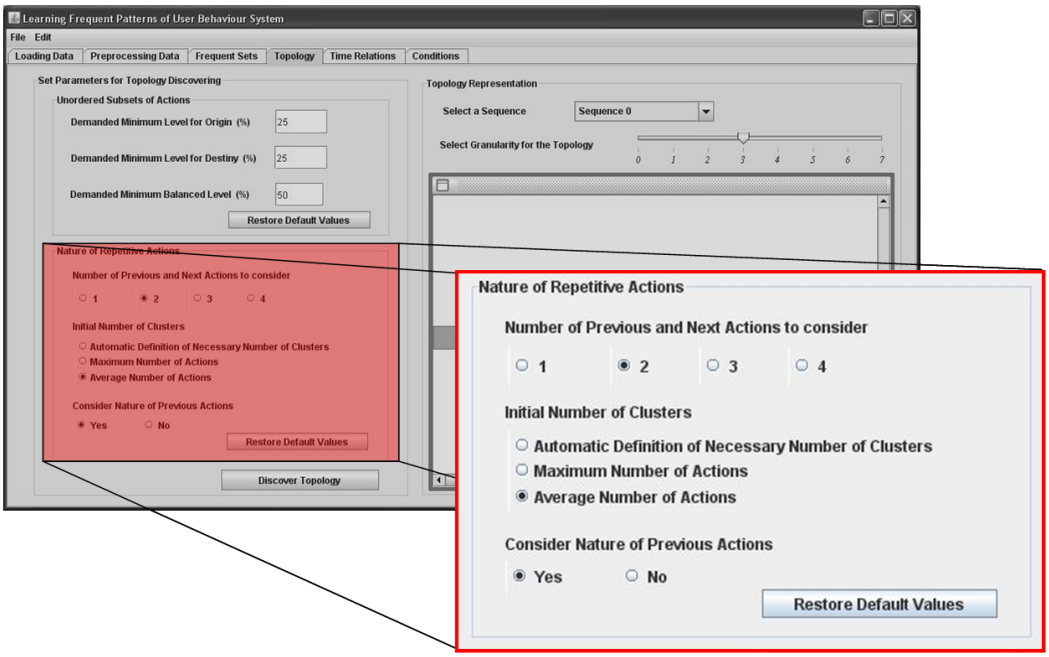

Figure 5.5: Defining the parameters to identify repetitive actions.

Unordered Subsets of Actions

Different works of the Workflow mining area also propose the idea of parallel subsets of actions. In that sense, the LFPUBS is aimed at detecting those actions that occur at a specific moment that are represented by means of a timestamp. Simple actions do not have duration, so the occurrence of parallel actions is impossible. Even so, the idea of parallel actions can be applied to discover *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, sometimes he 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 sets of actions shows bidirectional relationships between such actions.

To decide whether a bidirectional relationships (let us say between A and B) must be considered as an unordered set of actions, the LFPUBS includes a set of parameters:

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

$$(A \rightarrow B)/(OccurrencesA) > Minimum\ Level\ for\ Origin. \tag{5.1}$$

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

$$(B \rightarrow A)/(OccurrencesB) > Minimum\ Level\ for\ Destiny. \tag{5.2}$$

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

$$((A \rightarrow B)/(B \rightarrow A))\&((B \rightarrow A)/(A \rightarrow B)) > Minimum\ Balanced\ Level. \tag{5.3}$$

The GUI allows the definition of such parameters (see Figure 5.6).

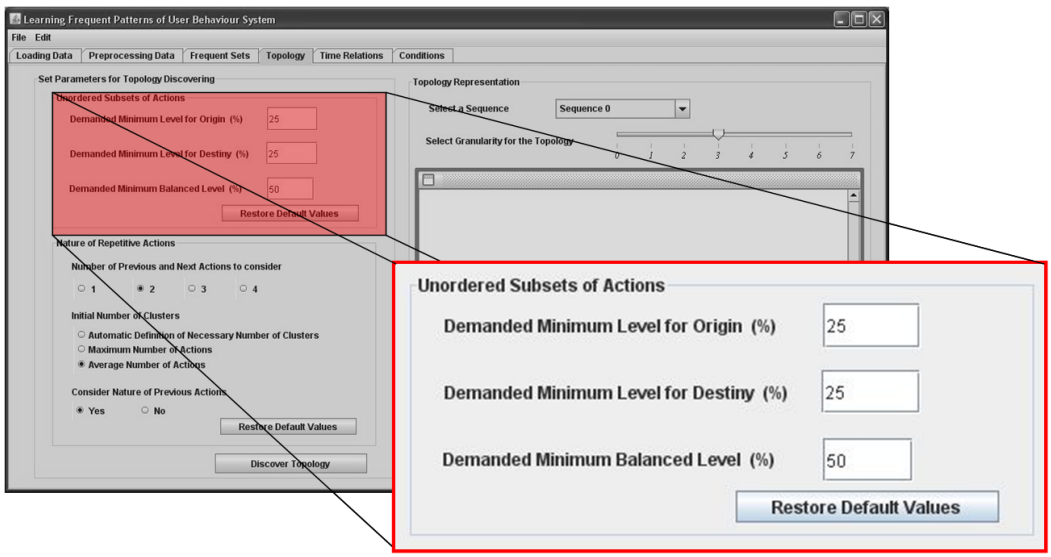

Figure 5.6: Defining the parameters to identify unordered subsets of actions.

Thus, all those bidirectional relationships that fulfil all the demanded levels are considered as unordered subsets of actions; each subset is treated as a unique action.

Granularity and Allowed Maximum Granularity

Once the behaviours of the user have been discovered and represented, the LFPUBS allows the user of the system to select the granularity of the Action Map. In this way, the user of the system can choose the complexity of the Action Map he/she wants the system to show. 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.

Selection of the granularity allows the user of the system to see Action Maps with different complexities, depending on his/her interests. It is clear that selecting granularities without any limit can result in illogical Action Maps in which actions are not related to any other action or Action Maps in which there is no path from *start* to *end*.

To avoid illogical representations, the LFPUBS includes the heuristic 'Uniform-cost search' [Rus03], 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.

Identifying Topology in Michael's scenario

Considering Michael's data (see Appendix A), only one Frequent Set was discovered by the 'Identifying Frequent Sets of Actions' step. The objective of this step is to identify in what order Michael usually carries out such a set of actions. First, all of the different actions as well as all of the relationships between actions are represented by the 'Basic Methodology'. The output is depicted in Figure 5.7.

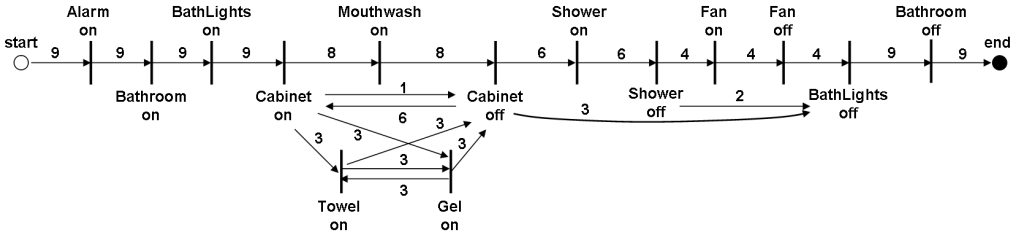

Figure 5.7: The basic representation of Michael's behaviour.

As stated above, the Basic Methodology parses the Frequent Sets into $\mathcal{L}_{LFPUBS}$. In that sense, the behaviour represented at this point covers all of the possible relationships defined by the data. An example of the status of an Action Pattern at this stage is shown below:

```
(Action Pattern 2)
 ON occurs (simple,(Alarm,On),t0)
 IF context ()
 THEN do (simple,(On,Bathroom),t) when ---
```

Once the initial representation is done, the next step is to define the nature of the repetitive actions. The only actions that are repeated in a Sequence are the actions 'Cabinet On' and 'Cabinet Off'. To discover the nature of each one of the occurrences of those actions, the previous and next actions of each occurrence must be analysed. Let us consider the action 'Cabinet On' and two previous and two next actions to explain the process that discovers the nature of each occurrence. In total, there are 15 occurrences of the action 'Cabinet On' and the previous and next actions of each one of them are as follows:

	Previous 2	Previous 1	Next 1	Next 2
O.1 (S1)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.2 (S1)	Mouthwash On	Cabinet Off	Towel On	Gel Off
O.3 (S2)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.4 (S3)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.5 (S3)	Mouthwash On	Cabinet Off	Gel On	Towel Off
O.6 (S4)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.7 (S5)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.8 (S5)	Mouthwash On	Cabinet Off	Gel On	Towel Off
O.9 (S6)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.10 (S6)	Mouthwash On	Cabinet Off	Towel On	Gel Off
O.11 (S7)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.12 (S8)	Bathroom On	Lights On	Mouthwash On	Cabinet Off
O.13 (S8)	Mouthwash On	Cabinet Off	Towel On	Gel Off
O.14 (S10)	Bathroom On	Lights On	Cabinet Off	Cabinet On
O.15 (S10)	Cabinet On	Cabinet Off	Gel On	Towel Off

Let us consider that the number of possible different groups is defined by the average of the occurrences in a Sequence (two in this case). Next, a clustering algorithm is used in order to decide which group each one of the occurrences belongs to. In this case, because of the similarities between different occurrences, it is clear that there is a group which covers the Occurrences 1, 3, 4, 6, 7, 9, 11, 12 and 14, whereas the second group covers the Occurrences 2, 5, 8, 10, 13 and 15. Once groups are defined, a particular instantiation of that action is created for each group. Thus, in Michael's case, there will be two different 'Cabinet On' actions ('Cabinet(1) On', 'Cabinet(2) On'). From this point on, each of the particular occurrences is related to either 'Cabinet(1) On' or 'Cabinet(2) On' actions, depending on the group they belong to. The same thing happens with the action 'Cabinet Off'.

Once the nature of repetitive actions is defined, the representation could vary. In Michael's case, the representation of his behaviour, taking into account repetitive actions, can be seen in Figure 5.8:

Regarding possible unordered subsets of actions, only one bidirectional relationship exists that involves the actions 'Towel On' and 'Gel On'. Let us consider the following minimum values for the parameters:

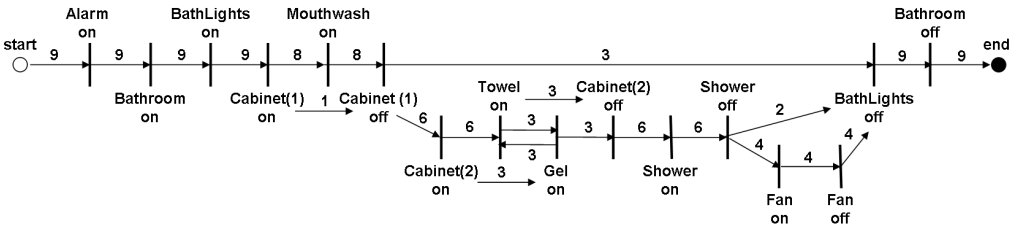

Figure 5.8: Michael's behaviour with repetitive actions.

Minimum Level for Origin: 25%

Minimum Level for Destiny: 25%

Minimum Balanced Level: 50%

The values of the relationship between 'Towel On' and 'Gel On' are:

Minimum Level for Origin: 50% (3/6)

Minimum Level for Destiny: 50% (3/6)

Minimum Balanced Level: 100% (3/3) & 100% (3/3)

Taking into account these values, it is clear that such a bidirectional relationship defines an unordered subset of actions that includes the actions 'Towel On' and 'Gel On'. In that sense, as mentioned in Section 5.2.1, $\mathcal{L}_{LFPUBS}$ allows the system to define an unordered subset of actions; for example, the Action Pattern that relates such an unordered subset with the action 'Cabinet Off' is represented as:

```
(Action Pattern n)
 ON occurs (unordered,((Towel,On)&(Gel,On)),t0)
 IF context ()
 THEN do (simple,(Off, Cabinet),t)
      when t = t0 + 1s
```

The final topology of Michael's behaviour, considering unordered subsets of actions, is depicted in Figure 5.9.

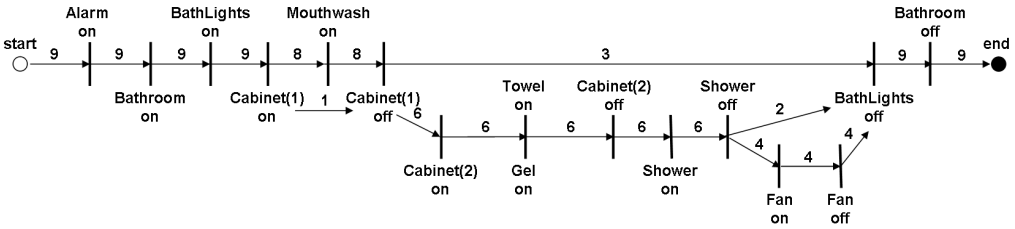

Figure 5.9: Michael's behaviour with unordered subsets of actions.

Finally, it is necessary to identify the Allowed Maximum Granularity that ensures that there will be a path from *start* to *end*. Let us assume that the 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 5.10, the representation of Michael's behaviour would lose its sense, making it difficult to understand it logically.

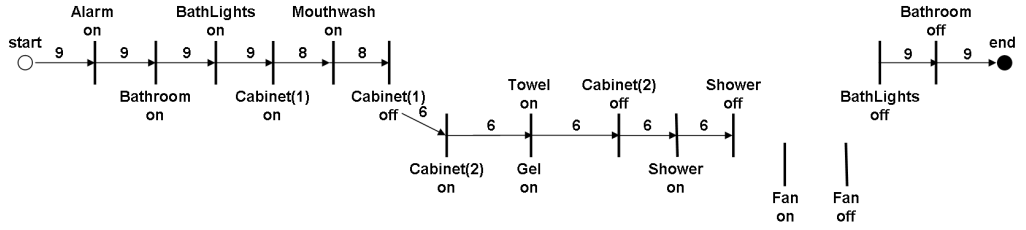

Figure 5.10: Michael's behaviour without considering the Allowed Maximum Granularity parameter.

Considering Michael's case, the need of identifying the Allowed Maximum Granularity is clear. In Michael's case, using the 'Uniform-cost search' heuristic, the LFPUBS discovered that the Allowed Maximum Granularity is of '4'. Thus, the user of the LFPUBS will not be able to select a granularity level higher than '4'. Michael's behaviour representation considering the Allowed Maximum Granularity is depicted in Figure 5.11.

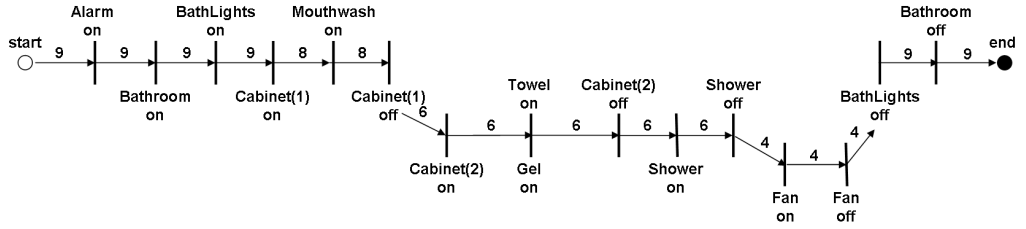

Figure 5.11: Michael's behaviour considering the Allowed Maximum Granularity parameter.

Once Topologies of different Frequent Sets have been identified, they are represented by means of $\mathcal{L}_{LFPUBS}$ (see Figure 5.12 for its graphical representation). Time Relations as well as particular and general Conditions are not yet discovered at this stage, but the relationships between actions (defined by ON and THEN clauses) can be defined. Michael's behaviour can be represented by means of Action Patterns as follows:

```
(Action Map 1)

(Action Pattern 0)                        (Action Pattern 1)
 ON occurs (start,--,t0)                   ON occurs (simple,(Alarm,On),t0)
 IF context ()                             IF context ()
 THEN do (simple,(On,Alarm),t)             THEN do (simple,(On,Bathroom),t)
      when ---                                  when ---

(Action Pattern 2)                        (Action Pattern 3)
 ON occurs (simple,                        ON occurs (simple,
           (Bathroom,On),t0)                         (BathroomLights,On),t0)
 IF context ()                             IF context ()
 THEN do (simple,(On,BathroomLights),t)    THEN do (simple,(On,Cabinet),t)
      when ---                                  when ---

 (...)

(Action Pattern 14)                       (Action Pattern 15)
 ON occurs (simple,                        ON occurs (simple,
           (Shower,Off),t0)                          (BathroomFan,On),t0)
 IF context ()                             IF context ()
 THEN do (simple,(On,BathroomFan),t)       THEN do (simple,(Off,BathroomFan),t)
      when ---                                  when ---

(Action Pattern 16)                       (Action Pattern 17)
 ON occurs (simple,                        ON occurs (simple,
           (BathroomFan,Off),t0)                     (BathroomLights,Off),t0)
 IF context ()                             IF context ()
 THEN do (simple,                          THEN do (simple,
         (Off,BathroomLights),t)                   (Off,Bathroom),t)
      when ---                                  when ---

(Action Pattern 18)
 ON occurs (simple,
           (Bathroom,Off),t0)
 IF context ()
 THEN do (--,end,t)
      when ---
```

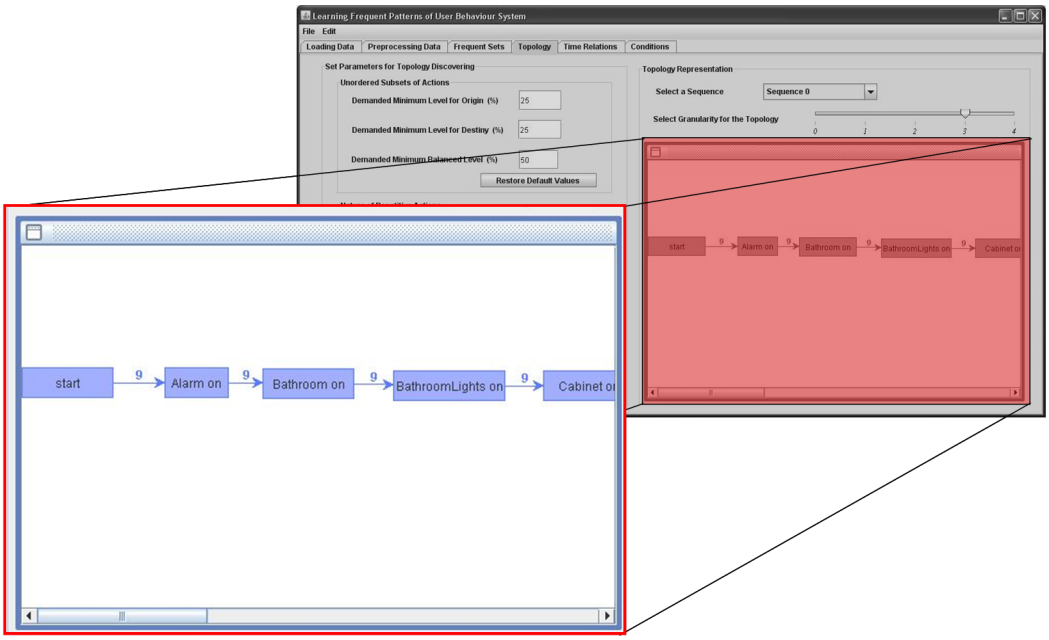

Figure 5.12: Graphical representation of the topology.

5.3.3 Identifying Time Relations

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

Similarities between this step and the same step of the Pairwise Approach are evident. In both cases, the problem and the objective are the same. Therefore the solution applied in the Pairwise Approach can be directly used in this case too. There are only two differences:

- It does not make sense to define a Time Relation in those Action Patterns in which either *start* or *end* actions are involved.
- In cases of unordered subsets of actions, the timestamp of either the first or the last action is considered.

The Action Map Approach includes both algorithms ('Basic Algorithm' and 'EM Algorithm') explained in Section 4.4.2.

Identifying Time Relations in Michael's scenario

Considering the Action Patterns defined by the Topology discovered in the previous step, the objective of this step is to identify frequent quantitative Time Relations. Let us consider 'Action Pattern 14' which shows how Michael usually turns on the fan after taking a shower.

The same example is considered in Section 4.4.2. The process is the same as the one explained in that section, and the results are the same also. Thus, 'Action Pattern 14' would be represented as follows:

```
(Action Pattern 14)
 ON occurs (simple,(Shower,Off),t0)
 IF context ()
 THEN do (simple,(On,BathroomFan),t)
      when t = t0 + 4s
```

Once possible quantitative Time Relations have been discovered using the 'Basic Algorithm', Michael's behaviour would be represented as (note that their graphical representation is the same as in the Pairwise Approach, which can be seen in Figure 4.9):

```
(Action Map 1)

(Action Pattern 0)                      (Action Pattern 1)
 ON occurs (start,--,t0)                 ON occurs (simple,(Alarm,On),t0)
 IF context ()                           IF context ()
 THEN do (simple,(On,Alarm),t)           THEN do (simple,(On,Bathroom),t)
      when ---                                 when t is after t0

(Action Pattern 2)                      (Action Pattern 3)
 ON occurs (simple,                      ON occurs (simple,
           (Bathroom,On),t0)                       (BathroomLights,On),t0)
 IF context ()                           IF context ()
 THEN do (simple,(On,BathroomLights),t) THEN do (simple,(On,Cabinet),t)
      when t = t0 + 2s                         when t is after t0

(...)

(Action Pattern 14)                     (Action Pattern 15)
 ON occurs (simple,                      ON occurs (simple,
           (Shower,Off),t0)                        (BathroomFan,On),t0)
 IF context ()                           IF context ()
 THEN do (simple,(On,BathroomFan),t)     THEN do (simple,(Off,BathroomFan),t)
      when t = t0 + 4s                         when t is after t0
```

```
(Action Pattern 16)
 ON occurs (simple,
           (BathroomFan,Off),t0)
 IF context ()
 THEN do (simple,
         (Off,BathroomLights),t)
      when t is after t0
```

```
(Action Pattern 17)
 ON occurs (simple,
           (BathroomLights,Off),t0)
 IF context ()
 THEN do (simple,
          (Off,Bathroom),t)
       when t = t0 + 1s
```

```
(Action Pattern 18)
 ON occurs (simple,
           (Bathroom,Off),t0)
 IF context ()
 THEN do (--,end,t)
      when ---
```

5.3.4 Identifying Conditions

Once Topology and Time Relations have been identified, Action Maps represent user's behaviours in a comprehensible way. Even so, 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.

On one hand, it is clear that all of the relations represented in an Action Map are supported by a number of occurrences. In that sense, a particularity to treat could be the case in which an action is followed by two (or more) different actions. These situations indicate that after an action, the user sometimes carries out one action and other times he/she carries out some other action. In an Action Map, these situations are easily identified because these situations are represented as splitting points from which more than one relation is created. In those cases, it is necessary to identify under what conditions each of those relations is true.

On the other hand, it is necessary to define the general context in which an Action Map occurs. General Conditions refer to calendar and context information that allow the user of the system to understand under what conditions the whole Action Map occurs.

Next, both processes of identifying Specific and General Conditions are explained in more detail.

Identifying Specific Conditions for Action Patterns

As stated above, different occurrences of an action can be followed by different actions defining different relations. These situations can be easily identified by analysing the different Action Patterns. Being aware that each Action Pattern defines a relation, if a specific action is the 'Event Definition' portion (defined by the ON clause) in more than one Action Pattern, such an action is followed by different actions (defined by the THEN clauses of such Action

Patterns). Graphically, these situations are easy to identify because they are splitting points from which more than one path starts.

Once such situations are identified, the process of identifying conditions for each relation is the same as the 'Identifying Conditions' step of the Pairwise Approach. The only difference is that instead of creating only two tables, in this case, as many tables as there are different relations must be created, i.e., one table for each possible relation. Then, conditions that separate different tables are discovered using the classification algorithm JRip [Wit05].

The conditions discovered at this stage represent the conditions under which a relation defined by an Action Pattern is true. Thus, such conditions are used to define the IF clause of different Action Patterns.

Identifying General Conditions for Action Maps

The concept of General Conditions is a new concept that did not exist in the Pairwise Approach. It refers to those context or calendar conditions that define the occurrences of different Action Maps. In this research work, 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, a simple algorithm has been developed. It is based on identifying the period of time, both in terms of 'Time of Day' and 'Day of Week', that covers all of the occurrences of such an Action Map. To do this, the timestamps of the first and the last actions of each occurrence are retrieved, and then starting from an empty value for both parameters, they are adapted in order to cover all of the occurrences. The adaptation technique is based on a simple rule that enhances the limits of the range to cover all of the occurrences.

This first version provides quite simple and 'excessive' conditions. For future version, some improvements are already being considered.

Identifying Conditions in Michael's Scenario

Considering the Action Map that represents Michael's morning behaviour, the process of discovering Specific and General Conditions is explained next.

The need for Specific Conditions is clear in two situations. On one hand, the 'Cabinet(1) Off' action can be followed by either 'Cabinet(2) On' or 'BathroomLights Off' actions. On the other hand, the action of 'Shower Off' can be followed by either 'BathroomFan On' or 'BathroomLights Off' actions.

Let us consider the Specific Conditions needed for the relations of 'Shower Off' with the actions 'BathroomFan On' and 'BathroomLights Off'. In this case, as there are two possible relations two tables are created - one for each possible relation. Thus, context and calendar information collected when occurrences of 'Shower Off' were followed by the action 'BathroomFan On' (Sequence 1, Sequence 5, Sequence 6 and Sequence 8) are recorded in the table '*ShowerOff-BathroomFanOn*'. In the same way, information collected when occurrences

of 'Shower Off' were followed by the action 'BathroomLights Off' are recorded in the table '*ShowerOff-BathroomLightsOff*'. Both tables are depicted in Figure 5.13.

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

Figure 5.13: 'Shower Off - BathroomFan On' and 'Shower Off - BathroomLights Off' tables with calendar and context information

There are different ways of separating both tables because of the small number of Sequences considered in Michael's case. Even so, 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 $\mathcal{A}_{LFPUBS}$ is:

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

In contrast, for the relation 'Shower Off'-'BathroomLights Off', the specific condition is:

```
IF context (Bathroom relative humidity level (<,70%))
```

The other situation that needs Specific Conditions also contains two relations, so two tables are created in this case as well; one for the relation 'Cabinet(1) Off'-'Cabinet(2) On' and another one for the relation 'Cabinet(1) Off'-'BathroomLights Off'. In this case, the condition that better separates both tables is the calendar information 'Day of Week', creating the following condition for the relation 'Cabinet(1) Off'-'Cabinet(2) On':

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

Regarding General Conditions, both 'Time of Day' and 'Day of Week' information for the first and last actions of each Sequence are collected. Below, this information is shown for all Sequences collected in Michael's case.

