Karolina Baras Dynamic world model for context-aware environments

UMinho | 2011

Universidade do Minho
Escola de Engenharia

Karolina Baras

Dynamic world model for context-aware environments

Outubro de 2011

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR Portugal

Universidade do Minho
Escola de Engenharia

Karolina Baras

Dynamic world model for context-aware environments

Tese de Doutoramento
Tecnologias e Sistemas de informação

Trabalho efectuado sob a orientação do
Professor Doutor Adriano Moreira

Outubro de 2011

Acknowledgements

The development of this project was only possible because of the collaboration and support of many people.

First of all, I owe my deepest gratitude to my supervisor Doctor Adriano Moreira who made this project possible in the first place. I thank him for his infinite patience, availability, optimism, encouragement, excellent sense of direction and constructive feedback and ideas. I am grateful to Doctor Filipe Meneses for his availability and friendship. In particular, I would like to thank him for the design and the creation of the campus map, the implementation of data cleaning and session duration analysis procedures. I would also like to thank all the members of the Ubicomp group at the School of Engineering, University of Minho for their cooperation, kindness and support.

I would like to thank Fundação para a Ciência e a Tecnologia for all the granted financial support under the doctoral grant number SFRH/BD/36046/2007.

I owe my deepest gratitude to the University of Madeira, for conceding me a three years long release from duty and a two year contract extension to finish my research project. I also thank my colleagues from Exact Sciences and Engineering Center for all the support, dialogue, friendship and fun.

Also, I would like to show my gratitude to all the participants in the user study without whom it could not be carried out successfully.

I am grateful to my friends from Minho region, especially to Sónia Ferreira for her hospitality.

Finally, I want to thank my family and friends for their infinite love and support.

Dynamic world model for context-aware environments

Abstract

The main subject of this dissertation is a dynamic symbolic world model which is connected to sensor networks through data acquisition and processing modules. Being dynamic, it is capable of accompanying the state of the represented spaces in real time. This model can be distributed over several servers as all the objects in the model are uniquely described by URI and relations may exist with objects located in remote servers. It also allows users to construct and share their own models of the world as they visit different places.

Symbolic world models are complementary to geometric world models which are much more widespread. They are similar to human mental models of space, allow for meaningful descriptions of locations and the creation of relations between them.

The main contributions of our research are: the space characterization based on traces from available networks, automatic feeding of the world model through a set of processing modules that receive information from a wireless network, a world model that resembles human mental models of space, an editing tool for manual creation of the model and a set of applications that contribute to the model expansion and/or use the existing data from the model.

Modelo de espaços dinâmico para ambientes sensíveis ao contexto

Resumo

O tema principal desta dissertação é um modelo de espaços simbólico e dinâmico ligado a uma rede de sensores através de módulos de aquisição e processamento de dados. Sendo dinâmico, acompnha o estado dos espaços representados em tempo real. Este modelo pode ser distribuído por vários servidores uma vez que todos os objctos no modelo estão univocamente identificados por um URI e poderão ser estabelecidas relações entre objectos localizados em servidores remotos. Os utilizadores podem criar e partilhar os seus próprios modelos na medida em que vão visitando os diversos locais de interesse.

Os modelos de espaço simbólicos complementam os modelos geométricos que são muito mais comuns. São semelhantes aos modelos mentais, permitem a introdução de descrições mais ricas dos objectos e a criação de relações entre eles.

Os contributos deste projecto de investigação são: caracterização dos espaços baseada nos dados provenientes de redes existentes, actualização automática do modelo de espaços através de um conjunto de módulos de procesamento que recebem os dados de uma rede sem fios, um modelo de espaços semelhante aos modelos mentais, uma ferramenta para criação manual do modelo de espaço e um conjunto de aplicações que actualizam o modelo e/ou utilizam os dados existentes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The only real voyage of discovery consists not in seeing new landscapes, but in having new eyes.
Marcel Proust

In the following pages, we propose a way of representing the world based on symbols that mimics the way we mentally represent spaces. In psychology, research has been done about the way humans build their mental representations of space. They found that human mental models consist of elements and relations between them as described in (Tversky, 2000).

In fact, when traveling by car, we do not actually remember the whole trajectory of a highway, but we do remember the starting and the ending points of our journey with some stopping in between on a gas station. The same occurs for a train trip, as we only want to know if there is a connection between an origin and a destination and at what time. The exact shape of the rail road, the existing slopes and curves are completely irrelevant for most of the passengers.

One of the most illustrative examples of a symbolic representation of space is a subway map. In this map, the essential knowledge is stored in the names of the stations and in the colors of the lines we need to follow if we wish to travel between two stations of our choice. Similarly, in our model, places are represented as the stations of a subway and relations as connections between them.

The main subject of this research is a dynamic symbolic world model which is connected to sensor networks through data acquisition and processing modules.

This feature makes it capable of accompanying the state of the represented spaces in real time. This model is meant to be distributed over several servers as all the objects in the model are uniquely described by an Uniform Resource Identifier (URI) and relations exist among them. It also allows users to construct and share their own models of the world as they visit different places.

When we talk about the space dynamics, we mean the changes of its context over time. This encompasses all factors that may contribute to the characterization of a place and that are variable. It can be the lighting conditions, the temperature, the number of people in a room, the number of tweets[1] submitted from a place, or something else.

1.1 World modeling

People create naturally their own mental models of space that surrounds them. From the early ages graphical representations of these models started to emerge. Maps, globes and plans were created for this purpose. With the fast technology advances, geographical information systems emerged and, more recently, the digital maps started to proliferate all over the Internet and on many hand-held devices.

With the advent of the ubiquitous computing in the early nineties, location models also became an interesting topic to be studied in this new area of computing. Once ubiquitous computing is all about disappearing technology and computers that are dissolved in the surroundings as described by Weiser in his visionary articles (Weiser, 1991, 1993), new ways of modeling the environment needed to be found in order to make the environment "visible" to the deployed devices and, through applications, to users.

World models, also referred to as location models or space models, are representations of the real world containing information about places, devices, things, people and the relations among them. World models are classified in three main groups: symbolic, geometric and hybrid (Becker & Dürr, 2005; Leonhardt, 1998).

[1] http://twitter.com/

Symbolic or topological models represent locations in the form of abstract symbols, e.g., Home, Dad's favorite restaurant, Library, and so forth. Geometric or geographic models define positions in the form of coordinate pairs within a reference coordinate system, e.g., $32°46'N, 17°0'W$. Geographic maps and building plans are the well known examples of these. Hybrid location models combine the two, symbolic and geometric representations of space, in order to take advantage of both.

Creating geometric world models is not straightforward, reading them neither. The creation of a geographic map from scratch demands an enormous effort and specialized knowledge. Today's huge proliferation of free maps and mobile devices with integrated GPS receivers is the response of the market to an ever increasing demand in this area. Due to such a great number of potential users, the investment in the creation of digital maps turns out to be worthwhile.

From our experience it follows that the symbolic world models, as opposed to geometric world models, allow for a much easier creation of the basic model itself, be it by an experienced user, be it by a novice user. Symbolic world models reflect the way people remember places and create relations among them mentally. However, people are so used to map viewing and following that when there is no geographical representation of the world, they feel "lost" as our short user study results will show. This is one of the reasons why the best of both worlds is achieved by the combination of symbolic and geometric models. The integration of the proposed world model with a geometric model of the same environment is one of the future steps for this project.

1.2 Thesis statement

The initial hypothesis of this research project is stated in the following terms:

> Context-aware environments and the applications they offer to users can benefit from the existence and availability of dynamic models of the world, as they introduce new sources of knowledge about spaces where they are deployed.

1. INTRODUCTION

In context-awareness, as stated by Schmidt *et al.* (1999), there are many more interesting things besides location to be aware of. The physiological state of the user or his/her personal history, physical or social conditions of the environment, for example.

In the area of context-awareness the greatest attention has been given to defining context of users which is completely comprehensible. Even more so when the computing devices accompany people permanently. It is expected that those devices behave according to the context of the user. Moreover, the context of a person, or the context in which an application is running on a personal device is, to a large extent, the context of the place where that person is located. In other words, the characteristics of a place, and its current and past states, are a fundamental part of a person's context – "tell me where you are and I'll tell you your context". This is why location and position have been ever since described as one of the most fundamental dimensions of context.

In our research, however, we are more interested in the characterization of the context of places than the characterization of the context of people. Nevertheless, the presence of people is very relevant to our work as we take advantage of the wireless network traces for the characterization of spaces according to the usage patterns.

Recently we have witnessed the appearance of several location based social networking platforms where people share the places they visit with their friends, collect pins and badges, become "mayors" if they check in very frequently and more than anyone else. foursquare[1] and Gowalla[2] are the most popular examples of these with millions of users all over the world. Commercial establishments have used them as a tool for advertising and drawing more clients by offering them prizes and discounts. In the scope of our research, these applications may also be seen as a tool for creating world models collaboratively, although without any predefined structure. The "check in" feature implemented in both these social networking mobile web applications is very close to one of our proposed features that detect frequent users of a particular space.

[1] https://foursquare.com/

[2] http://gowalla.com/

In an establishment using the above mentioned applications, the existence of a dynamic world model that takes advantage of the data coming from the shared profiles of its employees, visitors and clients could open a brand new perspective on the dynamics of the space.

Our approach started with the studying of the dynamic nature of spaces. This initial phase included the acquisition and the analysis of WiFi network data which provided us with a picture of the dynamics of an University campus. Then, based on an existing symbolic world model, we upgraded it to a new model capable of automatic recognition of the characteristics of a space (physical conditions, relations to other spaces, typical utilization patterns, and so on). An editing application was developed to support the manual creation of the elements and their relations in the model.

The validation process of this thesis included the development of a service that supports dynamic symbolic models of space, the development of a set of sensor data processing modules that feed the model and a set of applications that use the data available in the model. The model, being dynamic is capable of self-updating according to the physical, environmental or social changes in spaces it represents.

1.3 Contributions

The main contributions of this research project in the area of the space modeling are the following:

1. **To characterize spaces in terms of their dynamic nature and identify typical usage patterns.**

   For each space this includes the discovery of the parameters that describe it best and reflect the pulse of the space, their acquisition, processing and insertion in the model. Regarding data collection, the use of existing infrastructures was encouraged from the beginning of this project.

2. **To develop a conceptual model capable of accompanying the dynamic nature of spaces and the relations between them, being**

distributed and fed by a social network of users, as well as automatically by a set of processing modules.

Data collected from the environment needs to be processed and stored appropriately in the model. In order to achieve this, a set of processing modules is required and the model itself needs several enhancements to be able to receive dynamic data about its elements. For each user to be able to create and edit his/her own model and share it with other users, an editing application is needed as well as a mechanism of connecting the different personal models among them.

3. **To develop inference algorithms that, based on information collected by the sensors, allow for inference of the higher level parameters that describe the space dynamics.**

   Not all the knowledge about a space can be stored explicitly in the model. Thus, mechanisms are needed to infer that knowledge from the existing data. The foreseen processing modules and applications generate new knowledge about a space by inference, either based on the raw sensor data or, based on the already existing data in the space model or, even based on the combination of both.

1.4 Thesis outline

The thesis is structured in seven chapters and one appendix.

Chapter 1 (this chapter) introduces the motivation, the concept of world models, the thesis statement and the main contributions of this research project.

In Chapter 2, a selected number of publications from the area of context-awareness, location modeling and WiFi network data analysis are described and related to our work.

The fundamentals about the developed dynamic symbolic world model are introduced in Chapter 3.

Chapter 4 describes the general system architecture which integrates the proposed space model.

Chapter 5 gives an overview of the applications that may take advantage of the characteristics of the model.

Chapter 6 reports the obtained results and evaluates them in light of the initially proposed objectives.

Finally, Chapter 7 concludes the document by revisiting the thesis statement, providing a list of published papers and suggesting future developments.

Appendix A contains the Symbolic Contextualizer functional interface description.

Appendix B contains the materials that were used during the user study and a summary of the obtained results.

Chapter 2

Related work

The larger the island of knowledge,
the longer the shoreline of mystery.
Unknown author

Mark Weiser, the founder of the ubiquitous computing research area, idealized a world of invisible technology which is interwoven in the environment. Twenty years later, this vision is not exactly our reality, but a lot of effort has been done in that direction. Several authors have discussed whether the ubiquitous computing initial vision should continue to be the one to follow or should a new one be adopted according to our present reality which is rather different than the one of the late eighties. In (Bell & Dourish, 2007) the authors acknowledge that ubiquitous computing is already a reality. However, it turned out to be slightly different from the original vision. According to them, the initial vision of the computing of the future cannot continue to be pushed forward to the future as something unreachable. Of course, as the authors said, the "seamless interoperation and homogeneity" is not real yet and may never be, but there are evidence of real ubiquitous systems working in our everyday life.

The authors of the above article give the examples of Singapore and Korea where the technology pervades many spheres of people's lives, but there is no need to go so far away. We can think about our daily lives and see to what extent the technology has become interwoven in it. As Schmidt (2010) points out, we already use GPS without even being aware of it because it is already a part of our smartphone and of our car. The same goes for digital cameras, music players, etc. Also the wireless and mobile access to the Internet is becoming ever more widespread. Additionally, if we think of all the sensors that are embedded in our cars, smartphones and in the surrounding environment, as well as all the stuff related to the usage of so popular social networks, we may become more convinced about the reality of the ubiquitous computing in our world, different from the initial vision, but real.

In the following sections we focus on one of the subareas of ubiquitous computing – the context-awareness. We present a review of selected works from this area with a particular attention to the world modeling. We also refer to the research from the wireless network analysis and anomaly detection areas as these are also closely related to a part of our work.

2.1 Context and context-awareness

In order to understand the meaning of "context" in the scope of ubiquitous computing, we first explore the definitions given by several researchers from the area of context-aware computing. Second, we describe several options for context modeling. Third, we explore different types and categories of world models.

2.1.1 Defining context

Now we will have a look at some definitions of context in the scope of context-awareness and ubiquitous computing. Context is defined by Schilit & Theimer (1994) as being the "location and the identity of nearby people and objects". At the beginning, location was the dominant parameter for context description. Additionally, the identity of people started to be also considered as it allowed for application adaptation to their profiles and preferences. A few years later, Schmidt *et al.* (1999) suggested that instead of considering location only, additional parameters should be considered, such as those that characterize physical conditions of the environment. A similar definition of context is given by Dey (2001): "any information that can be used to characterize the situation of entities

(...) typically the location, identity and state of people, groups and computational and physical objects".

One of the broadest definitions describes context as encompassing all the characteristics and objects in the environment that are mobile and/or changeable, including computational resources, physical conditions and social situation. This definition is also one of the earliest and was published in (Schilit *et al.*, 1994):

> "Context encompasses more than just the user's location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth and even the social situation, e.g., whether you are with your manager or with a co-worker."

The term "context-aware" appears for the first time in (Schilit & Theimer, 1994). It emerges as a consequence of the mobile computing development. As the mobile devices move around in their environment, the running applications should become aware of these changes and adapt to them appropriately. A question that arises concerning context-aware applications is: how much autonomy should be given to the applications? This is quite interestingly discussed in (Erickson, 2002). The author defends that a context-aware system should focus mainly on the context acquisition and processing and leave the decision making to the user. Also, in (Rogers, 2006), the author defends an engaging rather than calm technology, instead of "making the environment smart and proactive, enable people to be smarter and more proactive". This means putting people in control and letting their creativity reinvent the everyday tasks instead of searching for the ideal invisible technological solutions that do it all for them.

In the scope of our project, context encompasses all the characteristics of a space, like its name, its purpose, the number of people present, the physical conditions and the relations to other spaces. For example, the context of a laboratory at some moment in time may be described as follows: *there is a room called Lab 1 with an area of 30 square meters, white walls, gray floor, 1 door, 3 windows; it is situated on the 2^{nd} floor of the building ABC, next to Lab 2 and it is accessible from the main corridor; at 19:00 of the 12^{th} December 2010, there were 5 people* *connected to the WiFi network access point AP1 which is situated near Lab 1; it is typically used from 9:00 to 18:00, by 8 people on average.*

Based on the above description, an object could be created in our model representing the described space. This object would have a name – *Lab 1* and a type – *room*. It would have several attributes that describe its physical characteristics: its area – *30 sq m*, the color of its walls – *white*, the color of its floor – *gray*, the number of doors – *1*, the number of windows – *3*. Besides these attributes that most probably would suffer no changes over time, it would have also several dynamic attributes: the number of connected devices to the nearest WiFi access point – *5 at 19:00 of the 12th December 2010*, its typical usage schedule – *from 9:00 to 18:00* and the average number of users – *8*. It would also have a set of relations with other objects: a containment relation with the *2nd floor* of the *building ABC*, an adjacency relation with the *Lab 2*, a connectedness relation with the *main corridor* and a proximity relation with the *AP1*.

Based on the above description, algorithms may be used to infer more knowledge about this space. For example, it may be pointed out that something different from the usual was happening on the 12th December 2010 as there were 5 people connected to the nearby access point at 19:00 while, normally, this place is used from 9:00 to 18:00.

2.1.2 Context modeling

Context models have been studied by a large number of researchers all over the world. Mostly, these are models describing context of people and applications. Initially, location was the dominant and sometimes the only aspect of context that was considered. Later, other parameters were added, such as people's identity, time stamp, physical conditions and so on as mentioned in the previous section.

A model for structuring the concept of context was proposed by Schmidt *et al.* (1999). It describes the situation and the environment in which a user or a device is found, having a unique name and a set of relevant parameters with their values. The proposed model divides the context in two parts, one is related to the physical characteristics of the environment and the other to human factors which include users' social interactions, performed tasks and so on.

In many cases context models are developed only for a specific application or a specific type of applications. However, it is important to develop generic context models that may be used by a wide range of applications. A review of the most common approaches in the area of context modeling is made by Strang & Linnhoff-Popien (2004). They propose the following classification of the existing models: attribute-value pair models, models based on schemes, graphic models, object oriented models, models based on logic and models based on ontologies.

One of the aims of the work hereby described is to represent not the context of individuals, but the context of spaces which Schmidt *et al.* (1999) call physical characteristics of the spaces, like dimensions, presence of people, noise level, temperature, etc. In other words, all the characteristics that describe a place and, additionally, all those that may be retrieved from the environment by sensor networks. In the scope of our research, when we refer to the dynamics of the space, we refer to its changing context.

The model proposed by Henricksen *et al.* (2002) is similar to ours as it is based on objects, attributes and relations. It is a context model capable of capturing the diversity, the quality and the complex relations existing within the context data. Based on a set of characteristics of the context data (temporal characteristics, information imperfections, alternative representations of context data and high interdependency of context data) an object model is proposed where the context information is structured within a set of entities, each one describing a conceptual or a physical object. The attributes represent the properties of each entity. Associations (unidirectional relations) are used to connect an entity to its attributes and to other entities. These associations may be static or dynamic. Static associations remain unchanged during the lifetime of the entity to which they belong so the context captured through this type of relations is known with a high level of confidence. Dynamic associations are classified according to their origin (sensors, inference algorithms, user). As the related parameters are highly dynamic, the level of confidence may often not be very high. This way of classification allows for inferring the quality of information available in the model.

As an alternative approach, Judd & Steenkiste (2003) propose the Contextual Information Services (CIS) that are organized as a virtual database containing information about location of people, location and the properties of printers, available network bandwidth and so on. In this way, applications may adapt to changes of resources without the user intervention.

Context models proposed by Henricksen *et al.* (2002) and Judd & Steenkiste (2003) emphasize the quality of information gathered from the surrounding environment and the way this information should be processed and supplied to the applications and the user. This is an important aspect especially when the context-aware system is allowed to make the decisions without the intervention of the user. However, in our work, we consider that the information gathered and stored by our system is not critical and at the present level of the development we did not provide any kind of quality of data verification mechanism. Nevertheless, this improvement should be considered in the future.

Extended research has also been done on the ontology development in the field of context-awareness. According to Strang & Linnhoff-Popien (2004) and Reichle *et al.* (2008), among others, the use of ontologies is the most suitable way of creating context models. It may well be so due to its capability of establishing a common vocabulary for a heterogeneous pervasive computing environment.

However, the existing ontologies have mostly been done from scratch in every research project as it seams to be much easier to create a new independent ontology to apply to a certain scenario than to build upon and integrate with the already existing ones. Some approaches provide a way of extending their default scheme in order to make it possible to add new elements to it. This is the case for the ontology defined by Lehmann *et al.* (2004). It allows any context provider to create an extension in order to go beyond the predefined object descriptions. An alternative approach that tries to take the advantage of the ontologies for the knowledge descriptions and of a relational context model for storing, processing and querying highly dynamic data gathered from the environment is proposed by Ejigu *et al.* (2008). After processing separately the context data and the context knowledge, both results are merged to improve the reasoning and the decision support of the whole system.

2.1.3 World modeling

In the first developments in the area of context-awareness, the location parameter played the leading role. Location models, also referred to as world models or space models are representations of the real world containing information about places, devices, things, people and the relations among them. In the research area of ubiquitous computing, location models have been developed mainly for tracking people and identifying the most suitable devices for the interaction with different context-aware applications.

According to Becker & Dürr (2005), a location model should provide a number of features, like, object positions in terms of geometric or symbolic coordinates and local and global reference systems, the distance between the objects and the topological relations of inclusion, connectedness and orientation of the objects.

In the literature, world models have been mostly classified in three groups: symbolic, geometric and hybrid (Becker & Dürr, 2005; Domnitcheva, 2001). Some authors, however, distinguish between topological (Brumitt & Shafer, 2001) or semantic (Pradhan, 2000) world models and metric models. Other, yet, call semantic only those location models that deploy an ontology (Hsieh & Yuan, 2003). Symbolic models represent locations in the form of abstract symbols, e.g., Lab3, Engineering building, Library, and so forth. Geometric models define positions in the form of coordinate tuples belonging to a reference coordinate system. Hybrid location models combine the two in order to take advantage of both, the symbolic and the geometric representation of space. In the following sections we describe examples of location models found in the literature grouped by type.

2.1.3.1 Symbolic world models

Symbolic world models are those models that represent locations and things in the form of abstract symbols. Symbolic models are classified in three different types by Becker & Dürr (2005): set-based models, hierarchical models and graph-based models.

The set based model relies on a set L containing symbolic coordinates. For example, L may contain all the divisions of a floor in a building: $L_{floor2} =$

$\{2.002, 2.003, ..., 2.067\}$. Subsets of this set may be created as needed, for example $A = \{2.002, 2.003\}$ and $B = \{2.003, 2.006\}$. Overlapping locations may be found by calculating the intersection of the subsets, $A \cap B = \{2.003\}$. The spatial inclusion is also easy do calculate, $A \subset L_{floor2}$. A qualitative notion of distance is also possible, but not straightforward. The authors define neighborhoods for this purpose. For example, $d(2.002, 2.003) < d(2.002, 2.006)$ because A is the smallest neighborhood containing locations 2.002 and 2.003 and is a proper subset of L_{floor2} which is the smallest neighborhood containing 2.002 and 2.006. One of the disadvantages of this approach, according to the authors, is the extremely high number of sets that are created and the related modeling effort.

Hierarchical models are constituted by a set of locations L, ordered according to the spatial inclusion relation. If there is no overlapping, it is a tree-based model. If there is overlapping, then it is a lattice-based model. Range queries, which return all the objects contained in a selected area, are straightforward as the model itself is created based on spatial inclusion. A simple qualitative notion of distance is also possible to implement. However, there is no way of modeling connections between locations.

In a graph-based model, symbolic coordinates define vertices of a graph. An edge is added between two vertices if there is a connection between the two locations. The edges may be weighted in order to model the distance between the locations. Due to the way it is constructed, this model supports spatial connection and an explicit definition of distance between locations. According to Becker & Dürr (2005), graph-based symbolic world models are adequate for nearest neighbor queries and navigation, but lack support for the range queries. Range queries are only possible based on the distance from a certain node. In order to get an improvement for the range queries, it is possible to combine set-based and graph-based models resulting in a more flexible model.

An example of a symbolic model representing only the indoor spaces is described in (Satoh, 2008). The main idea behind this approach is to create for each physical object its virtual counterpart. The framework presented in this paper is able to discover available services within the space where the user is located and execute them. It provides a location model that represents the structure of the space, knows the capabilities of existing computing devices and uses software to define services at appropriate devices whenever they are needed. This framework also supports those objects that do not have any computational resources or network interfaces to communicate with other objects, allows them to be listed and helps users to use these objects through their proxies that are running at home servers with abundant computational resources.

According to Kolodziej & Danado (2004), symbolic models are defined as those that describe localization and space in terms of names and abstractions. The authors propose a hierarchical decomposition of places until the required level of detail is achieved. This hierarchy may be represented as a graph or a tree in which each node is a space and each edge is a connection between spaces.

The symbolic model proposed by Kolodziej & Danado (2004) is capable of answering questions like: "Is location x contained in space y?" or "Is space x connected to space y?". In other words, it allows for relative positioning inside an existing hierarchy and for the discovery of adjacent spaces. The biggest drawback of this approach is the lack of precision due to the fact that there is no information about absolute location of a space nor the distance to other spaces. The way a space should be hierarchized is also very subjective and depends on the application that is going to be implemented. Nevertheless, it is very close to the space models we humans create in our minds.

The symbolic space model developed in our research share some similarities with this one. Just like in a graph, the vertices are the spaces themselves (rooms, corridors, campus, etc.) or the objects contained in those spaces (tables, telephones, cars, etc.). Our model is not centered on the location. Rather, its focus is on the context of spaces that it represents. The edges, that represent relations between objects, are, as well, much more generic. Some of the examples of implemented relations, in our model, are "Is_In" (containment), "Is_Accessible_From" (connectedness) and "Is_Next_To" (adjacency). The hierarchy in our model results naturally from the relation "Is_In". The "Is_Accessible_From" relation enables the support for pedestrian navigation applications as we shall demonstrate in Section 5.1. Moreover, in our model, any relation may have a set of attributes which enriches its description and facilitates inference processes.

2.1.3.2 Geometric world models

Maps are everywhere, from the historical city centers, airports, train stations, to our cars, computers and smartphones. They are the most familiar face of the geometric world models. The proliferation of GPS-enabled devices and the wide availability of free maps, have facilitated their acceptance and widespread usage by both professional and non-professional users.

Although the utilization of maps for car navigation purposes has become quite simple and painless, the creation of maps from scratch and their update have never been simple or straightforward tasks. In the past, the early cartographers needed several long term expeditions and explorations of the world to collect the data they needed to draw their representations of the known world. First, they only relied upon their own observations and experience, later many different instruments were invented to determine positions, distances and altitudes.

Today, the surveys are incomparably easier to carry out thanks to the available technology and all the knowledge that has been built over the centuries. GPS-enabled devices are becoming smaller, cheaper and more accurate. Moreover, the ever more popular smartphones frequently feature integrated GPS devices, maps and navigation software. Free maps are provided by many sites all over the Internet. Also, users are encouraged to create their own maps by adding new layers to a provided base map. In OpenStreetMap[1], users upload their own GPS traces and share them. This site allows people to view, edit and share street maps all around the world. The geotagged photos taken by millions of tourists traveling around and shared in community photo collections enable the creation of 3D models of the world as described in (Agarwal *et al.*, 2010).

Geographic Information Systems (GIS) combine cartography, statistical analysis and database technology. They provide tools for map creation, update, overlaying of different data, etc. GIS applications are available for corporate, government and individual use. They may be used for access and management of all kinds of geographically-related information, ranging from urban planning and telecommunications related terrain studies to shortest path finding and Point-Of-Interest (POI) location.

[1]http://www.openstreetmap.org/

In the scope of context-awareness, geometric models have been studied by several researchers. According to Kolodziej & Danado (2004), a geometric model defines a n-dimensional space and the locations are points in that space uniquely specified and represented through coordinates (x, y, z). Geometric location models contain points, lines and polygons that are characterized by metric and non metric attributes. These models allow for a straightforward positioning of the modeled objects as well as the determination of the topological relation of containment. However, connectedness needs to be modeled explicitly and when this happens, it also allows an improvement in distance calculation (Becker & Dürr, 2005).