	First Action's 'Time of Day'	First Action's 'Day of Week'	Last Action's 'Time of Day'	Last Action's 'Day of Week'
Sequence 1	08:02:12	Monday	08:33:41	Monday
Sequence 2	08:10:50	Tuesday	08:27:04	Tuesday
Sequence 3	08:06:19	Wednesday	08:36:10	Wednesday
Sequence 4	08:16:39	Thursday	08:36:02	Thursday
Sequence 5	08:05:40	Friday	08:39:12	Friday
Sequence 6	08:03:47	Monday	08:47:06	Monday
Sequence 7	08:10:39	Tuesday	08:24:46	Tuesday
Sequence 8	08:03:47	Wednesday	08:25:57	Wednesday
Sequence 10	08:10:27	Friday	08:36:38	Friday

Considering the attribute 'Time of Day', the LFPUBS starts from the information for Sequence 1 and creates a range of time. To do this, it takes the starting and ending time of the first Sequence's actions and creates a range, rounding off these values respectively. To round off, periods of time 15 minutes in length have been considered. Considering Sequence 1, for Michael's case, the created range is:

Time of Day: [08:00:00 - 08:45:00]

Next, the LFPUBS verifies, one by one, whether the information for the rest of the Sequences is covered by that range. If it is, the range is not modified. However, if the range does not cover any of the Sequences, the range is enhanced to cover those Sequences as well. In Michael's case, all of the Sequences are covered by the given range, except Sequence 6, whose last action's 'Time of Day' (08:47:06) is not covered. In this case, the range is enhanced up to 09:00:00, creating new general conditions for 'Time of Day':

Time of Day: [08:00:00 - 09:00:00]

Once all Sequences are covered by the range, it is used to define part of the General Condition. In Michael's case, the Condition addressing the attribute 'Time of Day' is:

```
context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<:09:00:00))
```

Regarding the 'Day of Week' attribute, the process of identifying such a condition is the same. The only difference is that instead of enhancing the range, all different days during which a Sequence occurred are added to the list. In Michael's case, there is at least one Sequence in the following days of the week:

Day of Week: Monday; Tuesday; Wednesday; Thursday; Friday;

Finally, this condition is added to the previous conditions to create General Conditions. Thus, in Michael's case the General Conditions is:

```
(General Condition)

   context (DayOfWeek (=,Monday,Tuesday,Wednesday,Thursday,Friday)) &
   context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<:09:00:00))
```

Graphical representation of different conditions can be seen in Figure 5.14.

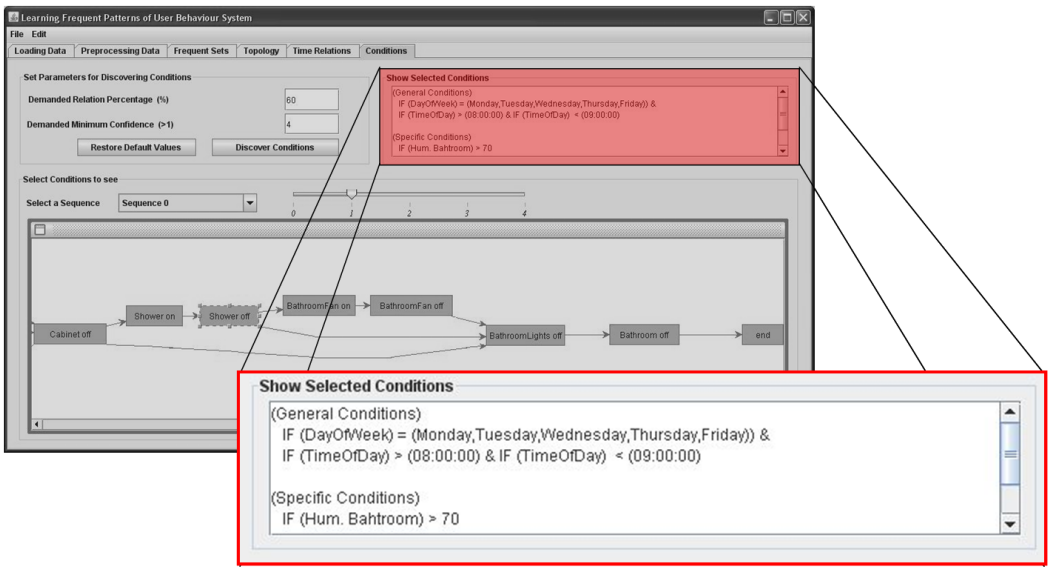

Figure 5.14: Graphical representation of the topology.

5.4 Summary

The Action Map Approach is an evolution of the Pairwise Approach. Its objective is to discover and represent whole behaviours of the users, without any limitation on the number of actions involved in a pattern.

To do this, both $\mathcal{L}_{LFPUBS}$ and $\mathcal{A}_{LFPUBS}$ have needed modifications. $\mathcal{L}_{LFPUBS}$ has been modified to allow the system to represent a whole behaviour in a unique pattern. In the case of the algorithm, $\mathcal{A}_{LFPUBS}$ has been modified to discover the sets of actions that define the frequent behaviours and specially to identify the frequent order of such actions. Moreover, the nature of the conditions is different, so both $\mathcal{L}_{LFPUBS}$ and $\mathcal{A}_{LFPUBS}$ have been adapted as well.

Chapter 6

Validation

In order to validate the system developed in this research work, the LFPUBS was applied to datasets collected from two real environments. Some of the validations were general, whereas other were focused on validating specific steps of the $\mathcal{A}_{LFPUBS}$. In addition, both approaches defined in Chapter 4 and Chapter 5 were validated using the same datasets, allowing the comparison of the results.

6.1 Validation Environments and Collected Data

The validation process was carried out using different datasets collected from two real environments (*MavPad* and *WSU Smart Apartment*). Below, both environments will be explained in more detail, defining the number and types of sensors, the actions and activities to be identified in each one of the environments and so on.

6.1.1 MavPad Environment

The MavPad is an on-campus student apartment located at University Village on the University of Texas at Arlington's campus [You05]. The apartment consists of a living/dining room combination, a kitchen, a bathroom, a large bedroom, and a walk-in closet. The sensors installed in MavPad are:

- 25 sensors on objects such as lamps, lights or outlets.
- 45 context sensors such as light, temperature or humidity.
- 36 motion sensors distributed in all of the rooms.

Sensors installed on objects indicate the actions of the users over such objects. Thus, for example, the 'A1' sensor indicates the status of the ceiling light installed in the living room, 'B6' indicates the status of the bathroom's fan and 'C8' the status of the bedroom's table lamp.

Different types of context sensors are installed in the kitchen, living room, bathroom and bedroom. Thus, for example, 'S1' detects the light level of the living room, 'S82' the temperature of the bedroom and 'S139' detects the humidity level of the bathroom.

Finally, motion sensors are installed through all of the rooms of the environment. The objective of the sensors is to detect where the user is in each moment.

The distribution of different sensors through MavPad can be seen in Figure 6.1. The sub-figure (a) shows the sensors installed on objects while sub-figure (b) shows the context and motion sensors.

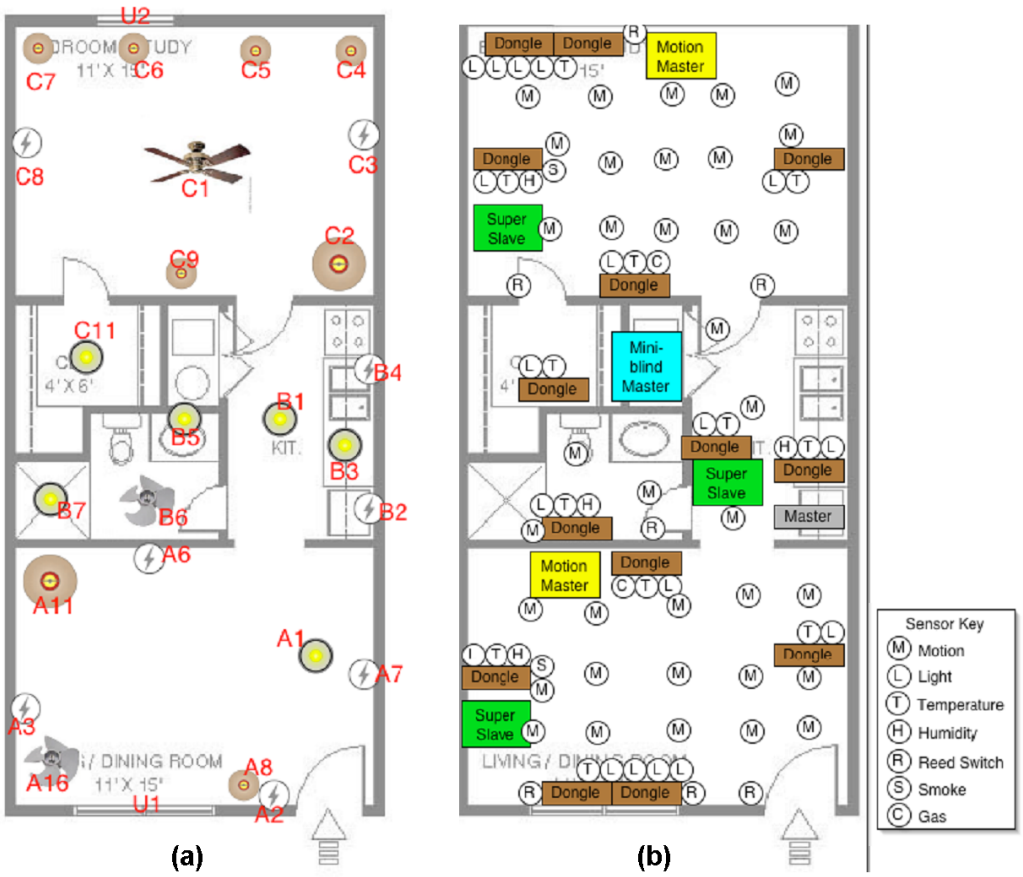

Figure 6.1: MavPad sensors on objects, context and motion sensors [You05].

The dataset used to validate the 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). It is worth mentioning that the frequent behaviours hidden in such trials were not known in advance because the users behaved as they liked. In addition, the data collected in each trial was totally independent from the other trials' data.

6.1.2 WSU Smart Apartment Environment

The testbed WSU Smart Apartment is an environment created at Washington State University [Coo08]. The sensors installed in it are:

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

Sensors installed on objects are aimed at detecting different actions of the participants. There are sensors that detect when the user takes the phone book ('I08'), the phone ('asterisk'), the medicine container ('I06') or item sensors ('I01'-'I05') for products like oatmeal, raisins, brown sugar, bowl or measuring spoon. There are also sensors to detect the use of water or the burner.

Regarding the motion sensors, their objective is to detect where the participants are located. The distribution of different motion sensors through the WSU apartment can be seen in Figure 6.2.

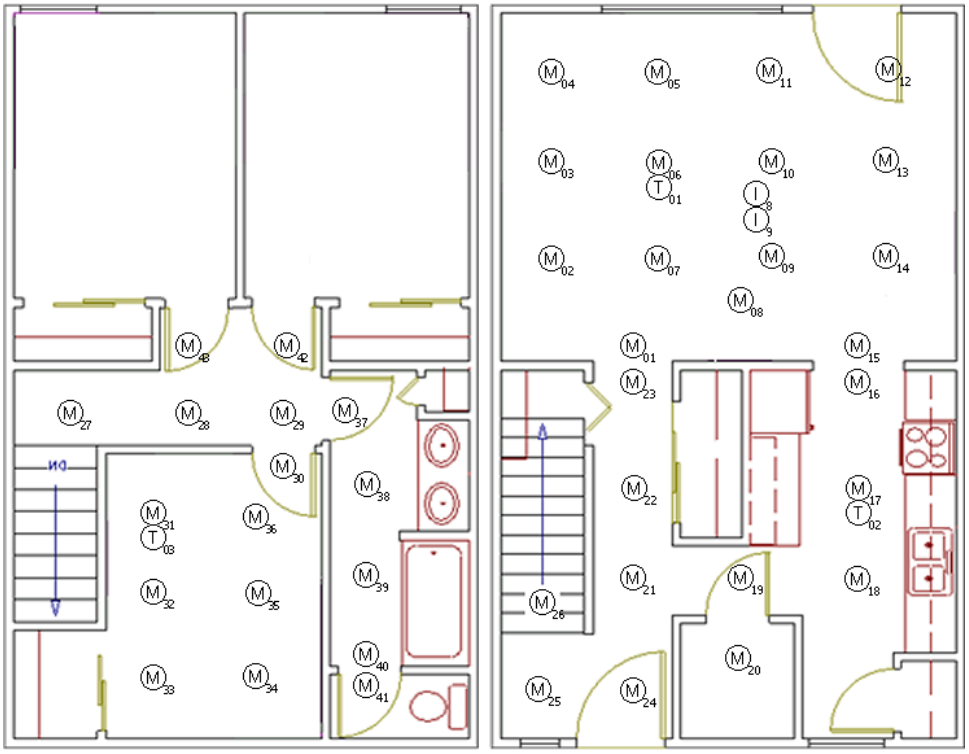

Figure 6.2: WSU Smart Apartment motion sensors [Coo08].

Unlike the MavPad environment, where the frequent behaviours hidden by different trials were unknown, 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 the LFPUBS should discover were known in advance. The five tasks were:

- Make a phone call. The participant moved to the phone in the dining room, looked a specific number in the phone book, dialled the number, and listened to the message. The recorded message provided cooking directions, which the participant summarised on a notepad.

- Wash hands. The participant moved into the kitchen sink and washed his/her hands in the sink, using hand soap and drying their hands with a paper towel.

- Cook. The participant cooked a pot of oatmeal according to the directions given in the phone message. To cook the oatmeal the participant had to measure water, pour the water into a pot and boil it, add oats, then put the oatmeal into a bowl with raisins and brown sugar.

- Eat. The participant took the oatmeal and a medicine container to the dining room and ate the food.

- Clean. The participant took all of the dishes to the sink and cleaned them with water and dish soap in the kitchen.

As stated above, the data collected from the WSU Smart Apartment showed participants performing the above-mentioned five ADLs. The actions involved in each one of the activities are shown in Table 6.1.

Table 6.1: 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'

6.2 Pairwise Approach

As described in Chapter 4, the objective of the first approach was to discover frequent pairwise relationships between the actions performed by the user. Thus, the first version of the LFPUBS was validated using data collected from both of the real environments described above. Below, the knowledge discovered in both cases is described in more detail.

6.2.1 Validating the Pairwise Approach with the MavPad dataset

As stated above, the data collected from MavPad were collected in three different trials. Given the possibility of configuring the LFPUBS, different tests were carried out by using different minimum confidence levels (25%, 50%, 75% and 100%). The minimum confidence level defines the number of occurrences (as a percentage) demanded for a relationship to consider it frequent (for further details see Section 4.4.1). To discover as many patterns as possible the minimum support parameter was eliminated and the minimum confidence level was used as a unique parameter to determine whether a relation was frequent.

Because of the nature of the data, some steps of the learning process could be better validated than others. Regarding the 'Identifying Frequent Relations' step, the number and the nature of the patterns to be discovered were unknow, so at this point, the tests were only able to confirm that the LFPUBS was able to discover Frequent Relations in unknown datasets. The number of Frequent Relations discovered with each minimum confidence level, together with the runtime of that experiment, is defined in Table 6.2. It is worth noting that as expected, the runtime of each experiment directly depends on the number of Frequent Relations discovered in it. When it was impossible to discover any Frequent Relation (because of the high (100%) confidence level demanded in the experiment), the runtime was almost imperceptible because the algorithm could not create candidates for it.

Once Frequent Relations were discovered, the next step was to identify possible quantitative Time Relations. To do that, the chosen algorithm was 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 6.3 shows the number of patterns in which it was possible to discover a quantitative Time Relation compared with the total number of patterns and the runtime. Regarding the runtime, it is directly proportional to the number of relations to analyse and the number of particular Time Distances to consider in each one of them. In those cases in which it was impossible to discover Frequent Relations (see Trial 1, Trial 2 and Trial 3 with confidence level of 100%), it was not necessary to calculate any Time Relation.

Once Frequent Relations and Time Relations were identified, the step of 'Identifying Conditions' discovered the necessary conditions to understand when each pairwise relationship becomes true. As in the previous steps, the Conditions to be discovered were not known in advance; however, the number of patterns in which it was possible to identify conditions

Table 6.2: Number of patterns obtained in different trials.

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Total Patterns**	**Total Patterns**	**Total Patterns**
25%	16 (78 ms)	40 (1347 ms)	20 (204 ms)
50%	5 (70 ms)	18 (188 ms)	10 (157 ms)
75%	1 (41 ms)	5 (69 ms)	6 (98 ms)
100%	0 (31 ms)	0 (31 ms)	0 (31 ms)

Table 6.3: Number of patterns with Time Relations (out of total patterns and the percentage) obtained in different trials.

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Patterns with Time Relations**	**Patterns with Time Relations**	**Patterns with Time Relations**
25%	14 (14/16 (87.5%)) (312 ms)	34 (34/40 (85%)) (1420 ms)	18 (18/20 (90%)) (504 ms)
50%	4 (4/5 (80%)) (267 ms)	16 (16/18 (89%)) (412 ms)	8 (8/10 (80%)) (359 ms)
75%	1 (1/1 (100%)) (58 ms)	4 (4/5 (80%)) (265 ms)	4 (4/6 (67%)) (106 ms)
100%	No Time Relations needed	No Time Relations needed	No Time Relations needed

shows the LFPUBS's ability to discover them (See Table 6.4). As expected, in this case too, the runtime directly depends on the number of conditions to discover.

Table 6.4: Number of patterns with Conditions (out of total patterns and the percentage) obtained in different trials.

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Patterns with Conditions**	**Patterns with Conditions**	**Patterns with Conditions**
25%	12 (12/16 (75%)) (2541 ms)	33 (33/40 (82.5%)) (7582 ms)	15 (15/20 (75%)) (4529 ms)
50%	3 (3/5 (60%)) (1847 ms)	14 (14/18 (78%)) (2841 ms)	4 (4/10 (40%)) (2147 ms)
75%	1 (1/1 (100%)) (58 ms)	3 (3/5 (60%)) (1024 ms)	2 (2/6 (33%)) (1129 ms)
100%	No Conditions needed	No Conditions needed	No Conditions needed

Finally, three of the patterns discovered in different trials are shown as an example (all patterns can be seen in Appendix F). The first pattern shows how at night (between 1:06 a.m. and 2:46 a.m.), the user switched on the luxo lamp as soon as (0 seconds) he/she turned off the bedroom light. This pattern was discovered in Trial 1 with a confidence level of 25%.

```
(Pattern 1 --> Trial 1, Confidence level: 25%)
  ON occurs (BedroomLight, Off,t0)
  IF context ((time (>,1:06:44) &
                    (<,2:46:32)))
  THEN do (On, BedroomLuxo1, t) when t = t0 + 0s
```

Another pattern shows how the user in Trial 2 switched off the luxo lamp after (30 seconds after) switching on the bedroom light if the light level of the bedroom was at a certain level (between 52 and 143) from 19:19 p.m. on.

```
(Pattern 2 --> Trial 2, Confidence level: 50%)
  ON occurs (BedroomLight, On,t0)
  IF context ((time (>,19:19:49)) &
              (Bedroom light level (>,52)) &
              (Bedroom light level (<,143)))
  THEN do (Off, BedroomLuxo2, t) when t=t0+30s
```

Finally, a very frequent pattern (>75%) discovered in Trial 3 shows how long the user took (115 seconds) to switch off the lights at night.

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

Discussion and trends

Because the frequent behaviours hidden in the data collected from MavPad were unknown, it can be said that the MavPad environment could be one of the real environments where the LFPUBS will be applied in the future, where it is impossible to define any previous knowledge to facilitate the process of learning. The results provided by the LFPUBS in different trials can be analysed in order to detect possible trends.

On one hand, it is interesting to analyse the possibility of identifying quantitative Time Relations and Conditions in the Frequent Relations discovered by the first step. Taking into account the percentages shown in Tables 6.3 and 6.4, the possibility of discovering quantitative Time Relations and Conditions depending on the demanded minimum confidence level are shown in Figures 6.3 and 6.4.

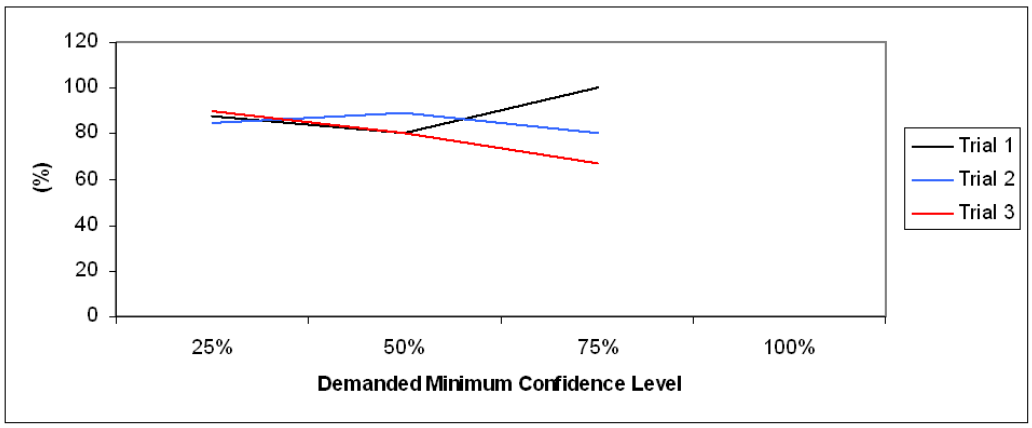

Figure 6.3: Percentage of the Patterns with Time Relations.

Regarding the percentage of the patterns in which it was possible to define a quantitative Time Relation and/or Conditions, in both cases, there was no clear trend that indicated how those percentages could vary, depending on the demanded minimum confidence level. In any case, they indicate that it is possible to define quantitative Time Relations and Conditions in most of the cases (87% and 78% respectively). It is worth emphasizing that the lack of Frequent Relations when considering a confidence level of 100% made it impossible to define quantitative Time Relations and Conditions, and therefore to calculate the percentage for such situations.

On the other hand, it is interesting to analyse the nature of the patterns in terms of frequency, i.e., the number of patterns that remain being frequent when a higher confidence

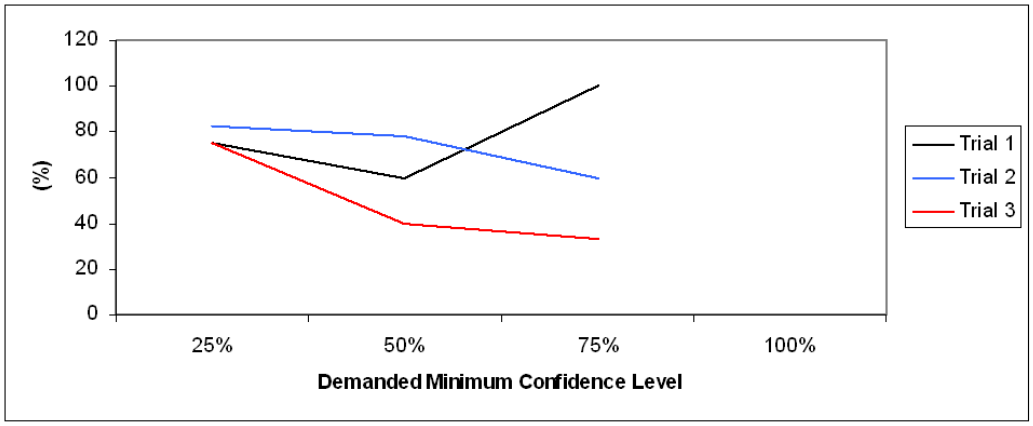

Figure 6.4: Percentage of the Patterns with Conditions.

level is demanded. Table 6.5 summarises the percentages of patterns that remain from the confidence level of 25% to 50% and from 50% to 75%. The minimum confidence level of 100% is ruled out because there are no patterns that achieve this level.

Table 6.5: The number of patterns that remain with different confidence levels.

	Trial 1	**Trial 2**	**Trial 3**
% of patterns that remain from 25% to 50% Confidence Level	31%	45%	50%
% of patterns that remain from 50% to 75% Confidence Level	20%	27%	60%

A trend that can be extracted in relation to the confidence level of patterns is that in most of the trials, less than half of the patterns discovered with a confidence level of 25% remain frequent with a confidence level of 50%. The percentage decreases when the confidence level increases to 75%. Trial 3 is the exception to such a trend because 60% of the total patterns discovered with a confidence level of 50% remain frequent with a confidence level of 75%. A high number of patterns with low frequency levels shows that most of the behaviours of the users were not performed with a monotonous regularity. In fact, this conclusion is reinforced by the fact that there was no a pattern with a confidence level of 100%.

In conclusion, it can be said that the MavPad data show a real environment where the user was not biased to behave in a certain way. Even so, the LFPUBS was able to discover frequent behaviours, although most of them were discovered with a low confidence level.

In addition, it was demonstrated that the LFPUBS was able to discover quantitative Time Relations as well as Conditions in a high percentage of the cases.

6.2.2 Validating the Pairwise Approach with the WSU Smart Apartment dataset

Unlike MavPad, participants of the WSU Smart Apartment behaved in a predefined way, so it was known in advance what knowledge the LFPUBS should discover. It can be thought of as an acid test, especially for the step 'Identifying Frequent Relations' because the sequence of activities to be carried out by the users was defined and the LFPUBS should discover the Frequent Relations defined by that sequence. Time Relations could also be identified, but the time spans between different actions were not defined in advance. Finally, it was not possible to discover Conditions because of the lack of context sensors that provided information about the status of the environment. Below, each one of the steps will be explained in more detail.

The objective of the 'Identifying Frequent Relations' step was to discover the relations defined by the sequence of actions described in Table 6.1. Considering a minimum confidence level of 60%, it discovered 23 Frequent Relations. Within those 23 patterns, there were the 17 Frequent Relations that described the performance of the 5 ADLs. To avoid misunderstandings about the number of Frequent Relations to be discovered, it must be noticed that the performance of those 5 activities demanded that some sets of actions be repeated (for example, 'Water On' - 'Water Off'), but such relationships, although they were repeated in different activities, were identified as a unique Frequent Relation because the Pairwise Approach does not consider different instantiations of the same action. Some of the Frequent Relations that were found are shown below:

```
(Pattern 1)                              (Pattern 2)
 ON occurs (Phone Book, On,t0)            ON occurs (Phone, On,t0)
 IF -                                     IF -
 THEN do (On, Phone, t) when --           THEN do (Off, Phone, t) when --

 (...)

(Pattern 15)                             (Pattern 16)
 ON occurs (Cabinet, On,t0)               ON occurs (Medicine, On,t0)
 IF -                                     IF -
 THEN do (On, Medicine, t) when --        THEN do (Off, Cabinet, t) when --

(Pattern 17)                             (Pattern 18)
 ON occurs (Cabinet, On,t0)               ON occurs (Medicine, Off,t0)
 IF -                                     IF -
 THEN do (Off, Medicine, t) when --       THEN do (Off, Cabinet, t) when --
```

Most of the patterns were logical relationships such as 'after speaking on the phone, he hangs up the phone' (See Pattern 2). Apart from those trivial relations, some other interesting relationships were also discovered, for example, the set of patterns (See Patterns 15-18) that shows how the users carried out the activity of taking their medicine.

Although the main contribution of this dataset was to validate the first step of the algorithm, it was also used to discover possible quantitative Time Relations. Unlike the Frequent Relations, quantitative Time Relations were not known in advance. Therefore, as in the MavPad environment, the only possible validation was to check the ability of the LFPUBS to discover such Time Relations. In that sense, it was able to define quantitative Time Relations in 16 out of 23 Frequent Relations (70%) that described the 5 ADLs. Below, previously shown patterns with the discovered Time Relations are shown:

```
(Pattern 1)
 ON occurs (Phone Book, On,t0)
 IF -
 THEN do (On, Phone, t)
      when t = t0 + 57s

 (...)

(Pattern 15)
 ON occurs (Cabinet, On,t0)
 IF -
 THEN do (On, Medicine, t)
      when t = t0 + 2s

(Pattern 17)
 ON occurs (Cabinet, On,t0)
 IF -
 THEN do (Off, Medicine, t)
      when t = t0 + 2s
```

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

(Pattern 16)
 ON occurs (Medicine, On,t0)
 IF -
 THEN do (Off, Cabinet, t)
      when t is after t0

(Pattern 18)
 ON occurs (Medicine, Off,t0)
 IF -
 THEN do (Off, Cabinet, t)
      when t = t0 + 2s
```

It is clear that the discovered Time Relations cannot be compared to anything to check their correctness. Even so, they can provide some interesting information such as the fact that users needed around 50 seconds to get the cooking instructions (Pattern 2).

Finally, because of the lack of context sensors, it was not possible to discover conditions that defined the occurrences of each pattern. By default, all patterns were considered as true under any condition.

Discussion and trends

The objective of this test was to guarantee that the LFPUBS was able to discover the Frequent Relations hidden in the dataset. In that sense, it was able to discover such patterns, and it was even able to discover quantitative Time Relations in 70% of the patterns.

Interesting conclusions can be extracted by analysing the nature of the discovered knowledge and the way it is represented. On one hand, it is clear that the users' behaviours are better described by relating user's actions among themselves instead of relating users' actions to global situations. On the other hand, it is clear that users' behaviours could be better described by means of Action Maps that represent the whole behaviours of the users without any limitation on the number of actions involved in each behaviour.

6.3 Action Map Approach

The analysis of the results obtained when applying the Pairwise Approach to different datasets showed that users' behaviours would be better described if the number of actions involved in a pattern were not limited to two actions. Thus, the Pairwise Approach evolved into the Action Map Approach, described in Chapter 5.

To validate this new approach, the same datasets used for the Pairwise Approach were considered. In this way, the output provided by the new approach was compared to the output provided by the first approach, and the advantages were identified.

6.3.1 Validating the Action Map Approach with the MavPad dataset

In this case too, the data collected in the three trials were used to validate the new approach of the LFPUBS. The objective of this validation process was twofold. The fist objective was to check if the Action Map Approach was able to discover users' frequent behaviours, and the second objective was to compare the knowledge extracted in this case with the knowledge extracted when using the Pairwise Approach.

The objective of the first step, 'Identifying Frequent Sets of Actions', was to discover those sets of actions that were more frequent than a demanded minimum confidence level. In order to be able to provide a clear comparison between both approaches, the LFPUBS was run by considering the same minimum confidence levels as in the Pairwise Approach (25%, 50%, 75% and 100%). In this case, the number of Frequent Sets to be discovered was also unknown because of the unbiased nature of the data. The number of Frequent Sets discovered in each trial, considering different minimum confidence levels, is shown in Table 6.6. Unlike the Pairwise Approach, the runtime of each experiment is not directly related to the number of Frequent Sets discovered in each experiment. In this case, the number of actions involved in each Frequent Set has a higher influence in the runtime than the number of Frequent Sets. This is because the most time consuming step of the Apriori algorithm (Candidate Generating step) is strongly influenced by the number of action it has to deal with.

For example, one of the Frequent Sets discovered in the Trial 1 with a minimum confidence level of 50% shows that the user performed the actions of 'BedroomLight On', 'BedroomLight Off', 'BedroomLuxo1 On' and 'BedroomLuxo1 Off' together (all Frequent Sets can be seen in Appendix H).

Table 6.6: The number of patterns obtained in different trials and the Experiments' runtimes (in milliseconds).

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Total Patterns**	**Total Patterns**	**Total Patterns**
25%	8 (156 ms)	3 (219 ms)	1 (153 ms)
50%	4 (78 ms)	1 (188 ms)	1 (68 ms)
75%	1 (62 ms)	1 (93 ms)	1 (56 ms)
100%	0 (35 ms)	0 (35 ms)	0 (35 ms)

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 (see Section 5.3.2 for the definition of these concepts).

Although some aspects of the process could not be validated, a comparison between the knowledge discovered by both approaches showed an interesting conclusion. The relations showed by different topologies were compared to the Frequent Relations discovered by the Pairwise Approach, and all of the relations (100%) represented by the different Topologies were also identified as Frequent Relations by the first approach.

The opposite statement, i.e., that all of the Frequent Relations were included in the Topologies, cannot be guaranteed. This is because Frequent Relations were discovered without considering any minimum support level. If a relation reached the demanded confidence level, it was considered a Frequent Relation even though it occurred very few times. In contrast, in the case of Frequent Sets, the minimum confidence level also worked as the minimum support level, so that an action had to fulfil both demanded minimum levels to be included in a Frequent Set. That is the reason why all of the Frequent Relations are not included in the Topologies.

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 light. This behaviour is represented by means of a unordered subset that groups the actions of 'BedroomLight Off' and 'BedroomLuxo1 On'. Finally, the user turned off the luxo lamp.

Once topology was defined, it was possible to represent such a behaviour by means of the $\mathcal{L}_{LFPUBS}$ (See Appendix H for the complete representations of all Action Maps).

```
(Action Map 1)

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

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

(Action Pattern 3)
 ON occurs (unordered,((BedroomLight,Off)&(BedroomLuxo1,On)),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)  when --

(Action Pattern 2)
 ON occurs (simple,(BedroomLuxo1,Off),t0)
 IF context ()
 THEN do (--,end,t)  when --
```

Table 6.7 shows the runtime of different experiments when considering the step of 'Identifying Topology'. The runtime of each experiment depends on many different parameters such as the number of actions involved in the Action Map, the number of repetitive actions or the number of unordered subsets of actions. An special mention deserves the Action Map discovered in the Trial 3. Considering confidence levels of 25%, 50% and 75%, in all those cases only one Frequent Set was discovered by the first step. Besides, the runtimes are equal, which could be an indication that the discovered Frequent Sets, as well as their topologies, are equal (see Appendix H for further details).

Regarding the Time Relations, they were calculated using the 'Basic Algorithm'. Compared with those Time Relations discovered in the first approach, the only difference was that the number of Time Relations increased because of repetitive actions. Moreover, some quantitative Time Relations can vary a little when they deal with unordered subsets of actions because in such cases, the Time Relations are calculated by taking into account the first occurrence of any of the actions involved in the subset. Table 6.8 shows the number of

Table 6.7: Experiments' runtimes (in milliseconds) when discovering the Topology of Frequent Sets.

	Trial 1	**Trial 2**	**Trial 3**
Confidence Level	**Total Patterns**	**Total Patterns**	**Total Patterns**
25%	172 ms	5750 ms	47 ms
50%	63 ms	5234 ms	47 ms
75%	47 ms	625 ms	47 ms
100%	No Topology needed	No Topology needed	No Topology needed

Time Relations (and the percentage they represent) discovered in each trial. In this case too, 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. Finally, 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%). On the other hand, 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%).

Considering the same Action Map presented above, once Time Relations were discovered, it was represented as follows:

```
(Action Map 1)

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

(Action Pattern 2)
 ON occurs (simple,(BedroomLight,On),t0)
 IF context ()
 THEN do (unordered,((Off,BedroomLight)&(On,BedroomLuxo1)),t)
      when t = t + 346s
```

```
(Action Pattern 3)
 ON occurs (unordered,((BedroomLight,Off)&(BedroomLuxo1,On)),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)  when t = t + 227s

(Action Pattern 2)
 ON occurs (simple,(BedroomLuxo1,Off),t0)
 IF context ()
 THEN do (--,end,t)  when --
```

Table 6.8: 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

Finally, 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 6.9 shows the percentage of times 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. That is why Trial 2 experiments' runtimes were higher than Trial 1 experiments' runtimes, although the situations that required conditions were not more.

Table 6.9: 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

Concerning the number and the percentage of the Specific Conditions discovered in each trial, it is worth noting that they cannot be compared to the Conditions discovered in the Pairwise Approach, because in the Action Map Approach Specific Conditions were discovered only when a situation required it, whereas in the Pairwise Approach, Conditions were discovered for each Frequent Relation.

Regarding General Conditions, because of the ability of the algorithm to generalise, it was possible to discover General Conditions for all of the Action Maps. For example, the 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 follows (See Appendix H for the complete representation of all of the Action Maps).

```
(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 2)
 ON occurs (simple,(BedroomLight,On),t0)
 IF context ()
 THEN do (unordered,((Off,BedroomLight)&(On,BedroomLuxo1)),t)
      when t = t + 346s

(Action Pattern 3)
 ON occurs (unordered,((BedroomLight,Off)&(BedroomLuxo1,On)),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)  when t = t + 227s

(Action Pattern 2)
 ON occurs (simple,(BedroomLuxo1,Off),t0)
 IF context ()
 THEN do (--,end,t)  when --
```

Discussion and trends

The objective of this validation process was twofold. The fist objective was to check if the Action Map Approach was able to discover users' frequent behaviours, and the second objective was to compare the knowledge extracted in this case with the knowledge extracted when using the Pairwise Approach.

It is clear that Action Maps, when compared with the knowledge provided by the pairwise relationships, facilitates the understanding of the users' behaviours. In this sense, the main concern was to guarantee that representing users' behaviour by means of Action Maps did not result in discovering different knwoledge. This was the most important aspect in the validation of the first two steps that discovered Frequent Sets and their Topologies. Once the results provided by the Action Map Approach were analysed, it could be said that all of the relationships (100%) defined by the topologies were also identified as Frequent Relations by the Pairwise Approach.

Regarding the Time Relations, Figure 6.5 shows, for each trial, the percentage of different Action Patterns in which quantitative Time Relations were identified. It is worth noting that in some cases it was not necessary to identify Time Relations, so that such situations will not be depicted in the figure. These situations occur because either there was not any Frequent Set (e.g., Trial 1, Trial 2 and Trial 3 with confidence level of 100%) or all the actions were grouped 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%). Analysing the percentages showed in the figure, it can be concluded that they do not indicate any clear trend that shows how such a percentage varies as a function of different minimum confidence levels. Although it seems that the percentage of Action Patterns with Time Relations decreases when the demanded minimum confidence level is higher, such a trend cannot be generalised.

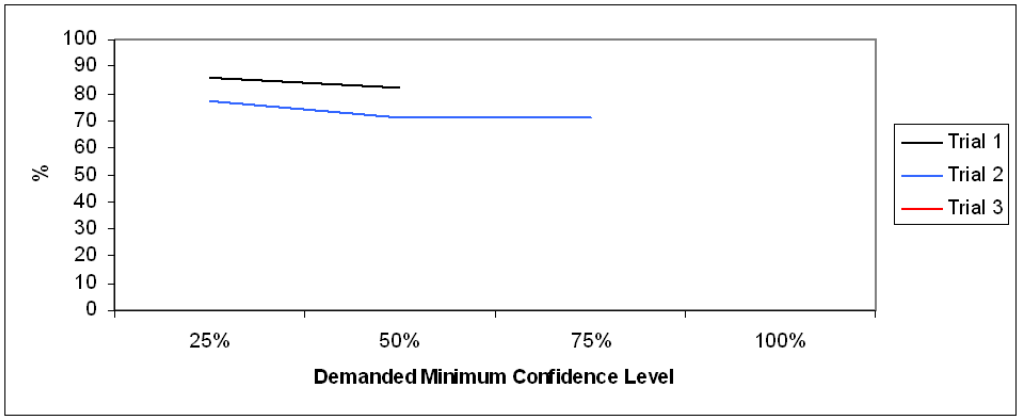

Figure 6.5: Percentage of the Patterns with Time Relations.

The percentages of Specific Conditions also demonstrate the ability of the Action Map Approach to discover Specific Conditions when needed. On average, it was able to identify Specific Conditions in 69% of the cases. The trend that shows how such percentages evolved depending on the different minimum confidence levels considered in different trials is depicted in Figure 6.6. In this case too, it was not necessary to discover Specific Conditions in some of the experiments, and therefore it was not possible to define the percentages for such cases.

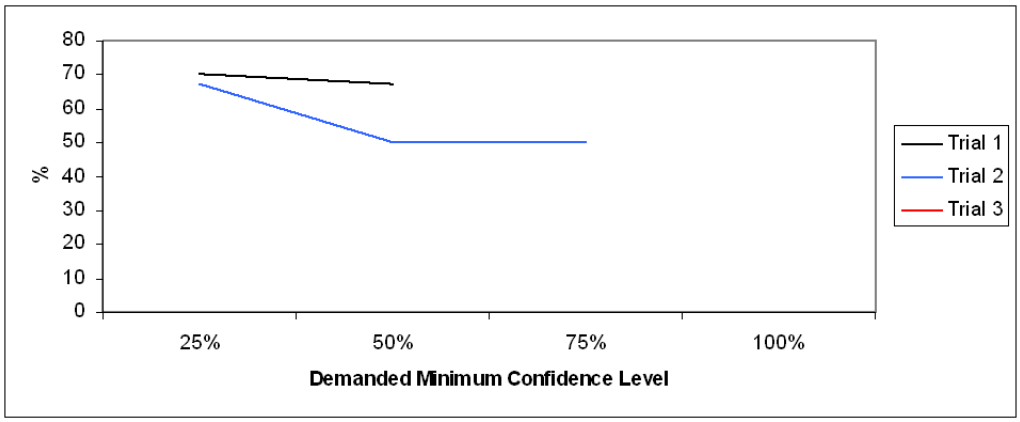

Figure 6.6: Percentage of the Patterns with Specific Conditions.

The Action Map discovered in Trial 3 deserves a special mention. Only one Action Map was discovered in Trial 3, a very frequent Action Map that remained frequent even for a confidence level of 75%. The special characteristic of this Action Map was that neither Time Relations nor Specific Conditions were needed. Although it is surprising, this fact has a very simple explanation; the different occurrences of the Action Map did not show a clear

topology, so all of the actions were included in an unordered subset of actions. The relations within an unordered subset of actions are not defined, so it was impossible to define Time Relations and Specific Conditions for such cases. The same thing happened with the Action Map discovered in Trial 1 for a confidence level of 75%.

At the moment, General Conditions provide very general information. Because General Conditions must cover all of the particular occurrences of the Action Maps, most of the time, they are so general that they do not separate the occurrences of different Action Maps. Undoubtedly this step of the LFPUBS is an aspect to improve in future versions of the system.

6.3.2 Validating the Action Map Approach with the WSU Smart Apartment dataset

Finally, the Action Map Approach was also validated using the data collected from the WSU Smart Apartment. As mentioned above, these data showed different participants performing 5 ADLs in a specific order. Thus, this validation was an acid test for the step of 'Identifying Topology' that had to model such a behaviour.

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 6.1, were identified as frequent.

Frequent Set 1: *'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'.*

Once the set of actions involved in the Action Map was identified, the next step was to discover the frequent order of such actions. In that sense, 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'. The nature and the purpose of such actions in each one of the activities is different; therefore, identifying repetitive actions was an important step to correctly model users' behaviours. In the case of the actions 'Water On' and 'Water Off' the LFPUBS was able to define that four different 'Water On' and 'Water Off' actions were needed (See Appendix I for the complete definition of the Action Map and repetitive actions).

It is true that 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 parameters defined below, 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' (See Appendix I for the complete definition).

- Minimum Level for Origin: 25%
- Minimum Level for Destiny: 25%
- Minimum Balanced Level: 50%

Once repetitive actions and unordered subsets of actions were identified, it was possible to define the topology that modelled participants' behaviour. The first part of the behaviour was defined as follows (the complete behaviour is detailed in Appendix I).

```
(Action Map 1)

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

(Action Pattern 2)
 ON occurs (simple,(PhoneBook,On),t0)
 IF context ()
 THEN do (simple,(On,Phone),t)  when --

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

(Action Pattern 4)
 ON occurs (simple,(Phone,Off),t0)
 IF context ()
 THEN do (simple,(On,Water),t)  when --

(Action Pattern 5)
 ON occurs (simple,(Water, On),t0)
 IF context ()
 THEN do (simple,(Off,Water),t)  when --

 (...)
```

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 of the relations defined by the Topology, the 'Basic Algorithm' was able to identify quantitative Time Relations in 25 out of 29 (86%) cases.

Compared with the quantitative Time Relations discovered in the 'Pairwise Approach' (See Section 6.2.2), two important conclusions can be extracted. First, when a relation did not involved either repetitive actions or actions involved in unordered subsets, the Time Relations discovered by both approaches were the same (for example, the Time Relation discovered for the relationship between 'PhoneBook On' and 'Phone On' was 57 seconds in both cases). Second, if either a repetitive action or an action involved in an unordered subset was present in the relationship, the discovered Time Relations varied slightly. This is because in the Action Map Approach, if repetitive actions existed, particular occurrences were considered, depending on the repetitive actions they belonged to, whereas in the Pairwise Approach all of the occurrences were considered together. A clear example of this was the relationship between the actions 'Water On' and 'Water Off'. In the 'Pairwise Approach', all of the occurrences of that relationship were collected together and the quantitative Time Relation was calculated by taking into account all of those occurrences. In contrast, in the Action Map Approach, more than one 'Water On' and 'Water Off' action was defined, so each process of discovering quantitative Time Relations was run by only considering the corresponding occurrences in each one of them. The other situation in which quantitative Time Relations varied slightly between the results of both approaches occurred when considering unordered subsets of actions. These slight variances are due to the fact that when dealing with unordered subsets of actions, it is considered the first occurrence of any of the actions involved in the subset is considered, whereas in the Pairwise Approach, this possibility was not considered. The representation of the Action Map once Time Relations were defined was as follows (see Appendix I for the complete definition of all Action Maps):

```
(Action Map 1)

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

(Action Pattern 2)
 ON occurs (simple,(PhoneBook,On),t0)
 IF context ()
 THEN do (simple,(On,Phone),t)  when t = t0 + 57s
```

```
(Action Pattern 3)
 ON occurs (simple,(Phone,On),t0)
 IF context ()
 THEN do (simple,(Off,Phone),t)  when t = t0 + 50s

(Action Pattern 4)
 ON occurs (simple,(Phone,Off),t0)
 IF context ()
 THEN do (simple,(On,Water),t)  when t is after t0

(Action Pattern 5)
 ON occurs (simple,(Water, On),t0)
 IF context ()
 THEN do (simple,(Off,Water),t)  when t = t0 + 23s

 (...)
```

In this case too, Specific and General Conditions were identified. Regarding the Specific Conditions, it is true that very few situations demanded Specific Conditions (only three). Besides, the lack of context information meant Specific Conditions could only be identified using calendar information. It is worth noting that using only calendar information, it was possible to identify conditions in two out of three (67%) cases. Even so, it should also be said that the discovered conditions were not very meaningful because it seems clear that different participants did not perform in different ways because of aspects such as 'Time of Day' or 'Day of Week', but rather some other context aspect such as temperature or light level was influencing their behaviour.

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.. The final representation of the Action Map was as follows (see Appendix I for the complete definition of all Action Maps):

```
(Action Map 1)

(General Condition)
 context (TimeOfDay(>,10:45:00)) & context (TimeOfDay(<:18:15:00)) &
 context (DayOfWeek(=,(Monday,Tuesday,Wednesday,Thursday,Friday)))


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

```
(Action Pattern 2)
 ON occurs (simple,(PhoneBook,On),t0)
 IF context ()
 THEN do (simple,(On,Phone),t)  when t = t0 + 57s

(Action Pattern 3)
 ON occurs (simple,(Phone,On),t0)
 IF context ()
 THEN do (simple,(Off,Phone),t)  when t = t0 + 50s

(Action Pattern 4)
 ON occurs (simple,(Phone,Off),t0)
 IF context ()
 THEN do (simple,(On,Water),t)  when t is after t0

(Action Pattern 5)
 ON occurs (simple,(Water, On),t0)
 IF context ()
 THEN do (simple,(Off,Water),t)  when t = t0 + 23s

 (...)
```

Finally, Table 6.10 shows the runtime of each of the steps of the algorithm.

Table 6.10: The runtime of different steps of the $\mathcal{A}_{LFPUBS}$.

Identifying Frequent Sets of Actions	Identifying Topology	Identifying Time Relations	Identifying Conditions
297 ms	969 ms	750 ms	422 ms

Discussion and trends

This validation process was mainly focused on the step of 'Identifying Topology' because the order of the actions involved in the Action Map were known in advance. Besides, repetitive actions as well as unordered subsets of actions had to be identified.

Within an Action Map, the definition of repetitive actions facilitated the understanding of the behaviour, separating the occurrences of the same action that had different meanings. At the same time, this meant that the number of relationships increased, influencing the runtime of the 'Identifying Time Relations' step. Although initially it can seem like a disadvantage, repetitive actions provide an important advantage when it comes to identifying quantitative

Time Relations, because repetitive actions made it possible to better identify the nature of particular occurrences, grouping such occurrences based on their similarities. The percentage of relations in which it was possible to identify a quantitative Time Relation increased from 70% in the Pairwise Approach up to 86% in the Action Map Approach.

The process of identifying Specific Conditions showed that it is possible that such conditions are able to separate the occurrences of different actions, but they do not provide a meaningful explanation if the nature of the actions is analysed.

6.4 Comparing both Approaches: The final Discussion

Validating both approaches using the same datasets allows one to compare the results and highlight the strengths and weaknesses of each approach. Each step of the Pairwise Approach will be compared to its equivalent in the Action Map Approach. All of them will be evaluated according to the following aspects:

- Advantages over the other approach.
- Efficiency in its task.
- Runtime.

6.4.1 Modelling Frequent Behaviours: A comparison

The first objective of the $\mathcal{A}_{LFPUBS}$ was to discover the actions that defined the frequent behaviours of the users. In the Pairwise Approach, this first task was carried out by the 'Identifying Frequent Relations' step, whereas in the Action Map Approach, it demanded the execution of two steps: 'Identifying Frequent Sets of Actions' and 'Identifying Topology'.

The main difference between the two approaches is defined by this first task. The Pairwise Approach models the frequent behaviours of the users by means of pairs of actions, so the topology is only represented by the relation between those two actions. In contrast, the Action Map Approach does not limit the number of actions involved in a pattern; therefore, it needs to identify the topology that defines how such actions are frequently related.

The difference between the two approaches is clear; the Pairwise Approach provides independent pieces of knowledge, while the Action Map Approach provides complete representations of the behaviours. Moreover, some aspects considered in the Action Map Approach, such as repetitive actions and unordered subsets of actions, facilitate comprehension instead of adding complexity to the patterns. This is because the Pairwise Approach considers all of the activations of a sensor as equal, whereas the Action Map Approach defines the nature of each particular activation, depending on the purpose of such an action. For example in the WSU Smart Apartment data, creating different instantiations of some actions clearly facilitated the identification of the frequent behaviour's topology.

Regarding the Frequent Relations discovered in the Pairwise Approach and the relationships defined as frequent by the topologies identified by the Action Map Approach, it must

be said that for both datasets, all of the relationships (100%) defined by the topologies were also identified as Frequent Relations by the Pairwise Approach. This demonstrates that the outputs provided by both approaches are consistent between them.

Regarding the runtime for the MavPad dataset, Figure 6.7 shows the runtime of different experiments as a function of different confidence levels. The data shown in Sections 6.2 and 6.3 have been summarised by taking into account the runtime of all of the trials together. A first analysis clearly shows that the main difference between the runtime of both approaches occurs in the step of 'Identifying Topology' because the runtimes of the steps 'Identifying Frequent Relations' and 'Identifying Frequent Sets of Actions' are very similar. The second conclusion that can be extracted is that in both approaches, the runtime strongly depends on the number of discovered Frequent Relations or Frequent Sets as well as the number of actions involved in the case of Frequent Sets. That is why the runtime is almost imperceptible when considering a minimum confidence level of 100%.

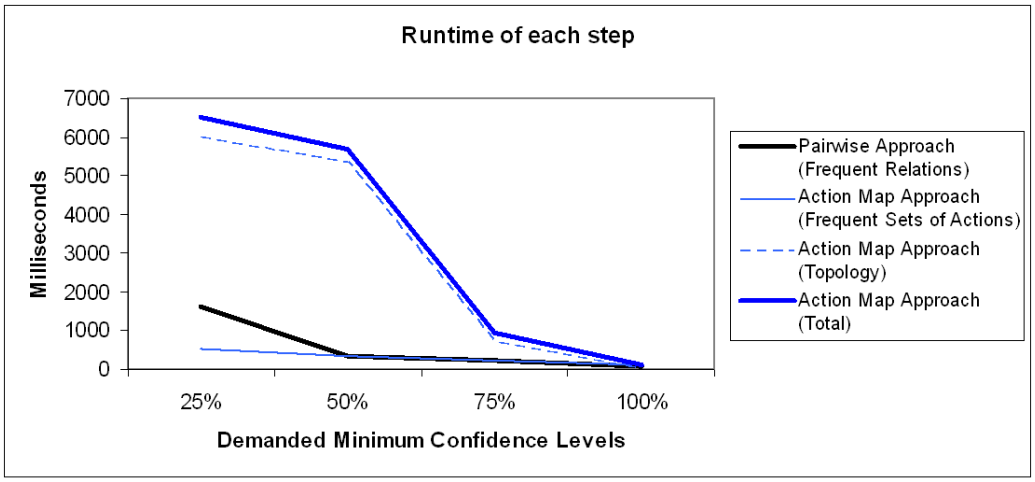

Figure 6.7: Both approaches' runtimes for the task of modelling frequent behaviours.

6.4.2 Identifying Time Relations: A comparison

When it comes to identifying quantitative Time Relations, the objective and the process were the same in both approaches. However, previous steps established some differences that influenced the process of identifying quantitative Time Relations.

First of all, in the Action Map Approach, it is not necessary to discover any type of Time Relations among the actions involved in an unordered subsets of actions.