The EasyLiving project (Brumitt *et al.*, 2000) deploys a geometric location model to provide location-based context in an intelligent environment. The authors are especially concerned with representing the devices present in the environment and with making their usage straightforward for the user. Location of the devices facilitates their recognition by the user. Like this, the user does not need to know the exact name of the device, but s/he can select it by its type and location. This model is adequate for applications running in the indoor environments and which may require sub-meter localization of objects. The model is fed by the sensors deployed in the environment and new data is dynamically added at run time. This characteristic makes it similar to our model although our objectives are different. While the EasyLiving aims to track people and devices inside the environment in order to make the interaction between users and devices smoother, we aim to understand the dynamics of the environment.

2.1.3.3 Hybrid world models

As we mentioned before, the best of both worlds is achieved by combining symbolic and geometric models. By doing this, a new model with better characteristics may be obtained for two main reasons. First, geometric information allows better precision in all distance related queries. Second, arbitrary geometric figures may be used for defining an area for the range queries. As such, the geometric model strengthens the weaker aspects of the symbolic models (lacking support for range queries and distance calculations). We will discuss this further in this section and we will also turn back to it in the final chapter when we talk about future developments of our research.

Two types of hybrid location models are presented in (Becker & Dürr, 2005): a hybrid model with subspaces and a hybrid model with partial subspaces. The former combines a symbolic and a geometric model while the the latter does not consider that the geometric content is modeled for all the locations in order to reduce the modeling effort. The geometric content is defined by using a global or a local reference system. The subspaces are created by integrating coordinate systems within other coordinate systems and defining the position and the orientation of the resulting system. This information allows for the conversion of coordinates from one system to the other.

A location model, called RAUM, is introduced by Beigl *et al.* (2002). It allows for an easier localization of digitally enhanced everyday objects in the environment. This is a hierarchical model organized as a tree. It combines symbolic names and descriptions and Cartesian coordinates. Each path in the tree gives the location of a specific object. This model is also dynamic as it is capable of localizing objects in real time and detect changes that occur in these objects.

In the scope of the Aura project a hybrid space model was developed (Hsieh *et al.*, 2004). It allows a user to insert data based on AutoCAD [1] map format. This model was extended to support the specification of connectivity points between spaces. The authors propose a model for the distance calculation between two locations in a museum as well as an estimate of the time needed to walk that distance.

The requirements for the space service of the Aura project are described in (Miller & Steenkiste, 2007). The authors defend the utility of the hybrid space models. The space service they describe is required to store diverse space information, from exact points (e.g., a printer location), closed named spaces (e.g., an office), unnamed undefined spaces (e.g., coverage area of a WiFi access point), to collections of spaces (e.g., a floor of offices). It should also support interior and exterior spaces. Information about spaces needs to be available to the applications which may use it for different purposes: to infer space properties (calculate

[1] http://usa.autodesk.com/autocad/

distances), for optimization (calculate the shortest path) and for effective communication with users (indicate a path). The space service must respond to queries in a reasonable amount of time even after being expanded. Furthermore, applications may need to take into account user preferences, other information not directly related to the physical space (e.g., class schedules), and the dynamic information (e.g., if the elevators are busy or not). This hybrid model is augmented with the information about the connectivity between spaces and the way they are perceived by humans. Some sample applications are described in order do demonstrate the utility and the efficiency of the model. One of these applications calculates the distance between two places situated inside the same building, on the same floor and on two different floors, and also in two different buildings.

Based on a relational model, Hu & Lee (2004) developed a hybrid location model for constrained environments with heterogeneous entities (buildings, rooms, halls). Each entity may have an arbitrary number of attributes. However, there are two compulsory attributes, one being the identity of a space and the other the geometric position of the entity within the model. They also define an exit hierarchy in order to model the connections and the distances between places.

A hybrid location model that combines lattice based and graph based representations of space is proposed by Ye *et al.* (2007). It describes containment, disjointness and overlapping relations using its lattice model and the adjacency and connectedness relations and notions of distance using its graph model.

The Nexus[1] project was initiated at the end of the nineties and is still extremely active. This is a multidisciplinary project developed at the University of Stuttgart by a large number of researchers divided in several research groups. As a multidisciplinary project involving hundreds of researchers, publications in areas of 3D modeling, urban computing, indoor navigation, navigation systems for blind people, event detection, quality of context and many more have been produced. This has allowed this project to grow and to tackle many important issues related to the area of ubiquitous computing.

The main idea of the Nexus platform for the spatial-aware applications is described by Hohl *et al.* (1999). The authors present the concept of an universal

[1] http://www.nexus.uni-stuttgart.de

open platform that should provide a common ground to location-aware applications worldwide. It consists of world models representing different regions and the objects and people that might happen to be there as well as it adds virtual objects that are only accessible to users of the platform. This creates an augmented world model which contains context data about the environment and offers services to users and applications.

The Nexus project provides a platform for context-aware applications based on a distributed world model. The aim of this platform is to facilitate the interaction between different context-aware applications that use the same underlying location data and the same query language as explained in (Bauer *et al.*, 2002). The world model deployed in this project is further described in (Lehmann *et al.*, 2004). It is a topological model enhanced with some geometric aspects of the modeled objects. Each location represents a vertex and each relation between locations an edge in a graph. Each location may have a set of attributes among which geometry attributes are allowed. It is also organized in multiple levels of detail which makes this model hierarchical. This model is fed by the sensors deployed in the environment and it supports real time detection of events related to the modeled environment as described in (Bauer & Rothermel, 2004, 2005). Nexus's ultimate aim is to develop a platform for the world wide federation of context models called *World Wide Space* (Lange *et al.*, 2009).

2.1.3.4 Other types of world models

An open, distributed collaborative system called *Web of Places* in which spaces and services are represented through HTML pages is introduced by Ostermaier & Bolliger (2008). A geographic search engine with ranking algorithms is used for crawling the web in order to index place and service pages, including their relations and classify them according to their relevance.

The utilization of mobile devices embedded sensors for the construction of a location model without any predefined structure is foreseen by Peddemors & Yoneki (2009). The authors expect that mobile devices share the acquired information about their surroundings and feed the model based on available data. The cooperation between different devices can make the data more accurate if the model takes into account what is seen in common by several devices. It also makes it possible to provide a model of the surrounding environment that is new for the user based upon data shared by other users. This location model should be able to accept different types of data coming from different sensors, capture spatial and temporal aspects and the uncertainty. The model is created from the elementary pattern, derived directly from sensor information and it should be able to produce knowledge at higher levels of abstraction. Frequent patterns and the simultaneous occurrence of these patterns should be detected and a probability should be associated for each one of them. Based on this a compositional hierarchy is created. Data is shared between peers (P2P) in order to improve the model, refine patterns or obtain patterns more rapidly in new environments.

As it follows from the above related research review on world models, there are several possible solutions. In our research we defend the utilization of symbolic world models mainly because of their simplicity, ease of use and great similarity to our human way of representing space mentally.

Next we talk about WiFi networks and the way they may be used for detecting context of spaces.

2.2 WiFi network data analysis

Since the early years of the implementation of the IEEE 802.11 standard, wireless networks have increasingly become commonplace on many university campuses, in enterprises and at other public venues providing access to the Internet on the move. As we shall see in this section, research results have shown that the traces of a wireless network may be really useful in the characterization of the covered areas in terms of space usage.

Extended research has been done on many different aspects related to the usage of the wireless networks. One of the earliest works that explores the way people use the wireless network in an academic environment is described in (Tang & Baker, 2000). The authors focus on the characterization of the user behavior, that is, which access points in a building are used and how much, what type of applications people use and how much traffic they generate.

With the aim of improving wireless network mobility studies, (McNett & Voelker, 2005) report their research conducted in a University campus based on WiFi enabled mobile devices instead of laptops. They found out that the users with hand-held devices moved about two times more than the laptop users. Their main contribution is the creation of two new wireless network topology models to be used in simulations.

Another user mobility study is presented by Kim & Kotz (2007). Besides the characterization of the user behavior, the authors also characterize access points. They found out that the popularity of access points depends on the academic calendar and on the type of the building where the access points are located.

The authors of (Henderson *et al.*, 2008) revisit their previous work and extend it with more recent data and results. This allows them to identify changes that happened with the evolution and the expansion of their campus wireless network. Some of the changes they point out are the increase in the number of users and the access point utilization, the increase of roaming (the same user being detected in more than one access point) and the fact that the same buildings continued to be the most popular.

The above mentioned researchers and many others are mainly striving to provide as many as possible useful elements that might help to improve a wireless network performance, its management and planning and to create more realistic models for simulation and evaluation of routing protocols.

As WiFi networks spread over many public and private places, the potential for this technology to capture the dynamics of the space is enormous. As for us, we are interested in metrics that have been used for the characterization of the wireless network usage and user mobility, but our prime objective is to characterize space dynamics and to understand the usage patterns and the pulse of each space we observe. For instance, data about the number of users per access point or the session length, as in (Hutchins & Zegura, 2002) and (McNett & Voelker, 2005), can provide us with some information about the popularity of an access point and, consequently, about the popularity of a space where it is deployed. As in (Henderson *et al.*, 2008), we infer the way a space is used and whether it is, normally, one of the most popular areas in the campus, through the analysis of access point utilization rate. Daily, weekly and seasonal trends give us a picture of the space pulse, contributing to the description of a place.

Our aim is to understand how the data collected from the wireless network reflects what is happening in its coverage area while the above mentioned research is more focused on how the user behavior affects the network performance and how it can be improved. Although we are using the same type of data source, our objectives are significantly different as we are using this data to construct and update a world model.

Research conducted at the MIT (Sevtsuk & Ratti, 2008; Sevtsuk *et al.*, 2008; Vaccari *et al.*, 2009) has shown that data from WiFi network may be used as a good indicator of human presence and that network usage reflects to some extent the space usage. The authors of the iSPOTS project (Sevtsuk *et al.*, 2008) focus on the impact that the introduction and growth of the WiFi network has on the campus space usage. They, actually, observed a change of habits of the students concerning the choice of a place to study or socialize as a consequence of the implementation and expansion of the WiFi network. Consequently, it is much closer to our work than the previously mentioned papers. Although the dimension of our campus is much smaller, daily usage patterns are similar. However, our approach has different objectives, as we are using the wireless network as a proxy for space usage observations.

Two experiments are described in (Sevtsuk & Ratti, 2008), one based on the utilization of a WiFi network on a university campus and the other based on the utilization of a mobile phone network in a city, with the aim to find out to what extent the similarities in the network usage correspond to similarities in the space usage.

Finally, Vaccari *et al.* (2009) links together data about the human presence in spaces based on the WiFi network data and a system that monitors HVAC (Heating, Ventilation and Air Conditioning) levels in order to improve energy efficiency in buildings.

The aim of our research is to understand how the usage of a wireless network is related to what is happening in the WiFi coverage areas.For us, the WiFi network is just another sensor network which happens to be widely deployed and used and which may provide us with useful information about the environment represented in our model.

2.3 Chapter summary

The term "context" has largely been defined since the beginning of the research in the area of context-awareness and the development of context-aware applications. There is some consensus that the broader the definition of the context, the better. This follows from the fact that people infer context from so many different factors, some of which are still unreachable to the existing sensors and to the limited intelligence of the computing devices.

Context models have been used for collecting relevant context information about users. Location has been one of the most frequent parameters for the context determination. Different ways of representing location have been developed. World models or location models are largely used for this purpose. We have examined different types of location models, symbolic, geometric and hybrid. Among these, hybrid seem to be the most promising ones due to the combination of the advantages of both, symbolic and geometric models.

The area of WiFi network data analysis has played an important role in the development and implementation of this type of networks all over the world. Beyond this significant contribution to the overall standard improvement and widespread usage, it has been also shown that a significant set of context data can be retrieved from the WiFi network traces, especially in the areas where these networks are widely deployed, like University campuses and some other urban areas.

In the next chapter we introduce the fundamentals of our dynamic symbolic world model and the service that provides an interface to it.

Chapter 3

Dynamic symbolic space model

Science is organized knowledge.
Wisdom is organized life.
Immanuel Kant

In this chapter we define a dynamic symbolic space model that represents real time context information about spaces. Our inspiration for the space model here described was taken from the human mental models of space that are studied in psychology. Based on these and on symbolic world models from the literature, we created a world model that consists of objects and relations between them.

The space dynamics, as we understand it, corresponds to any changes that may occur in a physical space. Different factors may contribute to it, but it is closely related to people and their activities as well as to changes in the existing infrastructure. Many characteristics contribute to space description, but not necessarily to its dynamics. Temperature, light, physical dimensions, the official or the popular name, relative or absolute position are some of the examples. In addition, a place is uniquely characterized by the established schedules, by its purpose, by the available infrastructure, by the access rules and so forth. Some of these do not change very often, other change more frequently.

From the very beginning of our lives we continuously build upon our mental models of the world that surrounds us. These models include representations of objects, people and spaces of different types and dimensions, like home, family, school, friends, beach, car, city, airport, etc. Psychologists have been studying the mechanisms related to the creation of these mental models for many years. In our research, we are only interested in a very small fraction of these human studies which gives us the definition of the mental spaces as they are created in human minds.

Human mental models are defined as "mental constructions, consisting of elements and the spatial relations among them" by (Tversky, 2000). Further, the author explains that "the elements may be objects, people, landmarks, regions, cities". All of these and other elements in our model are called *objects*. The spatial relations between the elements include "near", "next to", "in front of". Regarding relations, all of the quoted examples are perfectly valid relations in our model. However, human mental constructions are much more complex than our model will ever be. Its scope is limited as it just implements a very small fraction of all the existing spaces in the world. Nevertheless, it is open to grow further and to undergo future improvements, some of which we will discuss in the final chapter of this document.

In order to implement a representation of physical environments and reflect their changes, the dynamic symbolic space model we propose is integrated in a larger system that includes sensor networks, data acquisition and processing modules and applications. The model is accessed through a component that provides a query and I/O interface to processing modules and applications. This component is called Symbolic Contextualizer (SC) and it will be described in detail in Section 3.2.

The proposed system reproduces different ways of acquisition of context information about spaces as it allows for creating new objects manually, by direct sensing and by inference based on already existing data.

The processing modules are responsible for three different tasks. One is making the acquired context data suitable for the direct insertion in the model. The other is the automatic updating of data in the model. The third task of processing modules is using the existing data in the model to infer new knowledge. For fulfilling these tasks, the processing modules rely on sensor networks deployed in the environment, data acquisition modules that retrieve the context data from them and the Symbolic Contextualizer which provides the access to the model.

Another feature of our model is that it is meant to be distributed over a federation of servers and to be created and maintained not only automatically by the above mentioned processing modules, but also manually by a collaborative group of users.

Next we introduce the basic structure on which our model is built. We define the four components of the model and describe their main characteristics. After that we talk about the local and global models and the historical data. Then we describe the Symbolic Contextualizer service that supports the model by providing an interface for the users and applications, on one hand and, the modules that retrieve and process the context data gathered by the sensors that are deployed in the environment, on the other hand.

3.1 World model fundamentals

In this section we introduce the main definitions of the four structural parts of our model. These are *objects*, *object attributes*, *relations* and *relation attributes*. There was a need to define separately object attributes and relation attributes due to their distinct conceptual characteristics. Object attributes provide rich descriptions of objects. They store values of parameters like temperature or the number of people in a room. The relation attributes add details to relations. These details are parameters characterizing the relation, but they may also play an important role in the inference algorithms which are based on relation processing (e.g., the walking distance between two objects may be used for pedestrian navigation).

Further, we discuss the global and the local aspect of the model. Each user is encouraged to create his/her own local models, to populate them with as many details as s/he wants and to make key objects public and, as such, visible to other users and so allow that relations may be created between objects in different local models. As a result of this interrelation between several local models, a global model containing all areas represented in local models will emerge, as explained in Section 3.1.2. At the end of this section we shall see that keeping a record of the past states of the objects in the model is an interesting option as it may allow for studying the way a space changes as the time goes by. It creates a kind of memory for the model that can be used by the applications for different purposes.

3.1.1 Definitions

Similar to human mental models of space, the model hereby proposed represents the physical environment as a set of interrelated objects. In this way it makes it easier for a human user to create his/her own model describing the spaces s/he knows as an image of his/her mental model.

As shown in Figure 3.1, our model is a graph-based symbolic model in which vertices are objects and edges are relations among them. Each object may be related to several other objects that may be represented in the same local model or in a different local model as it will be explained in Section 3.1.2. Each object may be further described by a set of attributes. Each relation, in turn, is characterized by a set of relation attributes.

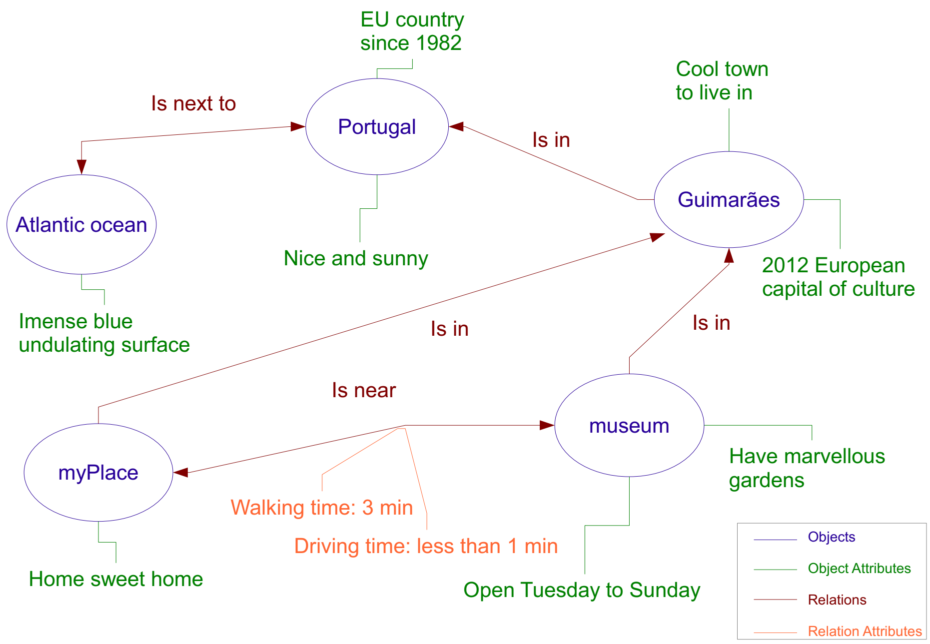


Figure 3.1: A snapshot of a symbolic world model

Definition 3.1 (Object). *An object is either a physical place (e.g., a room,* *a city square), a physical object (e.g., a table, an access point, a vehicle), an event (e.g., a conference) or an entity (e.g., an institution, a group of people).*

Each object is represented by an id, a name, a type, an author, a creation date, an availability (*public* or *private*) and a status (*active*, *inactive*). An object can have an arbitrary number of attributes and an arbitrary number of relations with other objects. Figure 3.2 shows a sample object description.

object = {objid, name, type, author, cdate, availability, status}

Figure 3.2: Example of an object

Objects are a part of a local model which is stored on a server. As such, every object can be uniquely identified through its URI (Uniform Resource Identifier) that consists of three elements: the object id, the host name or IP address of the server where the object is stored and the port from which it is accessible. The URI is not explicitly defined during the object creation. It is rather determined on the move by the Symbolic Contextualizer. The identification of objects through URIs allows that objects stored in separate local models may have a relation connecting them. We further explain this in Definition 3 when we introduce relations between objects and in Section 3.1.2 where we elaborate on global and local models.

A new object is created in the model manually by an authorized user or automatically by a processing module. In our implementation there is a web application, described in Section 4.2, available for the manual creation of objects. The user inserts the details about the object s/he wants to create and becomes the author of that object. The name an author gives to an object should be the name people usually use to refer to that object. In other words, the object names should be meaningful and easily recognized by the users. The same is true for the object types as they are not predefined. We explain this option in Section 3.1.4 and further discuss it in Section 4.2.

The availability of an object allows for distinguishing objects that are private from those that are public. Those objects in the model that have their availability set to *public* are visible to all users and may be related to any other objects, while those that are *private* are only visible to their author. For example, one user may create a detailed model for his/her school and make it *private* leaving only the object representing the school as a whole, *public*. By default, all new objects have their availability field set to *public*.

We assume that objects cannot be removed from the model but may have their status set to *inactive* if their attributes or relations are not updated for a certain period of time. By default, all new objects have their status field set to *active*.

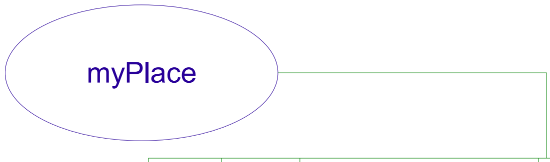
The management of the private and public domains of a local model as well as the management of the current status of each object are still open issues in our implementation. However, we will suggest possible solutions for these in the final chapter.

Definition 3.2 (Object attribute). *Object attributes are constant (e.g. the color of an object or the area of a room) or time varying characteristics (e.g. the temperature) of an object.*

An object attribute is defined by the attribute id, the id of the object which it describes, a time and a name-value pair. Attribute names should also be human readable just as object names. We further discuss this in Section 4.2. A sample set of attributes of an object is depicted in Figure 3.3.

The value of an attribute may also be a Uniform Resource Locator (URL) from which the instantaneous value is obtained when the corresponding object is retrieved from the model. An example is given in Figure 3.3. As stated in a recent article from the IEEE Spectrum magazine (Ackerman & Guizzo, 2011), more and more sensors are deployed in the environment and they are publishing their data on the web and, thus, creating the *web of things*. So as soon as adequate sensors are deployed in the environment represented in our models and their data made available on the Internet, we expect to be able to explore further this feature.

objAttribute = {attid, objid, time, name, value}



attid	objid	time	name	value
1	3	15:15:00 12/12/2006	description	Home sweet home
2	3	15:15:00 12/12/2006	Number of PCs	2
3	3	11:05:14 14/07/2008	Number of PCs	4
4	3	17:17:54 25/07/2011	temperature	http://someurl.xy

Figure 3.3: Example of object attributes. There are two attributes with the same name showing the history data for the object in question. The URL used in this example does not really exist.

There is an implemented mechanism that creates the historical data storage in the model. Each time a value of an attribute is newly calculated, a new record is inserted for it. As such, it is possible to retrieve the time series describing the evolution of an object's attribute value for further analysis of the object in question. We discuss this further in Section 3.1.3.

A new attribute, associated to an existing object may be created manually by a user or automatically by a processing module. It may, as well, be removed by the same actors. However, if the model is to store all the history data, the attributes are not removed when the new ones are inserted.

Definition 3.3 (Relation). *A relation is established between two objects and it represents their relationship in the real world.*

A relation is defined by a relation id, a name, a type, a creation date, the id of the object in the domain and the URI of the object in the co-domain. The object in the co-domain may belong to a different local model. This is why the complete URI is needed to reference it. A relation is further characterized by an arbitrary set of relation attributes. A relation can be created and removed manually by an authorized user or automatically by a processing module.

relation = {relid, name, type, cdate, objid, cdomURI}

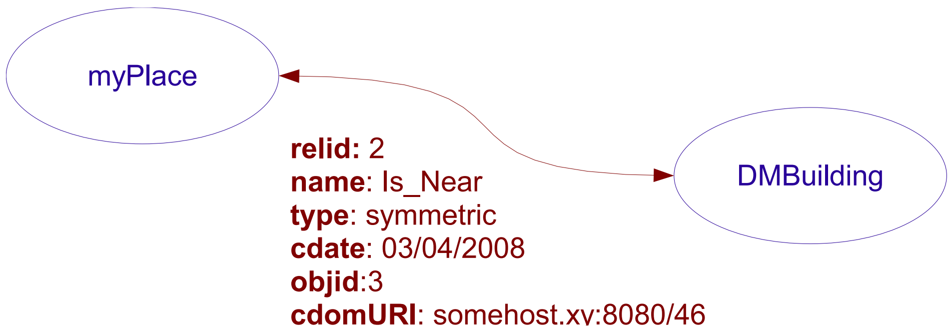


Figure 3.4: Example of a relation between two objects. The relation type is symmetric. It was created for the object called "myPlace" which is the domain object. The implemented inference algorithms allow that, if needed, this relation is also found for the "DMBuilding" object, the co-domain object.

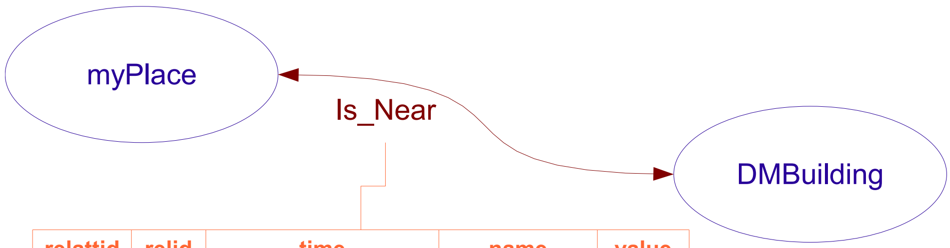
In our model, relation names have meanings and these are based on the way humans mentally relate objects or places. If there is a well known landmark, we will probably use it as a reference point for other objects, as in the following expression: *"the University is near the castle"*. An example of a relation between two objects is shown in Figure 3.4. The more relations one particular object (space) has with other objects, the better it is represented in the model.

Examples of the current relation names supported by our inference algorithms are: *Is_In*, which expresses spatial or administrative containment, *Is_Accessible_From*, which expresses accessibility or connectedness and *Is_Next_To* which expresses adjacency between two objects. Presently, the implemented inference algorithms support the relation types called *transitive* and *symmetric* (see Section 3.2.3). Other relation names and types are allowed but without any associated inference being performed.

Definition 3.4 (Relation attribute). *A relation attribute exists to characterize a relation. It is a constant or time varying characteristic of a relation.*

A relation attribute is defined by its own id, the id of the relation to which it belongs, a time and a name-value pair.

relAttribute = {relattid, relid, time, name, value}



relattid	relid	time	name	value
1	2	15:15:00 12/12/2006	Walking time	15 min
2	2	15:15:00 12/12/2006	Driving time	5 min

Figure 3.5: Example of relation attributes

In the example shown in Figure 3.5, a relation called *Is_Near* established between two objects has one attribute called *walking time* containing the average time a person may take to walk from one place to the other and another attribute called *driving time*.

A new relation attribute, associated to an existing relation may be created manually by the user or automatically when certain requirements are met. It may, as well, be removed in similar circumstances or when the relation to which it belongs is eliminated.

In the present implementation of our model, a relational database is used. Each one of the model's components is represented in a separate table. The fields of each table are those defined above for each element of the model. More details about the implementation of the model will be given in Section 4.1.

3.1.2 Global and local models

Every person experiences differently the surrounding reality and creates his/her own very original and personal mental model of it. This means that world models of the same environment created by different people are likely to be quite diverse.

This does not mean that these models will be contradictory or that one of them will be right and the other wrong. To the contrary, we believe that several models of the same reality will be complementary and will provide a richer and much more complete description of a place.

This is what we want to achieve with the so-called global model that is build upon a set of interrelated local models. By letting each user create and populate his/her own local model of space and share it with other people, more context information may be acquired for each space. When a local model is enriched with new data, its changes spread also to the global model, thus making it richer.

If we think of road maps or some other kind of maps provided by different manufacturers, they also exhibit differences. They may have different levels of detail and may not be updated simultaneously, meaning that, at a given time, they also provide different views of the same reality. However, these maps never happen to merge or become interconnected.

The idea behind the global model is to take advantage of all the differences that may exist in the local models, put them together and make them contribute to a richer description of objects.

Our space model is conceived as a distributed system in which there are several local or personal models. Although, these models may be located in many different and remote servers, the existing relations between their objects connect them and create a global model. The identification of the objects with URIs allows for distributing the model through a federation of servers. This enables an object stored on one server (one local model) to be related to another object stored on a different server (another local model). As each relation item has the id of the domain object and the URI of the co-domain object, the co-domain object can be easily retrieved by issuing a request (function f2_3) to the remote server as it will be explained in Section 3.2.2.

For creating a local model, users may choose to use remotely an existing Symbolic Contextualizer (SC) or install and maintain a personal or a corporate Symbolic Contextualizer, e.g., for privacy reasons. In any case, the local model can be linked to other public SCs to benefit from and to contribute to their models.

This approach creates an opportunity for the collaborative creation of a global model, stored in a federation of distributed models. Each user may create his/her own objects in the model and establish relations among them as well as with other people's objects residing elsewhere. In this way, a set of local models all together create a global model that comprises larger extensions of space due to relations that exist among the objects in different local models.