The identification of repetitive actions also influences the process of discovering quantitative Time Relations. On one hand, the number of relations to be analysed increases. The number of relationships discovered when considering the WSU Smart Apartment dataset is an example of this. In the Pairwise Approach, 23 relationships were defined in total, whereas when considering repetitive actions, the number increased to 29. However, the main influence

of the identification of repetitive actions was noticed in the accuracy of the quantitative Time Relations. Defining different instantiations of the same action allows the system to group particular occurrences based on their similarities, so Time Relations discovered for each one of those groups will be more accurate than the Time Relation identified when considering all of the particular occurrences together. A clear example of this is the relation between the actions 'Water On' and 'Water Off' discovered in the WSU Smart Apartment dataset. In the Pairwise Approach, only one Frequent Relation defined this relationship, so all of the Time Distances collected between both actions were used to identify the quantitative Time Relation. In fact, it was impossible to identify a quantitative Time Relation for such a relation. In contrast, in the Action Map Approach, considering the repetitive actions, four relationships between the actions 'Water On' and 'Water Off' were created (one for each one of the following activities: 'Wash hands', 'Cook', 'Eat' and 'Clean'). Thus, particular Time Distances were grouped based on their similarities and discovered Time Relations were more accurate. In this case, it was possible to define a quantitative Time Relation in three out of those four cases.

When it comes to efficiency, the same algorithm is used in both approaches, so it does not make any sense to compare them in terms of efficiency. Any possible small differences would only come from the previous steps.

Finally, considering the experiments' runtime with the MavPad dataset (See Figure 6.8), some conclusions can be extracted. As expected, the runtime of the Action Map Approach was a little higher because the consideration of repetitive actions created more relationships to be analysed. In this step, the runtime depends on the number of Time Relations to be analysed and the amount of data related to each one of the relations.

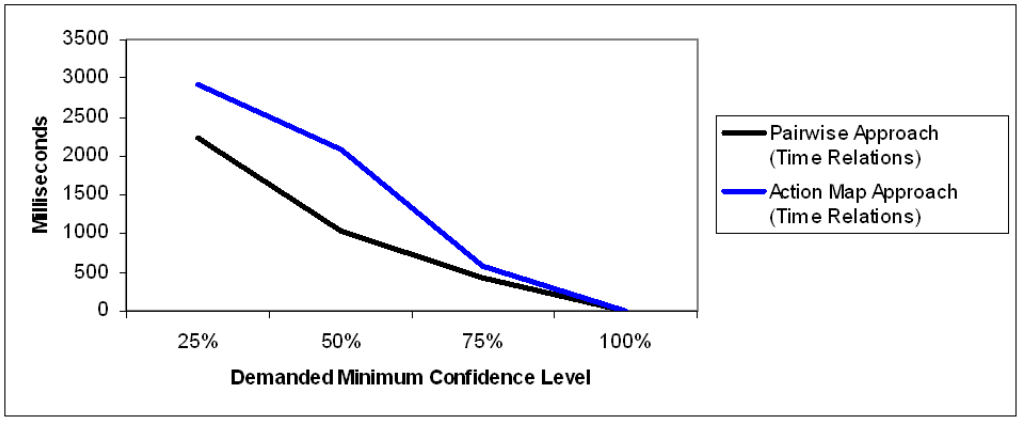

Figure 6.8: Both approaches' runtime considering the task of identifying Time Relations.

6.4.3 Identifying Conditions: A comparison

Although in both approaches the objective of this step was to contextualise the discovered patterns, the nature of the Conditions demanded by each approach is different.

In the Pairwise Approach, the objective of the Conditions was to define under which context and calendar conditions the action defined by the THEN clause was the correct response to the action defined by the ON clause.

In the Action Map Approach two types of conditions were considered. The Specific Conditions discovered the conditions that defined the occurrence of each relationship when a particular action could be followed by more than one option. The General Conditions defined the general context for the whole Action Patterns.

Because of the different natures of the conditions, they cannot be compared in terms of efficiency or runtime. For example, the Pairwise Approach identified conditions for all of the patterns, whereas Specific Conditions were only identified for those situations that required such conditions.

6.4.4 Runtime of different steps

Finally, the runtimes of the different steps were compared in order to identify the most demanding steps in terms of time. In the case of the MavPad dataset, the runtimes of the most demanding situation have been considered (confidence level = 25%). Figures 6.9 and 6.10 show the runtime of the different steps for the different approaches.

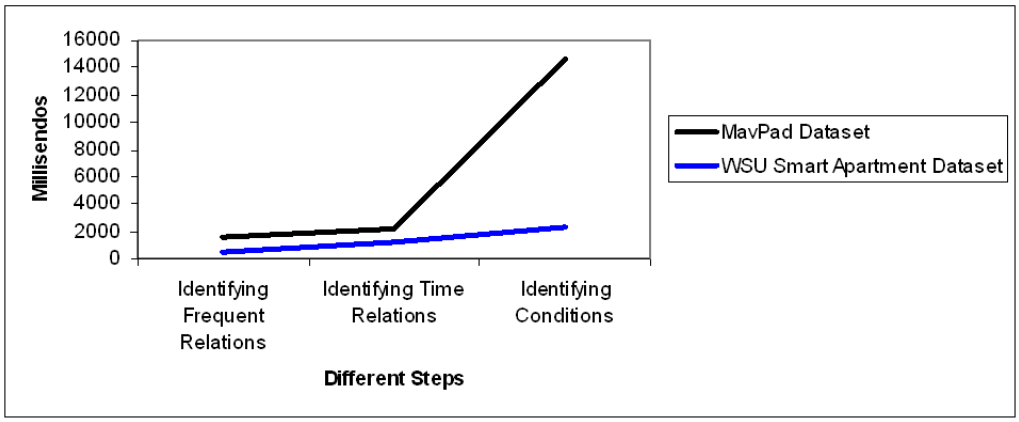

Figure 6.9: The runtime of different steps of the Pairwise Approach.

In the Pairwise Approach the most time-consuming step in both cases is the 'Identifying Condition' step. The explanation for this is that in that approach, the conditions are identified for each discovered pattern. In contrast, in the Action Map Approach, the runtimes of 'Identifying Topology' and 'Identifying Time Relations' are higher. This occurs because, on the one hand, conditions are only identified when needed, and on the other hand, the

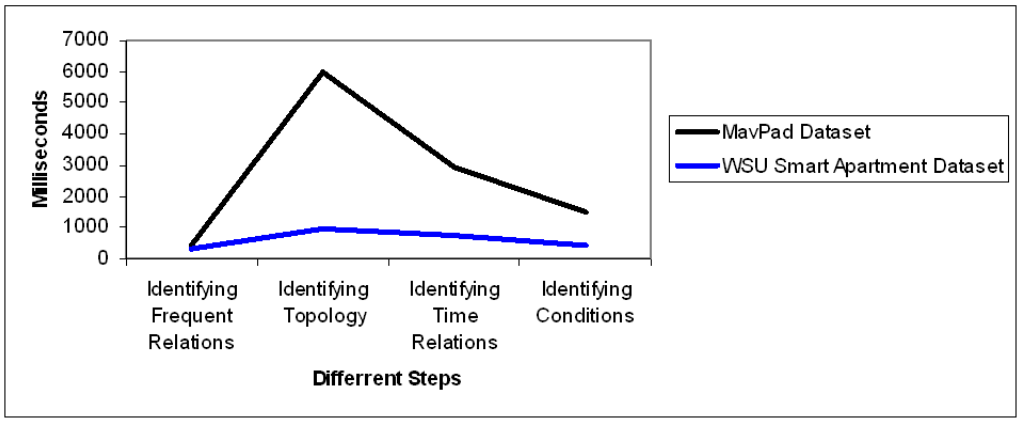

Figure 6.10: The runtime of different steps of the Action Map Approach.

'Identifying Topology' step includes many different subtasks such as discovering repetitive actions and unordered subsets of actions, which are very time-consuming.

6.5 Summary

Both the Pairwise Approach and the Action Map Approach were validated using data collected from two real environments (MavPad and WSU Smart Apartment). Different experiments were run in both cases to validate the efficiency of each one of the steps.

In addition, as different validations were carried out using the same datasets, it was possible to compare the output of both approaches in terms of the following aspects:

- Advantages over the other approach (e.g., knowledge representation).
- Efficiency in its task.
- Runtime.

Chapter 7

Conclusions and Further Research

A general abstract of the main results obtained from the attainment of the thesis is given in this chapter. Section 7.1 offers a global description of the research work. This work has resulted in a set of contributions, described in Section 7.2, in addition to a set of publications that are listed in Section 7.3. New research ideas and works that have been identified are described in Section 7.4. The final remarks are given in Section 7.5.

7.1 Conclusions

Intelligent Environments (IEs) are real environments (Smart Homes, Smart Classrooms etc.) that sensibly support people in their daily lives. This new term supposes a change of perspective in the relationships between humans 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. It is assumed that users' past and present frequent behaviours define their habits and preferences, so that IEs could provide personalised and adapted services if those behaviours were identified previously. Thus, the ability to discover users' frequent behaviours becomes an essential aspect for the successful implementation of IEs, allowing them to act proactively. Acting proactively could mean that the environment might automate some devices based on discovered users' habits, or it could also mean that unhealthy habits are detected.

IEs appear to be a new area where automatically learning algorithms used in other areas can be used in order to discover users' frequent behaviours; however, each area has different objectives, needs and features that can influence the learning process. In that sense, IEs also have some features which make these environments different from others. They were identified and analysed in Section 1.2.3.

Moreover, due to complexity of IEs (hardware, software and networks have to cooperate in an efficient and effective way to provide a suitable result to the user), the first developments were focused on needs associated with hardware and networking as the supporting infrastructure. This resulted in a simple automation that implements a reactive environment that did not take into account the personalised and adaptive features of IEs. Nevertheless, notable exceptions were found. Thus, Machine Learning techniques were identified as a possible so-

lution to automatically learn patterns and provide environments with intelligence. Different Machine Learning techniques, used for different research groups, were analysed taking into account the special features of IEs. It was concluded that still 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. Thus, the solution may rely on the combination of most of them, taking advantage of the strengths of each technique. The strengths and weaknesses of each technique were highlighted in Table 2.1.

Taking into account the need 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 the LFPUBS, and the Application Layer, which fills the gap from the LFPUBS to the real environment, are environment-dependent because their performance depends on specific aspects (e.g., types of sensors/actuators) of particular environments. On the contrary, 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 research work was to design and develop the necessary components of the Learning Layer in order to allow the LFPUBS to discover users' frequent behaviours. The internal architecture of the Learning Layer was made up of two components: the language component ($\mathcal{L}_{LFPUBS}$), which provides a standard conceptualisation of the patterns; and the algorithm component ($\mathcal{A}_{LFPUBS}$), which discovers the patterns. To this end, two different approaches have been developed.

The first approach, the Pairwise Approach, was aimed at discovering those pairs of actions that frequently occurred together. By relating users' actions among them, instead of relating them to context conditions, the understanding of their behaviours is facilitated. In addition to identifying the Frequent Relations, the Pairwise Approach was designed to discover the possible quantitative Time Relations between actions as well as the Conditions that defined under what situations the pattern occurred.

The second approach, the Action Map Approach, was an evolution of the Pairwise Approach. Pairwise relationships allowed one to relate only two actions, so that it had many limitations in order to represent a whole behaviour in a unique pattern. Thus, the Action Map Approach was designed to discover and represent users' frequent behaviours without any limitation on the number of actions involved in the pattern. These new requirements demanded a modification of both the language and the algorithm. The $\mathcal{L}_{LFPUBS}$ was extended to allow the system to represent a whole behaviour in an unique pattern; however, the component that needed a more substantial modification was the $\mathcal{A}_{LFPUBS}$. New steps that discovered frequent sets of actions and identified the frequent order (topology) of the actions involved in each of them were developed. These steps were undertaken while considering the inherent aspects of IEs where repetitive actions or unordered subsets of actions

can occur. Regarding the Conditions, two different types of conditions were needed in this new approach: Specific Conditions that identified under what conditions a relation between two actions becomes true; and General Conditions that contextualised, in terms of calendar information, the occurrence of the whole behaviour.

Both approaches were validated using data collected from two real environments: MavPad [You05] and WSU Smart Apartment [Coo08]. Validating both approaches using the same data allows one to compare the results and highlight the strengths and weaknesses of each approach.

- It was concluded that representing users' behaviours by means of Action Maps clearly facilitated the understanding of such behaviours. Whereas the Pairwise Approach provides independent pieces of knowledge, the Action Map Approach provides complete representation of the behaviours. Comparing the consistency of the knowledge discovered by both approaches at this stage, it was verified that the 100% of the relations defined by different topologies were also identified as Frequent Relations by the Pairwise Approach. Thus, although both approaches represent the discovered knowledge in different manners, it can be concluded that the outputs provided by both approaches were coherent among them.

- In order to identify quantitative Time Relations the same algorithm was used in both approaches; however, the effectiveness of the algorithm, that is, the percentage of the patterns where the algorithm was able to discover a quantitative Time Relation, increased with the Action Map Approach (e.g., 16% in the WSU Smart Apartment dataset). This is because, by considering repetitive actions, the system was able to better define the nature of each occurrence, and therefore more accurate Time Relations were identified.

- The objective of the conditions in both approaches was to contextualise the discovered knowledge. Even so, the nature of the conditions demanded for each one of the approaches was different. In the Pairwise Approach, conditions were identified for each one of the relations. In the Action Map Approach, however, two different types of conditions were demanded: Specific Conditions for situations where more than one possibility were considered; and General Conditions to contextualise the whole Action Map. However, it is worth noting that the effectiveness of this step was 78% and 69% (considering the MavPad dataset) in the Pairwise and in the Action Map Approaches, respectively.

Finally, the runtimes of different validation processes were compared. Thus, it was learned that in the Pairwise Approach the most time consuming step was the 'Identifying Condition' due to the fact that conditions were discovered for each one of the patterns. On the contrary, in the Action Map Approach, the most time consuming step was the 'Identifying Topology' step, mainly due to the fact that it includes many demanding subtasks such as 'Identifying Repetitive Actions' or 'Identifying Unordered Subsets of Actions'.

7.2 Contributions

Based on the aforementioned conclusions, a summary of the main contributions of the thesis is presented in this section:

- A detailed state of the art has been given (see Chapter 2):
  - Considering the special features of IEs.
  - Identifying the strengths and weaknesses of each one of the Machine Learning techniques.
- A general architecture for the LFPUBS has been proposed (see Chapter 3) that:
  - Allows the system to separate those environment-independent aspects from those that are environment-dependent.
  - Allows one to include in the system any transformation process identified in any particular environment.
  - Allows one to design and develop a Learning Layer free of any external influence.
  - Allows one to use discovered knowledge in order to achieve the objectives of particular environments. A speech-based interaction system was designed and implemented in order to facilitate the development of easy-to-use applications.
- A Pairwise Approach that discovers frequent relations between pairs of actions has been developed (see Chapter 4). It includes:
  - A language ($\mathcal{L}_{LFPUBS}$) that provides a standard conceptualisation of the patterns.
  - An algorithm ($\mathcal{A}_{LFPUBS}$) that discovers pairs of actions that are frequently related. In addition, $\mathcal{A}_{LFPUBS}$ discovers quantitative Time Relations and Conditions for each of the relations.
- An evolution of the Pairwise Approach (the Action Map Approach), that discovers users' frequent behaviours has been developed (see Chapter 5). It includes:
  - A new version of the $\mathcal{L}_{LFPUBS}$ that is able to represent whole behaviours.
  - A new $\mathcal{A}_{LFPUBS}$ that discovers the Frequent Set of Actions, the frequent Topology of such actions, the Time Relations and the Specific and General Conditions.
- A Graphical User Interface that allows the user of the LFPUBS to configure different parameters of the system to carry out the desired learning process.

7.3 Relevant Publications

The ideas of the research work presented here have been set forth in international research forums, such as: journals, book chapters and conferences.

7.3.1 International Journals

- A. Aztiria, A. Izaguirre and J. C. Augusto '*Learning patterns in Ambient Intelligence environments: A Survey*'. Artificial Intelligence Review, Springer. 2010.

7.3.2 Book Chapters

- A. Aztiria, A. Izaguirre, R. Basagoiti and J. C. Augusto '*Learning about preferences and common behaviours of the user in an intelligent environment*'. Behaviour Monitoring and Interpretation, 'Ambient Intelligence and Smart Environments' book series, ed. Björn Gottfried, Hamid Aghajan, V.3, pp. 289-315, 2009.

7.3.3 International Conferences

- A. Aztiria, A. Izaguirre, R. Basagoiti, J.C. Augusto and D. Cook '*Automatic Modeling of Frequent User Behaviours in Intelligent environments*'. 6th International Conference on Intelligents Environments, 2010.
- A. Aztiria, J.C. Augusto, R. Basagoiti and A. Izaguirre '*Accurate Temporal Relationships in Sequences of User Behaviours in Intelligent Environments*'. International Symposium on Ambient Intelligence (ISAmI'2010), 2010.
- J. M. Lucas-Cuesta, J. Ferreiros, A. Aztiria, J. C. Augusto and M. F. McTear '*Dialogue-based Management of user feedback in an autonomous preference learning system*'. 2nd International conference on Agents and Artificial Intelligence (ICAART'2010). pp. 330-337, 2010.
- J. M. Lucas-Cuesta, A. Aztiria, M. F. McTear, J. C. Augusto, and J. Ferreiros '*Facilitating Preference Revision through a Spoken Dialogue System*'. Workshop Designing ambient interactions for older users, Co-located with the AMI 2009. 2009.
- A. Aztiria, A. Izaguirre, R. Basagoiti, J. C. Augusto, and D. Cook '*Discovering of Frequent Sets of Actions in Intelligent Environments*'. Proceedings of the 5th International Conference on Intelligent Environments, pp 153-160, 2009.
- A. Aztiria, J. C. Augusto, A. Izaguirre and D. Cook '*Learning Accurate Temporal Relations from User Actions in Intelligent Environments*'. Proceedings of the 3th Symposium of Ubiquitous Computing and Ambient Intelligence. 2008.
- A. Aztiria, J. C. Augusto and A. Izaguirre '*Autonomous Learning of User's Preferences improved through User Feedback*'. Proceedings of the 2nd Workshop on Behaviour Monitoring and Interpretation (BMI'08), Co-located with the German Conference on AI. pp. 72-86, 2008.
- A. Aztiria, J. C. Augusto and A. Izaguirre '*Spatial and Temporal Aspects for Pattern Representation and Discovery in Intelligent Environments*'. Proceedings of the Workshop on Spatial and Temporal Reasoning, co-located with ECAI. 2008.

7.4 Future Work

Different topics that require deeper research have been detected throughout this research work. They are described below, grouped by each important chapter of the thesis.

7.4.1 Improving the State of the Art

The state of the art could be improved considering other techniques that have shown their utility in similar areas. Thus, techniques such as Markov Models, Bayesian networks or Support Vector Machines (SVMs) that have been widely used in context-awareness [Wu09; Duo06; Kas07] and pattern recognition [Bur04] areas could make an interesting contribution in the learning process. For that, it would be interesting to consider them together with the previous techniques and identify their strengths and weaknesses.

7.4.2 Improving the Architecture

This research work has been mainly focused on developing an environment-independent Learning Layer. Although Transformation and Application Layers depend on specific aspects of particular environments, some functionalities can be added to these layers to facilitate some tasks for LFPUBS's users.

Within the Transformation Layer, one of the main tasks that will probably take place in almost all the environments is the identification of actions and activities from collected data. For that, templates that define such inferences have been used in this research work (see Section 3.1) . Being aware of the effort that some research groups [Wu09; Brd05; Tap04] are doing in the area of context-awareness in IEs, including a context-awareness component in the Transformation Layer would help the LFPUBS serve as a holistic approach to discovering users' frequent behaviours.

The same idea can be extrapolated to the Application Layer. In that sense, an interesting functionality to add would be a translator or parser that translates the discovered knowledge into other types of representation that allow their use in particular applications. For example, it would be interesting to analyse the possibility of translating the discovered patterns into HMMs in order to use them to automate some devices. In that sense, the definition of the $\mathcal{L}_{LFPUBS}$ will greatly facilitate this task due to the fact that it provides a standard structure of the patterns.

7.4.3 Improving the Action Map Approach

Considering that the Action Map Approach is an evolution of the Pairwise Approach, some improvements are considered over this last approach. Regarding the steps defined in the Action Map Approach, some improvements related to Time Relations and Conditions were identified.

- So far, only exact quantitative Time Relations (e.g., 4 seconds) or qualitative Time Relations (e.g. 'after') have been considered. Other types of Time Relations can also

be considered (e.g., ranges such as 10-15 minutes) that provide more possibilities to define a Time Relation and allow the system to define a relation as accurate as possible.

- When it comes to discovering Specific Conditions, information provided by all of the context sensors is considered equally, without considering the nature of each of them. Thus, it could happen that the discovered conditions correctly separate the different occurrences, but analysing the nature of the actions to be separated does not provide a meaningful explanation. For that, it would be interesting to add semantic information [Dig09] to both actions and context information so that, in the process of discovering conditions, only context information that is related to the nature of the actions would be considered.
- One of the main shortcomings of the current approach is related to the General Conditions. As mentioned in the validation process (see Section 6.3), the current algorithm, with the objective of covering all the occurrences, creates General Conditions that do not allow one to discern different Action Maps. Thus, it is necessary to develop an algorithm that discovers more accurate General Conditions.

As mentioned in Section 4.1, other types of sensors (e.g., those that indicate the health status of the user [Erm08; Sta04]) can provide interesting information that would help to better define their behaviours. This would require adapting both $\mathcal{L}_{LFPUBS}$ and $\mathcal{A}_{LFPUBS}$ to include this new type of information in the process of learning.

7.4.4 Improving the Validation

The LFPUBS has been validated using data collected from two real environments. One of them showed users behaving without any predefined behaviour, whereas in the other participants behaved based on a predefined pattern. Behaving in a more controlled way seems difficult, although people could be predefined to even consider the Time Distances between the actions they perform, so that quantitative Time Relations to be discovered will also be known in advance.

Data collected from other Smart Homes or even from other types of IEs, such as Smart Cars or Smart Classrooms, could be used to validate the current (and future) approach of the LFPUBS.

7.4.5 More General Improvements

Finally, going a step further, some more ambitious research lines can be considered.

One of them would be to include in the LFPUBS all the learning periods defined in Section 1.2.3. So far, LFPUBS only considers to behave intelligently once patterns that define users' frequent behaviours have been discovered. However, two more learning periods were identified. The first one considered acting as intelligently as possible without patterns while the system is collecting data. The last learning period considered that, once the environment is acting in accordance with patterns discovered by the LFPUBS, it is necessary to adapt those patterns in a continuous way because users' frequent behaviours may change.

7.5 Final Remarks

With this research work we have tried to do our part in an area where few efforts have been made so far, although its importance is recognised by everyone. Interesting conclusions have been achieved and different future research lines have been defined to go a step further and get closer to truly Intelligent Environments.

Conclusiones y Líneas Futuras

En el presente capítulo se ofrece un resumen general de los principales resultados obtenidos a lo largo de la investigación llevada a cabo en la presente tesis. Primero se realiza una descripción global del trabajo realizado. Este ha dado lugar a una serie de contribuciones, las cuales son descritas a continuación, además de una serie de publicaciones. A continuación, se describen las diferentes líneas futuras que se han detectado y que pueden dar lugar a nuevas líneas de investigación. Por último se describen las consideraciones finales.

Conclusiones

Los Entornos Inteligentes (EIs) son entornos reales (Casas Inteligentes, Aulas Inteligentes, ...) que tratan de ayudar a las personas a llevar a cabo sus tareas diarias. Esta nuevo concepto de EIs lleva consigo un cambio de perspectiva en la relación entra la tecnología y sus usuarios. Desde una perspectiva centrada en la tecnología, donde el usuario tenía que aprender cómo utilizar esta, se pasa a una perspectiva centrada en el usuario, donde la tecnología es la que se adapta a este. Para ello, un EI debe aprender cómo actuar ante las acciones y necesidades del usuario, todo ello de una forma transparente y no obtrusiva. Se considera que las personas son 'animales de costumbres' por lo que sus comportamientos pasados y presentes definirán en gran medida sus futuras acciones. Así, la capacidad de descubrir los comportamientos frecuentes de los usuarios se convierte en un aspecto importante para los EIs para así poder actuar proactivamente (automatizando ciertas acciones o detectando hábitos no saludables).

La problemática de aprender comportamientos frecuentes de los usuarios en EIs parece un área propicia para la aplicación de las técnicas de aprendizaje automático utilizadas hasta la fecha en otras áreas. Sin embargo, cada área tiene diferentes objetivos, necesidades y características que pueden influenciar dicha utilización. En este sentido, los EIs también cuentan con ciertas particularidades que pueden influenciar el proceso de aprendizaje. Dichas características son identificadas y analizadas en la Sección 1.2.3.

La complejidad de dichos entornos viene dada por la necesidad de integrar de una forma eficiente el hardware, el software y las redes de comunicaciones, para proporcionar un resultado apropiado al usuario. En consecuencia, los primeros esfuerzos se centraron en desarrollar las infraestructuras necesarias (hardware y redes de comunicaciones). Dichos entornos funcionaban de forma reactiva sin proveer servicios personalizados y adaptados. Las técnicas de Aprendizaje Automático fueron identificadas como las técnicas idóneas para proveer de inteligencia dichos entornos. Las diferentes técnicas de Aprendizaje Automático utilizadas

por diferentes grupos de investigación fueron analizados teniendo en cuenta las características especiales de los EIs. Las fortalezas y debilidades de cada una de ellas han sido resumidas en la Tabla 2.1. Como conclusión de este análisis se puede decir que todavía no ha sido desarrollado un sistema que aprenda de forma rápida, a partir de poca información, con precisión, siendo independiente del entorno donde se aplica y proveyendo un modelo comprensible y novedoso.

Teniendo en cuenta la necesidad de aprendizaje, las características especiales de los EIs y el estado del arte actual, un sistema que descubre de forma automática los comportamientos frecuentes de los usuarios ha sido desarrollado. El sistema, Learning Frequent Patterns of User Behaviour System (LFPUBS), se basa en una arquitectura de tres capas que separa aquellos aspectos que dependen del entorno particular donde se esta aplicando de aquellos aspectos que son independientes. Tanto la capa de Transformación como la de Aplicación son dependientes del entorno particular debido a que su actuación depende de aspectos tales como los tipos de sensores/actuadores que hay en dicho entorno. Por el contrario, los componentes de la capa de Aprendizaje, que contiene todos los algoritmos que descubren los comportamientos frecuentes de los usuarios, son independientes del entorno particular donde se aplican.

El principal objetivo de este trabajo ha sido diseñar y desarrollar los componentes necesarios de la capa de Aprendizaje que permitan el descubrimiento automático de los comportamientos frecuentes de los usuarios. La arquitectura interna de la capa de Aprendizaje está compuesta por dos módulos principales: el lenguaje ($\mathcal{L}_{LFPUBS}$), que permite estandarizar la representación de los patrones descubiertos, y el algoritmo ($\mathcal{A}_{LFPUBS}$) que trata de descubrir dichos patrones. Para ello, dos diferentes propuestas fueron desarrolladas.

La primera propuesta, el Pairwise Approach, trata de descubrir parejas de acciones que frecuentemente occurren de forma consecutiva. Además de identificar las relaciones frecuentes, el Pairwise Approach descubre las relaciones temporales existentes entre dichas acciones así como las condiciones que determinan bajo qué circunstancias esa relación es cierta.

La segunda propuesta, el Action Map Approach, es una evolución del Pairwise Approach debido a que no se limita a descubrir sólo parejas de acciones, sino que su objetivo es descubrir aquellos conjuntos de acciones sin limitación en su número, que a su vez representan los comportamientos frecuentes de los usuarios. Ello implica que tanto el lenguaje como el algoritmo desarrollado para el Pairwise Approach hayan tenido que ser modificados. El $\mathcal{L}_{LFPUBS}$ fue extendido para permitir la representación del comportamiento global en un único patrón. El $\mathcal{A}_{LFPUBS}$ necesitó de modificaciones significativas; fue necesario desarrollar nuevos módulos para descubrir los conjuntos de acciones frecuentes e identificar el modelo (topología) para dichos conjuntos. En lo que se refiere a las condiciones, dos tipos fueron identificados en esta nueva versión: condiciones específicas que definen bajo qué circunstancias una relación definida por dos acciones es real, y condiciones generales que contextualizan el comportamiento global.

Ambas propuestas fueron validadas utilizando datos recogidos de dos entornos reales: MavPad [You05] y WSU Smart Apartment [Coo08]. La utilización de los mismos datos para

la validación de ambas propuestas permitió comparar los resultados obtenidos en cada una de ellas y definir sus fortalezas y debilidades.

- La representación del comportamiento mediante Action Maps facilita su comprensión dado que, mientras el Pairwise Approach provee pequeños patrones independientes, el Action Map Approach provee una representación global del comportamiento. En lo que se refiere a la consistencia de los resultados obtenidos, se comprobó que el total (100%) de las relaciones definidas por las diferentes topologías fueron identificadas como frecuentes por el Pairwise Approach. Así, aunque ambas propuestas representen de forma diferente el conocimiento adquirido, se puede concluir que ambas descubren el mismo conocimiento.

- Para la identificación de relaciones temporales el mismo algoritmo fue utilizado en ambas propuestas. Aún así, la efectividad de aplicar dicho algoritmo, es decir el porcentaje de relaciones donde el algoritmo fue capaz de identificar relaciones temporales, se incrementó en un 16% a la hora de aplicarlo en el Action Map Approach. Este incremento es debido a que el descubrimiento de acciones repetitivas permite definir mejor la naturaleza de cada una de ellas, de modo que relaciones temporales mucho más precisas son identificadas.

- El objetivo de las condiciones, en ambas propuestas, es la de contextualizar los patrones descubiertos. Aún así, la naturaleza de las condiciones exigidas en cada una de ellas varia de forma ostensible. En el Pairwise Approach, las condiciones fueron identificadas por cada relación. Sin embargo, en el Action Map Approach, dos tipos diferentes de condiciones fueron identificados. Por un lado, las condiciones específicas que definen aquellas situaciones donde más de una relación es posible, y por otro las condiciones generales que tratan de contextualizar el comportamiento global. Cabe mencionar la efectividad de ambas propuestas a la hora de descubrir tales condiciones (78% y 69% respectivamente).

Finalmente, los tiempos de ejecución de los diferentes experimentos fueron comparados. Se puede concluir que en el Pairwise Approach, la tarea más exigente en términos de tiempo de ejecución es la de identificar condiciones debido al hecho de que estas tenían que ser identificadas para todas las relaciones. Por el contrario, en el Action Map Approach, la tarea de identificar la topología fue la más exigente.

Contribuciones

Partiendo de las conclusiones obtenidas a la finalización de la tesis descritas previamente, en esta sección se ofrece una síntesis de las principales contribuciones de la tesis.

- Se ha realizado un análisis exhaustivo del estado del arte (ver Capítulo 2):
    - Considerando las características especiales de los EIs.

    * Identificando las fortalezas y debilidades de cada una de las técnicas de Aprendizaje Automático.

- Se ha propuesto una arquitectura general para el LFPUBS (ver Capítulo 3) que:
    * Permite al sistema separar aquellos aspectos dependientes de entornos particulares de aquellos que son independientes.
    * Permite incluir en el sistema cualquier proceso de transformación identificada en cualquier entorno.
    * Permite desarrollar una capa de Aprendizaje exenta de cualquier inferencia externa.
    * Permite utilizar el conocimiento descubierto para alcanzar los objetivos de cada entorno particular. En este sentido, un sistema, basado en la interacción por voz, ha sido desarrollado para facilitar el desarrollo de aplicaciones fáciles de utilizar.
- Un sistema (Pairwise Approach) que descubre relaciones frecuentes entre dos acciones ha sido desarrollado (ver Capítulo 4). Incluye:
    * Un lenguaje ($\mathcal{L}_{LFPUBS}$) que estandariza la representación de los patrones descubiertos.
    * Un algoritmo ($\mathcal{A}_{LFPUBS}$) que descubre pares de acciones relacionadas frecuentemente. Además, $\mathcal{A}_{LFPUBS}$ identifica a su vez las relaciones temporales y las condiciones para cada una de ellas.
- Una evolución del primer sistema (Action Map Approach), que descubre comportamientos frecuentes de los usuarios ha sido desarrollado (ver Capítulo 5). Incluye:
    * Una nueva versión del $\mathcal{L}_{LFPUBS}$ que es capaz de representar el comportamiento global en un único patrón.
    * Un nuevo $\mathcal{A}_{LFPUBS}$ que descubre los conjuntos de acciones que definen dicho comportamiento, el orden frecuente (topología) de dichas acciones así como las relaciones temporales y las condiciones necesarias.
- Se ha desarrollado un Interfaz Gráfico que permite al usuario del LFPUBS configurar los diferentes parámetros del proceso de aprendizaje.

Publicaciones Relevantes

El trabajo aquí presentado ha sido expuesto en foros de investigación internacionales. Entre ellos se encuentran revistas, capítulos de libros y conferencias.

Revistas Internacionales

- A. Aztiria, A. Izaguirre and J. C. Augusto '*Learning patterns in Ambient Intelligence environments: A Survey*'. Artificial Intelligence Review, Springer. 2010.

Capítulos de libro

- A. Aztiria, A. Izaguirre, R. Basagoiti and J. C. Augusto '*Learning about preferences and common behaviours of the user in an intelligent environment*'. Behaviour Monitoring and Interpretation, 'Ambient Intelligence and Smart Environments' book series, ed. Björn Gottfried, Hamid Aghajan, V.3, pp. 289-315, 2009.

Conferencias Internacionales

- A. Aztiria, A. Izaguirre, R. Basagoiti, J.C. Augusto and D. Cook '*Automatic Modeling of Frequent User Behaviours in Intelligent environments*'. 6th International Conference on Intelligents Environments, 2010.
- A. Aztiria, J.C. Augusto, R. Basagoiti and A. Izaguirre '*Accurate Temporal Relationships in Sequences of User Behaviours in Intelligent Environments*'. International Symposium on Ambient Intelligence (ISAmI'2010), 2010.
- J. M. Lucas-Cuesta, J. Ferreiros, A. Aztiria, J. C. Augusto and M. F. McTear '*Dialogue-based Management of user feedback in an autonomous preference learning system*'. 2nd International conference on Agents and Artificial Intelligence (ICAART'2010). pp. 330-337, 2010.
- J. M. Lucas-Cuesta, A. Aztiria, M. F. McTear, J. C. Augusto, and J. Ferreiros '*Facilitating Preference Revision through a Spoken Dialogue System*'. Workshop Designing ambient interactions for older users, Co-located with the AMI 2009. 2009.
- A. Aztiria, A. Izaguirre, R. Basagoiti, J. C. Augusto, and D. Cook '*Discovering of Frequent Sets of Actions in Intelligent Environments*'. Proceedings of the 5th International Conference on Intelligent Environments, pp 153-160, 2009.
- A. Aztiria, J. C. Augusto, A. Izaguirre and D. Cook '*Learning Accurate Temporal Relations from User Actions in Intelligent Environments*'. Proceedings of the 3th Symposium of Ubiquitous Computing and Ambient Intelligence. 2008.
- A. Aztiria, J. C. Augusto and A. Izaguirre '*Autonomous Learning of User's Preferences improved through User Feedback*'. Proceedings of the 2nd Workshop on Behaviour Monitoring and Interpretation (BMI'08), Co-located with the German Conference on AI. pp. 72-86, 2008.
- A. Aztiria, J. C. Augusto and A. Izaguirre '*Spatial and Temporal Aspects for Pattern Representation and Discovery in Intelligent Environments*'. Proceedings of the Workshop on Spatial and Temporal Reasoning, co-located with ECAI. 2008.

Líneas Futuras

A lo largo de la presente tesis se han ido detectando diferentes líneas que requieren profundización científica. A continuación son descritas agrupándolas según cada uno de los capítulos

principales de la tesis.

Estado del Arte

El estado del arte podría ser mejorado considerando otras técnicas que han mostrado su capacidad de aprendizaje en otras áreas. Técnicas tales como Modelos de Markov, Redes Bayesianas o Support Vector Machines (SVMs) han sido utilizados en otras áreas como la identificación de situaciones [Wu09; Duo06; Kas07] y reconocimiento de patrones [Bur04], y pueden aportar aspectos interesantes a la problemática tratada en este trabajo. Sería interesante considerarlos junto con el resto de las técnicas para identificar sus fortalezas y debilidades.

Arquitectura de LFPUBS

El objetivo principal de dicha arquitectura ha sido la de posibilitar el desarrollo de una capa de Aprendizaje libre de cualquier inferencia externa. Este trabajo se ha centrado en el desarrollo de dicha capa. Aunque las capas de Transformación y Aplicación sean dependientes de los entornos donde se aplican, algunas funcionalidades que puedan ser generalizables para muchos entornos podrían ser incluidas en el LFPUBS para facilitar su utilización.

Dentro de la capa de Transformación, una de las tareas cuya generalización aportaría grandes ventajas a la hora de utilizar el LFPUBS es la de identificar acciones y actividades que el usuario este llevando a cabo (context-awareness). Siendo conscientes del interés de esta área y de los trabajos realizados por varios grupos de investigación [Wu09; Brd05; Tap04], integrar un componente que facilite dicha tarea permitiría al LFPUBS proveer una solución general para el descubrimiento de comportamientos frecuentes en diferentes entornos.

La misma idea podría ser extrapolada a la capa de Aplicación. En este caso, desarrollar un componente que traduzca el conocimiento descubierto por LFPBUS a otros tipos de representaciones facilitaría alcanzar los objetivos particulares de cada entorno. Por ejemplo, sería interesante analizar la posibilidad de trasladar el conocimiento descubierto por LFPUBS a cadenas de Markov para poder automatizar la activación/desactivación de ciertos dispositivos. En este sentido, la estandarización de los patrones por parte del $\mathcal{L}_{LFPUBS}$ facilita de forma considerable dicha tarea.

Action Map Approach

Siendo la propuesta de Action Map la evolución de la primera propuesta, las líneas futuras han sido identificadas sobre esta última. La mayoría de las mejoras a considerar son relativas a las tareas de identificación de relaciones temporales e identificación de condiciones.

- LFPUBS, hasta la fecha, considera solamente relaciones temporales precisas (por ejemplo, 4 segundos) o relaciones temporales cualitativas (por ejemplo, 'despues'). A mayores de estas dos formas de representar las relaciones temporales, otras representaciones intermedias podrían ser consideradas. Por ejemplo, cuando no es posible definir

una relación temporal exacta, la consideración de rangos de tiempo (por ejemplo, 10-15 minutos) podrían proveer información más precisa que las relaciones temporales cualitativas.

- Cuando se trata de descubrir condiciones específicas, la información proveída por diferentes tipos de sensores es considerada por igual, sin tener en cuenta el significado de cada uno de ellos. De esta forma, podría suceder que las condiciones descubiertas por el sistema consigan su objetivo de separar pero que dicha separación no provea una explicación significativa teniendo en cuenta la naturaleza de las acciones involucradas. La utilización de información semántica para definir la naturaleza de las acciones y la información del entorno [Dig09] podría ayudar a descubrir condiciones mucho más precisas.

- Un aspecto a mejorar es el proceso que descubre las condiciones generales. El algoritmo actual, con el objetivo de cubrir todas las ocurrencias, genera condiciones demasiado amplias que dificultan discernir entre los diferentes comportamientos.

Como ha sido mencionado en la Sección 4.1, otros tipos de sensores, por ejemplo, aquellos que indican el estado actual del usuario (ritmo cardiaco, nivel de azúcar,...) [Erm08; Sta04], podrían proveer información que mejorase la calidad del conocimiento descubierto. Cabe mencionar, que esta consideración podría exigir la modificación tanto del $\mathcal{L}_{LFPUBS}$ como del $\mathcal{A}_{LFPUBS}$.

Validación

El LFPUBS ha sido validado utilizando datos provenientes de dos entornos reales. En uno de ellos (MavPad), los usuarios no tenían ningún comportamiento predefinido mientras que en el otro (WSU Smart Apartment) se sabía de antemano que los datos representaban a los usuarios realizando ciertas tareas. Definir un comportamiento más controlado se antoja difícil aunque siempre se puede considerar la posibilidad de definir los lapsos de tiempo entre acción y acción.

También sería interesante validar el actual (y futuro) sistema considerando otros tipos de EIs, como por ejemplo Coches Inteligentes, Aulas Inteligentes, etc.

Otras Líneas Futuras

Finalmente, se han identificado mejoras considerando aspectos más generales.

Como se ha definido en la Sección 1.2.3, se podrían considerar tres periodos de tiempo para que un entorno actúe de forma inteligente. Este trabajo se ha centrado en descubrir los patrones que definen los comportamientos frecuentes y actuar en base a ellos, pero como se ha mencionado con anterioridad otras situaciones demandan otras soluciones. El primer periodo considera la posibilidad de actuar de forma inteligente mientras se estén recogiendo los datos, es decir, sin descubrir los patrones. El último periodo de aprendizaje considera la necesidad de ir adaptando su conocimiento adquirido debido al hecho de que los usuarios

podrían modificar sus comportamientos frecuentes. Rediseñar y desarrollar una nueva versión de LFPUBS que actuase de forma inteligente en los tres periodos sería el último paso a dar.

Consideraciones Finales

Esta tesis ha tratado de aportar su granito de arena en un ámbito como el del aprendizaje automático de comportamientos frecuentes de los usuarios. Además de presentar una nueva propuesta, se observa que esta tesis abre nuevas vías de investigación para alcanzar entornos que actúen de forma inteligente.

Chapter 8

Appendix

Appendix A: Example of Data Collected in Michael's scenario

A-Type Information

(date;device;status;value)

2008-10-20 (Sequence 1)
08:02:12;Alarm;on;100
08:15:55;Bathroom;on;100
08:15:57;BathroomLights;on;100
08:17:10;Cabinet;on;100
08:17:15;Mouthwash;on;100
08:17:16;Cabinet;off;0
08:19:23;Cabinet;on;100
08:19:29;Towel;on;100
08:19:30;Gel;on;100
08:19:31;Cabinet;off;0
08:21:46;Shower;on;100
08:29:37;Shower;off;0
08:29:41;BathroomFan;on;100
08:32:57;BathroomFan;off;0
08:33:15;BathroomLights;off;0
08:33:41;Bathroom;off;0

C-Type Information

(date;device;status;value)

2008-10-20 (Sequence 1)
08:02:14;TempBathroom;on;18
08:05:19;HumBathroom;on;65
08:13:42;TempBathroom;on;19
08:18:40;TempBathroom;on;20
08:22:07;HumBathroom;on;66
08:27:19;TempBathroom;on;18
08:28:20;HumBathroom;on;70
08:29:27;TempBathroom;on;19
08:29:35;HumBathroom;on;72
08:29:58;HumBathroom;on;71
08:31:18;HumBathroom;on;68
08:33:27;TempBathroom;on;18

2008-10-21 (Sequence 2)

08:10:50;Alarm;on;100
08:23:18;Bathroom;on;100
08:23:20;BathroomLights;on;100
08:23:58;Cabinet;on;100
08:24:02;Mouthwash;on;100
08:24:03;Cabinet;off;0
08:25:04;Tap;on;100
08:25:54;Tap;off;0
08:26:48;BathroomLights;off;0
08:27:04;Bathroom;off;0

2008-10-21 (Sequence 2)

08:11:41;TempBedroom;on;22
08:12:50;HumBathroom;on;50
08:21:25;TempBathroom;on;19
08:22:49;TempBathroom;on;20
08:24:21;HumBathroom;on;53
08:25:18;TempBathroom;on;22
08:25:57;HumBathroom;on;54
08:26:22;HumBathroom;on;54

2008-10-22 (Sequence 3)

08:06:19;Alarm;on;100
08:21:55;Bathroom;on;100
08:21:58;BathroomLights;on;100
08:22:10;Cabinet;on;100
08:22:16;Mouthwash;on;100
08:22:17;Cabinet;off;0
08:23:07;Cabinet;on;100
08:23:08;Gel;on;100
08:23:09;Towel;on;100
08:23:10;Cabinet;off;0
08:24:50;Shower;on;100
08:28:18;Shower;off;0
08:35:43;BathroomLights;off;0
08:36:10;Bathroom;off;0

2008-10-22 (Sequence 3)

08:15:41;TempBathroom;on;18
08:16:16;HumBathroom;on;62
08:22:05;HumBathroom;on;63
08:22:41;TempBathroom;on;19
08:23:05;TempBathroom;on;20
08:24:58;HumBathroom;on;65
08:25:02;TempBathroom;on;21
08:25:51;HumBathroom;on;66
08:27:00;TempBathroom;on;22
08:28:17;HumBathroom;on;65
08:29:04;HumBathroom;on;66
08:29:57;TempBathroom;on;21
08:31:12;HumBathroom;on;65
08:33:41;TempBathroom;on;20

2008-10-23 (Sequence 4)

08:16:39;Alarm;on;100
08:31:25;Bathroom;on;100
08:31:26;BathroomLights;on;100
08:35:17;Cabinet;on;100
08:35:22;Mouthwash;on;100
08:35:23;Cabinet;off;0
08:36:00;BathroomLights;off;0
08:36:02;Bathroom;off;0

2008-10-23 (Sequence 4)

08:28:53;TempBathroom;on;20
08:29:16;HumBathroom;on;64
08:33:18;TempBathroom;on;22
08:34:05;HumBathroom;on;65
08:34:08;TempBathroom;on;21
08:35:28;TempBathroom;on;64
08:36:00;HumBathroom;on;19

2008-10-24 (Sequence 5)

08:05:40;Alarm;on;100
08:18:55;Bathroom;on;100
08:18:57;BathroomLights;on;100
08:19:58;Cabinet;on;100
08:20:04;Mouthwash;on;100
08:20:05;Cabinet;off;0
08:22:34;Cabinet;on;100
08:22:42;Gel;on;100
08:22:43;Towel;on;100
08:22:44;Cabinet;off;0
08:23:18;Shower;on;100
08:29:28;Shower;off;0
08:29:31;BathroomFan;on;100
08:36:17;BathroomFan;off;0
08:38:57;BathroomLights;off;0
08:39:12;Bathroom;off;0

2008-10-24 (Sequence 5)

08:17:17;TempBathroom;on;21
08:19:21;TempBathroom;on;20
08:20:26;TempBathroom;on;21
08:22:12;HumBathroom;on;65
08:23:04;TempBathroom;on;20
08:26:18;TempBathroom;on;21
08:28:57;HumBathroom;on;71
08:29:00;TempBathroom;on;22
08:29:25;HumBathroom;on;75
08:30:05;TempBathroom;on;20
08:31:41;HumBathroom;on;72
08:34:10;HumBathroom;on;68

2008-10-27 (Sequence 6)

08:03:47;Alarm;on;100
08:15:06;Bathroom;on;100
08:15:09;BathroomLights;on;100
08:17:30;Cabinet;on;100
08:17:35;Mouthwash;on;100
08:17:36;Cabinet;off;0
08:18:13;Cabinet;on;100
08:18:32;Towel;on;100
08:18:33;Gel;on;100
08:18:34;Cabinet;off;0
08:25:42;Shower;on;100
08:30:39;Shower;off;0
08:36:03;BathroomFan;on;100
08:38:53;BathroomFan;off;0
08:46:02;BathroomLights;off;0
08:47:06;Bathroom;off;0

2008-10-27 (Sequence 6)

08:16:27;HumBathroom;on;65
08:18:08;TempBathroom;on;19
08:24:25;TempBathroom;on;20
08:26:48;HumBathroom;on;69
08:27:41;TempBathroom;on;21
08:28:09;HumBathroom;on;70
08:30:25;HumBathroom;on;71
08:32:42;TempBathroom;on;20
08:37:07;TempBathroom;on;21
08:38:41;TempBathroom;on;20
08:42:12;HumBathroom;on;68

2008-10-28 (Sequence 7)
08:10:39;Alarm;on;100
08:21:30;Bathroom;on;100
08:21:32;BathroomLights;on;100
08:22:37;Cabinet;on;100
08:22:42;Mouthwash;on;100
08:22:44;Cabinet;off;0
08:24:40;BathroomLights;off;0
08:24:46;Bathroom;off;0

2008-10-28 (Sequence 7)
08:20:45;TempBathroom;on;18
08:22:20;HumBathroom;on;66
08:21:02;TempBathroom;on;20
08:21:41;HumBathroom;on;65
08:22:18;TempBathroom;on;19
08:22:35;HumBathroom;on;66
08:24:40;TempBathroom;on;20

2008-10-29 (Sequence 8)
08:03:47;Alarm;on;100
08:16:28;Bathroom;on;100
08:16:29;BathroomLights;on;100
08:16:39;Cabinet;on;100
08:16:43;Mouthwash;on;100
08:16:44;Cabinet;off;0
08:18:33;Cabinet;on;100
08:18:58;Towel;on;100
08:18:59;Gel;on;100
08:19:00;Cabinet;off;0
08:20:06;Shower;on;100
08:23:29;Shower;off;0
08:23:34;BathroomFan;on;100
08:24:39;BathroomFan;off;0
08:25:08;BathroomLights;off;0
08:25:57;Bathroom;off;0

2008-10-29 (Sequence 8)
08:15:53;HumBathroom;on;66
08:16:02;TempBathroom;on;18
08:17:08;HumBathroom;on;65
08:19:41;HumBathroom;on;66
08:19:45;TempBathroom;on;17
08:20:30;TempBathroom;on;16
08:20:35;HumBathroom;on;70
08:22:24;TempBathroom;on;15
08:22:47;HumBathroom;on;75
08:23:02;TempBathroom;on;17
08:23:20;HumBathroom;on;78
08:24:36;HumBathroom;on;73
08:24:19;TempBathroom;on;18
08:25:20;TempBathroom;on;17
08:25:22;HumBathroom;on;71

2008-10-30 (Sequence 9)
08:16:42;Alarm;on;100
08:31:02;Bathroom;on;100
08:31:04;BathroomLights;on;100
08:32:33;BathroomLights;off;0
08:32:47;Bathroom;off;0

2008-10-30 (Sequence 9)
08:30:12;TempBathroom;on;21
08:31:18;HumBathroom;on;65
08:31:25;TempBathroom;on;22
08:31:58;TempBathroom;on;20
08:32:08;HumBathroom;on;67

2008-10-31 (Sequence 10)

08:10:27;Alarm;on;100
08:20:54;Bathroom;on;100
08:20:56;BathroomLights;on;100
08:21:10;Cabinet;on;100
08:21:16;Cabinet;off;0
08:22:50;Cabinet;on;100
08:22:58;Gel;on;100
08:22:59;Towel;on;100
08:23:00;Cabinet;off;0
08:25:38;Shower;on;100
08:29:07;Shower;off;0
08:36:27;BathroomLights;off;0
08:36:38;Bathroom;off;0

2008-10-31 (Sequence 10)

08:18:53;HumBathroom;on;65
08:19:16;TempBathroom;on;18
08:20:58;HumBathroom;on;66
08:21:10;TempBathroom;on;17
08:23:41;TempBathroom;on;16
08:24:10;HumBathroom;on;65
08:25:40;TempBathroom;on;17
08:28:46;HumBathroom;on;69
08:28:55;TempBathroom;on;18
08:31:57;HumBathroom;on;68
08:32:29;TempBathroom;on;19
08:33:47;TempBathroom;on;20
08:35:49;TempBathroom;on;67

Appendix B: Pairwise Approach, Language Specification

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

Event_Definition::= Primitive_Event|Composite_Event
  Primitive_Event::= User_Action
    User_Action::= occurs(Device, Action, time)
      Device::=  device_1|device_2|...|device_n
      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|Context_Sensor
        Calendar::= time of day|day of week|...
        Context_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(Action, Device,time) when Relation
    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
```

Appendix C: Pairwise Approach, Discovered Patterns

```
(Pattern 1)
 ON occurs (Alarm, On,t0)
 IF context ()
 THEN do (On, Bathroom, t)
      when t is after t0
```

```
(Pattern 2)
 ON occurs (BathroomLights, Off,t0)
 IF context ()
 THEN do (Off, Bathroom, t)
      when t = t0 + 1s
```

```
(Pattern 3)
 ON occurs (Bathroom, On,t0)
 IF context ()
 THEN do (On, BathroomLights, t)
      when t = t0 + 2s
```

```
(Pattern 4)
 ON occurs (BathroomFan, Off,t0)
 IF context ()
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(Pattern 5)
 ON occurs (Shower, Off,t0)
 IF context (Bathroom humidity level
            (<,70%))
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(Pattern 6)
 ON occurs (Cabinet, Off,t0)
 IF context (DayOfWeek (<>,Tuesday,
             Thursday, Friday))
 THEN do (Off, BathroomLights, t)
      when t is after t0
```

```
(Pattern 7)
 ON occurs (BathroomLights, On,t0)
 IF context ()

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

```
(Pattern 8)
 ON occurs (Cabinet, Off,t0)
 IF context (DayOfWeek (=,Tuesday
             Thursday, Friday))
 THEN do (On, Cabinet, t)
      when t is after t0
```

```
(Pattern 9)
 ON occurs (Mouthwash, On,t0)
 IF context ()
 THEN do (Off, Cabinet, t)
      when t = t0 + 1s
```

```
(Pattern 10)
 ON occurs (Gel, On,t0)
 IF context ()
 THEN do (Off, Cabinet, t)
      when t = t0 + 1s
```

```
(Pattern 11)
 ON occurs (Towel, On,t0)
 IF context ()
 THEN do (Off, BathroomLights, t)
      when t = t0 + 1s
```

```
(Pattern 12)
 ON occurs (Cabinet, On,t0)
 IF context ()
 THEN do (Off, BathroomLights, t)
      when t = t0 + 5s
```

```
(Pattern 13)
 ON occurs (Cabinet, On,t0)
 IF context ()
 THEN do (On, Towel, t)
      when t is after t0
```

```
(Pattern 14)
 ON occurs (Gel, On,t0)
 IF context ()
 THEN do (On, Towel, t)
      when t = t0 + 1s
```

```
(Pattern 15)
 ON occurs (Cabinet, On,t0)
 IF context ()
 THEN do (On, Gel, t)
      when t is after t0
```

```
(Pattern 16)
 ON occurs (Towel, On,t0)
 IF context ()
 THEN do (On, Gel, t)
      when t = t0 + 1s
```

```
(Pattern 17)
 ON occurs (Cabinet, Off,t0)
 IF context ()
 THEN do (On, Shower, t)
      when t is after t0
```

```
(Pattern 18)
 ON occurs (Shower, On,t0)
 IF context ()
 THEN do (Off, Shower, t)
      when t is after t0
```

```
(Pattern 19)
 ON occurs (Shower, Off,t0
 IF context (Bathroom humidity level
            (>,70%))
 THEN do (On, BathroomFan, t)
      when t = t0 + 4s
```

```
(Pattern 20)
 ON occurs (BathroomFan, On,t0)
 IF context ()

 THEN do (Off, BathroomFan, t)
      when t is after t0
```