Similar approaches exist for geometric world models and the following are just three selected examples of web applications that allow for collaborative enrichment of maps. Wikimapia[1] invites registered users to draw contours of places they know over a map and to insert or update their descriptions. Google Maps[2] allows free visualization of maps and satellite images as well as creation of personalized markers, lines and polygons, over those maps. OpenStreetMap[3] provides maps of streets that were created based on the GPS traces shared by many different users.

The main difference between the above examples and our model is that in our case there is no centralized repository for the local models created by the users. Instead, there is a federation of servers storing the different models. In terms of privacy, this allows that each user decides which parts of the local model s/he wishes to share and which parts are to remain private.

In 2009, a new social networking concept emerged within a group of New York University students. They set out to a project that envisages the creation of a federation of servers hosting peoples's personal profile data and that allows for more control over sharing personal stuff over the Internet.

The project is called Diaspora[4] and is described in (Bleicher, 2011). The project's team claims that the existing social networking systems have serious privacy issues and defends that the creation of a federation of servers allows for a much better control over one's personal and shared data.

However, they face among others, one really difficult problem which has not yet found a solution. The problem is the way people search for each other's

[1]http://www.wikimapia.com
[2]http://maps.google.com
[3]http://www.openstreetmap.org
[4]https://joindiaspora.com/

profiles in a distributed system. How to find someone if you do not know which server hosts his/her profile? In a centralized system, it is quite straightforward as all the profiles are in the same database.

In our project, we raise the same question for a global query for objects: how can someone find shared objects in the global model? The only way we have to find an object hosted on a remote server is by its URI. This implies that we need to know its URI or at least the server's URL. But, if we are searching for objects in our distributed global world model, we cannot possibly know all the existing local models server's URLs. Well, we can if we maintain some kind of a list containing all the federated servers and perform a query in each one of them, but should not there be a more elegant solution? According to the above cited article, we are not alone in the quest for a solution to this problem, so, hopefully, it may be found in the near future.

3.1.3 Historical data

A space model with memory is closer to the human way of remembering spaces than the one that only provides instantaneous pictures of the environment. In terms of performance, it provides a much more reliable memory about past events as it keeps on recording all the events that occur in a space whereas people remember only a few of them. As pointed out by Baldauf *et al.* (2007), keeping track of the context history allows for recognizing utilization patterns or trends and predicting future values.

One possible solution for historical data storage is suggested by Lange *et al.* (2009). They propose that specialized history servers should be used for this purpose. In our implementation, all data is stored in the same database as the model itself. Every time a processing module calculates a new value for an attribute or adds a new object or a relation, a new record is created in the database that stores the model. If it shows to be likely that a local model grows rather quickly, it may be necessary to adopt the solution with specialized history servers or to think of a different one. A summary of the advantages and drawbacks of keeping the historical data in the model is provided in Table 3.1.

Model with history	Model without history
At any time, the dynamics of a space that led to a current state may be reconstructed.	The responsibility of keeping a record of the past events is referred to applications. The model continues to update in real time but the old data is overwritten which turns the model lightweight.
It is more similar to human mental model. It has the advantage over human mental models because it can keep track of all changes that occur whereas people remember only a few of those they witness.	It represents the instantaneous state of a space. If historical data is to be collected, a specialized application should be used for the purpose.
It provides data for space characterization and usage patterns analysis.	There is a need for a specialized application if space characterization and usage patterns are to be studied.
It is more adequate for supporting applications in mobile devices which need to remain lightweight.	The results of the processing and inference are wasted if there is no application storing them or analyzing them in real time.

Table 3.1: Advantages and disadvantages of keeping historical data in the model

Based on the data stored persistently in a database, an application may be able to reconstruct the whole dynamic evolution of the context of a particular space during any time interval in the past. This kind of results may provide a better understanding of the events that led to the present state of a space as well as the understanding of the usage pattern changes and their potential causes. In other words, history data provides information for the temporal analysis of the space dynamics.

3.1.4 Ontologies

An ontology is, usually hierarchical, organization of a knowledge domain containing all the relevant entities and the relationships between them. Extended research has been done on the ontology development in the field of context awareness. The authors of (Strang & Linnhoff-Popien, 2004) and (Reichle *et al.*, 2008), among others, defend that it is the most suitable way of creating context models. It may well be so due to its capability of establishing a common vocabulary for a heterogeneous pervasive computing environment.

Although we recognize the advantages of the pure or hybrid ontology approach, we do not follow it in our space model. We believe that letting the user define his/her own objects makes our implementation more similar to the way people create world models in their minds. Each person has his/her own way of remembering and modeling the places s/he visits. There may be some guiding suggestions based on the already existing objects in the model (as in our editing application, described in Section 4.2). However, the user remains free to define his/her own, especially when creating his/her private local models. The aim of using suggestions in our editing application during the creation of objects is to achieve a uniform nomenclature for object descriptions.

Although we do not use ontologies, one of the characteristics of our model is that it implements semantic relations. These relations have special properties that our system recognizes and processes by following the implemented inference algorithms as described in Section 3.2.3.

3.2 World model support

The technological support for our space model is built around a core component named Symbolic Contextualizer. The Symbolic Contextualizer provides a request-response mechanism based interface that allows processing modules and applications to access the data stored in the model.

As we have mentioned earlier, the world model may be distributed over a federation of servers and relations may exist between objects stored in different local models. Objects and their associated data are retrieved through the query interface provided by the SC that resides in each local server.

3.2.1 Internal architecture

The Symbolic Contextualizer is based on three modules as depicted in Figure 3.6. The first module, the I/O and Query Interface, provides external access to data stored in the model. It receives requests from processing modules or applications and hands them over to the second module, the Query Processor/Reasoner. In this module, the request is handed over to the Database Access module, inference algorithms are applied to the received data from the database and a response is generated and handed to the I/O and Query Interface. The third module, as its name suggests, handles the access to the database where the space model is stored.

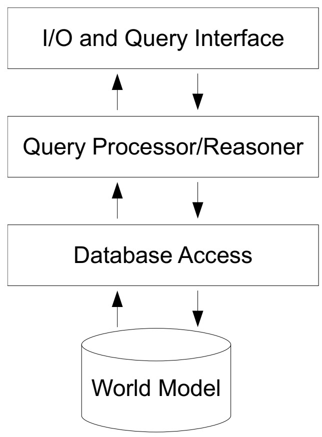


Figure 3.6: Symbolic Contextualizer internal architecture

3.2.1.1 I/O and Query Interface

The first module of the SC consists of two components, the I/O Interface and the Query Interface. The I/O interface handles insert and remove operations and the edition of the data in the model. It invokes appropriate methods from the Query Processor/Reasoner module in order to insert, remove or edit the descriptions of the symbolic objects. It replies with a XML description of the inserted or edited objects. The Query Interface accepts and validates requests for query operations on the data stored in the model. According to the received request, it calls the corresponding functions implemented in the Query Processor/Reasoner and replies with a received XML description of the found objects.

3.2.1.2 Query Processor/Reasoner

The Query Processor/Reasoner receives the requests from the I/O and Query Interface module and generates XML formatted responses. This module invokes appropriate methods from the Database Access module in order to insert, remove or obtain the descriptions of the symbolic objects needed for the response. Based on semantic rules established for each relation name and type, the Query Processor/Reasoner uses inference algorithms to decide which objects should be included in the response. All the inference algorithms are implemented in this module. It also implements the methods responsible for retrieving information from the URLs stored in the object attribute values.

3.2.1.3 Database Access module

The Database Access module abstracts the access to the relational database where the model is stored. It implements SQL statements allowing for querying and editing the model. A relational database provides an efficient way of storing and querying the model as well as its editing in runtime. More details about the database itself and its implementation will be given in Section 4.1.

3.2.2 Service functional specification

The current version of the SC supports a set of query functions and a set of data manipulation functions. The requests for these functions may be issued by an application or by a processing module. The supported functions are organized in two categories, query functions and editing functions. To issue a request the established syntax should be followed. The response consists of a XML description of the requested objects if they are found or an error message, otherwise. A more detailed description of the syntax and the examples for each one of the functions are provided in Appendix A.

Currently supported query functions are the following:

- **Function f2: Free query**. Given the name or the type of an object or the value of an attribute, the service returns a description of the matching object(s) if existent in the model. This function is particularly useful when a new model is about to be explored.

- **Function f2_1: Get an object by its name**. Given the name of an object, the service returns a description of the object(s), if they exist in the model. This query is useful when we know at least a part of the name of the object. This function is used in the path finding application described in Section 5.1 to identify the origin and the destination objects.

- **Function f2_2: Get an object by the value of one of its attributes**. Given the value of an object's attribute, the service returns the description of all the objects that have at least one attribute that matches the requested value. This function can be used for the purpose of self-location as labels may exist attached to some objects, like a door label, for example. It allows to get the description of the object in question, e.g., a room by searching for its attribute value.

- **Function f2_3: Get an object by its id**. Given the id of an object, the service returns the description of that object, if it exists in the model. This function is particularly useful because it allows to retrieve an object hosted on are remote server by its URI, as it happens for the co-domain object in a relation.

- **Function f2_4: Get an object by its type**. Given the type of an object, the service returns the description of all the objects, that match the requested type. This function allows to look for an object which name is not known, but we know what type of object it is, whether it is a room or a building, for instance.

- **Function f2_5: Get all the existing types of objects and their frequencies**. As a response to this request, the service returns a list of all existing types of objects and their frequencies. This function is used for the purpose of giving suggestions to the user during the manual insertion of new objects in the model (see Section 4.2).

- **Function f2_6: Get all objects having a given relation**. Given the name of a relation, the service returns the description of all the objects, that have the requested relation. This function is useful if we want to retrieve only one of the subgraphs representing our model. There may be more than one relation between each two objects. So if we query for the *Is_In* relation, we will have a subgraph containing a set of objects which are related among them with the *Is_In* relation. The number of vertices and edges in each subgraph vary according to the existing relations among objects. This function is presently used by the processing module described in Section 4.5.3.

- **Function f2_7: Get all objects having the same author**. Given the name of an author, the service returns the description of all the objects, that have been created by the requested author. This function allows for an author to browse his/her own local model (see Section 4.2).

Example 1: Query function request/response
Let us assume that the following HTTP request was issued:

```
http://host:port/sc/SCTX?function=f2_1&objname=my&infer=0
```

where `function = f2_1` defines the function f2_1 that queries the model by object name, `objname = my` defines a search parameter which can be partial or entire name of an object and `infer = 0` indicates the type of inference that should be applied for the relations this object owns.

If the requested object is found, then a XML document similar to the one in Listing 3.1 is returned.

Listing 3.1: XML object description as it is returned by the Symbolic Contextualizer.

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="39" type="house" name="myPlace">
<attribute-set>
<attribute name="Wall_Colour" value="White"/>
<attribute name="Door_Number" value="106"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```

In the given example, one object was found having a name that contains the word "my". The response consists of the full description of the found object. As `infer=0`, no inference was made and only the basic set of relations for this object is returned.

Currently supported editing functions are the following:

- **Function f3_1: Insert an object**. This function allows for creating new objects in the model. It receives a set of variables describing the object and inserts it in the model.

- **Function f3_2: Insert an object attribute**. This function allows for creating new attributes describing an existing object in the model. It receives a set of variables identifying an existing object and its new attribute and inserts it in the model.

- **Function f3_3: Insert a relation**. This function allows for creating new relations between two existing objects in the model. It receives a set of variables identifying the involved objects and describing the new relation and inserts it in the model.

- **Function f3_4: Insert a relation attribute**. This function allows for creating new relation attributes describing the existing relations in the model. It receives a set of variables describing the relation and its attribute and inserts it in the model.

Example 2: Insert object function request/response
Let us assume that the following HTTP request was issued:

```
http://host:port/sc/SCTX?function=f3_1&objname=MarysHouse&objtype=
Building
```

where `function = f3_1` defines the function f3_1 that inserts new objects in the model, `objname = MarysHouse` defines the new object name and `objtype= Building` describes the type of the object. All the other fields of the object description are filled in automatically.

If the insertion is successful, the SC replies with a full description of the newly created object like it is shown in Listing 3.2. Although, we refer to it as a "full" description, there are some fields from the object description that are not returned. This is the case for *cdate*, *author*, *availability* and *status* of an object. These fields are used for internal processing and are not returned by the implemented queries. The author field is used in our Space Editor application for showing his/her current model to an user. In conjunction with the availability field it will be used in the future developments to show only the authorized parts of a model (e.g., the author of a set of objects will be able to see all of his/her objects and only public objects from other authors). The status field is used to define if an object is active or inactive. If an object is set as inactive, it should not be returned by the queries. In the given example the description of the object has no attributes nor relations because the function f3_1 only creates the object itself. Functions f3_2 to f3_4 must be used in order to add more information about the newly created object.

Listing 3.2: XML object description as it is returned by the Symbolic Contextualizer after a new object insertion.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="397" type="Building" name="MarysHouse">
<attribute-set>
</attribute-set>
<relation-set>
</relation-set>
</object>
</context>
```

3.2.3 Inference algorithms

The currently implemented inference algorithms in our system are based on relation types. Their objective is to make the model browsing easier and to infer data that is not explicitly represented in the model. Two algorithms were implemented so far, one for the transitive relations and the other for the symmetric relations.

In order to run an inference algorithm, an additional parameter (`infer`) and its value must be included in each request made to the SC. Presently, there are four possible values (0, 1, 2 and 3) for the `infer` parameter. Their meaning is the following:

- `infer=0` – no inference (see Listing 3.3 for an example);
- `infer=1` – transitive only;
- `infer=2` – symmetric only;
- `infer=3` – both, transitive and symmetric.

Listing 3.3: XML object description with the `infer` parameter set to 0 – no inference. Only the basic set of relations defined for the requested object is returned.

```xml
<?xml version='1.0' encoding='iso-8859-1' standalone='no'?>
<context>
<object objid="27" type="green surface" name="CentralPark">
<attribute-set>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="14" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Next_To" objid="26" objhost="cityhall.cti" objport="8080"
objtype="Building" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="position" value="South"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="26" objhost="cityhall.cti"
objport="8080" objtype="square" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="walking distance" value="10 min"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="28" objhost="myhost.me"
objport="8080" objtype="building" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="walking distance" value="20 min"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="17" objhost="yourhost.you"
objport="8080" objtype="restaurant" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="driving distance" value="5 min"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```

An example of a transitive relation is the *Is_In* relation which models the topological relation *is contained in*. This relation may be used to express both, spatial and administrative containment, thus enabling range queries as required for a world model in (Becker & Dürr, 2005).

The algorithm that handles transitive relations was implemented in the scope of the LOCAL[1] project. When the requested object is found, the algorithm retrieves the complete set of the object's relations and extracts all those that are transitive, based on their type field. Then, it searches for all the related objects based on their id (URI, in the future). The list of all the found co-domain objects is visited sequentially while there are elements in it. Normally, this list of related objects finishes at the root object called *world*. This object exists in all local models as a default root object. In this way, for a given object, the whole hierarchy of related objects is returned when a query is performed with the inference parameter equal to 1 as shown in Listing 3.4. However, if there are missing relations and the root object cannot be reached, the list of found objects may be shorter.

The existence of the inference algorithm that handles symmetric relations avoids the duplication of relations between objects when these relations are valid in both directions. In our model, there are several examples of symmetric relations. Relations *Is_Next_To* and *Is_Near* may be used for the nearest neighbor queries, mentioned in (Becker & Dürr, 2005), as they express adjacency and proximity of the related objects, respectively. Additionally, and when necessary, they may have relation attributes containing the distance between the objects. These two relations are always symmetric.

The *Is_Accessible_From* relation models the topological relation *is connected to*. This relation may be used to express the existing paths and accessibilities between objects thus enabling navigation as required in (Becker & Dürr, 2005). This is an example of a relation that may be either, symmetric if a bidirectional path is available, or simple if there is only one way path between two objects. The application for pedestrian navigation, described in Section 5.1, exploits this

[1] http://ubicomp.algoritmi.uminho.pt/local/

relation.

Listing 3.4: XML object description with the `infer` parameter set to 1 – inference of transitive relations. Note that we omitted all the non-transitive relations as they appear in the same way as in the previous listing.

```
<?xml version='1.0' encoding='iso-8859-1' standalone='no'?>
<context>
<object objid="27" type="green surface" name="CentralPark">
<attribute-set>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="14" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_In" objid="2" objhost="cityhall.cti" objport="8080"
objtype="District" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_In" objid="44" objhost="cityhall.cti" objport="8080"
objtype="Country" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_In" objid="45" objhost="cityhall.cti" objport="8080"
objtype="Continent" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_In" objid="1" objhost="cityhall.cti" objport="8080"
objtype="world" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
...
</relation-set>
</object>
</context>
```

When a request for an object is made to the SC with the `infer` parameter equal to 2, the inference algorithm that handles symmetric relations is run. When searching for the relations of the requested object, this algorithm searches for the requested object id, both in the domain and in co-domain fields of a relation. It creates a list of related objects which are then returned. From the reply, one cannot distinguish the relations that belong to the requested object from those that were inferred. In order to see the difference, a request with the `infer` parameter equal to 0 may be issued and then compared to the one with the parameter equal to 2. Listing 3.5 only shows the inferred symmetric relations as the basic relation set is the same as in Listing 3.3.

Listing 3.5: XML object description with the `infer` parameter set to 2 – inference of symmetric relations. Note that the basic set of relations was omitted. Only the inferred symmetric relations are shown.

```
<?xml version='1.0' encoding='iso-8859-1' standalone='no'?>
<context>
<object objid="27" type="corridor" name="CentralArea">
<attribute-set>
</attribute-set>
<relation-set>
...
<relation relname="Is_Next_To" objid="28" objhost="herhost.she" objport="8080"
objtype="corridor" reltype="symmetric">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="22" objhost="herhost.she"
objport="8080" objtype="herPlace" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="walking distance" value="1 min"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="25" objhost="myRest.mr"
objport="8080" objtype="restaurant" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="walking distance" value="20 min"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
...
<relation relname="Is_Accessible_From" objid="31" objhost="cityhall.cti"
objport="8080" objtype="street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="type of accessibility" value="pedestrian only"
```

```
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="32" objhost="cityhall.cti"
objport="8080" objtype="street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="type of accessibility" value="pedestrian only"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="33" objhost="cityhall.cti"
objport="8080" objtype="street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="type of accessibility" value="pedestrian only"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="34" objhost="cityhall.cti"
objport="8080" objtype="street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="type of accessibility" value="vehicles and
pedestrian" time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
...
</relation-set>
</object>
</context>
```


When a request is issued with the `infer` parameter equal to 3, the resulting object description includes the complete set of its relations, both basic and inferred by the two existing inference algorithms. This results in a far more detailed description of an object as the more relations it has with other objects, the better.

3.3 Chapter summary

Symbolic world models are very close to the human mental models of space. They both consist of elements and relations among them. In our model we further describe the elements and the relations of the model by introducing their attributes. Object attributes essentially contain more details about the object to which they belong, while the relation attributes go beyond that. The relation attributes may be used by the inference algorithms. For instance, they can contain data about the distance between two objects and this information can be used in the shortest path calculation.

In our model, an object may represent a place, a physical object, an event or an entity. However, we are most interested in studying places and their changes over time. The changes of the characteristics of an object are also stored in the model as the time stamp in which a determined attribute value was observed is recorded. This provides a history data record that may be used for further analysis of the place dynamics over time.

The way the relations are implemented in our model allow for the interconnection of several local models into one global model. Each relation contains the id of the domain object and the URI of the co-domain object. This allows that the co-domain object may be located on a remote server and still be accessible to the local model. Each local model is created and maintained by one or more users and may be shared with other users. It is possible for each owner of a model to define which objects are private and which are public. In this way, only the public objects are included in the global model and made available to other users. The basis for this concept has been established as each object has in its definition the availability parameter and the relations contain the co-domain object URI. However, it is not yet possible to make a query in more than one local model at a time.

The interface to our model is provided by a service called Symbolic Contextualizer. It provides a set of querying functions and a set of editing functions. For each received request, it replies with a XML description of the found objects. It also implements two inference algorithms that allow for extracting information that is not explicitly contained in the model.

The proposed model is not standalone. It is a part of a system that consists of sensor networks, data acquisition and processing modules, world model services and applications. The system architecture is the topic of the next chapter.

Chapter 4

System architecture

Faith is taking the first step even when
you don't see the whole staircase.
Martin Luther King, Jr.

The world model described in the previous chapter is part of a larger system consisting of sensor networks, data acquisition and processing modules and applications. It acts as a repository of the context information about places, things and people which can be used by context-aware applications. Besides the manual edition performed by a user, the world model can also be updated automatically by a set of processing modules that handle context data received from sensor networks or inferred from the already existing data in the model.

The model can be fed by several sources of context data, provided that there are dedicated acquisition and processing modules to handle sensed data and to format them to fit the model interface requirements.

The system we propose is depicted in Figure 4.1 where all the components of the system are shown as well as the examples of processing modules which have been defined so far. This structure fits the five layered conceptual framework mentioned in (Baldauf *et al.*, 2007) and originally proposed by Ailisto *et al.* (2002).

The Ailisto et al. framework consists of a Physical Layer, which contains physical sensors deployed in the environment and/or virtual sensors, a Data Layer, which contains mechanisms for raw data retrieval and storing, a Semantic

Layer, which contains modules responsible for transforming the raw data into more meaningful data, an Inference Level, which uses the preprocessed data to infer new knowledge and an Application Layer, which receives this knowledge and uses it.

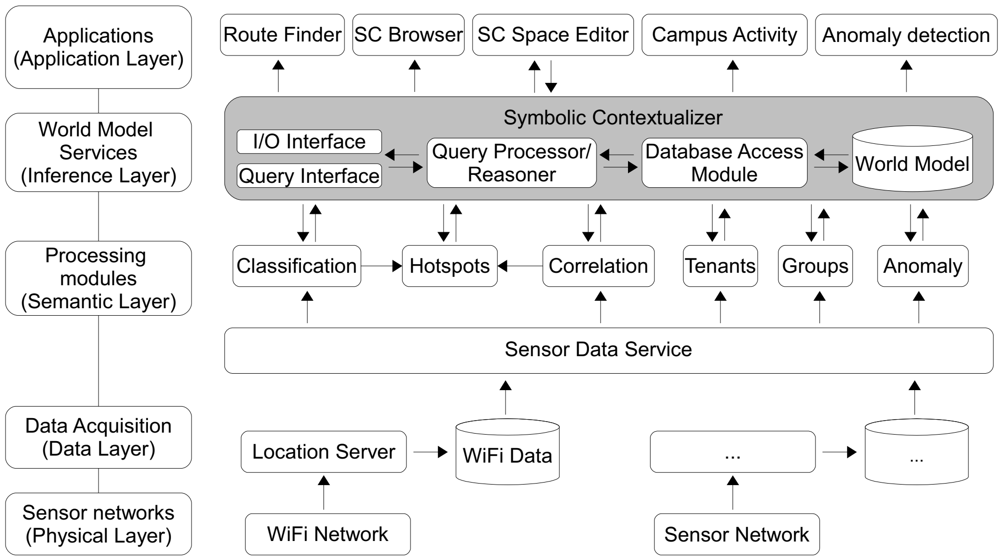


Figure 4.1: General system architecture fits the five layered framework proposed by Ailisto *et al.* (2002).

In our framework, the first layer also consists of sensor networks. Sensor networks can be physical or virtual. Physical sensor networks are deployed in the environment and include room temperature sensors, presence sensors, WiFi network, GSM network, Bluetooth devices and many more. Virtual sensors are data sources from social networks which can also be used as sources of context information about spaces based on the information people share in their profiles, for instance, during an event that is taking place somewhere, like a football match, a concert or an exhibition.

The second layer contains raw data acquisition modules (Location Server and Sensor Data Service). Each type of sensor network needs an adequate data acquisition module which is responsible for gathering and storing the sensed data for further processing. This component is also responsible for the storage of all the acquired data from sensors, thus maintaining a history of raw context data for future usage which was highlighted as an important feature in (Dey *et al.*, 2001). Keeping a history of data allows for studying trends and predicting future events. Besides this raw data history, our system also keeps the history of more meaningful context data in the fourth level, as we explained in Section 3.1.3.

The third layer consists of the processing modules. In this layer, data requested to the acquisition module are processed in order to fit the requirements for insertion in the world model. These modules are responsible for updating the model periodically with the results of the most recently processed data.

The fourth layer contains the central component of our system, which is the world model service acting as a repository of the rich context information about spaces. It also implements inference algorithms which allow for extracting non-explicit data from the model itself.

Finally, the fifth layer consists of applications that consume and combine the information stored in the model in order to fulfil user requests or needs.

Details about each level of our conceptual architecture are given in the next sections, starting by the central component which implements the model itself and its interface to applications and processing modules.

4.1 The world model services

The Symbolic Contextualizer is the core component of our system. It provides a HTTP request-response mechanism based interface that allows processing modules and applications to access data stored in the model.

The implementation of the Symbolic Contextualizer (version 0.9) described in this chapter was based on its previous implementation (version 0.5) developed in the scope of the LOCAL project as reported in (Moreira *et al.*, 2006).

As described in the previous chapter, the SC consists of three components: the I/O and Query Interface, the Query Processor/Reasoner and the Database Access module. The Symbolic Contextualizer resides on a web server running Apache Tomcat. The I/O and Query Interface was developed as a servlet using

Java technology. The service replies in XML format which is a W3C standard. All the other SC components are Java classes. The Database Access module uses SQL queries to access data from the model which is stored in a relational database.

In the scope of the LOCAL project, two different implementations were considered for the SC. One option was to represent the symbolic model in OWL format and the JENA framework. The other option was the relational database. The two approaches were tested and the decision was made in favor of the relational database representation. The main reasons for choosing a relational database were the possibility of editing the model in runtime and the scalability, as explained in (Moreira *et al.*, 2006). The latter is closely related to the model storage. While the OWL/JENA model is stored in memory, the database model is stored on the hard drive. The first solution performs faster for small models, but very large models may eventually be a problem.

As we foresee a fast growth of the model, a relational database continues to be our option in this project. Moreover, the runtime editing is becoming more and more important as the number of processing modules feeding the model is increasing. The present versions of our local models use MySQL databases, but other options are possible as the JDBC connection is used.

The database entity relationship diagram is depicted in Figure 4.2 and the details about the implementation of all the components of the model, objects, relations, object attributes and relation attributes are shown in Table 4.1.

Table Name	Attribute Name	Type	Description
OBJECTS	obj_id	integer, not null	Primary key. Symbolic object identifier.
	obj_name	varchar(100)	Symbolic object name.
	obj_type	varchar(30)	Symbolic object type.
	obj_author	varchar(30)	Symbolic object creator user name.
	obj_cdate	date/time	Symbolic object creation date and time.
	availability	varchar(10)	Indicates if the symbolic object is private or public.
	status	varchar(10)	Indicates whether the symbolic object is active or inactive.
RELATIONS	rel_id	integer, not null	Primary key. Relation identifier.
	rel_name	varchar(100)	Relation name.
	rel_cdate	dateTime	Relation creation date and time.
	rel_type	varchar(30)	Type of relation (symmetric, simple, transitive, etc.).
	obj_dom	integer	Foreign key. Identifies the symbolic object from OBJECTS table which is in domain of the relation.
	obj_cdom	integer	Identifies the symbolic object which is in the co-domain of the relation.
	host_cdom	varchar(30)	IP address where the co-domain symbolic object is stored.
	port_cdom	varchar(5)	Port number to access the co-domain symbolic object.
OBJATTRIBUTES	att_id	integer, not null	Primary key. Object attribute identifier.
	obj_id	integer	Foreign key. Identifies the symbolic object from OBJECTS table which is described by the attribute.
	att_time	date/time	Object attribute creation date and time.
	att_name	varchar(100)	Object attribute name.
	att_value	varchar(200)	Object attribute value. Allows URLs.
RELATTRIBUTES	rel_att_id	integer, not null	Primary key. Relation attribute identifier.
	rel_id	integer	Foreign key. Identifies the relation from RELATIONS table which is described by the attribute.
	rel_att_time	date/time	Relation attribute creation date and time.
	rel_att_name	varchar(100)	Relation attribute name.
	rel_att_value	varchar(30)	Relation attribute value.