Appendix D: Action Map Approach, Language Specification

```
ActionMap_Pattern::= ActionMap_Actions, General_Conditions
 ActionMap_Actions::= ActionPattern & ... & ActionPattern

 ActionPattern::= ON (Event_Definition)
                  IF (Condition_Definition)
                  THEN (Action_Definition)

 Event_Definition::= Primitive_Event|Composite_Event
  Primitive_Event::= User_Action
    User_Action::= occurs(Action, time)
     Action::= (Type, SetActions)|start
       Type::= simple|unordered
       SetActions::= SimpleActions|UnorderedActions
         SimpleActions::= Device, Action_Status
           Device::= device_1|device_2|...|device_n
           Action_Status::= on|off
         UnorderedActions::= SimpleActions &...& SimpleActions
  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::= User_Action when Relation
   User_Action::= do(Action)
    Action::= (Type, SetActions)|end
      Type::= simple|unordered
      SetActions::= SimpleActions|UnorderedActions
        SimpleActions::= Device, Action_Status
          Device::= device_1|device_2|...|device_n
          Action_Status::= on|off
        UnorderedActions::= SimpleActions &...& SimpleActions
   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

General_Conditions::= Condition_Definition
```

Appendix E: Action Map Approach, Discovered Patterns

```
(Action Map 1)

(General Condition)
 context (DayOfWeek (=,Monday,Tuesday,Wednesday,Thursday,Friday)) &
 context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<:09:00:00))

(Action Pattern 0)                      (Action Pattern 1)
 ON occurs (start,--,t0)                 ON occurs (simple,(Alarm,On),t0)
 IF context ()                           IF context ()
 THEN do (simple,(On,Alarm),t)           THEN do (simple,(On,Bathroom),t)
      when ---                                when t is after t0

(Action Pattern 2)                      (Action Pattern 3)
 ON occurs (simple,(Bathroom,On),t0)     ON occurs (simple,(BathroomLigts,On),t0)
 IF context ()                           IF context ()
 THEN do (simple,(On,BathroomLights),t) THEN do (simple,(On,Cabinet),t)
      when t = t0 + 2s                        when t is after t0

(Action Pattern 4)                      (Action Pattern 5)
 ON occurs (simple,(Cabinet(1),On),t0)   ON occurs (simple,(Cabinet(1),On),t0)
 IF context ()                           IF context ()
 THEN do (simple,(On,Mouthwash),t)       THEN do (simple,(Off,Cabinet(1)),t)
      when t = t0 + 5s                        when t is after t0

(Action Pattern 6)                      (Action Pattern 7)
 ON occurs (simple,(Mouthwash,On),t0)    ON occurs (simple,(Cabinet(1),Off),t0)
 IF context ()                           IF context (DayOfWeek (<>, Tuesday,
                                                    Thursday,Friday))
 THEN do (simple,(Off,Cabinet(1)),t)     THEN do (simple,(Off,BathroomLights),t)
      when t = t0 + 1s                        when t is after t0

(Action Pattern 8)                      (Action Pattern 9)
 ON occurs (simple,(Cabinet(1),Off),t0) ON occurs (simple,(Cabinet(2),On),t0)
 IF context (DayOfWeek (=,Tuesday,       IF context ()
             Thursday,Friday))
 THEN do (simple,(On,Cabinet(2)),t)      THEN do (unordered,((On,Towel)&(On,Gel)),t)
      when t is after t0                      when t is after t0
```

```
(Action Pattern 10)
 ON occurs (unordered,
           ((Towel,On)&(Gel,On)),t0)
 IF context ()
 THEN do (simple,(Off,Cabinet(2)),t)
      when t = t0 + 1s
```

```
(Action Pattern 11)
 ON occurs (simple,
           (Cabinet(2),Off),t0)
 IF context ()
 THEN do (simple,(On,Shower),t)
      when t is after t0
```

```
(Action Pattern 12)
 ON occurs (simple,(Shower,On),t0)
 IF context ()

 THEN do (simple,(Off,Shower),t)
      when t is after t0
```

```
(Action Pattern 13)
 ON occurs (simple,(Shower,Off),t0)
 IF context (Bathroom humidity level
              (<,70%))
 THEN do (simple,(Off,BathroomLights),t)
      when t is after t0
```

```
(Action Pattern 14)
 ON occurs (simple,(Shower,Off),t0)
 IF context (Bathroom humidity level
             (>,70%))
 THEN do (simple,(On,BathroomFan),t)
      when t = t0 + 4s
```

```
(Action Pattern 15)
 ON occurs (simple,(BathroomFan,On),t0)
 IF context ()

 THEN do (simple,(Off,BathroomFan),t)
      when t is after t0
```

```
(Action Pattern 16)
 ON occurs (simple,
           (BathroomFan,Off),t0)
 IF context ()
 THEN do (simple,
         (Off,BathroomLights),t)
      when t is after t0
```

```
(Action Pattern 17)
 ON occurs (simple,
           (BathroomLights,Off),t0)
 IF context ()
 THEN do (simple,
          (Off,Bathroom),t)
      when t = t0 + 1s
```

```
(Action Pattern 18)
 ON occurs (simple,(Bathroom,Off),t0)
 IF context ()
 THEN do (--,end,t) when --
```

Appendix F: Validating the Pairwise Approach with MavPad data

Note: For each trial, discovered Patterns are shown. Besides, for each Pattern the highest confidence level where they were discovered appears.

```
-----------
  Trial 1
-----------

(Pattern 1 (25%))
 ON occurs (TableLamp,On,t0)
 IF context (Temperature (<,318))
 THEN do (On,FloorLamp,t)
      when t = t0 + 1s

(Pattern 2 (25%))
 ON occurs (TableLamp,Off,t0)
 IF context (TimeOfDay (<,23:34:00))
 THEN do (Off,FloorLamp,t)
      when t = t0 + 2s

(Pattern 3 (25%))
 ON occurs (FloorLamp,On,t0)
 IF context (TimeOfDay(<,20:48:00))
 THEN do (On,TableLamp,t)
      when t = t0 + 0s

(Pattern 4 (50%))
 ON occurs (BedroomFan,On,t0)
 IF context ()
 THEN do (Off,BedroomFan,t)
      when t = t0 + 3s

(Pattern 5 (25%))
 ON occurs (ClosetLight, Off,t0)
 IF context ()

 THEN do (Off,BedroomFan,t)
      when t = t0 + 20s

(Pattern 6 (75%))
 ON occurs (BedroomLuxo1,On,t0)
 IF context (TimeOfDay (>,01:06:00)
                    & (<,02:46:00))
 THEN do (Off,BedroomLight,t)
      when t = t0 + 0s

(Pattern 7 (25%))
 ON occurs (BedroomLight,Off,t0)
 IF context (TimeOfDay (>,01:06:44)
                   & (<,02:46:32))
 THEN do (On,BedroomLuxo1,t)
      when t = t0 + 0s

(Pattern 8 (50%))
 ON occurs (BedroomLight,Off,t0)
 IF context ()

 THEN do (Off,BedroomLuxo1,t)
      when t = t0 + 61s

(Pattern 9 (25%))
 ON occurs (ClosetLight,Off,t0)
 IF context (LightLevel (<,44))
 THEN do (On,ClosetLight,t)
      when t = t0 + 15s

(Pattern 10 (25%))
 ON occurs (LivingRoomLight,Off,t0)
 IF context (LightLevel (>,36))
 THEN do (On,ClosetLight,t)
      when t = t0 + 573s
```

```
(Pattern 11 (25%))
 ON occurs (ClosetLight,On,t0)
 IF context (Temperature (>,149) & (<,151))
 THEN do (Off,ClosetLight,t)
      when t = t0 + 138s

(Pattern 12 (25%))
 ON occurs (ClosetLight,On,t0)
 IF context (LightLevel (>,36)& (<,44))
 THEN do (Off,ClosetLight,t)
      when t = t0 + 26s

(Pattern 13 (50%))
 ON occurs (BedroomLamp1,On,t0)
 IF context (LightLevel (=,105))
 THEN do (On,BedroomLamp2,t)
      when t is after t0

(Pattern 14 (25%))
 ON occurs (BedroomLamp1,Off,t0)
 IF context (LightLevel (<, 105))
 THEN do (On,BedroomLamp2,t)
      when t is after t0

(Pattern 15 (25%))
 ON occurs (BedroomLamp1,On,t0)
 IF context ()
 THEN do (Off,BedroomLamp2,t)
      when t is after t0

(Pattern 16 (50%))
 ON occurs (BedroomLamp1,Off,t0)
 IF context (LightLevel (=,105))
 THEN do (Off,BedroomLamp2,t)
      when t is after t0

-----------
  Trial 2
-----------

(Pattern 1 (25%))
 ON occurs (FloorLamp,Off,t0)
 IF context (LightLevel (=,151))
 THEN do (On,FloorLamp,t)
      when t = t0 + 5s

(Pattern 2 (25%))
 ON occurs (TableLamp,On,t0)
 IF context (TimeOfDay(>,21:21:00))
 THEN do (On,FloorLamp,t)
      when t = t0 + 6s

(Pattern 3 (50%))
 ON occurs (FloorLamp,On,t0)
 IF context (TimeOfDay(<,18:17:00))
 THEN do (Off,FloorLamp,t)
      when t = t0 + 9s

(Pattern 4 (50%))
 ON occurs (BathroomLight,Off,t0)
 IF context (TimeOfDay(>,20:06:00))
 THEN do (On,LivingRoomLight,t)
      when t = t0 + 71s

(Pattern 5 (25%))
 ON occurs (CounterLights,Off,t0)
 IF context (LightLevel (>,45))
 THEN do (Off,LivingRoomLight,t)
      when t = t0 + 19s

(Pattern 6 (50%))
 ON occurs (LivingRoomLight,Off,t0)
 IF context (LightLevel (>,9))
 THEN do (On,TableLamp,t)
      when t = t0 + 18s

(Pattern 7 (25%))
 ON occurs (CounterLights,On,t0)
 IF context (DayOfWeek (=,Sunday))
 THEN do (On,TableLamp,t)
      when t is after t0

(Pattern 8 (50%))
 ON occurs (TableLamp,On,t0)
 IF context (Temperature (>,148))
 THEN do (Off,TableLamp,t)
      when t = t0 + 6s
```

```
(Pattern 9 (50%))
 ON occurs (LivingRoomLight,Off,t0)
 IF context (DayOfWeek (=,Thursday))
 THEN do (Off,TableLamp,t)
      when t = t0 + 18s

(Pattern 10 (25%))
 ON occurs (BathroomLight,On,t0)
 IF context ()
 THEN do (On,CounterLights,t)
      when t = t0 + 18s

(Pattern 11 (25%))
 ON occurs (ShowerLight,Off,t0)
 IF context (TimeOfDay(>,21:44:00))
 THEN do (On,CounterLights,t)
      when t = t0 + 15s

(Pattern 12 (75%))
 ON occurs (CounterLights,On,t0)
 IF context (LightLevel (>,22))
 THEN do (Off,CounterLights,t)
      when t = t0 + 32s

(Pattern 13 (50%))
 ON occurs (ShowerLight,On,t0)
 IF context ()
 THEN do (On,BathroomLight,t)
      when t = t0 + 34s

(Pattern 14 (50%))
 ON occurs (ShowerLight,Off,t0)
 IF context (LightLevel (>,50))
 THEN do (Off,BathroomLight,t)
      when t = t0 + 6s

(Pattern 15 (75%))
 ON occurs (BathroomLight,On,t0)
 IF context (LightLevel (<,204))
 THEN do (On,BathroomFan,t)
      when t = t0 + 4s

(Pattern 16 (25%))
 ON occurs (ShowerLight,Off,t0)
 IF context (HumidityLevel (>,98))
 THEN do (Off,BathroomFan,t)
      when t = t0 + 0s

(Pattern 17 (25%))
 ON occurs (ShowerLight,On,t0)
 IF context (HumidityLevel (>,85)))
 THEN do (Off,ShowerLight,t)
      when t = t0 + 48s

(Pattern 18 (25%))
 ON occurs (CounterLights,On,t0)
 IF context (LightLevel (<,195))
 THEN do (Off,ShowerLight,t)
      when t = t0 + 26s

(Pattern 19 (75%))
 ON occurs (BathroomLight,Off,t0)
 IF context ()
 THEN do (Off,ShowerLight,t)
      when t = t0 + 3s

(Pattern 20 (25%))
 ON occurs (BedroomLight,On,t0)
 IF context (ReedSwitch (>,249))
 THEN do (On,BedroomFan,t)
      when t = t0 + 18s

(Pattern 21 (25%))
 ON occurs (BedroomFan,On,t0)
 IF context (Lightlevel (<,51))
 THEN do (Off,BedroomFan,t)
      when t is after t0

(Pattern 22 (25%))
 ON occurs (BedroomLight,Off,t0)
 IF context (LightLevel (<,53))
 THEN do (Off, BedroomFan,t)
      when t = t0 + 0s
```

```
(Pattern 23 (50%))
 ON occurs (BedroomLight,On,t0)
 IF context ()
 THEN do (Off,BedroomLight,t)
      when t = t0 + 43s
```

```
(Pattern 24 (25%))
 ON occurs (BedroomLight,On,t0)
 IF context (LightLevel (=,211))
 THEN do (On,BedroomLuxo1,t)
      when t = t0 + 5s
```

```
(Pattern 25  (50%))
 ON occurs (BedroomLight,On,t0)
 IF context (TimeOfDay (>,19:19:49)
            & (LightLevel (>,52)
            & (LightLevel (<,143))
 THEN do (Off,BedroomLuxo2,t)
      when t = t0 + 30s
```

```
(Pattern 26 (25%))
 ON occurs (BedroomLight,On,t0)
 IF context (LightLevel (=,6))


 THEN do (On,BedroomLuxo3,t)
      when t = t0 + 23s
```

```
(Pattern 27 (25%))
 ON occurs (BedroomLight,Off,t0)
 IF context (LightLevel (>,11))
 THEN do (On,BedroomLuxo3,t)
      when t = t0 + 9s
```

```
(Pattern 28 (75%))
 ON occurs (BedroomLuxo3,On,t0)
 IF context (Temperature (<,81))
 THEN do (Off,BedroomLuxo3,t)
      when t = t0 + 12s
```

```
(Pattern 29 (50%))
 ON occurs (BedroomLuxo2,Off,t0)
 IF context (Temperature (>,146))
 THEN do (Off,BedroomLuxo3,t)
      when t = t0 + 5s
```

```
(Pattern 30 (25%))
 ON occurs (BedroomLuxo4,Off,t0)
 IF context (LightLevel (<,109))
 THEN do (Off,BedroomLuxo3,t)
      when t = t0 + 19s
```

```
(Pattern 31 (25%))
 ON occurs (BedroomLuxo4,Off,t0)
 IF context (LightLevel (<,111))
 THEN do (On,BedroomLuxo4,t)
      when t = t0 + 14s
```

```
(Pattern 32 (25%))
 ON occurs (BedroomLuxo4,On,t0)
 IF context (ReedSwitch (>,242))
 THEN do (Off, BedroomLuxo4,t)
      when t = t0 + 23s
```

```
(Pattern 33 (25%))
 ON occurs (BedroomLuxo2,Off,t0)
 IF context (LightLevel (<,40))
 THEN do (Off,BedroomLuxo4,t)
      when t = t0 + 9s
```

```
(Pattern 34 (50%))
 ON occurs (BedroomLuxo3,On,t0)
 IF context (LightLevel (<,48))
 THEN do (Off,BedroomLuxo4,t)
      when t = t0 + 24s
```

```
(Pattern 35 (25%))
 ON occurs (BedroomLuxo3,Off,t0)
 IF context (LightLevel (>,12))
 THEN do (Off,BedroomLuxo4,t)
      when t = t0 + 11s
```

```
(Pattern 36 (50%))
 ON occurs (BedroomLight,Off,t0)
 IF context (LightLevel (<,14))
 THEN do (Off,BedroomLuxo4,t)
      when t = t0 + 5s
```

```
(Pattern 37 (25%))
 ON occurs (CloseLight,On,t0)
 IF context (ReedSwitch (<,228))
 THEN do (Off,CloseLight,t)
      when t = t0 + 5s

(Pattern 38 (25%))
 ON occurs (BedroomLamp1,Off,t0)
 IF context ()
 THEN do (On,BedroomLamp1,t)
      when t = t0 + 11s

(Pattern 39 (50%))
 ON occurs (BedroomLamp1,On,t0)
 IF context ()
 THEN do (Off,BedroomLamp1,t)
      when t = t0 + 0s

(Pattern 40 (75%))
 ON occurs (BedroomLamp1,On,t0)
 IF context ()
 THEN do (Off,BedroomLamp2,t)
      when t = t0 + 0s

-----------
  Trial 3
-----------

(Pattern 1 (50%))
 ON occurs (LivingRoomLight,Off,t0)
 IF context ()
 THEN do (On,LivingRoomLight,t)
      when t = t0 + 5s

(Pattern 2 (75%))
 ON occurs (LivingRoomLight,Off,t0)
 IF context (TimeOfDay (>,22:10:27))
 THEN do (On,LivingRoomLight,t)
      when t = t0 + 115s

(Pattern 3 (25%))
 ON occurs (BedroomLight,Off,t0)
 IF context (LightLevel (<,34))
 THEN do (On,LivingRoomLight,t)
      when t is after t0

(Pattern 4 (75%))
 ON occurs (LivingRoomLight,On,t0)
 IF context ()
 THEN do (Off,LivingRoomLight,t)
      when t = t0 + 4s

(Pattern 5 (50%))
 ON occurs (KitchenLight,Off,t0)
 IF context (HumidityLevel (<,108))
 THEN do (Off,LivingRoomLight,t)
      when t = t0 + 51s

(Pattern 6 (25%))
 ON occurs (BedroomLight,Off,t0)
 IF context (LightLevel (<,34))
 THEN do (Off,LivingRoomLight,t)
      when t is after t0

(Pattern 7 (75%))
 ON occurs (FloorLamp,On,t0)
 IF context (TimeOfDay (>,22:10:00))
 THEN do (Off,LivingRoomLight,t)
      when t = t0 + 40s

(Pattern 8 (50%))
 ON occurs (CounterLights,On,t0)
 IF context (DayOfWeek (=,Saturday))
 THEN do (Off,KitchenLight,t)
      when t = t0 + 4s

(Pattern 9 (25%))
 ON occurs (BathroomLight,Off,t0)
 IF context (TimeOfDay (>,22:13:00))
 THEN do (On,CounterLights,t)
      when t = t0 + 51s

(Pattern 10 (25%))
 ON occurs (ShowerLight,Off,t0)
 IF context (Temperature (<,171))
 THEN do (On,CounterLights,t)
      when t = t0 + 78s
```

```
(Pattern 11 (25%))
 ON occurs (CounterLights,On,t0)
 IF context (Temperature (<,171))
 THEN do (Off,CounterLights,t)
      when t = t0 + 78s
```

```
(Pattern 12 (50%))
 ON occurs (ShowerLight,On,t0)
 IF context (TimeOfDay (<,13:49:00))
 THEN do (On,BathroomLight,t)
      when t = t0 + 23s
```

```
(Pattern 13 (75%))
 ON occurs (ShowerLight,Off,t0)
 IF context ()
 THEN do (Off,BathroomLight,t)
      when t = t0 + 9s
```

```
(Pattern 14 (25%))
 ON occurs (ShowerLight,On,t0)
 IF context (LightLevel (<,26))
 THEN do (Off,ShowerLight,t)
      when t = t0 + 1s
```

```
(Pattern 15 (75%))
 ON occurs (ShowerLight,On,t0)
 IF context ()
 THEN do (Off, ShowerLight,t)
      when t = t0 + 78s
```

```
(Pattern 16 (75%))
 ON occurs (ShowerLight,On,t0)
 IF context ()
 THEN do (Off, ShowerLight,t)
      when t is after t0
```

```
(Pattern 17 (25%))
 ON occurs (BathroomLight,Off,t0)
 IF context (LightLevel (<,31))
 THEN do (Off, ShowerLight,t)
      when t = t0 + 0s
```

```
(Pattern 18 (25%))
 ON occurs (BedroomLight,On,t0)
 IF context (LightLevel (>,10))
 THEN do (Off, BedroomLight,t)
      when t = t0 + 24s
```

```
(Pattern 19 (25%))
 ON occurs (BedroomLight,On,t0)
 IF context (TimeOfDay (>,13:24:00))
 THEN do (On, BedroomLuxo1,t)
      when t = t0 + 9s
```

```
(Pattern 20 (25%))
 ON occurs (BedroomLuxo1,On,t0)
 IF context (LightLevel (>,6))
 THEN do (Off, BedroomLuxo1,t)
      when t = t0 + 27s
```

Appendix G: Validating the Pairwise Approach with WSU data

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

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

(Pattern 3)
 ON occurs (Cabinet,On,t0)
 IF context ()
 THEN do (On,Raisins,t) when t = t0 + 3s

(Pattern 4)
 ON occurs (Raisins,On,t0)
 IF context ()
 THEN do (On,Oatmeal,t) when t is after t0

(Pattern 5)
 ON occurs (Oatmeal,On,t0)
 IF context ()
 THEN do (On,Raisins,t) when t is after t0

(Pattern 6)
 ON occurs (Cabinet,Off,t0)
 IF context ()
 THEN do (On,Cabinet,t) when t = t0 + 34s

(Pattern 7)
 ON occurs (Phone,Off,t0)
 IF context ()
 THEN do (On,Water,t) when t is after t0

(Pattern 8)
 ON occurs (Cabinet,Off,t0)
 IF context ()
 THEN do (On,Water,t) when t = t0 + 6s

(Pattern 9)
 ON occurs (Water,Off,t0)
 IF context ()
 THEN do (On,Cabinet,t) when t = t0 + 39s

(Pattern 10)
 ON occurs (Water,On,t0)
 IF context ()
 THEN do (Off,Water,t) when t is after t0

(Pattern 11)
 ON occurs (Water,Off,t0)
 IF context ()
 THEN do (On,Pot,t) when t is after t0

(Pattern 12)
 ON occurs (Sugar,On,t0)
 IF context ()
 THEN do (Off,Sugar,t) when t = t0 + 3s

(Pattern 13)
 ON occurs (Bowl,On,t0)
 IF context ()
 THEN do (On,MeasuringSpoon,t)
      when t is after t0

(Pattern 14)
 ON occurs (Bowl,On,t0)
 IF context ()
 THEN do (Off,Cabinet,t)
      when t = t0 + 3s
```

```
(Pattern 15)
 ON occurs (Cabinet,On,t0)
 IF context ()
 THEN do (On, Medicine,t)
      when t = t0 + 2s
```

```
(Pattern 16)
 ON occurs (Medicine,On,t0)
 IF context ()
 THEN do (Off, Cabinet,t)
      when t is after t0
```

```
(Pattern 17)
 ON occurs (Cabinet,On,t0)
 IF context ()
 THEN do (Off, Medicine,t)
      when t = t0 + 2s
```

```
(Pattern 18)
 ON occurs (Medicine,Off,t0)
 IF context ()
 THEN do (Off, Cabinet,t)
      when t = t0 + 2s
```

```
(Pattern 19)
 ON occurs (Cabinet,Off,t0)
 IF context ()
 THEN do (On, Burner,t) when t = t0 + 5s
```

```
(Pattern 20)
 ON occurs (Water,On,t0)
 IF context ()
 THEN do (On, Burner,t) when t = t0 + 3s
```

```
(Pattern 21)
 ON occurs (Burner,Off,t0)
 IF context ()
 THEN do (On, Cabinet,t)
      when t = t0 + 12s
```

```
(Pattern 22)
 ON occurs (Burner,Off,t0)
 IF context ()
 THEN do (Off, Water,t)
      when t = t0 + 1s
```

```
(Pattern 23)
 ON occurs (Burner,On,t0)
 IF context ()
 THEN do (Off, Burner,t) when t = t0 + 9s
```

Appendix H: Validating the Action Map Approach with MavPad data

Note: For each trial and demanded confidence level, discovered Action Maps are shown.

```
----------------------------------------
  Trial 1 (Demanded confidence level: 25%)
----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,03:30:00)) |
  context (TimeOfDay(>,10:45:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern -)
  ON occurs (start,--,t0)                     ON occurs (start,--,t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BedroomLight),t)        THEN do (simple,(On,BedroomLamp1),t)
       when --                                     when --

 (Action Pattern 1)                          (Action Pattern 2)
  ON occurs (simple,(On,BedroomLamp1),t0)     ON occurs (simple,
                                                         (Off,BedroomLamp1),t0)
  IF context ()                               IF context (TimeOfDay (<,11:51:00))
  THEN do (simple,(Off,BedroomLamp1),t)       THEN do (simple,(On,BedroomLamp1),t)
       when t = t0 + 0s                            when t = t0 + 11s

 (Action Pattern 3)                          (Action Pattern 4)
  ON occurs (simple,(On,BedroomLamp1),t0)     ON occurs (simple,(On,BedroomLamp2),t0)
  IF context ()                               IF context ()
  THEN do (simple,(Off,BedroomLamp2),t)       THEN do (simple,(Off,BedroomLamp2),t)
       when t is after t0                          when t = t0 + 0s

 (Action Pattern 5)                          (Action Pattern 6)
  ON occurs (simple,(Off,BedroomLamp1),t0)    ON occurs (simple,(On,BedroomLight),t0)
  IF context (TimeOfDay (>,11:51:00))         IF context (TimeOfDay(<,21:14:00))
  THEN do (simple,(On,BedroomLamp2),t)        THEN do (simple,(On,BedroomLuxo1),t)
       when t is after t0                          when t = t0 + 5s
```

```
(Action Pattern 7)
 ON occurs (simple,(On,BedroomLight),t0)
 IF context (TimeOfDay(>,21:14:00))
 THEN do (simple,(Off,BedroomLight),t)
      when t =t0 + 43s

(Action Pattern 8)
 ON occurs (simple,(On,BedroomLuxo1),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 2s

(Action Pattern 9)
 ON occurs (simple,(Off,BedroomLight),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)
      when t =t0 + 61s

(Action Pattern 10)
 ON occurs (simple,(On,BedroomLuxo1),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)
      when t = t0 + 27s

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

(Action Pattern -)
 ON occurs (simple,
           (Off,BedroomLamp2),t0)
 IF context ()
 THEN do (--,end,t) when --


(Action Map 2)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,03:15:00)) |
  context (TimeOfDay(>,11:15:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)
  ON occurs (start,--,t0)
  IF context ()
  THEN do (unordered,((Off,BedroomLuxo1)&(Off,BedroomLight)&(On,BedroomLuxo1)&
          (On,BedroomLight)& (Off,TableLamp)&(On,TableLamp)&(Off,LivingRoomLight)&
          (On,LivingRoomLight)),t) when --

 (Action Pattern -)
  ON occurs (unordered,((Off,BedroomLuxo1)&(Off,BedroomLight)&(On,BedroomLuxo1)&
            (On,BedroomLight)& (Off,TableLamp)&(On,TableLamp)&(Off,LivingRoomLight)&
            (On,LivingRoomLight)),t0)
  IF context ()
  THEN do (--,end,t) when --


(Action Map 3)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,03:15:00)) |
  context (TimeOfDay(>,11:15:00)) & context (TimeOfDay(<,23:59:59))
```

```
(Action Pattern -)
 ON occurs (start,--,t0)
 IF context ()
 THEN do (simple,(On,TableLamp),t)
      when --
```

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

```
(Action Pattern 1)
 ON occurs (simple,(On,TableLamp),t0)

 IF context ()
 THEN do (unordered,((Off,LivingRoomLight)&
         (Off,FloorLamp)&(Off,TableLamp)),t)


      when t = t0 + 6s
```

```
(Action Pattern 2)
 ON occurs (simple,
           (Off,BedroomLight),t0)
 IF context (TimeOfDay (<,11:51:00))
 THEN do (unordered,
          ((Off,LivingRoomLight)&
          (Off,FloorLamp)&
          (Off,TableLamp)),t)
      when t is after t0
```

```
(Action Pattern 3)
 ON occurs (simple,(On,BedroomLight),t0)
 IF context ()
 THEN do (simple,(On,BedroomLuxo1),t)
      when t = t0 + 5s
```

```
(Action Pattern 4)
 ON occurs (simple,(On,BedroomLight),t0)
 IF context ()
 THEN do (simple,(On,BedroomLuxo1),t)
      when t = t0 + 9s
```

```
(Action Pattern 5)
 ON occurs (simple,(On,BedroomLight),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 43s
```

```
(Action Pattern 6)
 ON occurs (simple,(On,BedroomLight),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 24s
```

```
(Action Pattern 7)
 ON occurs (simple,(Off,BedroomLight),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)
      when t is after t0
```

```
(Action Pattern 8)
 ON occurs (simple,(On,BedroomLuxo1),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)
      when t = t0 + 27s
```

```
(Action Pattern -)
 ON occurs (unordered,((Off,LivingRoomLight)
           &(Off,FloorLamp)&(Off,TableLamp))
           ,t0)
 IF context ()
 THEN do (--,end,t) when --
```

```
(Action Pattern -)
 ON occurs (simple,
           (Off,BedroomLuxo1),t0)

 IF context ()
 THEN do (--,end,t) when --
```

```
(Action Map 4)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,03:15:00)) |
  context (TimeOfDay(>,11:15:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern -)
  ON occurs (start,--,t0)                     ON occurs (start,--,t0)
  IF context ()                               IF context ()
  THEN do (unordered,((On,FloorLamp)&         THEN do (simple,(On,BedroomLight),t)
          (On,TableLamp)),t) when --                  when --

 (Action Pattern 1)                          (Action Pattern 2)
  ON occurs (simple,(Off,TableLamp),t0)       ON occurs (unordered,((On,FloorLamp)&
  IF context ()                                          (On,TableLamp)),t0)
  THEN do (unordered,((Off,FloorLamp)&        IF context ()
           (On,TableLamp)),t)                 THEN do (simple,((Off,TableLamp)&
       when t = t0 + 6s                               when t = t0 + 6s

 (Action Pattern 3)                          (Action Pattern 4)
  ON occurs (simple,(On,BedroomLight),t0)     ON occurs (simple,(On,BedroomLight),t0)
  IF context (TimeOfDay(>,20:00:00))          IF context ()
  THEN do (simple,(On,BedroomLuxo1(0)),t)     THEN do (simple,(On,BedroomLuxo1(0)),t)
       when t = t0 + 5s                               when t = t0 + 9s

 (Action Pattern 5)                          (Action Pattern 6)
  ON occurs (simple,(Off,BedroomLight),t0)    ON occurs (simple,
                                                         (On,BedroomLuxo1(1)),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BedroomLuxo1(1)),t)     THEN do (simple,(Off,BedroomLight),t)
       when t = t0 + 0s                               when t = t0 + 0s

 (Action Pattern 7)                          (Action Pattern 8)
  ON occurs (simple,(On,BedroomLight),t0)     ON occurs (simple,(On,BedroomLight),t0)
  IF context ()                               IF context (TimeOfDay(<,20:00:00))
  THEN do (simple,(Off,BedroomLight),t)       THEN do (simple,(Off,BedroomLight),t)
       when t = t0 + 43s                              when t = t0 + 24s

 (Action Pattern 9)                          (Action Pattern 10)
  ON occurs (simple,(Off,BedroomLight),t0)    ON occurs (simple,
                                                         (On,BedroomLuxo1(1)),t0)
  IF context ()                               IF context ()
  THEN do (simple,(Off,BedroomLuxo1),t)       THEN do (simple,(Off,BedroomLuxo1),t)
       when t = t0 + 61s                              when t = t0 + 27s
```

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

(Action Map 5)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,04:15:00)) |
  context (TimeOfDay(>,11:15:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern -)
  ON occurs (start,--,t0)                     ON occurs (start,--,t0)
  IF context ()                               IF context ()
  THEN do (unordered,((On,ClosetLight)&       THEN do (simple,(On,BedroomLight),t)
          (Off,ClosetLight)),t) when --            when --

 (Action Pattern 1)                          (Action Pattern 2)
  ON occurs (simple,(On,BedroomLight),t0)     ON occurs (simple,(On,BedroomLight),t0)
  IF context (TimeOfDay(>,11:59:00))          IF context ()
  THEN do (simple,(On,BedroomLuxo1(0)),t)     THEN do (simple,(On,BedroomLuxo1(0)),t)
       when t = t0 + 5s                            when t = t0 + 9s

 (Action Pattern 3)                          (Action Pattern 4)
  ON occurs (simple,(Off,BedroomLight),t0)    ON occurs (simple,
                                                        (On,BedroomLuxo1(1)),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BedroomLuxo1(1)),t)     THEN do (simple,(Off,BedroomLight),t)
       when t = t0 + 0s                            when t = t0 + 1s

 (Action Pattern 5)                          (Action Pattern 6)
  ON occurs (simple,(On,BedroomLight),t0)     ON occurs (simple,(On,BedroomLight),t0)
  IF context ()                               IF context (TimeOfDay(<,11:59:00))
  THEN do (simple,(Off,BedroomLight),t)       THEN do (simple,(Off,BedroomLight),t)
       when t = t0 + 43s                           when t = t0 + 24s

 (Action Pattern 7)                          (Action Pattern 8)
  ON occurs (simple,(Off,BedroomLight),t0)    ON occurs (simple,
                                                        (On,BedroomLuxo1(1)),t0)
  IF context ()                               IF context ()
  THEN do (simple,(Off,BedroomLuxo1),t)       THEN do (simple,(Off,BedroomLuxo1),t)
       when t = t0 + 61s                           when t = t0 + 27s
```

```
(Action Pattern 9)
 ON occurs (simple,(Off,BedroomLight),t0)

 IF context ()
 THEN do (simple,(Off,LivingRoomLight),t)
      when t is after t0
```

```
(Action Pattern 10)
 ON occurs (simple,
    (Off,BedroomLuxo1),t0)
 IF context ()
 THEN do (--,end,t) when --
```

```
(Action Pattern -)
 ON occurs (unordered,((On,ClosetLight)& (Off,ClosetLight)),t0)
 IF context ()
 THEN do (--,end,t) when --
```

```
(Action Map 6)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,03:00:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))
```

```
(Action Pattern -)
 ON occurs (start,--,t0)
 IF context ()
 THEN do (unordered,((On,LivingRoomLight)&
         (Off,LivingRoomLight)&
         (On,ClosetLight)&
         (Off,ClosetLight)),t) when --
```

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

```
(Action Pattern 1)
 ON occurs (simple,(On,BedroomLamp1),t0)

 IF context ()
 THEN do (simple,(Off,BedroomLamp1)
      when t = t0 + 0s
```

```
(Action Pattern 2)
 ON occurs (simple,
           (Off,BedroomLamp1),t0)
 IF context (TimeOfDay(>,11:39:00))
 THEN do (simple,((On,BedroomLamp1)&
      when t = t0 + 11s
```

```
(Action Pattern 3)
 ON occurs (simple,(On,BedroomLamp1),t0)
 IF context (TimeOfDay(<,11:39:00))
 THEN do (simple,(Off,BedroomLamp2),t)
      when t = t0 + 8s
```

```
(Action Pattern 4)
 ON occurs (simple,(On,BedroomLamp2),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLamp2),t)
      when t = t0 + 0s
```

```
(Action Pattern 5)
 ON occurs (simple,(Off,BedroomLamp1),t0)

 IF context ()
 THEN do (simple,(Off,BedroomLamp2),t)
      when t is after t0
```

```
(Action Pattern 6)
 ON occurs (simple,
           (Off,BedroomLamp2),t0)
 IF context ()
 THEN do (simple,(On,BedroomLamp2),t)
      when t = t0 + 2s
```

```
(Action Pattern 7)
 ON occurs (simple,(Off,BedroomLamp1),t0)
 IF context ()
 THEN do (simple,(On,BedroomLamp2),t)
      when t = t0 + 43s
```

```
(Action Pattern 8)
 ON occurs (simple,
           (Off,BedroomLight),t0)
 IF context ()
 THEN do (unordered,(On,LivingRoomLight)
         & (Off,LivingRoomLight)
         & (On,ClosetLight)
         & (Off,ClosetLight)),t)
      when t = t0 + 24s
```

```
(Action Pattern 9)
 ON occurs (unordered,((On,LivingRoomLight)&
           (Off,LivingRoomLight)&
           (On,ClosetLight)&
           (Off,ClosetLight)),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 43s
```

```
(Action Pattern 10)
 ON occurs (unordered,
           ((On,LivingRoomLight) &
           (Off,LivingRoomLight)&
           (On,ClosetLight)&
           (Off,ClosetLight)),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 24s
```

```
(Action Pattern -)
 ON occurs (simple,(Off,BedroomLamp2),t0)
 IF context ()
 THEN do (--,end,t) when --
```

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

```
(Action Map 7)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,04:15:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)
  ON occurs (start,--,t0)
  IF context ()
  THEN do (unordered,((On,LivingRoomLight)&(On,BedroomLight)&
          (On,ClosetLight)&(Off,ClosetLight),t) when --

 (Action Pattern 1)
  ON occurs (unordered,((On,LivingRoomLight)&(On,BedroomLight)&
          (On,ClosetLight)&(Off,ClosetLight),t0)
  IF context ()
  THEN do (simple,(Off,BedroomLight),t) when t = t0 + 43s
```

```
 (Action Pattern 2)
  ON occurs (unordered,((On,LivingRoomLight)&(On,BedroomLight)&
          (On,ClosetLight)&(Off,ClosetLight),t0)
  IF context ()
  THEN do (simple,(Off,BedroomLight),t) when t = t0 + 24s

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

(Action Map 8)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,04:15:00)) |
  context (TimeOfDay(>,10:45:00)) & context (TimeOfDay(<,23:59:59))

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

 (Action Pattern 1)
  ON occurs (simple,(On,BedroomLight),t0)
  IF context ()
  THEN do (unordered,((On,BedroomLuxo1)&(Off,BedroomLight)),t) when t = t0 + 43s

 (Action Pattern 2)
  ON occurs (simple,(On,BedroomLight),t0)
  IF context ()
  THEN do (unordered,((On,BedroomLuxo1)&(Off,BedroomLight)),t) when t = t0 + 5s

 (Action Pattern 3)
  ON occurs (simple,(On,BedroomLight),t0)
  IF context ()
  THEN do (unordered,((On,BedroomLuxo1)&(Off,BedroomLight)),t) when t = t0 + 24s

 (Action Pattern 4)
  ON occurs (simple,(On,BedroomLight),t0)
  IF context ()
  THEN do (unordered,((On,BedroomLuxo1)&(Off,BedroomLight)),t) when t = t0 + 9s
```

```
 (Action Pattern 5)
  ON occurs (unordered,((On,BedroomLuxo1)&(Off,BedroomLight)),t0)
  IF context ()
  THEN do (simple,(Off,BedroomLuxo1),t) when t = t0 + 61s

 (Action Pattern 6)
  ON occurs (unordered,((On,BedroomLuxo1)&(Off,BedroomLight)),t0)
  IF context ()
  THEN do (simple,(Off,BedroomLuxo1),t) when t = t0 + 27s

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

-----------------------------------------
  Trial 1 (Demanded confidence level: 50%)
-----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,04:15:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern 1)
  ON occurs (start,--,t0)                     ON occurs (simple,(On,BedroomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BedroomLight),t)        THEN do (unordered,((On,BedroomLuxo1)&
       when --                                          (Off,BedroomLight)),t)
                                                    when t = t0 + 43s

 (Action Pattern 2)                          (Action Pattern 3)
  ON occurs (simple,(On,BedroomLight),t0)     ON occurs (simple,(On,BedroomLight),t0)
  IF context ()                               IF context ()
  THEN do (unordered,((On,BedroomLuxo1)&      THEN do (unordered,((On,BedroomLuxo1)&
          (Off,BedroomLight)),t)                      (Off,BedroomLight)),t)
       when t = t0 + 5s                            when t = t0 + 24s
```

```
(Action Pattern 4)
 ON occurs (simple,(On,BedroomLight),t0)

 IF context ()
 THEN do (unordered,((On,BedroomLuxo1)&
         (Off,BedroomLight)),t)
      when t = t0 + 9s

(Action Pattern 5)
 ON occurs (unordered,((On,BedroomLuxo1)
           & (Off,BedroomLight)),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)
      when t = t0 + 61s

(Action Pattern 6)
 ON occurs (unordered,((On,BedroomLuxo1)&
           (Off,BedroomLight)),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLuxo1),t)
      when t = t0 + 27s

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

(Action Map 2)

(General Conditions)
 context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,03:15:00)) |
 context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

(Action Pattern -)
 ON occurs (start,--,t0)
 IF context ()
 THEN do (simple,(On,BedroomLamp1),t)
      when --

(Action Pattern 1)
 ON occurs (simple,(On,BedroomLamp1),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLamp1),t)
      when t = t0 + 0s

(Action Pattern 2)
 ON occurs (simple,(Off,BedroomLamp1),t0)
 IF context ()
 THEN do (simple,(On,BedroomLamp1),t)
      when t = t0 + 11s

(Action Pattern 3)
 ON occurs (simple,(On,BedroomLamp1),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLamp2),t)
      when t is after t0

(Action Pattern 4)
 ON occurs (simple,(On,BedroomLamp1),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLamp2),t)
      when t = t0 + 8s

(Action Pattern 5)
 ON occurs (simple,(On,BedroomLamp2),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLamp2),t)
      when t = t0 + 0s

(Action Pattern 6)
 ON occurs (simple,(Off,BedroomLamp1),t0)

 IF context ()
 THEN do (simple,(Off,BedroomLamp2),t)
      when t is after t0

(Action Pattern 7)
 ON occurs (simple,
           (Off,BedroomLamp1),t0)
 IF context ()
 THEN do (simple,(On,BedroomLamp2),t)
      when t is after t0
```

```
(Action Pattern 8)                          (Action Pattern -)
 ON occurs (simple,(On,BedroomLamp1),t0)     ON occurs (simple,
                                                        (Off,BedroomLamp2),t0)
 IF context ()                               IF context ()
 THEN do (simple,(On,BedroomLamp2),t)        THEN do (--,end,t) when --
      when t is after t0

(Action Map 3)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,03:15:00)) |
  context (TimeOfDay(>,11:15:00)) & context (TimeOfDay(<,23:59:59))

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

 (Action Pattern 1)                          (Action Pattern 2)
  ON occurs (simple,(On,TableLamp),t0)        ON occurs (simple,(On,BedroomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(Off,TableLamp),t)          THEN do (simple,(On,BedroomLuxo1),t)
       when t = t0 + 6s                            when t = t0 + 5s

 (Action Pattern 3)                          (Action Pattern 4)
  ON occurs (simple,(On,BedroomLight),t0)     ON occurs (simple,
                                                         (Off,BedroomLight),t0)
  IF context (TimeOfDay(<,21:00:00))          IF context (TimeOfDay(>,01:43:00))
  THEN do (simple,(On,BedroomLuxo1),t)        THEN do (simple,(On,BedroomLuxo1),t)
       when t = t0 + 9s                            when t = t0 + 0s

 (Action Pattern 5)                          (Action Pattern 6)
  ON occurs (simple,(On,BedroomLight),t0)     ON occurs (simple,(On,BedroomLight),t0)
  IF context (TimeOfDay(>,21:00:00))          IF context ()
  THEN do (simple,(Off,BedroomLight),t)       THEN do (simple,(Off,BedroomLight),t)
       when t =t0 + 43s                            when t = t0 + 24s

 (Action Pattern 7)                          (Action Pattern 8)
  ON occurs (simple,(Off,BedroomLight),t0)    ON occurs (simple,(On,BedroomLuxo1),t0)
  IF context (TimeOfDay(<,01:43:00))          IF context ()
  THEN do (simple,(Off,BedroomLuxo1),t)       THEN do (simple,(Off,BedroomLuxo1),t)
       when t =t0 + 61s                            when t = t0 + 27s
```

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

(Action Map 3)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,04:15:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern -)
  ON occurs (start,--,t0)                     ON occurs (unordered,((On,ClosetLight)&
  IF context ()                                          (Off,ClosetLight)),t0)
  THEN do (unordered,((On,ClosetLight)&       IF context ()
          (Off,ClosetLight)),t) when --       THEN do (--,end,t) when --


-----------------------------------------
  Trial 1 (Demanded confidence level: 75%)
-----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,04:15:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern -)
  ON occurs (start,--,t0)                     ON occurs (unordered,((On,BedroomLight)
  IF context ()                                          & (Off,BedroomLight)&
                                                         (On,BedroomLuxo1)&
  THEN do (unordered,((On,BedroomLight)&                 (Off,BedroomLuxo1)),t0)
        (Off,BedroomLight)&(On,BedroomLuxo1)  IF context ()
        & (Off,BedroomLuxo1)),t) when --      THEN do (--,end,t) when --
```

```
-----------------------------------------
  Trial 2 (Demanded confidence level: 25%)
-----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,05:30:00)) |
  context (TimeOfDay(>,09:45:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern -)
  ON occurs (start,--,t0)                     ON occurs (start,--,t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BathroomLight),t)       THEN do (unordered,((On,ShowerLights)&
       when --                                        (Off,ShowerLights)),t) when --

 (Action Pattern 1)                          (Action Pattern 2)
  ON occurs (simple,(On,ShowerLights),t0)     ON occurs (simple,(On,ShowerLights),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BathroomLight),t)       THEN do (simple,(On,BathroomLight),t)
       when t = t0 + 34s                           when t = t0 + 23s

 (Action Pattern 3)                          (Action Pattern 4)
  ON occurs (simple,(On,BathroomLight),t0)    ON occurs (simple,
                                                         (Off,ShowerLights),t0)
  IF context ()                               IF context ()
  THEN do (simple,(Off,BathroomLight),t)      THEN do (simple,(Off,BathroomLight),t)
       when t = t0 + 64s                           when t = t0 + 6s

 (Action Pattern 5)                          (Action Pattern 6)
  ON occurs (simple,(Off,ShowerLights),t0)    ON occurs (simple,
                                                         (On,BathroomLight),t0)
  IF context (TimeOfDay(<,22:11:00))          IF context (TimeOfDay(>,02:43:00))
  THEN do (simple,(Off,BathroomLight),t)      THEN do (simple,(Off,BathroomLight),t)
       when t = t0 + 9s                            when t is after t0

 (Action Pattern 7)                          (Action Pattern 8)
  ON occurs (simple,(Off,LivingRoomLight),t0) ON occurs (simple,
                                                         (Off,BathroomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,LivingRoomLight),t)     THEN do (simple,(On,LivingRoomLight),t)
       when t = t0 + 5s                            when t = t0 + 71s
```

```
(Action Pattern 9)
 ON occurs (simple,(Off,LivingRoomLight),t0)

 IF context ()
 THEN do (simple,(On,LivingRoomLight),t)
      when t = t0 + 18s
```

```
(Action Pattern 10)
ON occurs (simple,
          (Off,BedroomLight),t0)
 IF context ()
 THEN do (simple,(On,LivingRoomLight),t)
      when t is after t0
```

```
(Action Pattern 11)
 ON occurs (simple,(On,BathroomLight),t0)

 IF context (TimeOfDay(<,02:43:00))
 THEN do (simple,(On,CounterLights),t)
      when t = t0 + 18s
```

```
(Action Pattern 12)
 ON occurs (simple,
           (On,BathroomLight),t0)
 IF context ()
 THEN do (simple,(On,CounterLights),t)
      when t = t0 + 51s
```

```
(Action Pattern 13)
 ON occurs (simple,(Off,ShowerLights),t0)

 IF context ()
 THEN do (simple,(On,CounterLights),t)
      when t = t0 + 15s
```

```
(Action Pattern 14)
 ON occurs (simple,
           (Off,ShowerLights),t0)
 IF context (TimeOfDay(>,22:11:00))
 THEN do (simple,(On,CounterLights),t)
      when t = t0 + 78s
```

```
(Action Pattern 15)
 ON occurs (simple,(On,CounterLights),t0)

 IF context (TimeOfDay(>,18:31:00))
 THEN do (simple,(Off,CounterLights),t)
      when t = t0 + 32s
```

```
(Action Pattern 16)
 ON occurs (simple,
           (On,CounterLights),t0)
 IF context ()
 THEN do (simple,(Off,CounterLights),t)
      when t = t0 + 78s
```

```
(Action Pattern 17)
 ON occurs (simple,(On,LivingRoomLight),t0)

 IF context ()
 THEN do (simple,(On,ShowerLights),t)
      when t is after t0
```

```
(Action Pattern 18)
 ON occurs (simple,
           (On,BathroomLight),t0)
 IF context ()
 THEN do (simple,(On,ShowerLights),t)
      when t is after t0
```

```
(Action Pattern 19)
 ON occurs (simple,(On,LivingRoomLight),t0)

 IF context ()
 THEN do (simple,(Off,LivingRoomLight),t)
      when t = t0 + 46s
```

```
(Action Pattern 20)
 ON occurs (simple,
           (Off,CounterLights),t0)
 IF context ()
 THEN do (simple,
         (Off,LivingRoomLight),t)
      when t = t0 + 19s
```

```
(Action Pattern 21)
 ON occurs (simple,(On,BedroomLight),t0)

 IF context ()
 THEN do (simple,(Off,LivingRoomLight),t)
      when t = t0 + 1s
```

```
(Action Pattern 22)
 ON occurs (simple,
           (Off,BedroomLight),t0)
 IF context ()
 THEN do (simple,(On,BedroomLight),t)
      when t = t0 + 189s
```

```
(Action Pattern 23)
 ON occurs (simple,(Off,BathroomLight),t0)

 IF context ()
 THEN do (simple,(Off,ShowerLights),t)
      when t = t0 + 5s
```

```
(Action Pattern 24)
 ON occurs (simple,
           (Off,BathroomLight),t0)
 IF context ()
 THEN do (simple,(Off,ShowerLights),t)
      when t = t0 + 12s
```

```
(Action Pattern 25)
 ON occurs (simple,(On,CounterLights),t0)

 IF context (TimeOfDay(<,18:31:00))
 THEN do (simple,(Off,ShowerLights),t)
      when t = t0 + 26s
```

```
(Action Pattern 26)
 ON occurs (simple,
           (On,ShowerLights),t0)
 IF context ()
 THEN do (simple,(Off,ShowerLights),t)
      when t = t0 + 48s
```

```
(Action Pattern 27)
 ON occurs (simple,(On,ShowerLights),t0)

 IF context ()
 THEN do (simple,(Off,ShowerLights),t)
      when t = t0 + 1s
```

```
(Action Pattern 28)
 ON occurs (simple,
           (On,ShowerLights),t0)
 IF context ()
 THEN do (simple,(Off,ShowerLights),t)
      when t = t0 + 55s
```

```
(Action Pattern 29)
 ON occurs (simple,(On,BedroomLight),t0)

 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 43s
```

```
(Action Pattern 30)
 ON occurs (simple,
           (On,BedroomLight),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 24s
```

```
(Action Pattern -)
 ON occurs (simple,(Off,LivingRoomLight),t0)

 IF context ()
 THEN do (--,end,t) when --
```

```
(Action Pattern -)
 ON occurs (unordered,((On,ShowerLights)
           & (Off,ShowerLights)),t0)
 IF context ()
 THEN do (--,end,t) when --
```

```
(Action Map 2)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,04:45:00)) |
  context (TimeOfDay(>,10:15:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern 1)
  ON occurs (start,--,t0)                     ON occurs (simple,(On,BedroomLight),t0)
  IF context ()                               IF context (TimeOfDay(<,22:48:00))
  THEN do (simple,(On,BathroomLight),t)       THEN do (unordered,((On,BedroomLamp1)&
       when --                                        (Off,BedroomLamp1)),t)
                                                   when t = t0 + 225s

 (Action Pattern 2)                          (Action Pattern 3)
  ON occurs (simple,(On,BedroomLamp1),t0)     ON occurs (simple,(On,BedroomLamp1),t0)
  IF context ()                               IF context ()
  THEN do (unordered,(Off,BedroomLamp1),t)    THEN do (simple,(Off,BedroomLamp1),t)
       when t = t0 + 1s                            when t = t0 + 5s

 (Action Pattern 4)                          (Action Pattern 5)
  ON occurs (unordered,((On,BedroomLamp1)&    ON occurs (simple,
            (Off,BedroomLamp1)),t0)                    (Off,BedroomLamp1),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BedroomLamp1),t)        THEN do (simple,(On,LivingRoomLight),t)
       when t is after t0                          when t is after t0

 (Action Pattern 6)                          (Action Pattern 7)
  ON occurs (simple,(Off,BedroomLight),t0)    ON occurs (simple,
                                                         (Off,BedroomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,LivingRoomLight),t)     THEN do (simple,(On,LivingRoomLight),t)
       when t = t0 + 3s                            when t = t0 + 15s

 (Action Pattern 8)                          (Action Pattern 9)
  ON occurs (simple,(Off,BathroomLight),t0)   ON occurs (simple,
                                                         (On,BathroomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,LivingRoomLight),t)     THEN do (simple,(Off,BathroomLight),t)
       when t is after t0                          when t is after t0
```