Table 4.1: Database tables description.

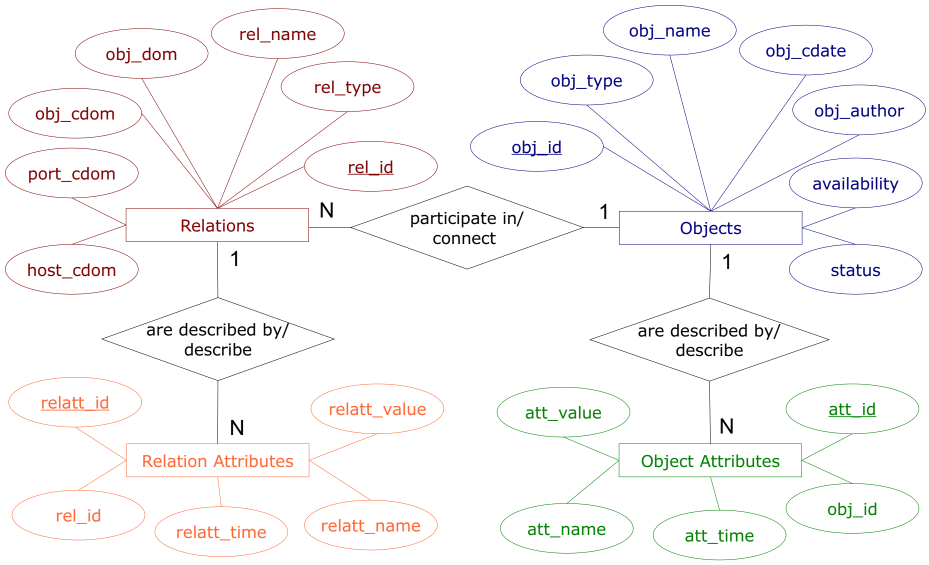


Figure 4.2: Database entity relationship diagram.

4.2 Editing the model manually

When a user wants to create a new local model, s/he needs to access an editing application. The Symbolic Contextualizer Space Editor (version 0.3) is the one we developed. It is a web application that allows the user to create and describe his/her objects and the relations between them and to visualize and browse the created model. After logging in, the user is invited to chose whether s/he wants to insert a new object or visualize the already existing objects in his/her model as depicted in Figure 4.3.

The first option guides the user through three simple steps: 1. Create an object; 2. Add attributes to the created object; 3. Relate the created object with other objects. For a new object to be created, a name and a type are the minimum requirements. All other fields (id, creation date, author, availability and status) are completed automatically in the current version. In the future, the user will be able to define the availability of the objects s/he creates in order to separate the private part of his/her personal model from the public one.

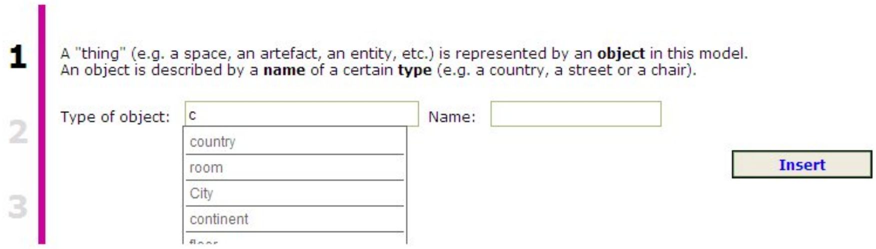
Names given to objects, especially in shared models, should be human read-

Figure 4.3: Symbolic Contextualizer Space Editor allows a user to insert a new object to his/her model or to visualize the existing ones with their descriptions and relations.

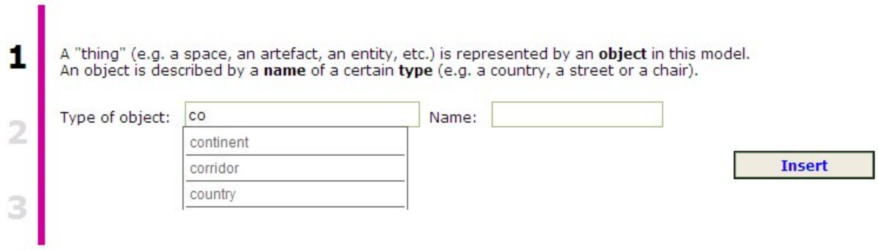
able and should correspond to their commonly used names in order to make the model browsing easier and the self-location possible. The self-location is based on recognizing nearby landmarks and features in the environment and is used in our application for pedestrian navigation, described in Section 5.1. For example, an object named ZXCV would not be very useful for someone searching for a path, except if it had a really good description in the type field and in terms of attributes and relations.

When the user starts creating an object, suggestions are given for the object type according to the frequency that they appear in the model. After inserting the first letter in the object type form field, the list of suggested types of objects shows the most frequent object type first, as illustrated in Figure 4.4(a). This functionality is supported by the SC query function f2_5 (see Section 3.2.2 and

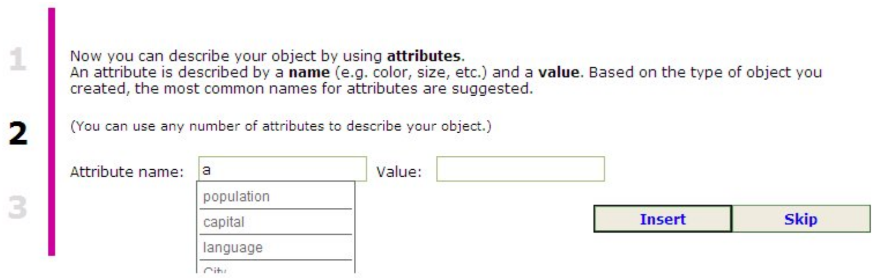
Appendix A). In turn, after inserting the second letter, the list of the existing matching types is ordered alphabetically as shown in Figure 4.4(b). However, the user is not forced to select one of the suggested types. S/he may insert a new one.



(a) The suggested object types are those that most frequently appear in the model.



(b) After inserting the second letter, the list of matching object types is ordered alphabetically.



(c) A list of suggested attributes is created based on their frequency for the given type of object. In the example the chosen object type is *country*.

Figure 4.4: SC Space Editor suggestions for types of objects are based on the already existing elements in the model.

Attributes and relations may be added to better describe the created object and to help the user to easily recognize it in the real environment. Attribute names are not predefined, but should also be human readable. Suggestions for the attribute names in our current implementation are given according to the type of object that is being created as illustrated in Figure 4.4(c). As such, the list of suggested attributes contains all the existing attributes belonging to the objects of the same type. By suggesting names for the attributes, or for the type of an object, users are influenced, while creating a model, to create progressively an informal ontology.

In the present implementation, relation names are restricted to a predefined list, but in the future, the user should also be able to insert new relation names.

As the new object is being created, a textual description of it is shown on the upper part of the screen. When the user finishes the creation of a new object, s/he is shown the final description of the inserted object and prompted to make any changes. After choosing a different option, the edition of the objects is no longer available. In the future versions of this editing tool the user will be given more control over his/her objects and an easier way of managing the model's elements.

The user can visualize all the objects contained in his/her model by choosing the second option of the main menu: "My Space Model". This allows user to browse through his/her objects and to examine the existing attributes and relations as illustrated in Figure 4.5. This feature relies on a tool that was developed during the course of our project and which is called Symbolic Contextualizer Browser.

The SC Browser, as its name indicates, allows for browsing a symbolic space model starting with a name of one of the existing objects and following the links to other related objects. In the textual version of the tool the objects are shown in a list. Their descriptions and the related objects are obtained by clicking on the provided links. In the graphical version of the application, which is in development, the model is shown as a graph with the objects as the vertices and the relations between objects as the edges. Further descriptions of the objects and relations are also obtained by clicking on them.

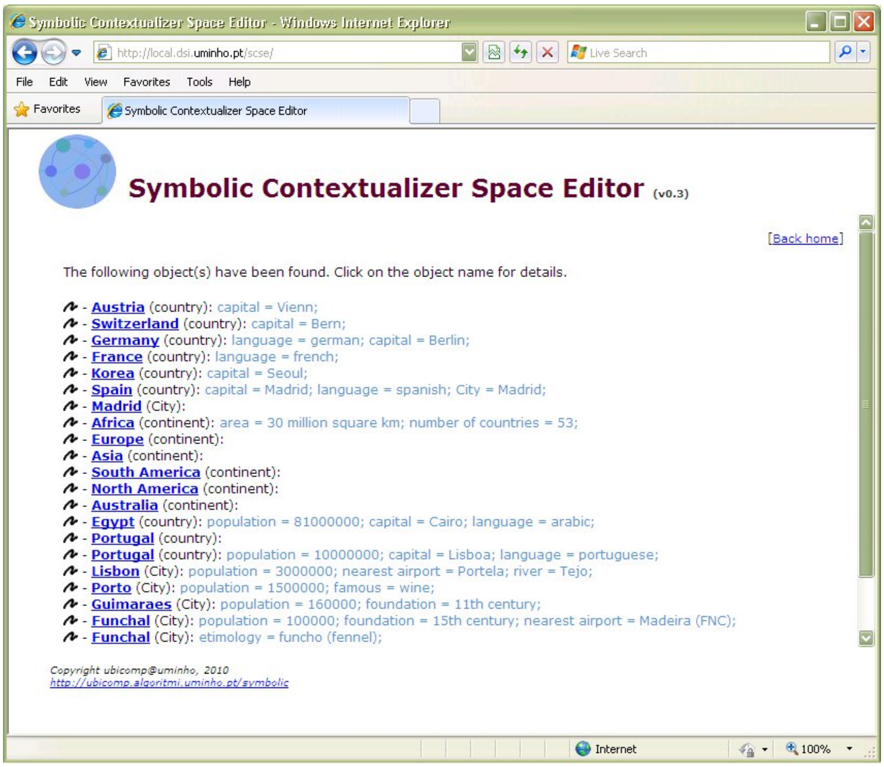


Figure 4.5: The user can visualize all the objects contained in his/her model, their descriptions and their relations.

A solution for an indoor navigation system based on hybrid geometric/symbolic space model and indoor positioning systems is proposed by (Stahl & Haupert, 2006). However, this solution requires the creation of geometric models for the buildings, which demands some level of technical expertise, access to CAD drawings, and a considerable amount of time (”The modeling of the rooms in the ground floor took a few hours.”). Our experience, however, showed that very little time is needed to create a local model for a building or a part of a building with the proposed model editor. According to our user study results (see Section 6.2.1 and Appendix B), someone that is using the application for the first time takes around 3 minutes to create an object, its attributes and relations. With a hand-held device, a more experienced user may walk through the building and create his/her local model in a very short time. No previous or specialized knowledge is needed as the objects are given their common names and the relations are established in a way which is very close to the one people use when creating their mental models.

Still, our current implementation lacks a simple way of managing the already existing objects, their attributes and relations. This limitation forces the user to create the model in a very organized manner, with the especial attention to the relations that will be created. For example, to create a room, and to relate it to a floor, the latter needs to be created first. In turn, to relate the floor to a building, the building must exist already. So, presently, the user needs to have in mind the whole structure of the model s/he wants to create which is not very practical. Also, the insertion of relation attributes has not yet been implemented. These are some of the issues that are discussed in Section 6.2.1 and that, hopefully, will be solved in the future versions of the SC Space Editor.

4.3 Sensor networks

Sensor networks are becoming more and more widespread in our world. They allow for gathering the context information from the environment which can then be processed further or stored for future analyses. Besides the physical sensors which are deployed in the monitored environments, there are also virtual sensors which rely on the data shared on the move by the users of social networking applications. These can also provide interesting insights about places and their dynamics as, for example, in the MIT project Real Time Rome[1].

We chose to use the wireless network as our primary source of context data because it is the most widespread network in the studied area and because other projects showed that it was quite a rich source of data about the way spaces are used, e.g., (Sevtsuk *et al.*, 2008).

Our analysis of the WiFi network as a proxy to space usage was conducted aiming to use it as a means for the characterization of physical spaces and, consequently, as a source of information for a dynamic symbolic model representing those spaces. Other data sources may be considered in the future developments

[1] http://senseable.mit.edu/realtimerome/

of this project.

When we started our research, there were 233 active access points spread over the two University of Minho campuses. In 2009, this number reached 411 access points and continues to grow as the network is being expanded according to the needs of the academic community. The collection of data from the considered wireless network was initiated prior to this project. It has been in course since 2005.

On-site observations were conducted in several areas of the Azurém campus in order to assess to what extent the WiFi network utilization captures the dynamics of those spaces. The obtained results were then compared to those captured by the WiFi network data acquisition system.

The selected locations were: the library and one studying room nearby, the main bar, two corridors with classrooms and the center for the computing support to students. Observations were made during 30 minute periods in each location by annotating the approximate number of people present, the number of open laptops and the number of people who enter and leave the place. One of the indicators used for the accuracy of the WiFi observations was the fraction of people using laptops in each space. We assumed that once a laptop is being used, its wireless connection was active.

We found out that in places like the library, the studying rooms and the laboratories, data collected from the WiFi network reflects quite well the usage of those spaces as nearly 90% of people use the WiFi network. However, in a place like the cafeteria, where the space dynamics is very high, and where people are constantly coming and going, especially at coffee-breaks, very few people use a laptop (only about 15%, according to our observations). In this case other types of context information (the number of Bluetooth devices or the number of served coffees, for example) would be helpful to enrich the knowledge about what is happening in these areas. The significant increase in WiFi enabled mobile phones may influence significantly these figures and make them change in the near future.

4.4 Data acquisition modules

WiFi data acquisition is performed by a server which owns permissions to access the wireless network infrastructure. The acquisition of data from the WiFi network is performed at a rate of one sample every five minutes.

The Location Server was developed in the scope of the LOCAL project. Its role is to query the WiFi network for data about the usage of the access points and to store this raw data in a database for future needs and also for the Sensor Data Service to use them.

Presently, the acquisition modules are stored in two servers, one for each campus wireless network as shown in Figure 4.6.

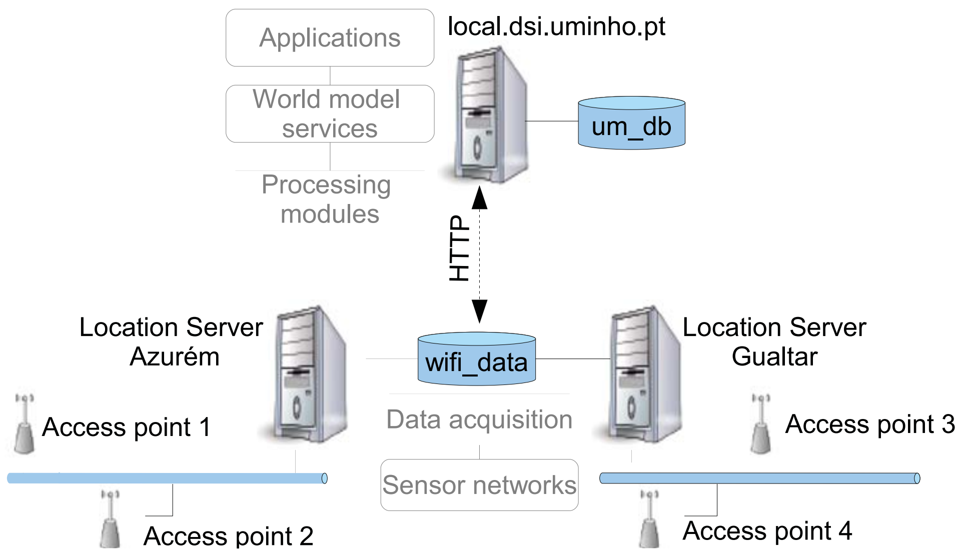


Figure 4.6: System architecture

Sensor Data Service (SenDS, version 0.1) is a Java servlet that was developed to act as a mediator between the Location Server and the processing modules. It receives requests for a specified amount of data ranging between two given time stamps and it delivers this data to the processing modules. The requested data is retrieved from a database which is populated by the Location Servers and which contains all raw data from the access points belonging to the WiFi network (wifi_data in Figure 4.6).

The requests for data made by the processing modules to the SenDS have the following format:

```
http://host:port/sends/SenDS?function=f1&fromdate=yyyy-MM-dd HH:mm:ss
&todate=yyyy-MM-dd HH:mm:ss
```

where `fromdate` contains the initial date of the data set and `todate` contains the final date. Like this, each processing module can make requests for data sets of variable lengths.

4.5 Processing modules

Figure 4.1 shows the modules that have been envisaged for the processing of the WiFi network data in order to feed the model. The analysis of the WiFi data containing the number of users connected to each access point, and the lists of MAC addresses that were observed, showed us that useful information may be extracted from there and fed into the model. In the following sections we propose five examples of processing modules, three of which rely on the number of users data and two rely on the lists of MAC addresses.

The role of data processing modules is that of making the sensor networks data adequate for insertion into the model. As well, these modules share with the users the responsibility for maintaining the model up to date. Each processing module performs specific operations based on present or past datasets of a variable length. The results produced by the processing modules are injected into the world model through its I/O interface in real time. Like this, the model is kept up to date based on the sensed data from the covered areas and the context-aware applications may use it as a source of context information.

4.5.1 Classification of access points

The first processing module that we are going to describe is the one that performs the automatic classification of the access points based on two parameters that characterize their usage patterns, the utilization rate, which we define next, and the average number of users.

The module performing the classification of access points implements the automatic update of object attributes in the model. Presently, this module only updates attributes of those access points that are already represented in the model. This means all the objects with their type defined as *access point* or containing these words. In the future, it may also become enabled to create and insert automatically new objects in the model representing the most recently added access points in the wireless network.

We would like to clearly distinguish the terms *utilization rate* and *utilization level*, which will be used a lot in the following sections. The former is defined next and the latter corresponds simply to the number of users that were connected to an access point at some time interval.

Definition 4.1 (Utilization Rate). *The utilization rate is the percentage of the observed time intervals for an access point in which its utilization level is greater than zero.* For instance, if an access point data is collected during 288 time intervals and there are 176 time intervals in which the utilization level is greater than zero, then the utilization rate is $176/288 = 0.61$, this is, 61%. Table 4.2 shows the variation of the average utilization rate between 2007 and 2010 and the percentage of access points belonging to different intervals of the utilization rate values.

Utilization rate	Percentage of access points		
	2007	2009	2010
More than 50%	13%	22%	22%
Between 40% and 50%	4%	7%	6%
Between 30% and 40%	9%	17%	12%
Less than 30%	74%	54%	60%
Average utilization rate	25%	32%	31%

Table 4.2: WiFi network utilization rate comparison between 2007 and 2010.

An initial analysis of the utilization level variation in 23 week long time series of around 230 access points resulted in the identification of four different usage patterns. At first, this classification was performed manually, mainly based on the visual observation and comparison of curve shapes. Although the analyzed data is discrete, continuous graphics were used for an easier perception of the data evolution over time. Sample plots obtained for a set of access points are shown in Figure 4.7.

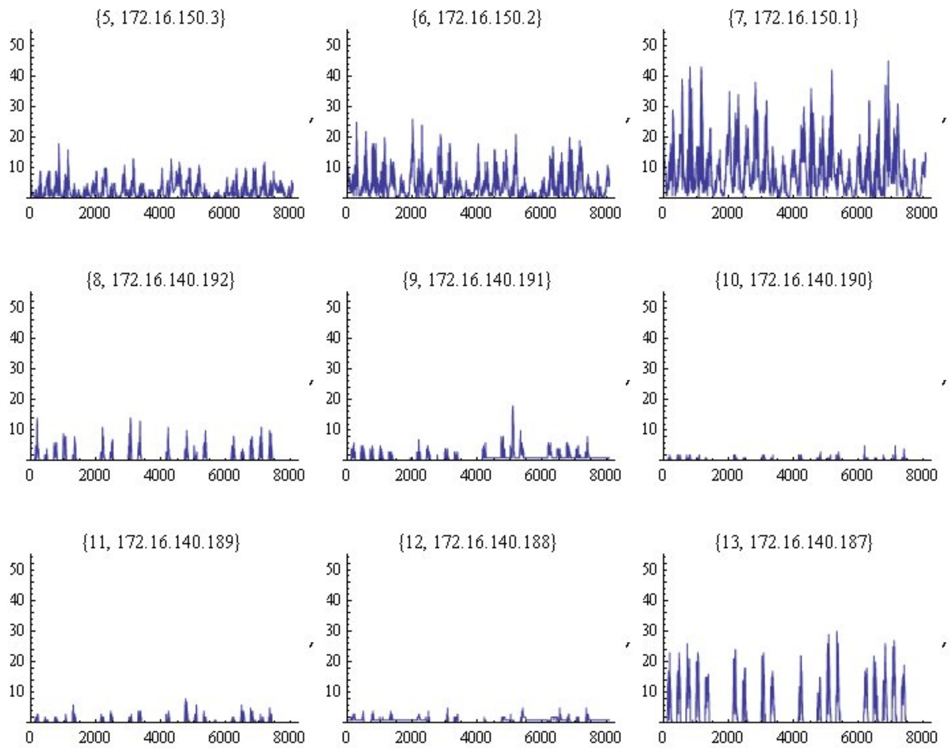
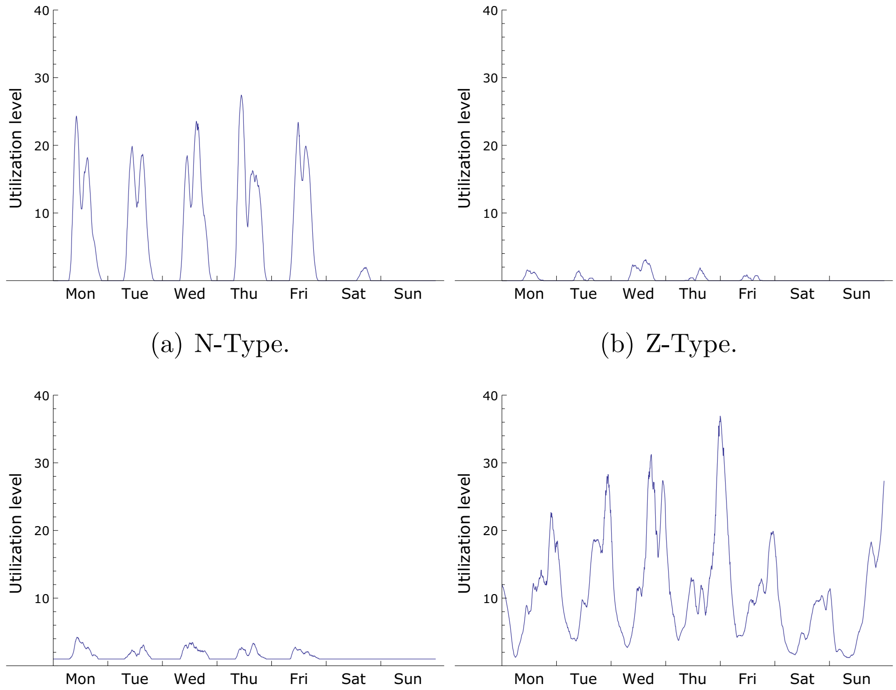


Figure 4.7: Access points data used for manual classification. The classes associated to this set of data time series were (from the top left): R-Type, R-Type, R-Type, N-Type, N-Type, Z-Type, Z-Type, C-Type, N-Type.

The patterns shown in Figure 4.8 were obtained after the smoothing of the curve representing one week data of one access point for each class. The smoothing was performed with a moving average computed by averaging runs of 24 samples which correspond to two hours (there is one sample for every five minute interval).

N-Type (normal type) is one of the most common usage patterns, corresponding to 40% of all observed access points, altough the range of their utilization level



(a) N-Type.

(b) Z-Type.

(c) C-Type.

(d) R-Type.

Figure 4.8: Four access point usage patterns.

may vary considerably. Two high peaks during working days are identified and sometimes another smaller peak in the evening. There may also exist a small peak on Saturday morning.

Z-Type (zero type) is the most common usage pattern as around 50% of all access points in the studied WiFi network seem to be used very rarely and by very few users. This fact is also confirmed by the data shown in Table 4.3.

C-Type (constant type) usage pattern is characterized by an utilization rate higher than 70% and a fairly low average number of users. It is usually found in access points belonging to computation centers or laboratories.

R-Type (residence type) usage pattern is characterized by a very high utilization rate during the whole week. It exhibits several peaks during the day but the highest ones appear in the evening or during the night.

Table 4.3 shows how the percentage of access points belonging to each class varied between 2007 and 2010.

AP Class	%		
	2007	2009	2010
C-Type	8%	6%	8%
N-Type	40%	46%	40%
R-Type	3%	3%	2%
Z-Type	50%	46%	49%

Table 4.3: Percentage of access points belonging to each class during the analyzed periods.

In order to obtain the rules for the automatic classification of access points, values of the utilization rate and the average number of users were calculated for a total of 411 access points. Additionally, 155 access points were classified manually by the observation of their usage patterns. Half of these were used to train a decision tree and the other half to test it. The CR&T algorithm was used to obtain the rules for the automatic classification process which are given in Table 4.4 and illustrated in Figure 4.9.

Utilization rate (U)	Average number of users ($\overline{N}_{user}$)	Class
$U < 0.712$	$\overline{N}_{user} \geq 0.513$	N-Type
$U \geq 0.712$	$\overline{N}_{user} \geq 2.692$	R-Type
$U \geq 0.712$	$0.513 \leq \overline{N}_{user} < 2.692$	C-Type
any	$\overline{N}_{user} < 0.513$	Z-Type

Table 4.4: The rules for the classification of access points.

After we obtained the automatic classification rules, we set out to compare the results of the manual and the automatic classification process. So we selected a set of 86 access points from the Azurém campus and classified them both manually and automatically. We compared the outcome of the two and obtained a matching of 87%. That is, 75 out of 86 access points were classified in the same way manually and automatically. The results are depicted in Figures 4.10(a) and 4.10(b) which show the spatial distribution of the different types of access points at the Azurém campus as obtained by the automatic and the manual classification process, respectively.

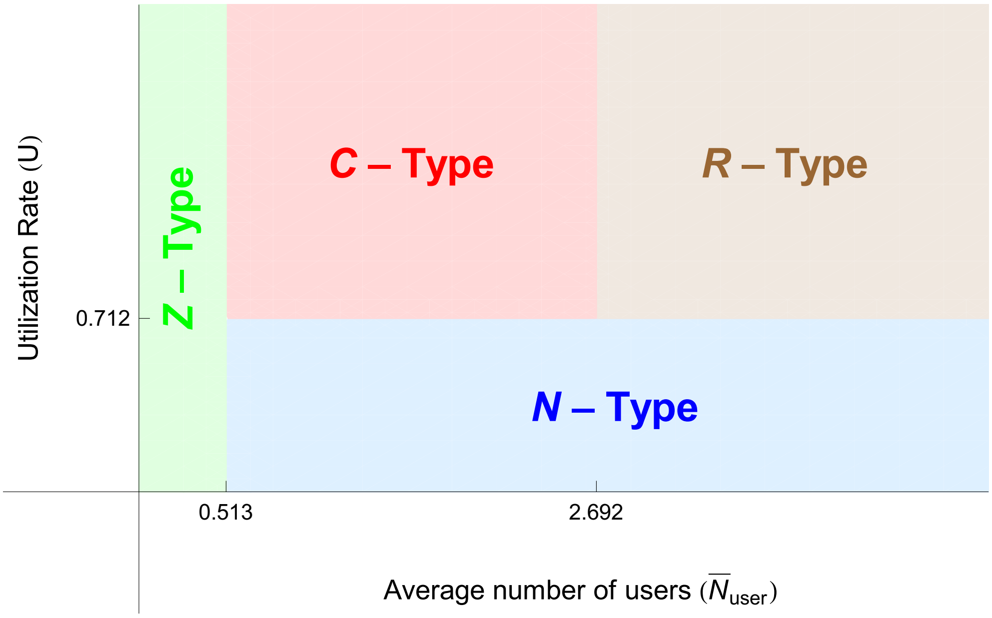
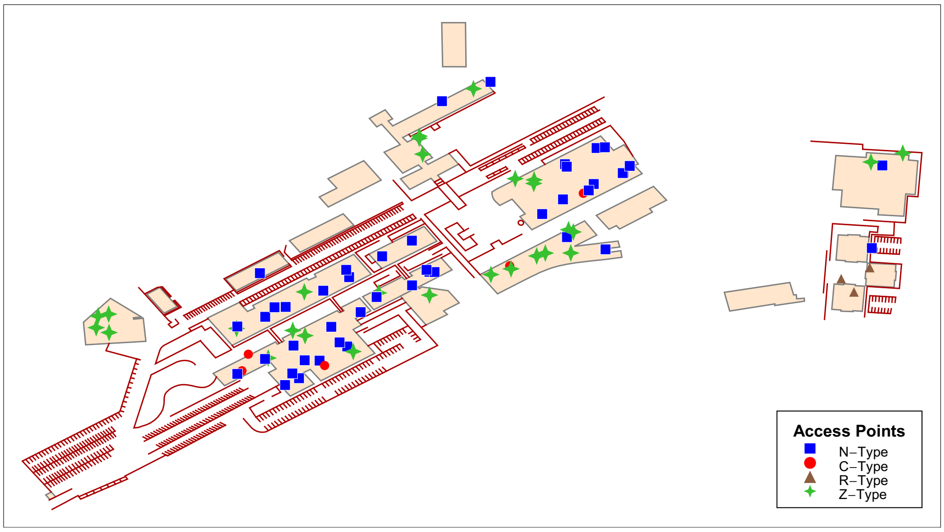


Figure 4.9: Access point classification.

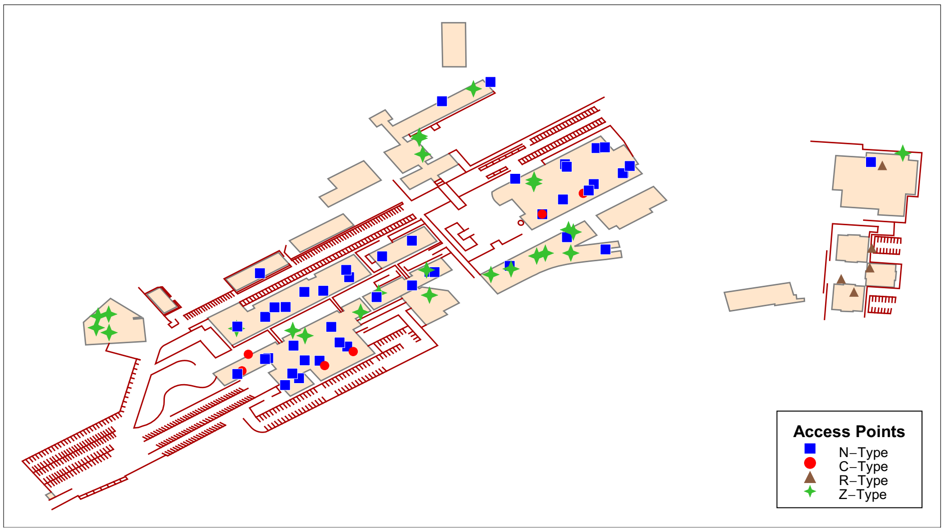
From Figure 4.10 it is quite clear that the N-Type and Z-Type are the most frequent access point classes. All the R-Type access points are concentrated at the dormitories (the isolated buildings on the right). In the refectory (the small building on the left side of the figure) all access points are of the Z-Type. This is an expected result as people usually go there only to have lunch or dinner and just a few of them switch their laptops on. Eventually, this may change over time as an ever increasing number of people starts to use WiFi-enabled smartphones.

The classification module is a Java application that periodically requests, from the SenDS, one week long time series of all access points in the network, calculates the utilization rate and the average number of users and classifies the access points according to the established set of rules. It is responsible for including new data in the model about the following attributes that describe objects of type *access point*: *avgNumUsers*, *utRate* and *apClass*. For each attribute, besides its id and the id of the object it describes, a time stamp, a name and a value are inserted in the model.

The classification of access points allows for the identification of areas with similar characteristics. Detecting changes in the usage patterns of the access



(a) Automatic classification of access points.



(b) Manual classification of access points.

Figure 4.10: Spatial distribution of access point classes at the Azurém campus.

points may be interesting for an application like the one we describe in Section 5.3. As well, the results of the classification of access points combined with those of the correlation, which is presented next, are used in the process of the creation of a new type of objects – the hotspots – as we will explain in Section 4.5.3.

4.5.2 Correlation between access points

Based on the performed analysis of the WiFi data time series, it appeared to be likely that access points belonging to the same class were strongly correlated, that is, with similar usage patterns, representing places with the same "pulse". This being true, it would allow us to identify spaces with similar characteristics based on the value of the correlation coefficient between access points covering those areas. In order to exploit this, two sets of two week long data sets were used. The Pearson correlation coefficient was calculated between each one of a total of 88 access points and all the other 87 access point at the Azurém campus.

The Pearson correlation coefficient is given by

$$r_{XY} = \frac{1}{N-1} \sum_{i=1}^{N} \frac{(X_i - \overline{X})(Y_i - \overline{Y})}{s_X s_Y} \tag{4.1}$$

where N is the length of the time series, $\overline{X}$ and $\overline{Y}$ are the averages of the two time series containing number of users for the two access points, X_i and Y_i, respectively and s_X and s_Y are their standard deviations.

The first results showed that among the access points of the same class, 60% of the N-Type access points have the correlation coefficient higher than 0.75, while the C-Type and R-Type highest values are 0.55 and 0.65, respectively.

Figure 4.11 shows N-Type access points with correlation coefficients higher than 0.85. We use such a high value here to better illustrate the connections between the correlated access points. A lower value would result in a highly dense mesh of lines that would render the figure unreadable. These connections give origin to relations between objects after the processing module finishes its work.

The correlation module shows how new relations and their attributes may be inserted automatically in our world model. It is a Java application that periodically requests one week long time series from the SenDS (described in Section 4.4) and calculates the Pearson correlation coefficient between all pairs of access points according to Equation (4.1). This processing module implements the way relations and their attributes can be inserted automatically in the model. It is

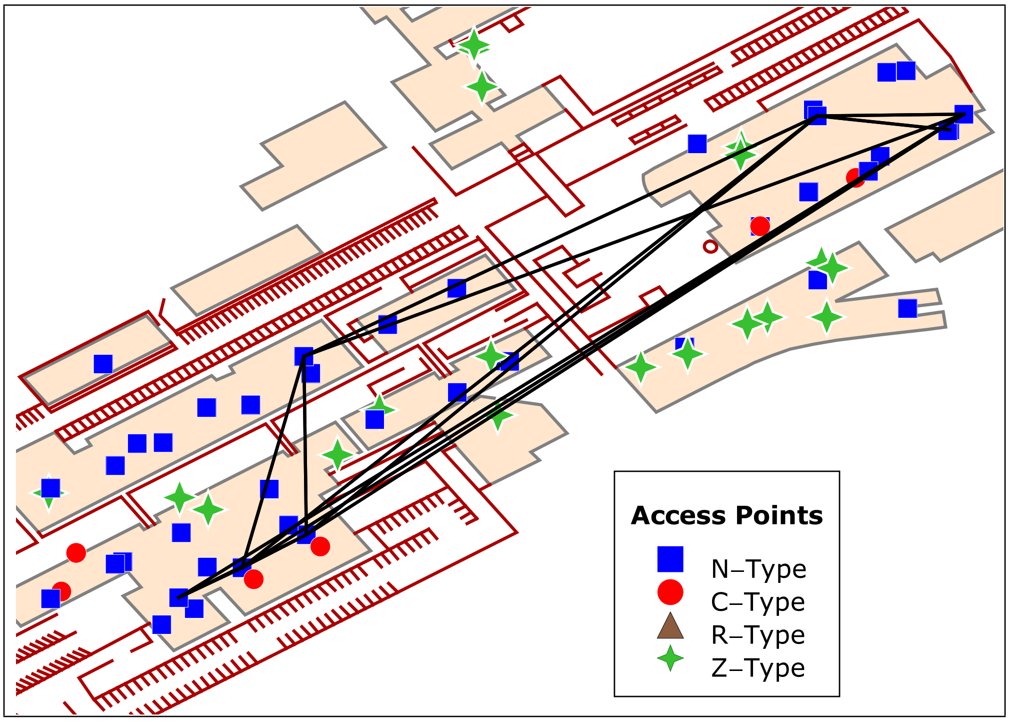


Figure 4.11: A set of highly correlated access points ($r_{XY} > 0.85$).

responsible for inserting a relation called *Is_Similar_To* between two objects of type *access point* as well as the relation's attribute *correlationCoef* which contains the calculated value. This relation and its attribute are only inserted when the correlation coefficient is higher than a given threshold. Presently used threshold is 0.65. All the results that yield a value lower than the threshold are discarded.

The obtained results for the correlation coefficient calculations between access points indicate that it may be used to a certain extent to identify spaces with similar utilization patterns in different parts of the campus or even in different campuses. In conjunction with the results of the classification module, the results of the correlation module are used in the process of detection and creation of a new type of objects as we will describe in the next section.

4.5.3 Searching for popular areas in the campus

Creation of higher level objects in the model based on the characteristics of the objects representing access points, may allow us to obtain a new vision of the observed space. When we add the analysis of groups of users and the information about frequent visitors for the detected hotspots, described in the next section, we may well get a new and interesting insight into the pulse and the dynamics of our campus.

The proposed module is an example of how new objects can be created automatically in the model. For this purpose, it uses data generated and inserted in the model by the two previously described modules, the classification and the correlation module.

Definition 4.2 (Hotspot object). *A hotspot represents a popular area in the campus or elsewhere, covered by one or more nearby access points of the same type which are highly correlated and have the average number of users ($\overline{N}_{users}$) higher than a certain threshold.*

Figure 4.12 shows the first three steps in hotspot detection which is the grouping of access points according to their type, correlation coefficient and distance between them. The next step is to obtain $\overline{N}_{users}$ for the identified set of access points (the sum of $\overline{N}_{users}$ of all involved access points) and check if it is higher than the threshold. Figure 4.13 shows an example where groups of two and three N-Type access points and four R-Type access points gave origin to three new hotspots.

After grouping the access points and checking if they meet the requirements, a new object representing a hotspot is created in the model. Also, new relations are created between the hotspot object and the access point objects that were involved in it its creation.

This module is also a Java application that periodically requests from the SC the available data on all pairs of access points having a relation *Is_Similar_To* and then groups them together according to their class (N, Z, C, or R). Then, it calculates the distance between them based on their coordinates. If this value is higher than a threshold, it discards some of the access point pairs because they

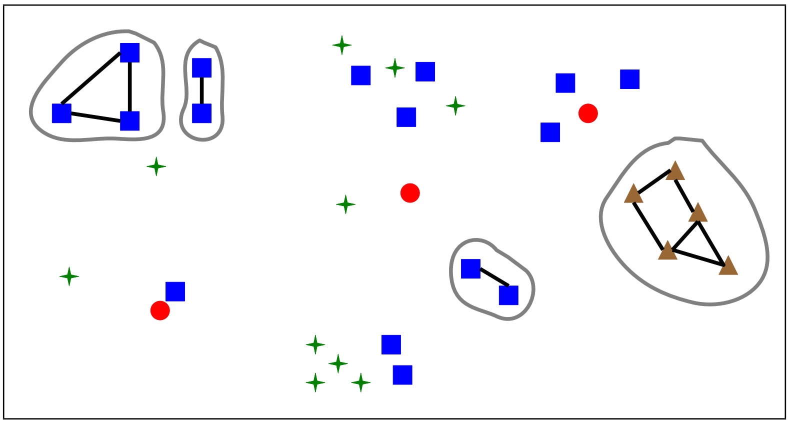


Figure 4.12: Grouping of access points according to their type, correlation coefficient and distance between them.

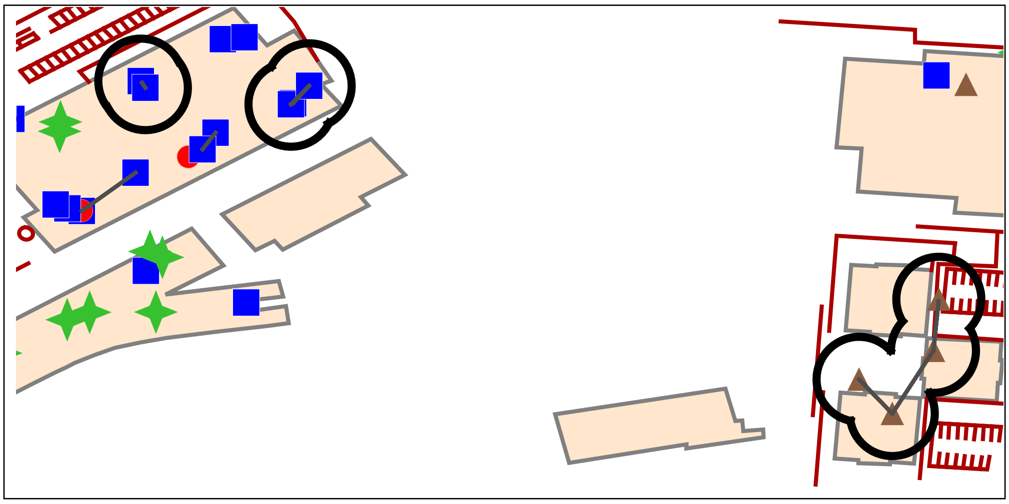


Figure 4.13: Hotspots for $r_{XY} > 0.65$, $d < 30m$ and $\overline{N}_{users} \geq 5$ (for N-Type) and $\overline{N}_{users} \geq 25$ (for R-Type).

are too far apart to belong to the same hotspot. For each group of access points, a sum of their average number of users ($\overline{N}_{users}$) is calculated and if it is higher than a minimum value, a new object of the type *hotspot* is created. This object establishes relations called *Is_Served_By* with each of the members of the final set of access points that participated in its creation. When $\overline{N}_{users}$ of a hotspot goes below the threshold, the hotspot object's *status* field is set to *inactive*.

In the next two sections we explore the detection of frequent users and groups of users as two more examples of processing modules that may feed the world model with data that contributes to the space characterization. For this purpose we use the lists that contain all the MAC addresses that connected to each access point instead of just the number of connected users. We assume that one MAC address corresponds to one person. Our objective is to observe if the users in a particular area of the campus are the usual visitors of that place or not by calculating the *tenant level* of a place as well as find and characterize the places in the campus where the groups of users emerge and spend their time.

4.5.4 Hotspot tenants, visitors and strangers

We propose a simple way of modeling our human perception of the usual users of a place. For instance, when we enter a familiar space, we easily notice whether the usual people are there or if there is someone new or someone missing. We propose a way of measuring this based on the number of occurrences of each MAC address during a given time period. The process of calculating the *tenant level* of an access point is the responsibility of one of the processing modules dedicated to this particular operation. Besides the calculations, it is also responsible for inserting the results in the model.

Detecting a set of MAC addresses that repeatedly appear at some nodes of a network may allow for the characterization of a place in terms of its users. Keeping track of the type of visitors of a particular place (place visitor profile) may let us discover whether a place is normally visited by its tenants, frequent or occasional visitors or complete strangers. This kind of knowledge may be used by context-aware applications running in the infrastructure.

This module is responsible for the update of two attributes of a hotspot object or an access point object, one called *long term tenant level* and the other called *short term tenant level* with the values between 0 and 1.

For each hotspot or access point, the list of connected devices' MAC addresses is recorded every five minutes. Frequencies of each detected MAC address are stored and updated. Figure 4.14 shows an example in which data was stored during 24h resulting in 288 samples. In each sample the number of connected

MACs per access point is recorded and the counter of occurrences for each present MAC is incremented. After the acquisition of samples during a time period, the available data consists of the frequencies of each detected MAC address. In our experiments the considered time periods were 20 days (corresponds to 5760 samples) for the long term and 5 minutes (a single sample) for the short term.

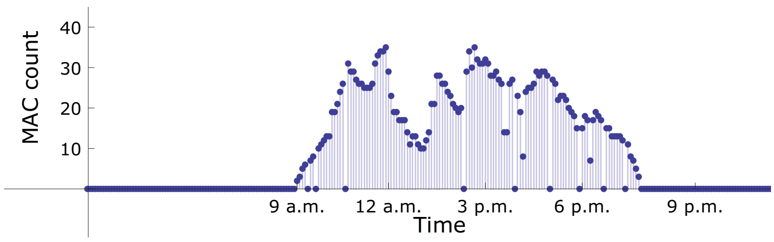


Figure 4.14: 24h hours of MAC counts for an access point.

4.5.4.1 Long term tenant level

The *long term tenant level* shows if a place is mostly visited by the same people, people frequently dropping in or just occasionally passing by. It is calculated for the data collected during a fairly long time period, like several weeks, in the following way:

1. Calculate absolute frequencies for each observed MAC address i ($f_{\text{abs},i}$);

2. Calculate normalized frequencies of observed MACs: $f_{\text{norm},i} = f_{\text{abs},i}/S$, where S is the number of samples in which at least one MAC was detected (valid samples). This value is chosen instead of the total number of samples that are taken during the considered time interval in order to discard periods in which nobody connects, e.g. during the night or during the weekend. In places that never close and where there are always people, it may be more suitable to consider the total number of samples. In Figure 4.14, S is given by the number of time intervals between 9 a.m. and 7.30 p.m.;

3. Classify MACs based on their normalized frequencies and according to the rules described in Table 4.5. The result is depicted in Figure 4.15 for a set

of 19 MACs randomly chosen out of the total of 720 MACs detected during the observation period (20 days).

Table 4.5: Place visitor profile based on detected MACs.

Normalized frequency (f_{norm})	MAC Class
$f_{\text{norm}} \geq 0.2$	Tenants
$0.1 \leq f_{\text{norm}} < 0.2$	Frequent Visitors
$0.01 \leq f_{\text{norm}} < 0.1$	Visitors
$f_{\text{norm}} < 0.01$	Strangers

4. Count the total number of detected MACs (n_{MACs}).

5. Calculate the final value for the *long term tenant level*:

$$tenantLevel_{\text{LT}} = \frac{f_{\text{normT}} + f_{\text{normFV}} + f_{\text{normV}}}{n_{\text{MACs}}} \tag{4.2}$$

which is the sum of all the normalized frequencies higher or equal to 0.01 (all tenants, frequent visitors and visitors) divided by the total number of detected MACs. This results in a value between 0 and 1 for the *long term tenant level* of a place.

For the example illustrated in Figure 4.15, the calculated normalized frequencies reveal the detection of 6 tenants, 1 frequent visitor, 7 visitors, and 5 strangers. The resulting tenant level follows from Equation (4.2): $tenantLevel_{\text{LT}} = 0.13$.

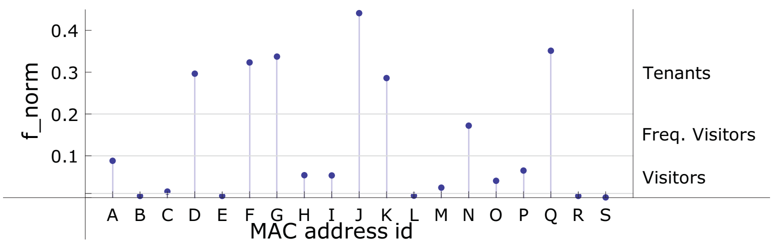


Figure 4.15: An example of a place visitor profile based on a set of MACs detected at an access point during an observation period of 20 days.

4.5.4.2 Short term tenant level

The *short term tenant level* shows, in turn, if the considered area is being visited at a particular moment by its tenants, frequent visitors, visitors or strangers. The *short term tenant level* is calculated for a single sample as follows:

1. Count the number of tenants, frequent visitors, visitors and strangers in the current sample. All the detected MACs in the current sample are compared to their class, if any, in the long term place visitor profile. MACs appearing for the first time are considered strangers;

2. Calculate the value for the *short term tenant level* as:

$$tenantLevel_{\mathrm{ST}} = \frac{f_{\mathrm{normT}} + f_{\mathrm{normFV}} + f_{\mathrm{normV}}}{m_{\mathrm{MACs}}} \tag{4.3}$$

   which is the sum of the normalized frequencies higher or equal to 0.01 divided by the total number of detected MACs (tenants, frequent visitors, visitors and strangers) in the current sample. This results in a value between 0 and 1 for the *short term tenant level* of a place.

For the same case from the Figure 4.15, where we calculated the *long term tenant level* for an access point with 19 detected devices, we will now calculate the *short term tenant level*. To do this, we will consider that the following devices were detected in a single sample: A, D, K, L and N. According to the figure, we have 2 tenants (D, K), 1 frequent visitor (N), 1 visitor (A) and 1 stranger (L). If we use the same values for their normalized frequencies as those shown in the figure, Equation (4.3) yields: $tenantLevel_{\mathrm{ST}} = 0.21$. The resulting slightly different value than the one obtained for the long term means that, although a place may be mostly visited by occasional visitors, there exist some time periods when it is visited by its tenants and frequent visitors.

For the following examples the *long term tenant level* was calculated for a 20 days long data set and the *short term tenant level* was calculated for each single sample during a 48h time period (from Sunday, 5 p.m. to Tuesday 5 p.m.). The first example (Figure 4.16(a)) is from one of the library access points which is usually visited by a very large number of users during the opening hours. The second example (Figure 4.16(b)) is from a rarely used access point, localized near the design rooms in the department of architecture. The third example (Figure 4.16(c)) shows the results for the set of four access points in the students' dormitories which are usually heavily used during all week. It shows strong frequent visitor, and sometimes, tenant profile.

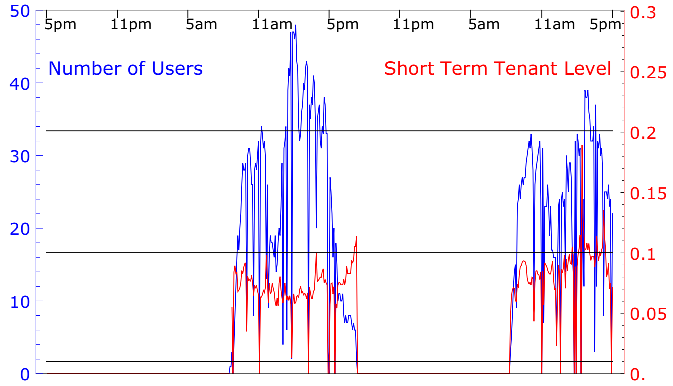
It makes sense that in the library the visitor profile is characterized by many occasional visitors as it is a place where a large portion of the academic population spends some time during the week. However, the characterization of the design rooms profile is not as clear, because there are many fluctuations during the day and during the week. At some hours of the day only tenants are found, but as soon as the number of users increases, the tenant level value lowers. In turn, the dormitories exhibit quite a clear profile. Although there is a large number of users like in the library, this place is mostly used by its residents. Although it is not always clear whether the users connected to a particular access point are its tenants or occasional visitors, the results show a new perspective on the space usage. New knowledge about a space may be extracted from graphics like those shown in Figure 4.16.

4.5.5 Groups of users

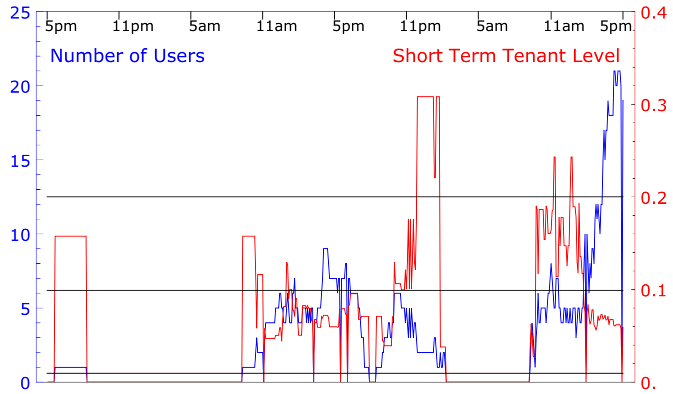
The purpose of detecting groups of users is to extract more characteristics of the spaces where these groups are created, according to their number and their dimension. The identification of groups of devices, and to some extent, groups of users, allows for the detection of the areas in the campus where groups of users work or meet. It also may add more dynamics to our model as it reflects their presence and movement within the area represented in the model.

First we define the "group" and what we mean by "iteration" in order to make it clear in the rest of the section.

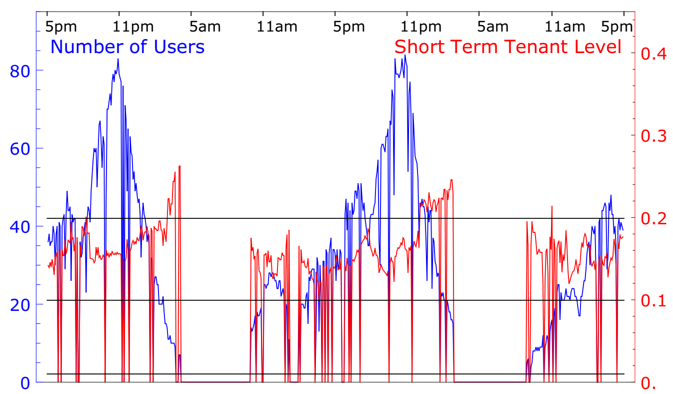
Definition 4.3 (Iteration). *An iteration consists of six consecutive samples, taken every five minutes.* This means that a single iteration is 30 minutes long. Due to overlapping (see Figure 4.17) the duration of two consecutive iterations is



(a) The library is characterized by many occasional visitors.



(b) Design rooms are characterized by a few tenants when no one else is there.



(c) Dormitories are characterized by frequent visitors and tenants.

Figure 4.16: Short term tenant level variation during two days time period. Left Y-axis represents the number of users and the right Y-axis, the short term tenant level.

35 minutes.

Definition 4.4 (Group). *A group is defined as a set of two or more MAC addresses that are detected together at two different access points, during at least four iterations (*$N_{\text{iter}} = 4$*) at one of them.* As explained next and shown in Figure 4.17, the duration of this time interval may vary, depending on whether the MACs X and Y are found together in consecutive or non-consecutive iterations.

The minimum number of iterations for the creation of a group determines the number of detected groups. Four iterations for a week time is a reasonable value because it means that the members of the group are found together during at least 45 minutes, which is the minimum duration of a lecture. Also, the constraint of at least two different access points is important for it helps to eliminate laboratory equipment that is connected to the WiFi network unattended. An enhancement could be made by establishing a minimum distance between the two different access points in order to eliminate the "ping-pong" effect which may occur between nearby access points. When the density of access points is high and a device is within the range of several access points, it often changes its connections from one access point to another without the user actually moving physically to a different place.

Two or more MAC addresses should be found connected to at least two access points during at least N_{iter} iterations for an object of type *group* to be created. Once created, this object will have a relation *Is_In_Range_Of* established with an access point or a relation *Is_At* with a hotspot every time it connects to the network. These two are the examples of what we call dynamic relations as they are ephemeral. They are created when the group connect to the network and removed when they disconnect.

Figure 4.18(a) shows a graph representation of a group in which vertices correspond to MAC addresses that were found connected to an access point during a considered time interval. The weight associated with each edge represents the number of iterations during which the pair of MACs was observed. If we set a threshold to 4 iterations ($N_{\text{iter}} = 4$), as we did in our experiments, all the edges with weights lower than 4 are removed as shown in Figure 4.18(b).

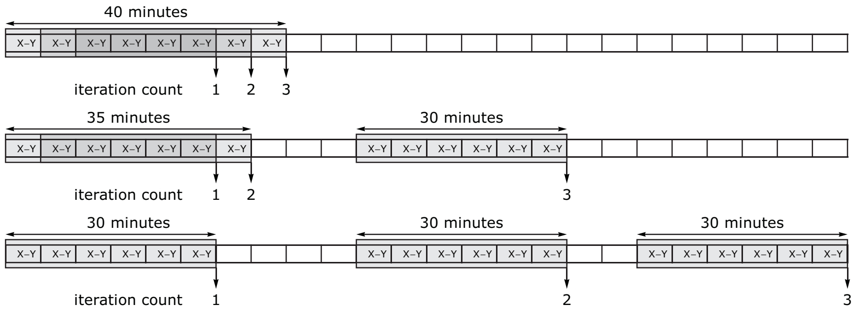
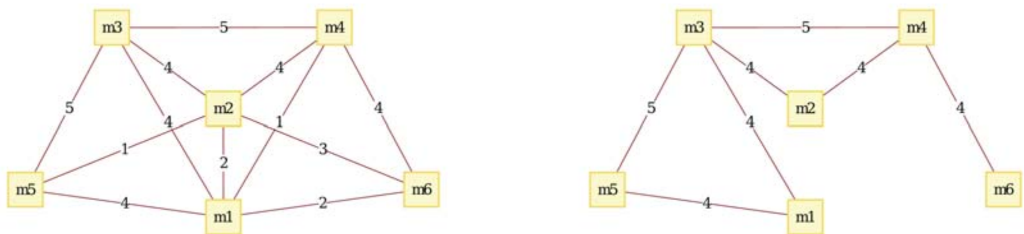


Figure 4.17: Duration of the total number of counted iterations is variable. Three iterations may mean that the members of a group were found together during 40 min, 65 min or 90 min. The minimum of minutes, given by $30 + (N_{\text{iter}} - 1) \times 5$, is obtained if all the iterations are consecutive (partially overlapping) and the maximum, equal to $N_{\text{iter}} \times 30$ is obtained if there is no overlapping between iterations.