```
(Action Pattern 10)
 ON occurs (simple,(On,LivingRoomLight),t0)

 IF context ()
 THEN do (simple,(On,BedroomLight),t)
      when t is after t0

(Action Pattern 12)
 ON occurs (simple,(On,BedroomLight),t0)

 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 24s
```

```
(Action Pattern 11)
 ON occurs (simple,
           (On,BedroomLight),t0)
 IF context (TimeOfDay(>,22:48:00))
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 43s

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

```
(Action Map 3)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,05:30:00)) |
  context (TimeOfDay(>,09:45:00)) & context (TimeOfDay(<,23:59:59))

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

 (Action Pattern 1)
  ON occurs (simple,(On,LivingRoomLight),t0)

  IF context ()
  THEN do (simple,(On,BathroomLight),t)
       when t is after t0

 (Action Pattern 3)
  ON occurs (simple,(Off,LivingRoomLight),t0)

  IF context ()
  THEN do (simple,(On,LivingRoomLight),t)
       when t = t0 + 5s
```

```
(Action Pattern -)
 ON occurs (start,--,t0)
 IF context ()
 THEN do (simple,(On,LivingRoomLight),t)
      when --

(Action Pattern 2)
 ON occurs (simple,
           (Off,BathroomLight),t0)
 IF context ()
 THEN do (simple,(On,BathroomLight),t)
      when t is after t0

(Action Pattern 4)
 ON occurs (simple,
           (On,BathroomLight),t0)
 IF context ()
 THEN do (simple,(Off,BathroomLight),t)
      when t = t0 + 492s
```

```
(Action Pattern 5)
 ON occurs (simple,(Off,LivingRoomLight),t0)

 IF context ()
 THEN do (simple,(On,LivingRoomLight),t)
      when t = t0 + 115s
```

```
(Action Pattern 6)
 ON occurs (simple,
            (Off,BathroomLight),t0)
 IF context (TimeOfDay(<,17:31:00))
 THEN do (simple,(On,LivingRoomLight),t)
      when t = t0 + 71s
```

```
(Action Pattern 7)
 ON occurs (simple,(Off,BedroomLight),t0)

 IF context (TimeOfDay(<,13:20:00))
 THEN do (simple,(On,LivingRoomLight),t)
      when t is after t0
```

```
(Action Pattern 8)
 ON occurs (simple,
            (Off,BathroomLight),t0)
 IF context (TimeOfDay(>,17:31:00))
 THEN do (unordered,
          ((On,LivingRoomLight)&
          (Off,LivingRoomLight)),t)
      when t is after t0
```

```
(Action Pattern 9)
 ON occurs (unordered,((On,LivingRoomLight)&
           (Off,LivingRoomLight)),t0)
 IF context ()
 THEN do (simple,(Off,LivingRoomLight),t)
      when t = t0 + 3s
```

```
(Action Pattern 10)
 ON occurs (simple,(Off,BedroomLight),t)

 IF context ()
 THEN do (simple,
          (Off,LivingRoomLight),t)
      when t = t0 + 386s
```

```
(Action Pattern 11)
 ON occurs (simple,(Off,BedroomLight),t0)

 IF context (TimeOfDay(>,13:20:00))
 THEN do (simple,(On,BedroomLight),t)
      when t = t0 + 428s
```

```
(Action Pattern 12)
 ON occurs (simple,
            (On,BedroomLight),t0)
 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 43s
```

```
(Action Pattern 13)
 ON occurs (simple,(On,BedroomLight),t0)

 IF context ()
 THEN do (simple,(Off,BedroomLight),t)
      when t = t0 + 24s
```

```
(Action Pattern 14)
 ON occurs (simple,
            (Off,BedroomLight),t0)
 IF context ()
 THEN do (--,end,t)
      when --
```

```
-----------------------------------------
  Trial 2 (Demanded confidence level: 50%)
-----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,05:30:00)) |
  context (TimeOfDay(>,09:45:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern 1)
  ON occurs (start,--,t0)                     ON occurs (simple,
                                                         (On,LivingRoomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BathroomLight),t)       THEN do (simple,(On,BathroomLight),t)
       when --                                     when t is after t0

 (Action Pattern 2)                          (Action Pattern 3)
  ON occurs (simple,(On,BathroomLight),t0)    ON occurs (simple,
                                                         (Off,BathroomLight),t0)
  IF context ()                               IF context (TimeOfDay(<,17:49:00))
  THEN do (unordered,(Off,BathroomLight),t)   THEN do (simple,(On,LivingRoomLight),t)
       when t is after t0                          when t = t0 + 71s

 (Action Pattern 4)                          (Action Pattern 5)
  ON occurs (simple,(Off,LivingRoomLight),t0) ON occurs (simple,
                                                         (Off,BathroomLight),t0)
  IF context ()                               IF context (TimeOfDay(>,17:49:00))
  THEN do (simple,(On,LivingRoomLight),t)     THEN do (simple,
       when t = t0 + 5s                                  (Off,LivingRoomLight),t)
                                                    when t = t0 + 169s

 (Action Pattern 6)                          (Action Pattern 7)
  ON occurs (simple,(Off,LivingRoomLight),t0) ON occurs (simple,
                                                         (On,LivingRoomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,LivingRoomLight),t)     THEN do (simple,
       when t = t0 + 115s                                (Off,LivingRoomLight),t)
                                                    when t = t0 + 4s

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

```
----------------------------------------
  Trial 2 (Demanded confidence level: 75%
----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,05:30:00)) |
  context (TimeOfDay(>,09:45:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)                          (Action Pattern 1)
  ON occurs (start,--,t0)                     ON occurs (simple,
                                                         (On,LivingRoomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(On,BathroomLight),t)       THEN do (simple,(On,BathroomLight),t)
       when --                                     when t is after t0

 (Action Pattern 2)                          (Action Pattern 3)
  ON occurs (simple,(On,BathroomLight),t0)    ON occurs (simple,
                                                         (Off,BathroomLight),t0)
  IF context ()                               IF context (TimeOfDay(<,17:49:00))
  THEN do (unordered,(Off,BathroomLight),t)   THEN do (simple,(On,LivingRoomLight),t)
       when t is after t0                          when t = t0 + 71s

 (Action Pattern 4)                          (Action Pattern 5)
  ON occurs (simple,(Off,LivingRoomLight),t0) ON occurs (simple,
                                                         (Off,BathroomLight),t0)
  IF context ()                               IF context (TimeOfDay(>,17:49:00))
  THEN do (simple,(On,LivingRoomLight),t)     THEN do (simple,
       when t = t0 + 5s                                  (Off,LivingRoomLight),t)
                                                   when t = t0 + 169s

 (Action Pattern 6)                          (Action Pattern 7)
  ON occurs (simple,(Off,LivingRoomLight),t0) ON occurs (simple,
                                                         (On,LivingRoomLight),t0)
  IF context ()                               IF context ()
  THEN do (simple,(Off,LivingRoomLight),t)    THEN do (simple,
       when t = t0 + 115s                                (Off,LivingRoomLight),t)
                                                   when t = t0 + 4s
```

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


----------------------------------------
  Trial 3 (Demanded confidence level: 25%
----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,05:00:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)
  ON occurs (start,--,t0)
  IF context ()
  THEN do (unordered,((On,BedroomLight)&(Off,BedroomLight)),t) when --


 (Action Pattern -)
  ON occurs (unordered,((On,BedroomLight)&(Off,BedroomLight)),t0)
  IF context ()
  THEN do (--,end,t) when --


----------------------------------------
  Trial 3 (Demanded confidence level: 50%
----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,05:00:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)
  ON occurs (start,--,t0)
  IF context ()
  THEN do (unordered,((On,BedroomLight)&(Off,BedroomLight)),t) when --
```

```
 (Action Pattern -)
  ON occurs (unordered,((On,BedroomLight)&(Off,BedroomLight)),t0)
  IF context ()
  THEN do (--,end,t) when --

-----------------------------------------
  Trial 3 (Demanded confidence level: 75%
-----------------------------------------

(Action Map 1)

 (General Conditions)
  context (TimeOfDay(>,00:00:00)) & context (TimeOfDay(<,05:00:00)) |
  context (TimeOfDay(>,08:00:00)) & context (TimeOfDay(<,23:59:59))

 (Action Pattern -)
  ON occurs (start,--,t0)
  IF context ()
  THEN do (unordered,((On,BedroomLight)&(Off,BedroomLight)),t) when --


 (Action Pattern -)
  ON occurs (unordered,((On,BedroomLight)&(Off,BedroomLight)),t0)
  IF context ()
  THEN do (--,end,t) when --
```

Appendix I: Validating the Action Map Approach with WSU data

```
(Action Map 1)

 (General Condition)
   context (TimeOfDay(>,10:45:00)) & context (TimeOfDay(<:18:15:00)) &
   context (DayOfWeek(=,(Monday,Tuesday,Wednesday,Thursday,Friday)))

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

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

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

 (Action Pattern 3)
  ON occurs (simple,(Off,Phone),t0)
  IF context ()
  THEN do (simple,(On,Water(0)),t)
       when t is after t0

 (Action Pattern 4)
  ON occurs (simple,(On,Water(0)),t0)
  IF context ()
  THEN do (simple,(Off,Water(0)),t)
       when t = t0 + 23s

 (Action Pattern 5)
  ON occurs (simple,(Off,Water(0)),t0)
  IF context ()
  THEN do (unordered,((On,Cabinet(0))&
          (Off,Cabinet(0))),t)
       when t = t0 + 39s

 (Action Pattern 6)
  ON occurs (unordered,((On,Cabinet(0))&
            (Off,Cabinet(0))),t0)
  IF context ()
  THEN do (unordered,((On,Oatmeal)&
          (On,Raisins)),t)
       when t = t0 + 3s

 (Action Pattern 7)
  ON occurs (unordered,((On,Oatmeal)&
                    (On,Raisins)),t0)
  IF context ()
  THEN do (simple,(On,Sugar),t)
       when t = t0 + 2s
```

```
(Action Pattern 8)
 ON occurs (unordered,((On,Oatmeal)&
           (On,Raisins)),t0)
 IF context ()
 THEN do (simple,(On,MeasuringSpoon,t)
      when t = t0 + 3s
```

```
(Action Pattern 9)
 ON occurs (simple,(On,Sugar),t0)

 IF context ()
 THEN do (simple,(On,MeasuringSpoon),t)
      when t = t0 + 5s
```

```
(Action Pattern 10)
 ON occurs (simple,(On,MeasuringSpoon),t0)
 IF context ()
 THEN do (simple,(On,Bowl),t)
      when t = t0 + 18s
```

```
(Action Pattern 11)
 ON occurs (simple,(On,Bowl),t0)
 IF context ()
 THEN do (simple,(Off,Cabinet(0)),t)
      when t = t + 7s
```

```
(Action Pattern 12)
 ON occurs (simple,(Off,Cabinet(0)),t0)
 IF context ()
 THEN do (simple,(On,Water(1)),t)
      when t = t0 + 10s
```

```
(Action Pattern 13)
 ON occurs (simple,(On,Water(1)),t0)
 IF context ()
 THEN do (simple,(Off,Water(1)),t)
      when t = t + 5s
```

```
(Action Pattern 14)
 ON occurs (simple,(Off,Water(1)),t0)
 IF context (TimeOfDay(<,14:17:00))
 THEN do (simple,(On,Pot),t)
      when t = t0 + 4s
```

```
(Action Pattern 15)
 ON occurs (simple,(Off,Water(1)),t0)
 IF context (TimeOfDay(<,14:17:00))
 THEN do (simple,(On,Burner),t)
      when t = t + 12s
```

```
(Action Pattern 16)
 ON occurs (simple,(On,Pot),t0)
 IF context ()
 THEN do (simple,(On,Burner),t)
      when t = t0 + 10s
```

```
(Action Pattern 17)
 ON occurs (simple,(On,Burner),t0)
 IF context ()
 THEN do (simple,(Off,Burner),t)
      when t is after t0
```

```
(Action Pattern 18)
 ON occurs (simple,(Off,Burner),t0)
 IF context ()
 THEN do (simple,(On,Cabinet(1)),t)
      when t is after t0
```

```
(Action Pattern 19)
 ON occurs (simple,(On,Cabinet(1)),t0)
 IF context ()
 THEN do (simple,(On,Medicine),t)
      when t = t + 2s
```

```
(Action Pattern 20)
 ON occurs (simple,(On,Medicine),t0)
 IF context ()
 THEN do (simple,(Off,Cabinet(1)),t)
      when t = t0 + 2s
```

```
(Action Pattern 21)
 ON occurs (simple,(Off,Cabinet(1)),t0)
 IF context (DayOfWeek(<>,Tuesday))
 THEN do (simple,(On,Water(2)),t)
      when t = t + 9s
```

```
(Action Pattern 22)
 ON occurs (simple,(Off,Cabinet(1)),t0)
 IF context (DayOfWeek(=,Tuesday))
 THEN do (simple,(On,ColdWater),t)
      when t = t0 + 4s
```

```
(Action Pattern 23)
 ON occurs (simple,(On,Water(2)),t0)
 IF context ()
 THEN do (simple,(On,ColdWater),t)
      when t = t + 40s
```

```
(Action Pattern 24)
 ON occurs (simple,(On,ColdWater),t0)

 IF context ()
 THEN do (unordered,((On,Cabinet(2))&
         (On,Water(2))&(Off,Water(2))),t)
      when t = t0 + 8s
```

```
(Action Pattern 25)
 ON occurs (unordered,((On,Cabinet(2))&
   (On,Water(2))&(Off,Water(2))),t0)
 IF context ()

 THEN do (simple,(On,Cabinet(2)),t)
      when t = t + 6s
```

```
(Action Pattern 26)
 ON occurs (simple,(On,Cabinet(2)),t0)
 IF context ()
 THEN do (simple,(Off,Medicine),t)
      when t = t0 + 2s
```

```
(Action Pattern 27)
 ON occurs (simple,(Off,Medicine),t0)
 IF context ()
 THEN do (simple,(Off,Cabinet(2)),t)
      when t = t + 8s
```

```
(Action Pattern 28)
 ON occurs (simple,(Off,Cabinet(2)),t0)
 IF context ()
 THEN do (simple,(On,Water(3)),t)
      when t = t0 + 15s
```

```
(Action Pattern 29)
 ON occurs (simple,(On,Water(3)),t0)
 IF context ()
 THEN do (simple,(Off,Water(3)),t)
      when t is after t0
```

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

Bibliography

[Aal04] W.M.P. van der Aalst, A.J.M.M. Weitjers, and L. Maruster. Workflow mining discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, vol. 18(9):pp. 1128–1142, 2004.

[Agh09] H. Aghajan, J.C. Augusto, and R. Lopez Cozar Delgado. *Human-centric interfaces for ambient intelligence*. Academic Press, 2009.

[Agr95] R. Agrawal and R. Srikant. Mining sequential patterns. In *Pro. 11th International Conference on Data Engineering*, pp. 3–14. 1995.

[Aha91] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, vol. 6:pp. 37–66, 1991.

[All84] J. Allen. Towards a general theory of action and time. In *Artificial Intelligence*, vol. 23, pp. 123–154. 1984.

[Aug04] 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)*, pp. 778–782. IO Press, 2004.

[Aug06a] J. C. Augusto and C. D. Nugent, editors. *Designing Smart Homes. The Role of Artificial Intelligence*. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[Aug06b] J. C. Augusto and C. D. Nugent. *Smart homes can be smarter*, pp. 1–15. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[Aug07a] J.C. Augusto and D. Cook. *Ambient Intelligence: applications in society and opportunities for AI. Tutorial Lecture Notes delivered at 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*. IJCAI, Hyderabad, India, January 2007.

[Aug07b] J. C. Augusto. *Ambient Intelligence: the Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence*, pp. 213–234. Intelligent Computing Everywhere. Springer London, 2007.

[Aug07c] J. C. Augusto and P. McCullagh. Ambient intelligence: Concepts and applications. In *Computer Science and Information Systems*, vol. 4, pp. 1–28. ComSIS Consortium, 2007.

[Aug09] J. C. Augusto. Past, present and future of ambient intelligence and smart environments. In *1st International Conference on Agents and Artificial Intelligence (ICAART)*. 2009.

[Azt08] 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)*. 2008.

[Beg06] R. Begg and R. Hassan. *Artificial neural networks in smart homes*, pp. 146–164. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[Boi99] A. Boisvert and R. B. Rubio. Architecture for intelligent thermostats that learn from occupants' behavior. In *ASHRAE Transactions*, pp. 124–130. 1999.

[Brd05] O. Brdiczka, P. Reignier, and J. L. Crowley. Supervised learning of an abstract context model for an intelligent environment. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, vol. 121, pp. 259–264. ACM, 2005.

[Bur04] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, vol. 2(2):pp. 121–167, 2004.

[Cam06] E. Campo, S. Bonhomme, M. Chan, and D. Esteve. Learning life habits and practices: an issue to the smart home. In *International Conference on Smart Homes and health Telematic*, pp. 355–358. 2006.

[Cas02] G. Castellano, A.M. Fanelli, and C. Mencar. Generation of interpretable fuzzy granules by a double-clustering technique. *Arch Contr Sci*, vol. 12:pp. 397–410, 2002.

[Cer99] N. Cercone, A. An, and C. Chan. Rule-induction and case-based reasoning: hybrid architectures appear advantageous. *IEEE Transactions on Knowledge and Data Engineering*, vol. 11:pp. 166–174, 1999.

[Cha95] M. Chan, C. Hariton, P. Ringeard, and E. Campo. Smart house automation system for the elderly and the disabled. In *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1586–1589. 1995.

[Coe98] M. H. Coen. Design principles for intelligent environments. In *Proceedings of the 1998 15th National Conference on Artificial Intelligence, AAAI*, pp. 547–554. AAAI Press, 1998.

[Coo07] 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, pp. 53–73. Elsevier Science, 2007.

[Coo08] D. Cook and M. Schmitter-Edgecombe. Activity profiling using pervasive sensing in smart homes. *IEEE Transactions on Information Technology for Biomedicine*, 2008.

[Das06] S. K. Das and Diane J. Cook. Designing and modeling smart environments. In *Proc. of International Sysmposium on Wireless, Mobile and Multimedia Networks*, pp. 490–494. 2006.

[Dig09] R.M. van Eijk J. van Diggelen, R.J. Beun and P.J. Werkhoven. Efficient semantic information exchange for ambient intelligence. *The computer journal*, 2009.

[Doc05] 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, pp. 55–65. 2005.

[Doo06] 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*, pp. 51–59. 2006.

[Duc01] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J. C. Burgelman. Scenarios for ambient intelligence in 2010. Tech. rep., 2001.

[Dum08] H. Duman, H. Hagras, and V. Callaghan. Intelligent association exploration and exploitation of fuzzy agents in ambient intelligent environments. *Journal of Uncertain Systems*, vol. 2(2):pp. 133–143, 2008.

[Duo06] V. Duong, Q. Phung, H. Bui, and S. Venkatesh. Human behavior recognition with generic exponential family duration modeling in the hidden semi-markov model. In *18th International Conference on Pattern Recognition, ICPR 2006*, vol. 3, pp. 202–207. 2006.

[Erm08] M Ermes, J Parkka, J Mantyjarvi, and I Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled contitions. *IEEE Transactions on Information Technology in Biomedicine*, vol. 12:pp. 20–26, 2008.

[Fri05] M. Friedwald, O. M. Da Costa, Y. Punie, P. Alahuhta, and S. Heinonen. Perspectives of ambient intelligence in the home environment. In *Telematics and Informatics*, vol. 22, pp. 221–238. Pergamon Press, 2005.

[Gal01] C. Le Gal, J. Martin, A. Lux, and J. L. Crowley. Smartoffice: Design of an intelligent environment. *IEEE Intelligent Systems*, vol. 16(4):pp. 60–66, 2001.

[Gal06] M. Galushka, D. Patterson, and N. Rooney. *Temporal data mining for smart homes*, pp. 85–108. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[Gop04] K. Gopalratnam and D.J. Cook. Active lezi: An incremental parsing algorithm for sequential prediction. *International Journal of Artificial Intelligence*, vol. 14:pp. 917–930, 2004.

[Got06] B. Gottfried, H. W. Guesgen, and S. Hubner. *Spatiotemporal reasoning for smart homes*, pp. 16–34. Designing Smart Homes. The Role of Artificial Intelligence. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[Hag04] 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(6):pp. 12–20, 2004.

[Hei02] 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*, pp. 537–540. 2002.

[Hog05] R. Hogg, J. McKean, and Allen Craig. *Introduction to Mathematical Statistics*, pp. 359–364. Pearson Prentice Hall, 2005.

[Jak07a] 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.

[Jak07b] V. R. Jakkula, A. S. Crandall, and D. J. Cook. Knowledge discovery in entity based smart environment resident data using temporal relation based data mining. In *7th IEEE International Conference on DataMining*, pp. 625–630. 2007.

[Jan93] J.S.R. Jang. Anfis: Adaptive-network-based fuzzy inference system. *IEEE Systems, man and cybernetics*, vol. 23:pp. 665–684, 1993.

[Jia04] Li Jiang, Da-You Liu, and Bo Yang. Smart home research. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, vol. 2, pp. 659–663. 2004.

[Kas07] T.L.M. van Kasteren and B.J.A. Kröse. Bayesian activity recognition in residence for elders. In *Proceedings of the 3rd International Intelligent Environments Conference*, pp. 209–212. 2007.

[Kus04] N. Kushwaha, M. Kim, D. Y. Kim, and W. Cho. An intelligent agent for ubiquitous computing environments: Smart home ut-agent. In *Proceedings of the 2nd IEEE Workshop on Software Technologies for future Embedded and Ubiquitous Systems*, pp. 157–159. 2004.

[Lea06] D. Leake, A. Maguitman, and T. Reichherzer. *Cases, context, and comfort: opportunities for case-based reasoning in smart homes*, pp. 109–131. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006.

[Mit97] T. M. Mitchell. *Machine Learning*. The McGraw-Hill and MIT Press, 1997.

[Moz95] M. C. Mozer, R. H. Dodier, M. Anderson, L. Vidmar, R. F. Cruickshank, and D. Miller. *The neural network house: an overview*, pp. 371–380. Current trends in connectionism. Erlbaum, 1995.

[Mul04] M. E. Muller. Can user models be learned at all? inherent problems in machine learning for user modelling. In *Knowledge Engineering Review*, vol. 19, pp. 61–88. Cambridge University Press, 2004.

[Pan06] M. Pantic, A. Pentland, A. Nijholt, and T. Huang. Human computing and machine understanding of human behavior: A survey. In *Proceedings of the 8th international conference on Multimodal interfaces*, pp. 239–248. ACM, 2006.

[Par06] T. Partala, V. Surakka, and T. Vanhala. Real-time estimation of emotional experiences from facial expressions. *Interacting with Computers*, vol. 18(2):pp. 208–226, 2006.

[Ram08] Carlos Ramos, Juan Augusto, , and Daniel Shapiro. Ambient intelligence - the next step for artificial intelligence (guest editors' introduction to the special issue on ambient intelligence). *IEEE Intelligent Systems*, vol. 23(2):pp. 15–18, Mar/Apr 2008.

[Rao04] S. P. Rao and D. J. Cook. Predicting inhabitant action using action and task models with application to smart homes. *International Journal on Artificial Intelligence Tools (Architectures, Languages, Algorithms)*, vol. 13(1):pp. 81–99, 2004.

[Rus03] S.J. Russell and P. Norvig. *Artificial Intelligence: A modern approach, 2nd edition*. Prentice Hall, 2003.

[Sad05] N. M. Sadeh, F. L. Gandom, and O. B. Kwon. Ambient intelligence: The mycampus experience. Tech. Rep. CMU-ISRI-05-123, ISRI, 2005.

[Sah03] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *IEEE Computer*, vol. 36:pp. 25–31, 2003.

[San09] L.A. SanMartin, V.M. Pelaez, R. Gonzalez, and A.M. Campos. Environmental user preference learning for smart homes. In *Proceedings of the 5th International Conference on Intelligent Environments*, pp. 177–184. 2009.

[Sha07] H. Sharp, Y. Rogers, and J. Preece. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley and Sons Ltd., 2007.

[Sta04] V Stanford. Biosignals offer potential for direct interfaces and health monitoring. *IEEE Pervasive Computing*, vol. 3:pp. 99–103, 2004.

[Sta06] V. Stankovski and J. Trnkoczy. *Application of decision trees to smart homes*, pp. 132–45. Designing Smart Homes. The Role of Artificial Intelligence, ed. Augusto,J. C. and Nugent,C. D. Springer-Verlag, 2006. M1: Copyright 2006, The Institution of Engineering and Technology.

[Tap04] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Proceedings of Pervasive*, pp. 158–175. 2004.

[Tur07] M. Turunen, J. Hakulinen, A. Kainulainen, A. Melto, and T. Hurtig. Design of a rich multimodal interface for mobile spoken route guidance. In *Proceedings of Interspeech 2007 - Eurospeech*. 2007.

[Vai08] A. M. Vainio, M. Valtonen, and J. Vanhala. Proactive fuzzy control and adaptation methods for smart homes. *IEEE Intelligent Systems*, vol. 23(2):pp. 42–49, 2008.

[Wan92] L.X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Systems, man and cybernetics*, vol. 22:pp. 1414–1427, 1992.

[Wan03] L.X. Wang. The mw method completed: A flexible system approach to data mining. *IEEE Transactions fuzzy systems*, vol. 11:pp. 768–782, 2003.

[Wat92] C.J.C.H. Watkins and P. Dayan. Q learning. *Machine Learning*, vol. 8:pp. 279–292, 1992.

[Wei91] M. Weiser. The computer for the 21st century. *Scientific American*, vol. 265(3):pp. 94–104, 1991.

[Wei93] M. Weiser. Hot topics: Ubiquitous computing. *IEEE Computer*, vol. 26:pp. 71–72, 1993.

[Wei01] A.J.M.M. Weijters and W.M.P. van der Aalst. Process mining discovering workflow models from event-based data. In *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pp. 283–290. 2001.

[Wen07] L. Wen, W.M.P. van der Aalst, J.Wang, and J. Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, vol. 15(2):pp. 145–180, 2007.

[Wit05] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed.* Elsevier, 2005.

[Wu09] C. Wu and H. Aghajan. Using context with statistical relational models - object recognition from observing user activity in home environment. In *Workshop on Use of Context in Vision Processing (UCVP), ICMI-MLMI*. 2009.

[You05] G. M. Youngblood, D. J. Cook, and L. B. Holder. Managing adaptive versatile environments. In *IEEE International Conference on Pervasive Computing and Communications*. 2005.

[Zad65] L.A. Zadeh. Fuzzy sets. *Information and Control*, vol. 8:pp. 338–353, 1965.

[Zai08] S. Zaidenberg, P. Reignier, and J. L. Crowley. Reinforcement learning of context models for a ubiquitous personal assistant. In *Proceedings of the 3rd Symposium of Ubiquitous Computing and Ambient Intelligence*, vol. 51/2009, pp. 254–264. 2008.