(a) Edges between MAC addresses correspond to the number of iterations.

(b) When a threshold of 4 is set for the minimum number of iterations, edges with lower weights are removed.

Figure 4.18: Graph representation of a group of users.

In our experiments, we detected 50 groups on average during one day in the campus. During one week we detected 830 groups on average. We also identified places in the campus where the largest number of groups emerge and also where the largest groups appear. Each of the considered places is covered by at least two and at most four access points. As shown in Figure 4.19, the most frequent group dimension is 2 which is the minimum. The largest detected group had 42 members and it was found in the dormitories.

Compared to the total number of the detected unique MACs (3222), those par-

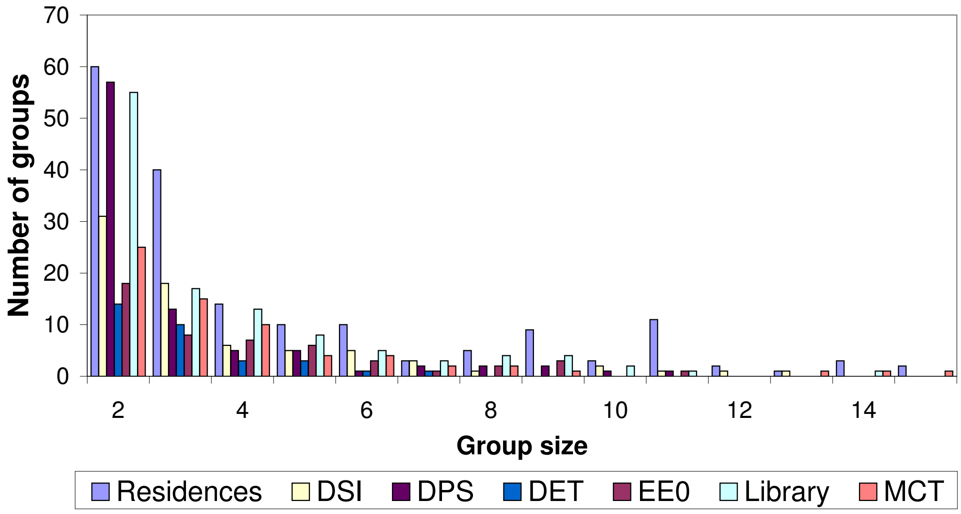


Figure 4.19: Dimensions and number of groups for a week long data set. Larger groups were detected, 42 being the biggest. They are not shown here as they are not very frequent.

ticipating in groups (945) represent nearly 30%. As expected, the most crowded places are the library and the dormitories and there appear the largest number of groups. We identified 7 places in the campus where 70% of the groups emerge (see Figure 4.20). If we do not consider the dormitories, than we have 50% of all groups emerging in 6 places.

In Figure 4.21 we compare values for three different weeks. There are slight differences, but the general picture is quite the same from one week to another. In the Azurém campus the residences are, undoubtedly, the place where the largest number of groups emerge, followed by the library and the Department of Information Systems.

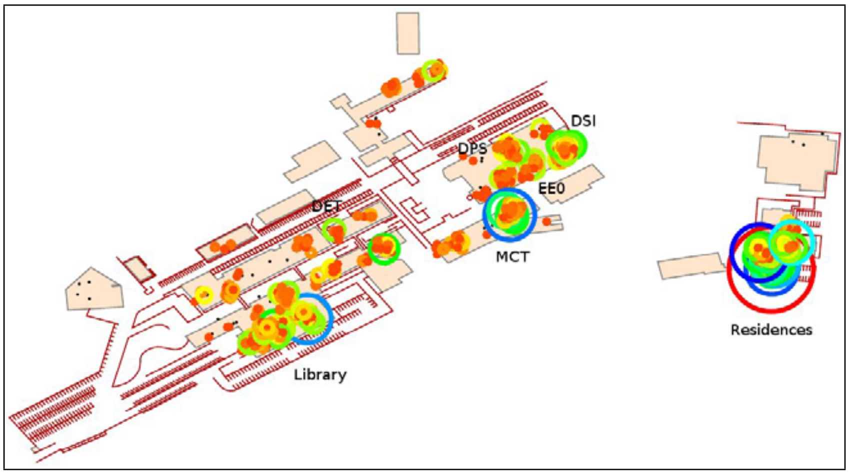


Figure 4.20: Groups at the campus after analyzing a week long data set. The smallest groups have 2 members and the largest 42 (red circle). The radius of the circle is proportional to the dimension of the group.

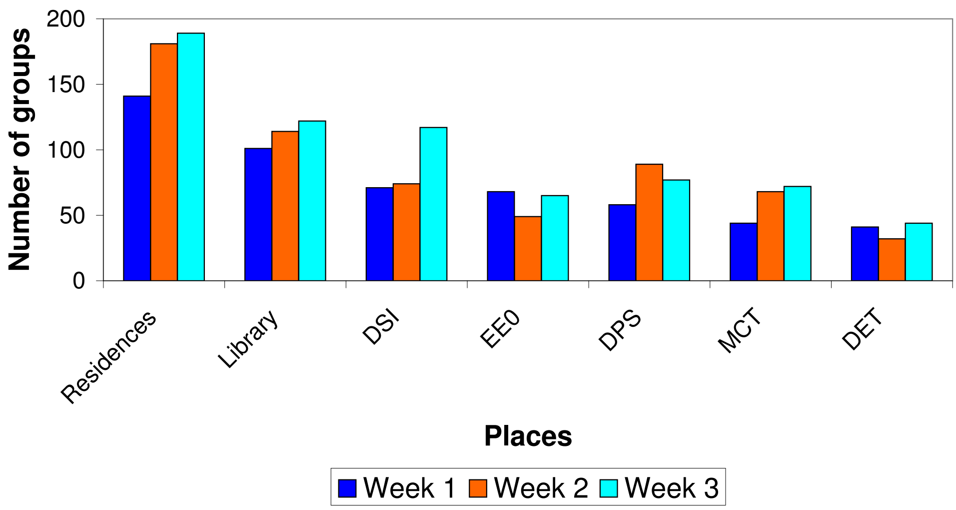


Figure 4.21: Comparison of the number of groups in three different week long data sets.

4.6 Chapter summary

In this chapter we have described our system architecture that fits the five layered framework proposed by Ailisto *et al.* (2002). The first four layers of our system were presented in detail. These comprise the world model services, the processing modules, the raw data acquisition modules and the sensor networks. The applications, which are part of the fifth layer, will be described in the next chapter.

The main component of our system is the Symbolic Contextualizer which provides a service for accessing the space model. The model itself consists of objects and relations between them and is stored in a relational database. Applications and processing modules may make requests for objects from the model to the SC. The SC replies with a list of matching objects descriptions in XML format.

The model can be edited either manually through the SC Space Editor web application or automatically through the processing modules. The former allows a user to easily create his/her own model. The latter retrieve data from the data acquisition modules and process them in such a way that they become suitable for the insertion in the model.

In the scope of our research we used exclusively data gathered from an University wireless network. The processing modules were developed according to the type of data we had. However, other sensor networks may be used for context data extraction from the environment. When this happens, new processing modules need to be developed in order to be able to feed new data into the model. One of our main objectives was to identify and define new parameters based on WiFi data that may allow us to describe better the spaces contained in the coverage area of the access points. For this purpose we used two different types of data, one containing the number of connected users per access point and the other containing the lists of users' MAC addresses.

We defined three processing modules that use the first type of data and two processing modules that use the second type of data. There is an additional processing module that does not use the raw data but relies on data inserted in the model by two other processing modules. All modules transform data in objects, relations or attributes for the space model. WiFi network data showed to be a very rich source of information about spaces. With the six proposed modules we certainly did not exhaust all the possibilities and in the future many more processing modules may be defined in order to collect even more information about the characteristics of the WiFi zones, and not only in the University campuses, but also in urban areas in general.

Chapter 5

Applications

The beginning of knowledge is the discovery
of something we do not understand.
Frank Herbert

In this chapter we describe the fifth level of our system architecture - the application level. We propose three different applications. One of them was implemented and deployed and the other two are left as future work.

These applications aim to show the usefulness of the developed space model as they rely on data that is stored in it. Data from the environment has been gathered using the WiFi network as a proxy to space usage and fed to the model by the means of the set of processing modules described in the previous chapter.

The first application we introduce here is the Route Finder. It was developed for pedestrian navigation, especially in the indoor environments where the GPS signal is too weak or is not detected at all. It is based on the symbolic space model, the user self-location and the recognition of nearby features. It uses the properties of the relation *Is_Accessible_From* that, in our space model, represents the existence of a path between two objects (e.g., the restaurant *Is_Accessible_From* the beach).

The second application implements anomaly detection techniques applied to the data from the model. This application includes a set of specialized processing modules created for it. Based on several anomaly detection techniques applied to the time series gathered from the access points, we try to identify significant deviations from the usual usage patterns of an environment.

The third application aggregates data that is injected in the model by all the processing modules or only by a chosen subset. Based on this data, it provides a visualization of the overall campus dynamics over time.

5.1 Pedestrian navigation

The Route Finder (version 0.4) is a web-based application which relies on the previously described symbolic world model to assist pedestrian navigation in the indoor and outdoor environments. The application interface allows the user to specify his/her current location and the desired destination. As for the SC Space Editor and the SC Browser, the Route Finder application uses the Symbolic Contextualizer query interface to retrieve the descriptions of the selected objects. Based on the existing *Is_Accessible_From* relations, it calculates the shortest path between the two places.

Presently, this application is available on two different servers that store two different local models:

- http://scontext.uma.pt:8080/route/index.html
- http://local.dsi.uminho.pt:8080/route/index.html

The application provides the description of the path the user has to follow if s/he wants to go from his/her current location to the specified destination. The path is the ordered list of places the user has to walk through, with the available description for each place (a set of *object attributes*). This solution allows the user to be guided based on what s/he sees in the nearby environment. It can be simply described as being a digital version of the hints people receive when they ask for directions.

Initially, the user selects an origin and a destination, eventually using partial descriptions of the two places if s/he is not sure about the exact name, as in Figure 5.1. A place can be identified by different names and there may be several places with similar names. In order to avoid ambiguities, the interface provides a list of objects that match the user's search. In this way, the user may select correctly his/her current location and the desired destination (Figure 5.2).

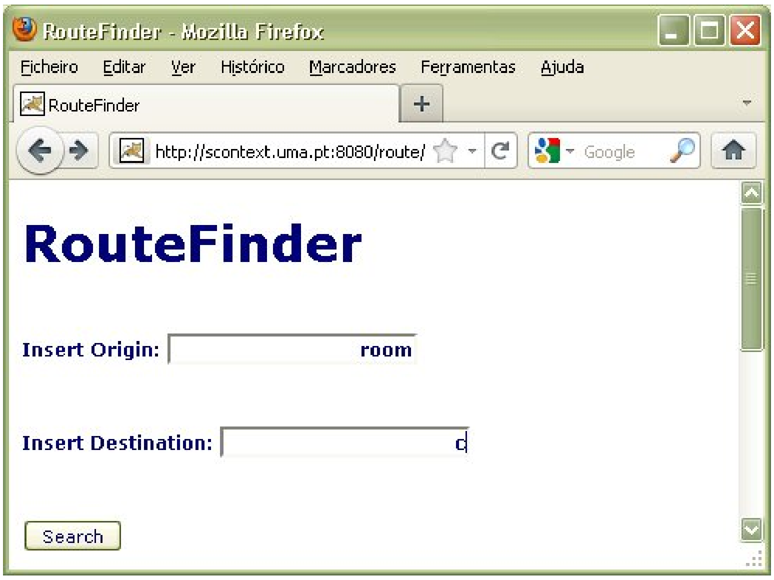


Figure 5.1: The Route Finder homepage.

The Route Finder queries the Symbolic Contextualizer to find the shortest path between the origin and the destination based on the *Is_Accessible_From* relation. It also takes into account the value of the relation *type*, which may be *symmetric* or *simple*, depending on the path being bidirectional or unidirectional.

The implemented path search algorithm is based on the Breadth First Search algorithm from the graph theory. It starts with a search for the requested objects descriptions in the model (origin and destination objects). Once found, it extracts all the relations *Is_Accessible_From* in which the destination object appears. The resulting list of existing pairs of related objects is saved and is used for path search. All the nodes are searched exhaustively following the First In, First Out (FIFO) technique until the origin is found.

When the origin is finally reached, the final path from the origin to the destination is obtained from the list of visited objects which is read backwards. The resulting directions are shown to the user as illustrated in Figure 5.3. If there is

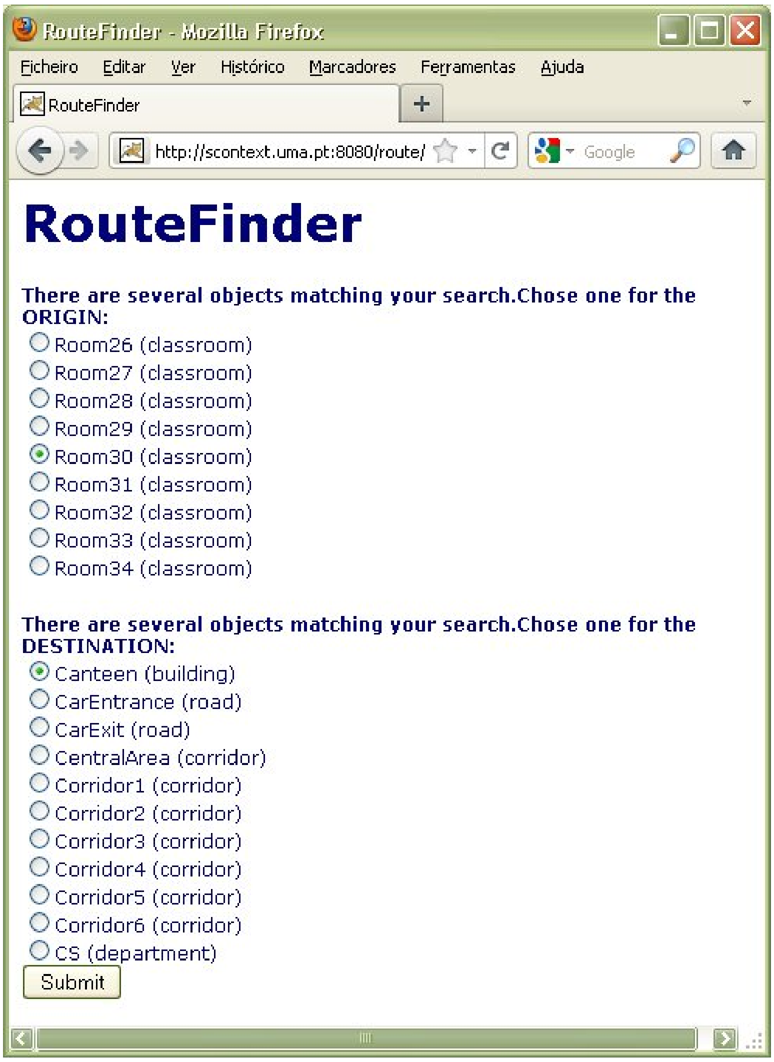


Figure 5.2: The Route Finder interface provides a list of objects that match the user's search. In this way, the user may select correctly his/her current location and the desired destination.

no path between the two objects, the user is informed about that and is invited to submit a new search as shown in Figure 5.4.

Our future plans, to further enhance the routing capabilities, include the exploitation of other relations between objects, such as *Is_Next_To*, to improve the route description provided to the users. For computing estimates of the trip time, the values of some attributes, such as the distance between objects or travel

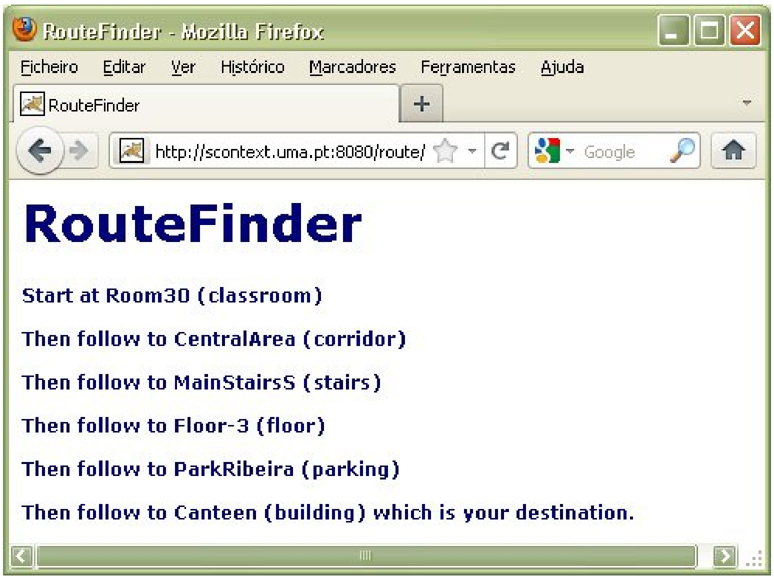


Figure 5.3: The Route Finder shows the directions to the user according to the chosen origin and destination objects.

Figure 5.4: Route Finder reply when there is no path between the requested objects.

time attributes of the *Is_Accessible_From* relations, may also be explored.

Although the described approach is entirely supported by symbolic space models and human self-location, some navigation applications may benefit from automatic positioning to help users locate themselves. Integrating positioning systems providing position in a geometric referential into the symbolic space model's framework is a possibility in the future.

An evaluation of this application is provided in Section 6.2.1.

5.2 Anomaly detection

In this section, we propose a set of anomaly detection techniques that may be used to develop an application for the detection of unusual situations in physical spaces, such as high concentration of people in a certain place (e.g., a conference room) or the absence of people in what is usually a crowded place (e.g., a city square). Since we are using the WiFi network as a proxy to capture space dynamics, an unusual happening in the covered area should have a certain impact on the utilization patterns of the network itself.

The collected data from our campus, the time series describing the evolution of the number of users connected to a particular access point, exhibit a periodic behavior, with daily and weekly trends as we saw in Section 4.5.1. Taking these characteristics of our data into account, we exploited four simple techniques in an attempt to detect anomalies in the space usage.

The detection of anomalies in time series has been addressed in many application areas. Regarding the analysis of data about the utilization of communication networks, quite a lot of attention has been dedicated to network security and, in particular, intrusion detection. There are many techniques that have been developed and used for this purpose. For instance, (Kumar, 1995) evaluates several different approaches of anomaly and intrusion detection, such as statistical approaches which rely on the creation of profiles and predictive pattern generation which establishes that there is a pattern in events sequences, among others. A signal analysis technique based on wavelets which allows to effectively expose features of the network traffic is proposed by (Barford *et al.*, 2002). In (Thottan & Ji, 2003) a review of anomaly detection methods is presented and a statistical signal processing technique based on abrupt change detection is described.

Our aim is to understand how the usage of our network is related to what is happening in the coverage area while the above mentioned research is more focused on the network traffic and on what is happening in the infrastructure itself. We, however, also explore the entropy concept as suggested in (Nucci & Bannerman, 2007) as it will be explained further on.

Using the WiFi network data as a proxy to what is happening in the physical spaces may provide us with data about unusual events or situations if we find a way to record the usual utilization patterns for each space or access point covering that area. In this way, it should be possible to detect anomalies, like a very high number of people in a place that has never been crowded before or, a very small number of users in a usually very popular place. Four techniques for anomaly detection in physical spaces were experimented and the following sections report the obtained results.

5.2.1 Utilization level difference

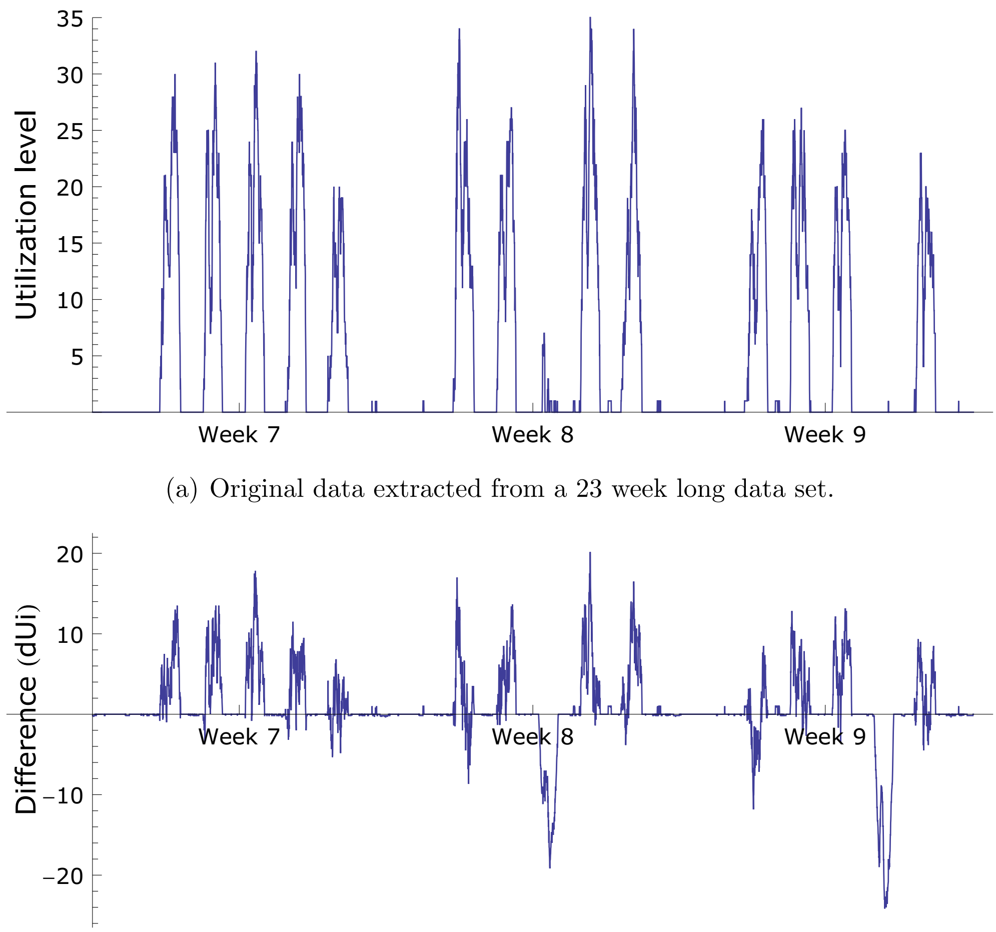
Our first approach in anomaly detection was to exploit the periodic behavior of a single access point by comparing its current utilization level, u_i, with the average value of the N previous weeks at the same time instant. In other words, we compared the utilization level at, say, Tuesday 9:30 a.m., to the average value of the N previous Tuesdays at 9:30 a.m. Equation 5.1 models this approach through the calculation of the difference of utilization levels (dU_i) defined as:

$$dU_i = u_i - \frac{1}{N} \sum_{j=1}^{N} u_{i-j \times SW} \tag{5.1}$$

where u_i is the current utilization level, N is the number of past weeks (history), and SW is the number of samples per week.

Figure 5.5(a) shows an example where the used data set is extracted from a 23 week long data time series of one N-Type access point. Figure 5.5(b), in turn, shows the resulting difference which was calculated with $N = 6$ and $SW = 2016$. One day with a significantly lower utilization and one day with no utilization at all (Wednesday of week 8 and Thursday of week 9, respectively) were detected as anomalous by this technique. However, there are also some false positives, like the Mondays of week 9 where a slightly lower utilization level relative to the previous weeks gave origin to a high negative value of dU_i.

This technique is suggested by the periodic patterns observed in a great number of access points, but suffers from an undesired characteristic of our data - its high variance. The major consequence of the high variation of the data is that



(a) Original data extracted from a 23 week long data set.

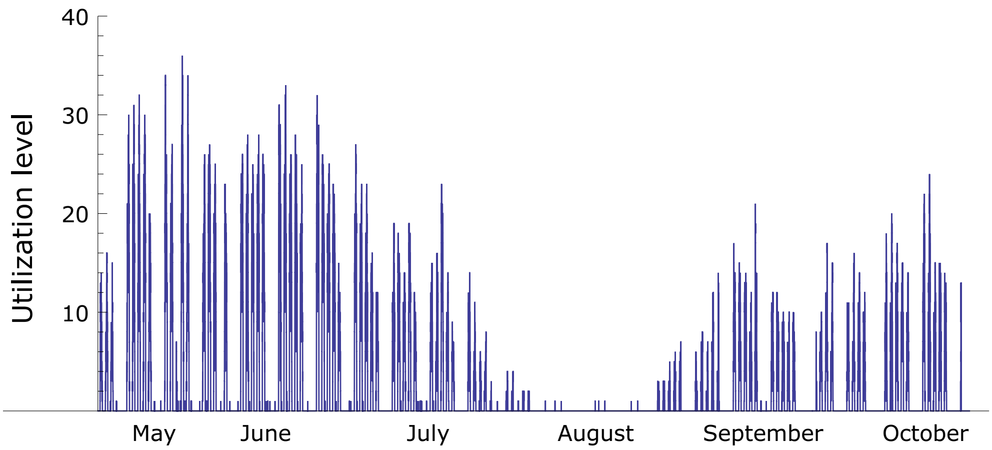
(b) Resulting utilization level difference calculated with $N = 6$ and $SW = 2016$.

Figure 5.5: Calculating the utilization level difference for a three week long subset of data for a N-Type access point.

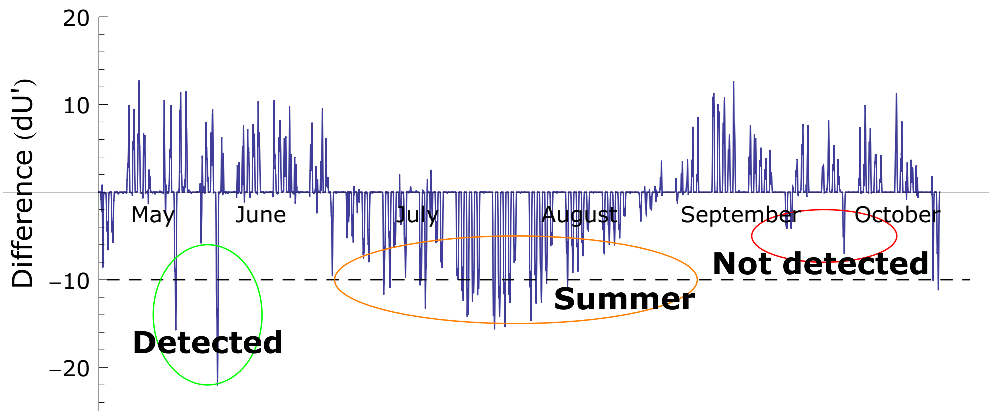
dU_i reflects that variance and, therefore, cannot be used to detect anomalies. In order to reduce this effect, we pre-processed our data samples by replacing each sample by the average of the M most recent samples (a moving average), thus turning dU_i into:

$$dU_i' = \frac{1}{M} \times \left(\sum_{k=0}^{M-1} u_k - \frac{1}{N} \sum_{j=1}^{N} \sum_{k=0}^{M-1} u_{i-k-j\times SW} \right) \tag{5.2}$$

With this filtering technique, the comparison enabled us to detect the same two anomalies as before, but much more clearly. Figure 5.6 shows an example where two holidays are detected (Wednesday and Thursday in third and fourth weeks, which correspond to week 8 and 9 of the overall dataset; weeks 1 to 5 are not shown) as anomalous utilizations of the place covered by a particular access point (by setting a threshold at the value of -10).



(a) The utilization level (original data).



(b) The detection results after the application of a filtering technique (dU').

Figure 5.6: Detecting anomalies by comparison to behavior in previous weeks (equation 5.2: $N = 6; M = 12; SW = 2016$).

These are, however, two cases where the changes in the WiFi network load are very strong and clear, and also where this change represents a negative variation. More subtle variations and, mostly, unusually high utilization levels cannot be detected due to the high variance of the data. The choice of a correct value for the threshold is also a problem. Moreover, during seasonal time periods, such as summer holidays, the problem of setting a proper threshold becomes even more serious.

A simple observation of the data for long periods explains why one such a simple technique cannot be used. Figure 5.7 shows the observed utilization levels for one single access point during 10 "normal" weeks. Here, "normal" means that no anomalous behavior was observed during these weeks.

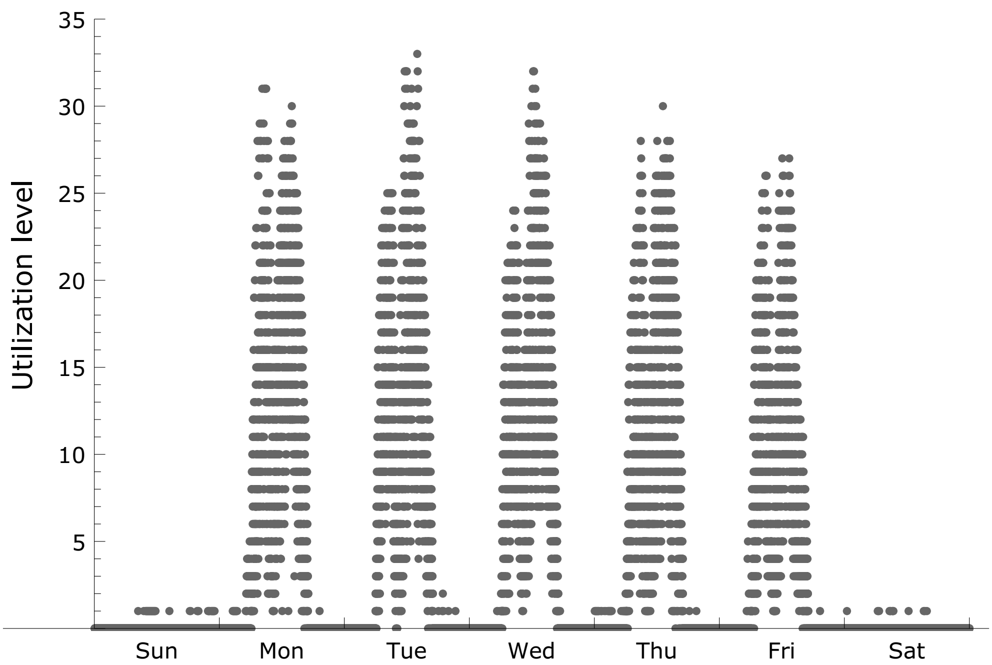


Figure 5.7: The values observed during 10 "normal" weeks of data exhibit a large variation (this picture includes a total of 20160 samples).

A great variation observed in our data suggests that normality may eventually be characterized by large variations and unpredictability, instead of a more "well behaved" and periodic function. This impelled us to exploit the concept of entropy.

5.2.2 Entropy

If one detects a large variation in the entropy of an access point, this is because a large variation occurred in the typical utilization of the corresponding place. For this, the recent entropy $H(U_M, i)$ (Equation (5.6)) has been compared to the corresponding values for the N past weeks to produce a difference metric dH_i given by:

$$dH_i = \frac{1}{\log_2[M]} \times H(U_{M,i}) - \frac{1}{\log_2[M \times N]} \times H(U_{M,N,i}) \tag{5.3}$$

where

$$U_{M,i} = \{u_{i-M-1}, ..., u_{i-1}, u_i\} \tag{5.4}$$

is a vector of the M most recent samples up to sample u_i,

$$\begin{aligned} U_{M,N,i} = \{&u_{i-M-1-N*SW}, \ldots, u_{i-N*SW}, \\ &\ldots, u_{i-M-1-SW}, u_{i-SW}\} \end{aligned} \tag{5.5}$$

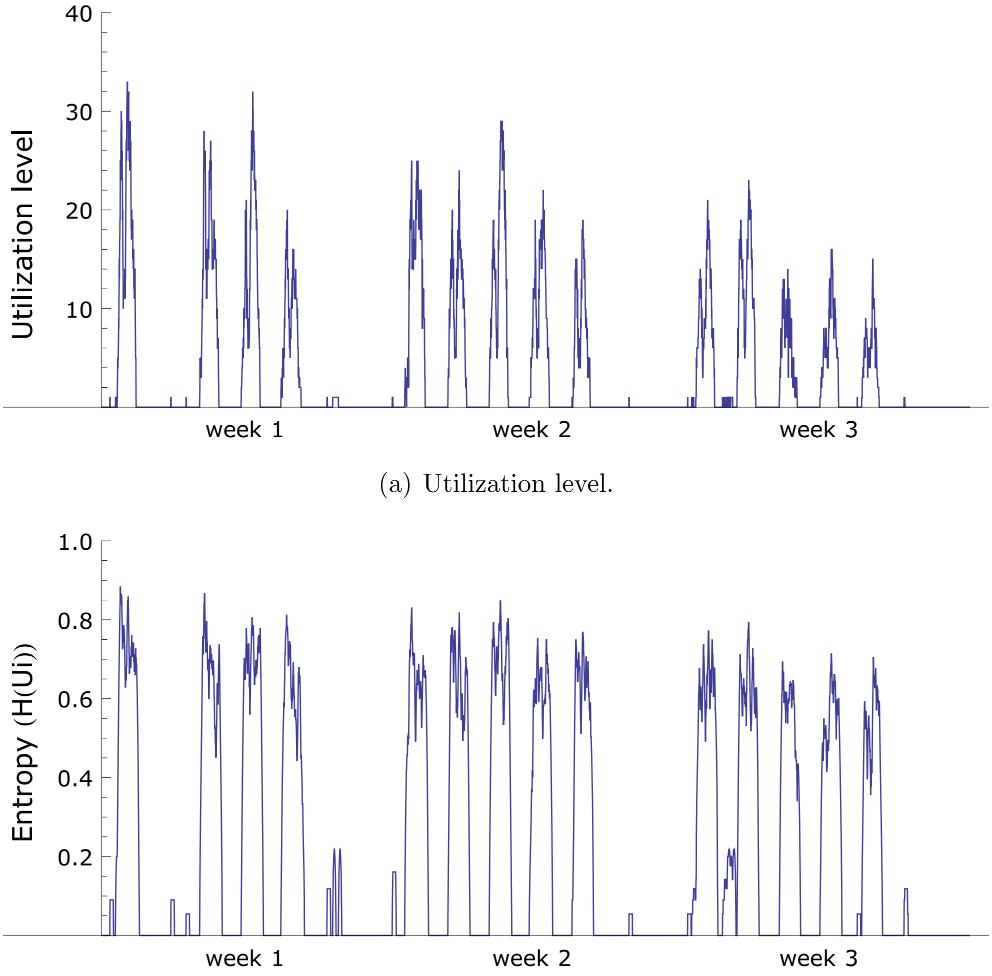
is a vector of the M corresponding samples for the N previous weeks and $H(U)$ is the Shannon entropy, given by:

$$H(U_M) = \sum_{j=1}^{S} p_{M,k} \times \log_2\left[\frac{1}{p_{M,k}}\right] \tag{5.6}$$

where $p_{M,k}$ is the *a posteriori* probability of each utilization value observed in an interval with M samples (Taub & Schilling, 1986).

The entropy value is very high during the day and approaches zero during the night when no activity is observed, as illustrated in Figure 5.8(b). On Tuesday of the first week, a holiday occurred and, consequently, the utilization level (number of connected devices) is negligible as we can notice in Figure 5.8(a). Figure 5.8(b) also shows that the entropy measure is quite insensitive to the absolute values of the utilization level. Note the differences between the utilization levels observed in week 1 and week 3, and the corresponding values of the entropy.

By comparing this entropy measure with the entropy of the corresponding values in previous weeks (Equation (5.3)), we observed that strong reductions of



(a) Utilization level.

(b) Entropy.

Figure 5.8: Entropy as a characteristic of a space (equation 5.5: $M = 24$).

the entropy (high values of the metric provided by the Equation (5.3)) actually correspond to the unusual usage of an access point, as verified during holidays. In order to deal with noise, the dH_i values were filtered using a moving average of the 6 most recent samples to produce a list of dH'_i values. The obtained results are illustrated in the Figure 5.9 for a period of 23 consecutive weeks, where the dH' metric is plotted as $dH''_i = \exp(5 \times dH'_i) - 1$ in order to make the differences more visible.

Note that six anomalies were detected. When compared to the previous re-

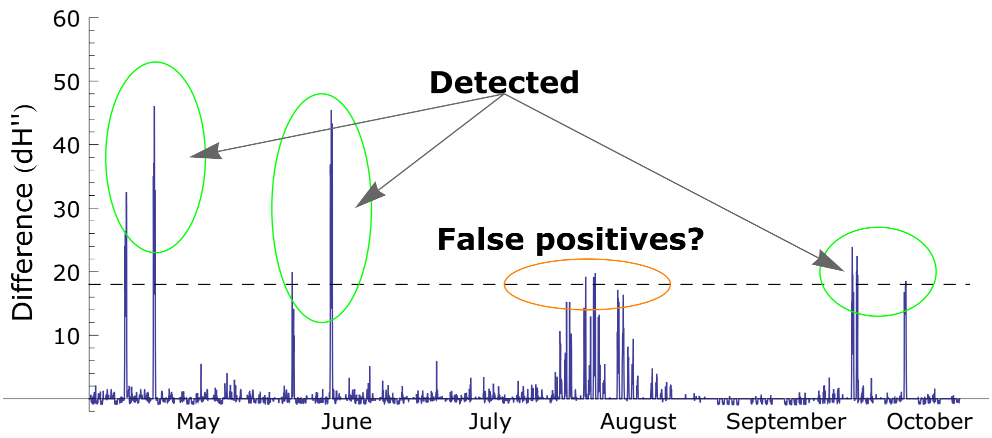


Figure 5.9: Comparing the current entropy to its previous values (equation 5.3 plotted as $dH''_i = \exp(5 \times dH'_i) - 1$: $M = 24, N = 2$).

sults depicted in Figure 5.6 (the first two are not visible in Figure 5.6, though), the entropy approach proves to be much more efficient. During this period of 23 weeks, no other anomalies occurred besides the six detected. However, the problem of setting a proper value for the threshold still persists.

Regarding the seasonal period of the summer holidays, the entropy approach also seams to perform much better. Actually, the two moments when the dH'' metric crosses the threshold do not correspond to the summer holidays weeks but to the last week before the holidays, when a strong reduction of the utilization of the network was observed, as shown in Figure 5.6(a).

Whether the detected anomalies in the beginning of the summer holidays should be considered false positives or true anomalies depends on the particular application. In our case, since we are looking for changes in the typical use of spaces, they should not be considered false positives but a sign that the space is actually being used differently. From the statistical point of view, since this behavior is expected every year at the beginning of summer holidays, these should not be considered anomalies.

Similar experiments were conducted for other access points of different classes. As an example, Figure 5.10 shows the utilization level and the entropy values for a R-Type access point for a three week period and the final result for the anomaly detection. For this class of access points, we were, actually, able to detect anomalies. However, we could not relate them to any known event, like in the case of N-Type access points where we easily detected non-working days. Only the summer holidays are easily identified as it is the time when the students massively leave the residences and go back home.

5.2.3 Pearson and Spearman correlation coefficient

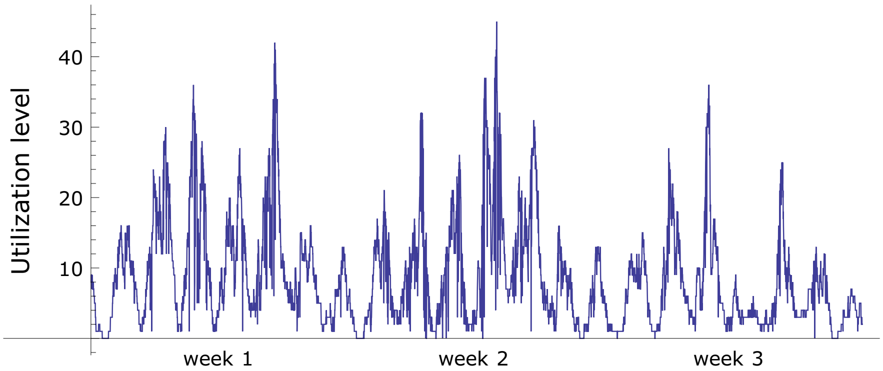
Through the analysis of our data we have found that sometimes nearby access points exhibit similar temporal behavior and usually are of the same type. This observation is in agreement with our assumption that the utilization of the WiFi network reflects some aspects of the space characteristics. Therefore, access points that serve the same space are expected to behave similarly and when this does not happen it means that an anomaly might have occurred.

The first approach here was to select a group of access points that cover the same area and measure the extent to which their utilization levels are linearly related. For this purpose, we calculated the global cross-correlation coefficient for a 4 week period between three N-Type access points (two belonging to the library and one to the studying room next to it). The results were the following: $r_{1,2} = 0.87$, $r_{2,3} = 0.94$ and $r_{1,3} = 0.85$. This shows, so far, that the above assumption of similarity between nearby access points is correct to some extent as the correlation coefficients are very high.

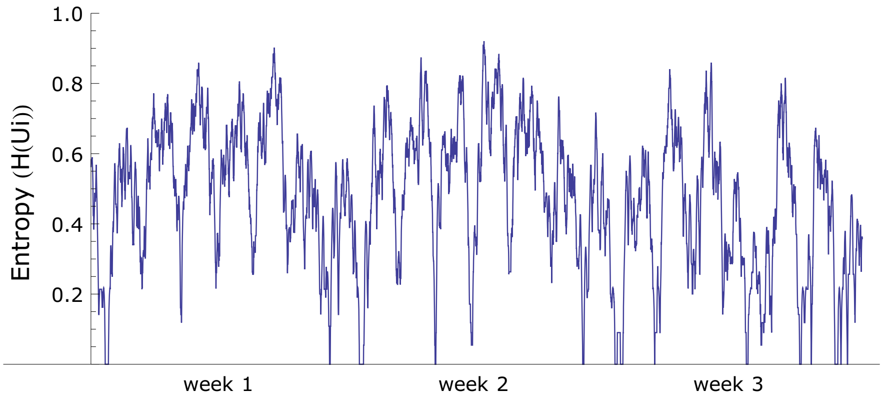
In order to track in more detail the evolution of the correlation between the utilization levels of a group of access points, we used the Pearson correlation coefficient for 12 hour long data time series. The Pearson correlation coefficient is given by

$$r_{XY} = \frac{1}{N-1} \sum_{i=1}^{N} \frac{(X_i - \overline{X})(Y_i - \overline{Y})}{s_X s_Y} \tag{5.7}$$

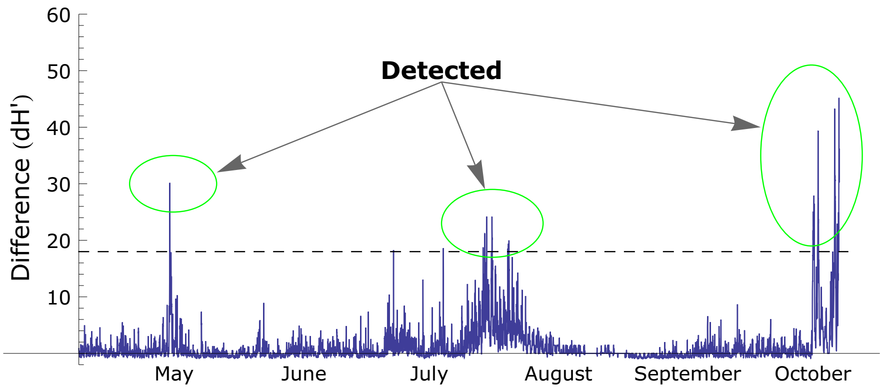
where $\overline{X}$ is the mean value of X_i (time series containing utilization values of an access point) and s_X is its standard deviation.



(a) R-Type utilization level.



(b) R-Type entropy.



(c) Anomaly detection.

Figure 5.10: Anomaly detection based on entropy for a R-Type access point.

Due to the fact that our data exhibits long periods of constant values, especially during the night, the corresponding standard deviation is zero and the Pearson coefficient cannot be calculated. In order to avoid this problem, we added to it a discrete sinusoid of a very low amplitude given by $z(t_i) = 10^{-15} sin(\frac{t_i}{10})$. Due to such a low amplitude, the addition of this function to the original time series does not affect the results when the original amplitude is different from zero.

Figure 5.11 shows utilization levels of the three observed access points. AP1 exhibits slight differences from the other two access points. For example, its utilization level is different from zero on all Saturdays except for the week 2 and also on Wednesday of the week 4.

In Figure 5.12(a) we show how AP1 is correlated with AP2 and AP3 by plotting the average of the two correlations: $r_{1,2}$ and $r_{1,3}$. The Pearson correlation coefficient exhibits abrupt changes at the beginning of each working day. This is due to the long periods of constant and close to zero values during the night followed by higher values at the beginning of the working day. The Pearson coefficient is highly sensitive to extreme data points. The length of the chosen time period (12 hours) causes some delay in the correlation plot, but a shorter time period, say of 6 or 8 hours, makes it too noisy as illustrated in Figure 5.13.

In our second approach, we tried the Spearman rank coefficient which is more resistant to extreme values. The Spearman correlation coefficient r_S is a special case of the Pearson r (Equation (5.7)) for ranked data where X_i and Y_i are the ranks of corresponding values in X and Y (Myers & Well, 2003).

This solution performs much better than the Pearson coefficient. The results are shown in Figure 5.12(b). By setting a threshold at 0.5, four anomalies were detected: three Saturdays and Wednesday of week 4, as expected. However, in week 2, there is a possible false positive which is due to the lower utilization levels in all access points.

A similar study was conducted for a group of R-Type access points. However, for this class of access points, it is difficult to identify possible anomalies as data is much more random than the one described above and there is almost

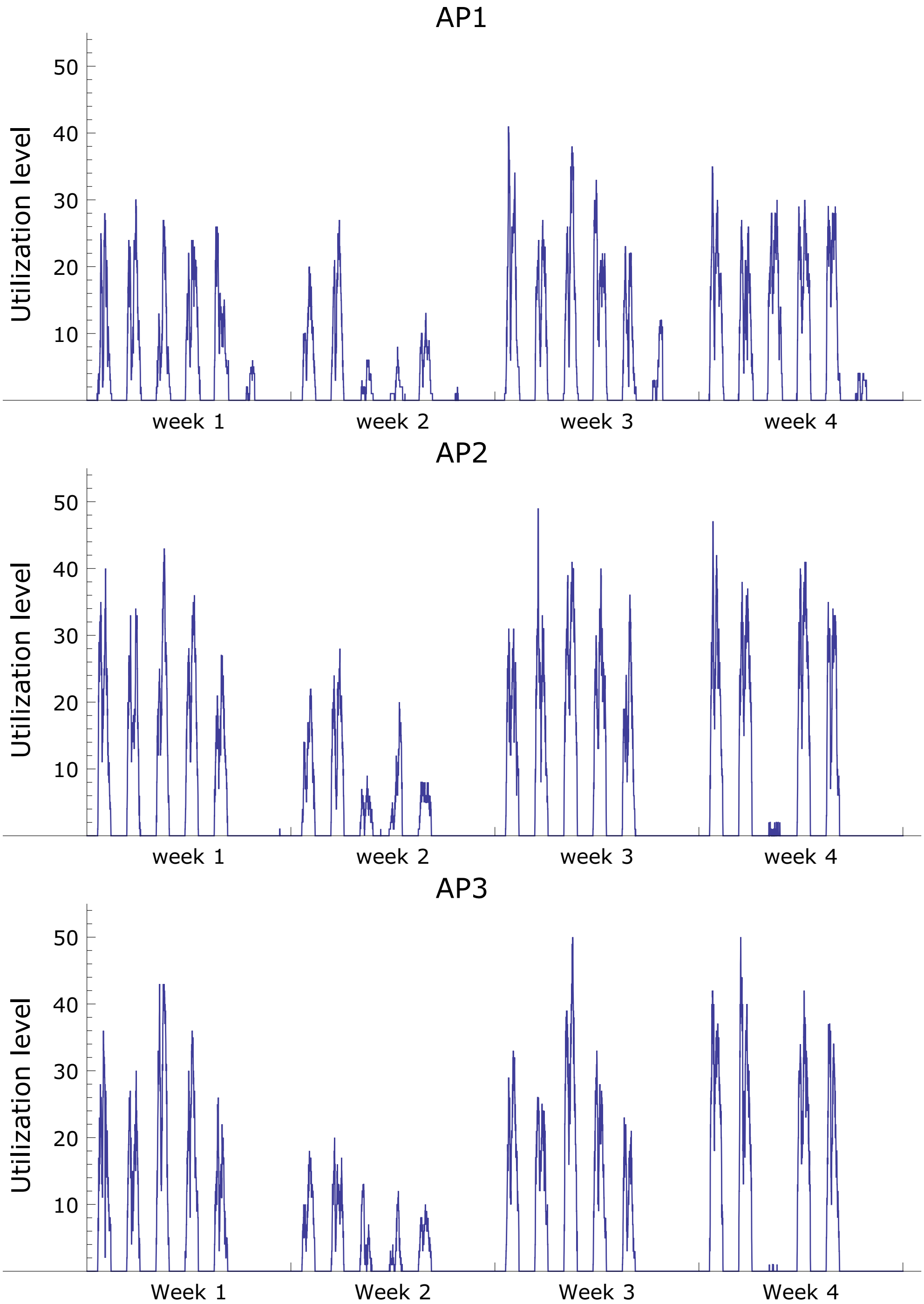
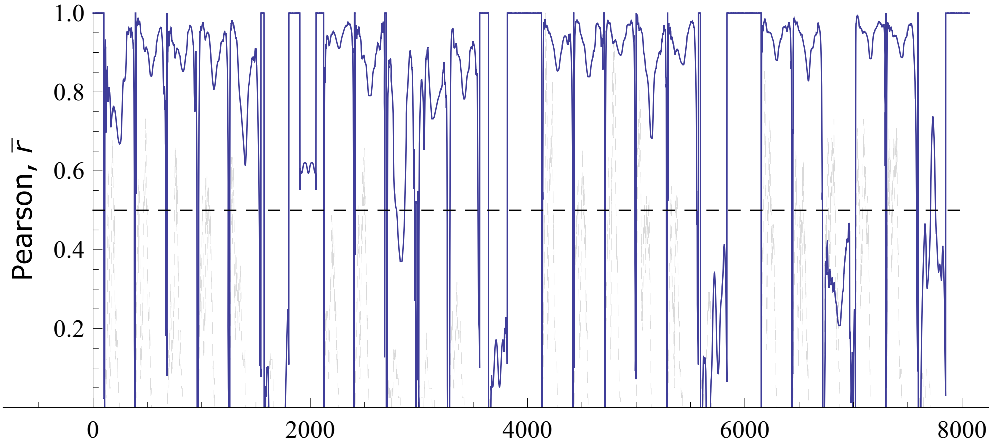
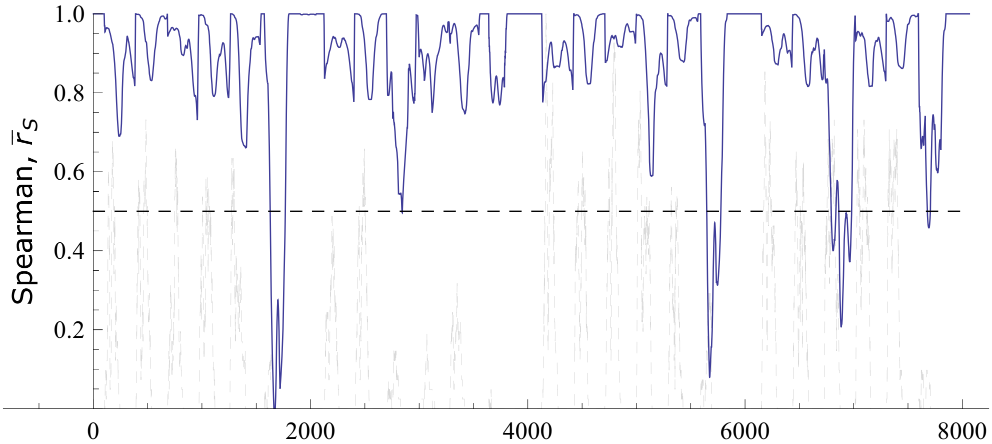


Figure 5.11: Utilization levels of three nearby N-Type access points.



(a) Pearson correlation with the 12 hours time period.



(b) Spearman rank correlation.

Figure 5.12: Mean correlation between an access point and its two neighbors.

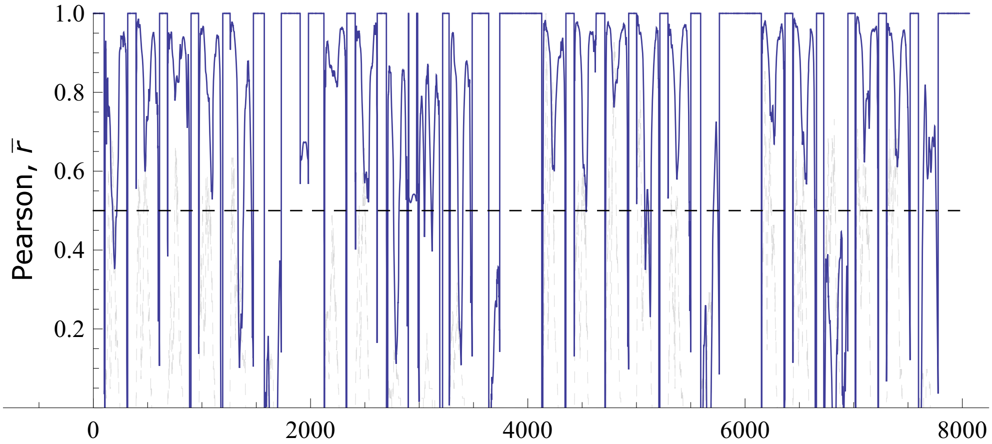


Figure 5.13: Pearson correlation with the 6 hours time period.

no distinction between holiday, weekend or working days. This is illustrated in Figures 5.14 and 5.15. Neither of the two above described approaches showed to be adequate for detecting anomalies for this class of access points.

The above analysis showed that access point that cover the same area, like those in the library, exhibit similar temporal behavior and that the correlation coefficient between them is high. However, the library area is just a specific case where the assumption that the nearby access points are similar is true. The following will show that the distance between access points is not as important as it may seemed to be in the above analysis. More than the distance, the purpose of a space dictates the level of similarity between spaces. As such, we may find in our two campuses which are around 20 km apart, individual access points and groups of access points with a very high correlation coefficient. For example, the average correlation coefficient between the campus libraries access points is 0.76.

The set of access points with known coordinates was considered in both campuses, 87 in Azurém and 112 in Gualtar. First we calculated the correlation coefficient and the distance between all pairs of access points in each campus. Figure 5.16 shows the histogram of calculated distances in each campus. The next step was to draw a plot of correlation vs. distance which is shown in Figure 5.17.

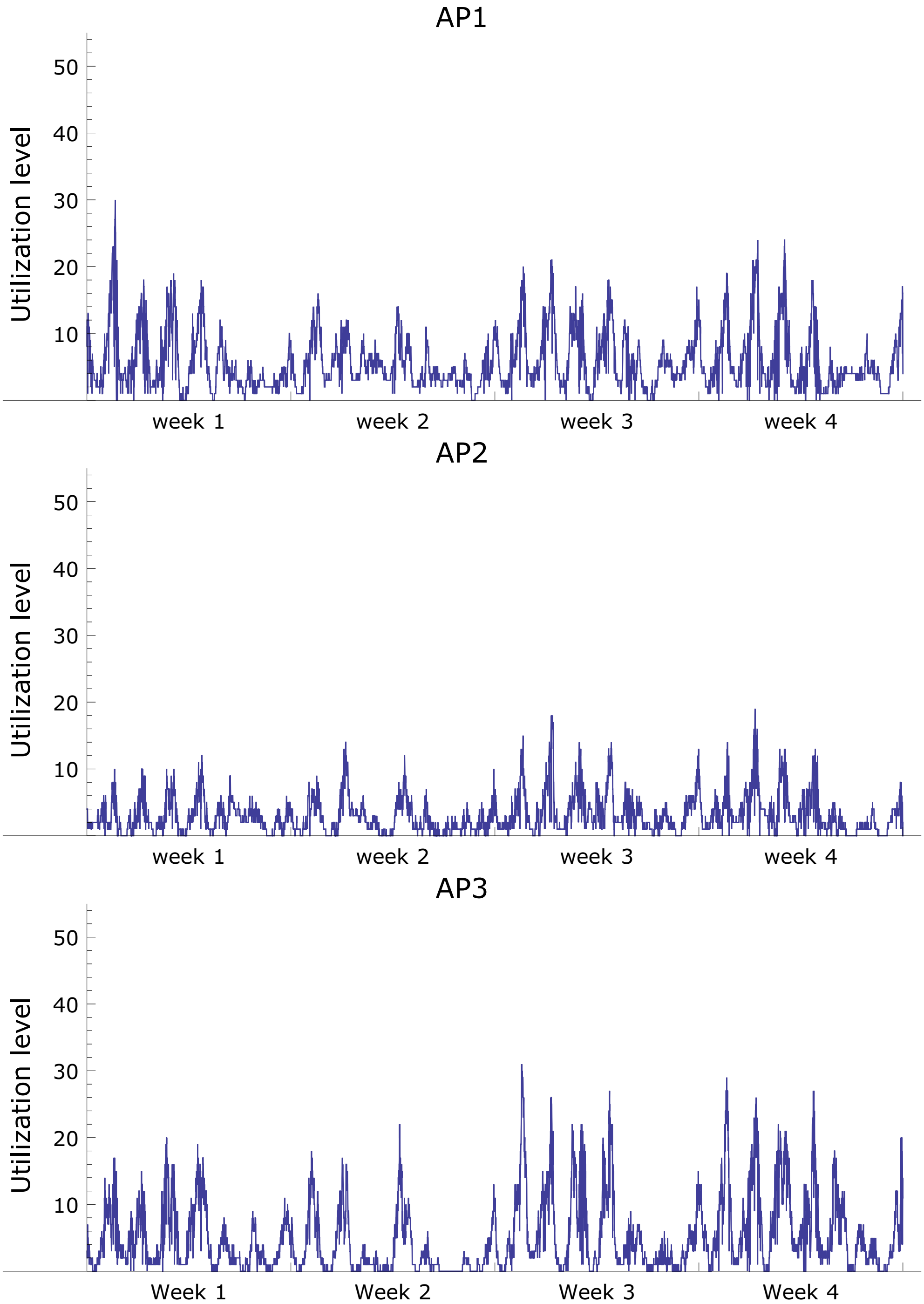
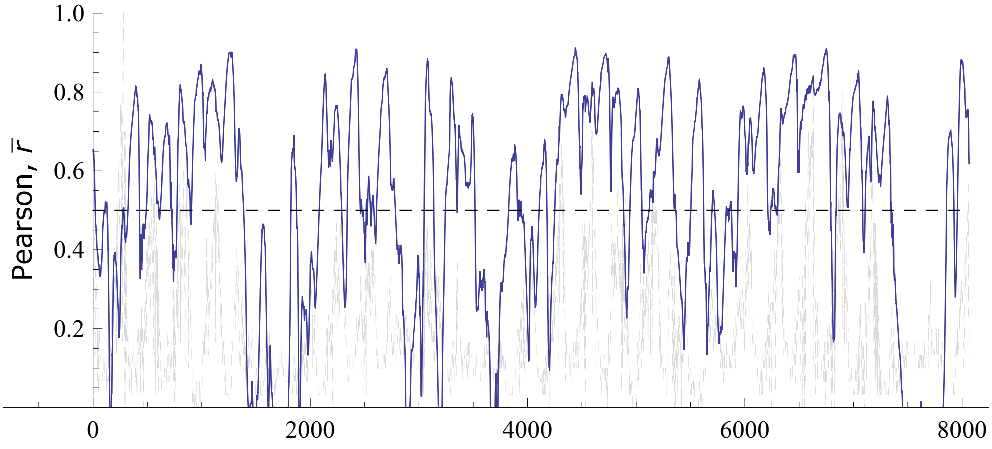
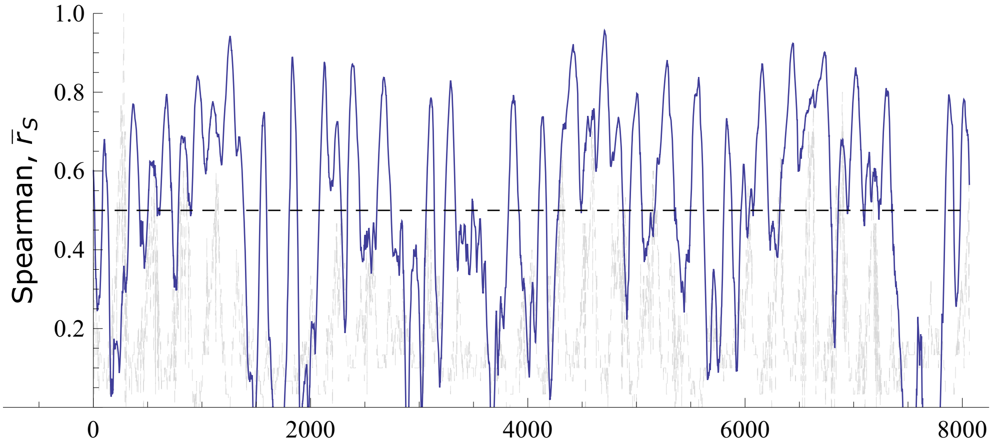


Figure 5.14: Utilization levels of three nearby R-Type access points.



(a) Pearson correlation.



(b) Spearman rank correlation.

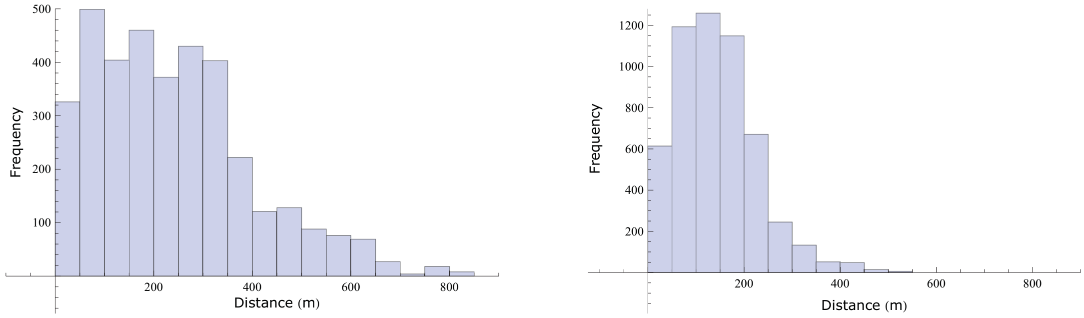
Figure 5.15: Mean correlation between an access point and its two neighbors.

5.2.4 Individual access point contribution

Definition 5.1 (Contribution of an access point). *The contribution of an access point k is the quotient between its current utilization level u_{ki} and the sum of the utilization levels of N access points in the campus*:

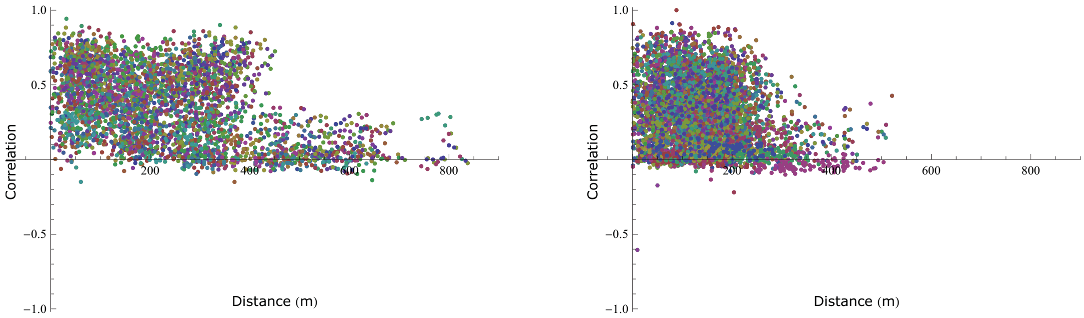
$$C_{ki} = \frac{u_{ki}}{\sum_{k=1}^{N} u_{ki}} \tag{5.8}$$

Figure 5.18 shows the contributions of two access points. Each color represents



(a) Azurém campus. (b) Gualtar campus.

Figure 5.16: Distribution of calculated distances between all pairs of access points.



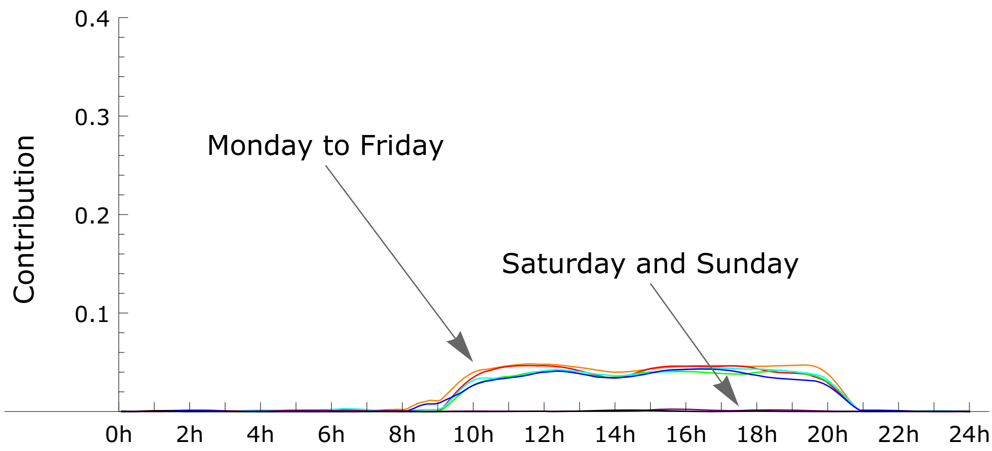
(a) Azurém campus. (b) Gualtar campus.

Figure 5.17: Correlation vs distance.

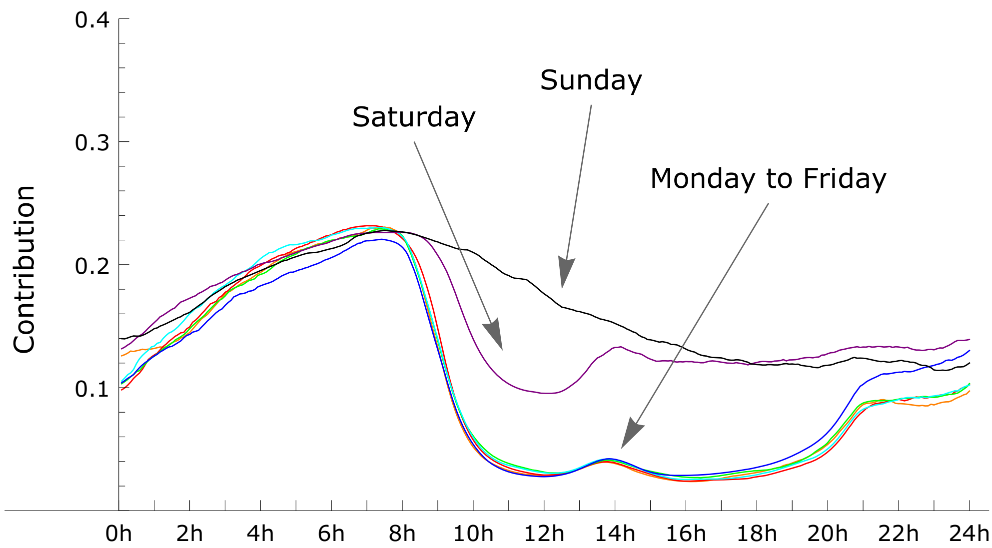
a day of the week and each daily plot corresponds to the average of 17 weeks. This means that, for example, the plot for Monday resulted from the average of 17 Mondays' contributions of an access point.

An access point contribution varies in time and in some cases exhibits a periodic behavior. For an R-Type access point it is more significant as well as noisier than for an N-Type access point. In Figure 5.18(a), between 8 a.m. and 9 p.m. we can observe an increase of the N-Type access point contribution and, in Figure 5.18(b), a decrease of the R-Type access point contribution. This is due to the movement of students from the residences to other parts of the campus.

We expected that the contribution of an access point to the overall network utilization varied according to the time of the day and whether it is a working



(a) N-Type.



(b) R-Type.

Figure 5.18: Access points average daily contributions calculated based on 17 week long data set.

day or the weekend. If this was the case, the level of the contribution parameter would allow us to identify spots of more intense network utilization which are likely to correspond to places where people usually gather to work or socialize or where some kind of event is taking place. However, it is only the case for a small subset of access points. Instead of anomaly detection, the contribution signal may be used to clarify the class of an access point and to characterize the daily time span of its utilization.

The application of anomaly detection techniques showed to be the most suitable for the N-type access points with high utilization rates. In some cases, the R-Type access points also allow for it although it showed to be much more difficult to identify possible causes for the detected anomalies. As for the C-Type and Z-Type access points, none of the experimented anomaly detection techniques showed to be particularly useful as their utilization level is very low. We believe that only a very significant change in the utilization of these access points may be successfully detected as an anomaly.

5.3 Campus Activity

In this section, we propose an application that shows the campus dynamics through visual effects over the campus map that may be available in the most popular areas of the campus or on individual mobile devices.

This application could have two different aspects. On one hand, a more informative and entertaining aspect available to users on their smartphones and laptops and, on the other hand, an aspect that aims at studying the flow of students around the campus or their favorite routes or places to dwell. This second aspect could help to improve the available services in the campus. As well, this application, if made available to visitors, may give an interesting live picture of the campus, by showing the most popular places or the most crowded or calm places as the result of the WiFi network usage. This concept can also be extended to wider urban areas as soon as sensor networks become widely deployed there to capture the human spatio-temporal behavior or the data shared by the social network users is analyzed and translated into some kind of visual representation of the space dynamics.

The application could also provide interactive time lapse looking animations with several hours or days of campus activity at different speeds so that one can perceive the changes on the timely basis.

An early prototype of this application was built based on GoogleMaps. Figure 5.19 shows a screenshot of this application. Four test access points were used. Their color on the map changes according to the number of connected users.

The four access points are represented in the world model database with a special object type (*testAP*) in order to distinguish them from the regular access points. Their attributes contain their coordinates, MAC addresses and IP addresses. There is an attribute for the number of clients that contains as its value an URL to the Location Server. This URL is used only for testing. The reply from the LS is a XML description containing a time stamp and the number of clients which is randomly generated.

The WiFiViewer application requests data from the Symbolic Contextualizer which, in turn, requests data from the Location Server. So, the reply that is received by the WiFiViewer contains the XML descriptions of the four test access points and the details returned by the LS (time stamp and the number of connecrted users) as the value of the attribute called *numberOfClients*. The WiFiViewer parses the response, and generates the colored circles over the map showing the campus area.

The future Campus Activity application should take advantage of the hotspot, groups of users and tenant level processing modules and should enable a real time visualization of the campus dynamics. Also, it may be combined with the Route Finder and with the anomaly detection to provide more complete picture of the activities happening at the campus.

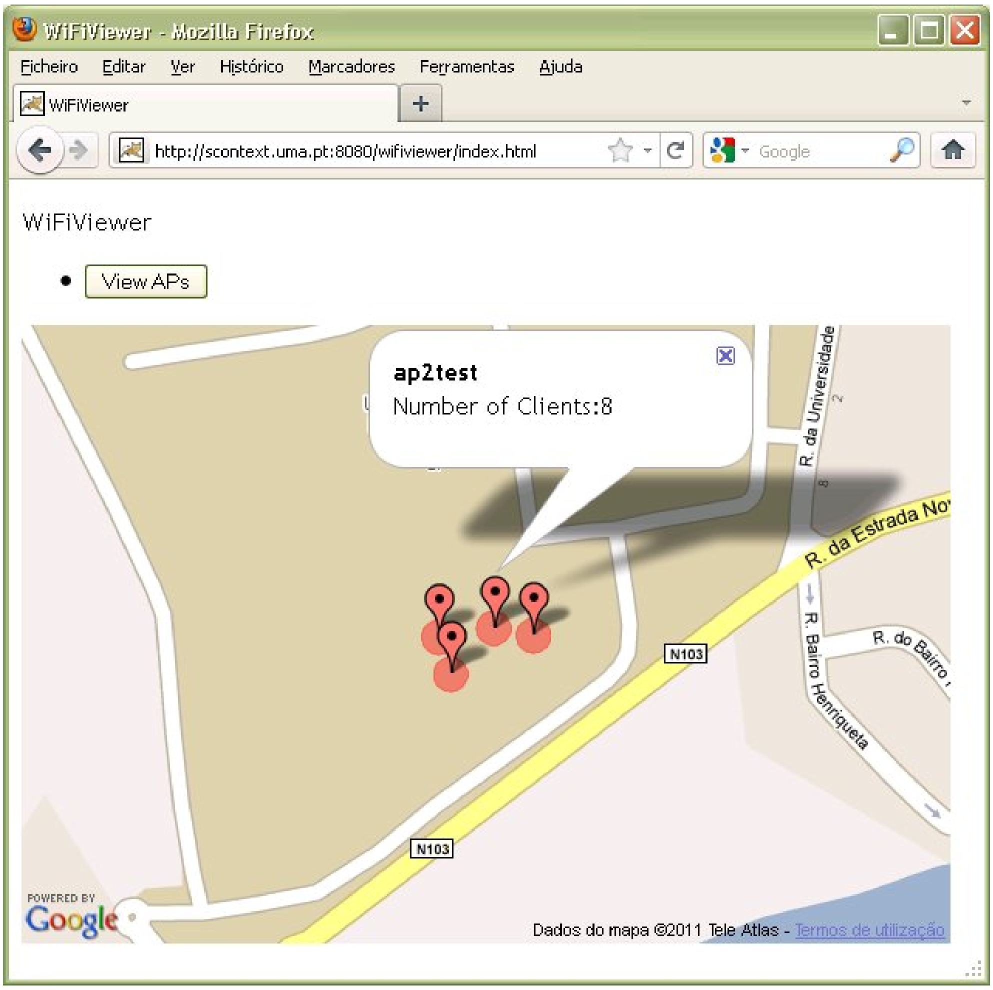


Figure 5.19: The application prototype showing four test access points. Their color changes according to the number of connected users.

5.4 Chapter summary

In this chapter, we proposed three examples of applications that may benefit from the existence of a dynamic symbolic space model. The main requirement for them is to comply to the interface format imposed by the Symbolic Contextualizer. Several processing modules can be associated to an application in order to allow for data generation according to its needs.

We briefly described an indoor routing web application supported by symbolic space models that can be created collaboratively by a group of users based on their mental space models. This approach represents an alternative to existing systems for indoor and pedestrian navigation that are based on geometric space models.

The aim of the anomaly detection study was to explore the possibilities of detecting utilization patterns that diverge from the usual ones in spaces covered by wireless networks. When the anomaly detection becomes automatic, it may be used for identification of events or happenings as well as for feeding a world model by triggering the update of the attributes of existing objects in the model or the creation of the new ones.

Data collected from a WiFi network can provide us with information about what is happening in its coverage area. This information is more accurate in places which are equipped with adequate infrastructure for working on a laptop computer and which, consequently, attract more users. The proposed anomaly detection techniques are more accurate for a subset of N-Type access points, i.e., those that have clear daily and weekly patterns and higher utilization levels. Negative variations are more easily detected than the positive ones due to great variance of the data acquired from the wireless network. Combining the described techniques is one possible future approach that might bring to light new results in this area of our research.

Finally, the Campus Activity application is meant to be a way of showing another perspective of the campus dynamics. By exploiting data from the sensor networks, new context data may be extracted from the environment and presented to the users in the form of a campus interactive map which is updated in real time.

Chapter 6

Discussion

Results! Why, man, I have gotten a lot of results.
I know several thousand things that won't work.
Thomas A. Edison

In this chapter we discuss the results of the developed project and evaluate the overall research in the light of the initially established objectives. For this purpose, first, we examine each one of the aims individually, describe the actions undertaken for their accomplishment and discuss the obtained results. Second, we state a set of requirements location models should comply with, as described by Becker & Dürr (2005) and we evaluate our model according to these requirements. We also report the results of a user study that was conducted based on the defined world model concept, the SC Space Editor, SC Browser and Route Finder applications. Finally, we describe the current state of the overall system, referring to all of its components, developed and proposed, their current state of development and performance.

6.1 Evaluation of the established objectives

In this section, we revisit the three contributions we proposed us to deliver with this project to the area of the space modeling and we describe the specific actions taken to reach each one of our objectives.

As stated in the beginning of this document, the proposed contributions of this research project were:

1. **To characterize spaces in terms of their dynamic nature and identify typical usage patterns.**

   For each space this includes the discovery of the parameters that describe it best and reflect the pulse of the space, their acquisition, processing and insertion in the model. Regarding data collection, the use of existing infrastructures was encouraged from the beginning of this project.

2. **To develop a conceptual model capable of accompanying the dynamic nature of spaces and the relations between them, being distributed and fed by a social network of users, as well as automatically by a set of processing modules.**

   Data collected from the environment needs to be processed and stored appropriately in the model. In order to achieve this, a set of processing modules was required and the model itself needed several enhancements to be able to receive dynamic data about its elements. For each user to be able to create and edit his/her own model and share it with other users, an editing application was needed and also a mechanism for connecting the different personal models among them.

3. **To develop inference algorithms that, based on information collected by the sensors, allow for inference of the higher level parameters that describe the space dynamics.**

   Not all the knowledge about a space can be stored explicitly in the model. Thus, mechanisms were needed to infer that knowledge from the existing data. The proposed processing modules and applications generate new knowledge about a space by inference, either based on the raw sensor data or, based on the already existing data in the space model or, even based on the combination of both.

6.1.1 Objective 1: characterization of spaces

A preliminary decision concerning this objective guided our research in this specific area. As only the already existing sensing infrastructures were to be used and being the WiFi network the only widely available sensing infrastructure present in all parts of our campus, the University of Minho wireless network was chosen as our primary source of context information. This choice was also based on other research projects that showed the potential of WiFi networks for space usage characterization, as already mentioned before.

On-site observations were made in a number of selected places of the Azurém campus in order to assess to what extent the data gathered from the access points reflected the real world. The results were quite encouraging as we found places where more than 70% of people used the WiFi network. As such, the data gathered from the access points were reflecting the approximate number of people present in each space.

An existing wireless network data acquisition system was used – Location Server (LS), shown in Figure 6.1. During the course of this research some changes were introduced and a second server was installed as shown in the figure. The database, "wifi_data" in Figure 6.1, was created to store data sets containing the number of connected clients at each access point and the lists of clients' physical addresses, instead of storing them to log files as in the previous version of the LS.

The "wifi_data" database consists of two tables, one for storing data about access points of the wireless network (the access point's IP and MAC addresses) and the other for storing the retrieved data, with a time stamp, the access point reference, the list of connected devices' MAC addresses and the number of connected devices). The database's entity relationship diagram is shown in Figure 6.2. This database stores data for all the available access points that belong to the University of Minho wireless network.

The wireless network data collection was initiated in 2005. A summary of all the existing data sets is given in Table 6.1. The column "# records" indicates the number of time intervals for each access point during which the number of connected users was registered. This means that the total number of records is

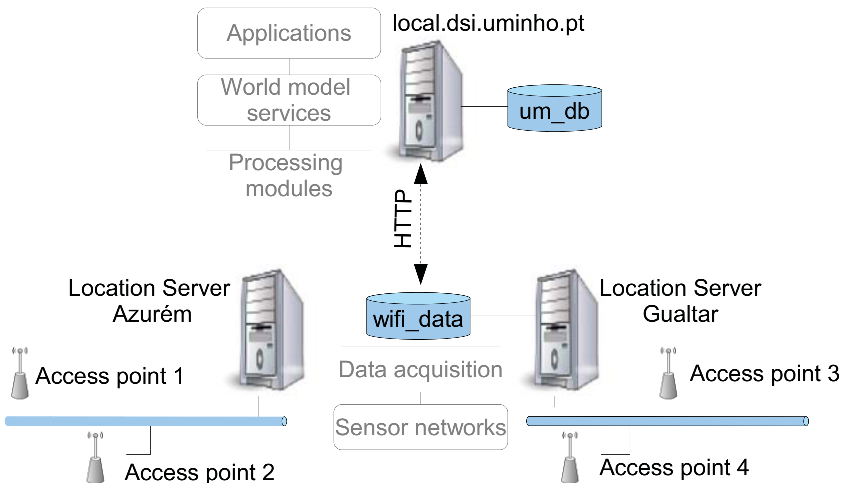


Figure 6.1: System architecture

actually the number of access points times #records.

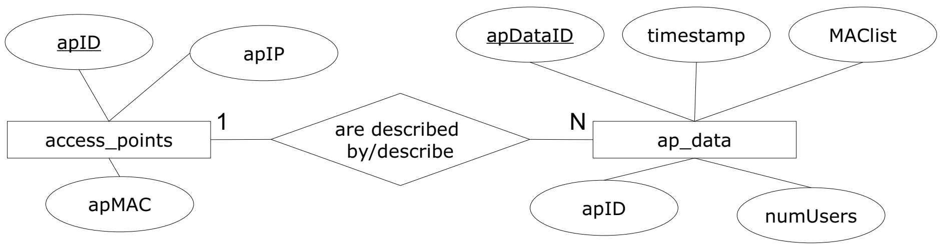


Figure 6.2: wifi_data database entity relationship diagram.

Extensive data analyses were conducted based on a part of the available data referring to around 230 access points of the University of Minho's wireless network situated in its two campuses, Azurém and Gualtar. For this purpose, Wolfram Mathematica tools were used. Several operations were applied to sets of data in order to recognize data characteristics, their periodicity, statistics and visualize the data time series. Data sets, ranging from one week to several months for a single access point or for all access points of one or both University of Minho campuses, were analyzed. As the samples are taken every 5 minutes, one day of data corresponds to 288 samples and one week to a maximum of 2016 samples.

From	To	# records	Observations
23-04-2005 20:06:44	24-04-2005 12:19:22	143	First data set for both campuses.
24-04-2005 12:29:21	21-03-2007 10:43:35	119838	8 data sets for both campuses, with small gaps.
03-04-2007 00:11:48	05-01-2008 16:20:31	77966	Individual AP analysis, classification, correlation, anomaly and hotspot detection.
05-01-2008 16:37:14	27-06-2008 11:20:50	48106	Begins the collection of lists of clients' MAC addresses for both campuses. 2008 data sets suffer from a collection failure and were not used for analysis.
16-01-2009 16:57:41	08-05-2009 15:20:32	30166	5 data sets for both campuses.
08-05-2009 15:23:06	28-05-2009 12:33:53	5662	Data set for Azurém.
08-05-2009 16:10:11	28-05-2009 12:24:44	5496	Data set for Gualtar.
28-05-2009 12:45:21	03-09-2009 11:38:22	25983	Last 2 data sets collected globally for both campuses.
03-09-2009 11:56:15	06-10-2009 22:03:41	9548	Data set for Azurém.
03-09-2009 14:13:23	19-11-2010 17:05:30	105980	Data sets for Gualtar.
26-10-2009 17:12:16	17-11-2009 17:45:30	6289	Data set for Azurém used for the detection of tenants and groups.
17-11-2009 17:53:18	19-11-2010 03:13:36	103992	Data sets for Azurém.
02-11-2010 11:08:07	30-03-2011 15:55:08	39315	Data sets for Gualtar.
29-04-2011 17:51:30	30-07-2011 13:35:02	25947	Data sets for Azurém.
02-05-2011 18:24:09	30-07-2011 13:31:30	24545	Data sets for Gualtar.

Table 6.1: WiFi network data collection milestones and existing datasets.

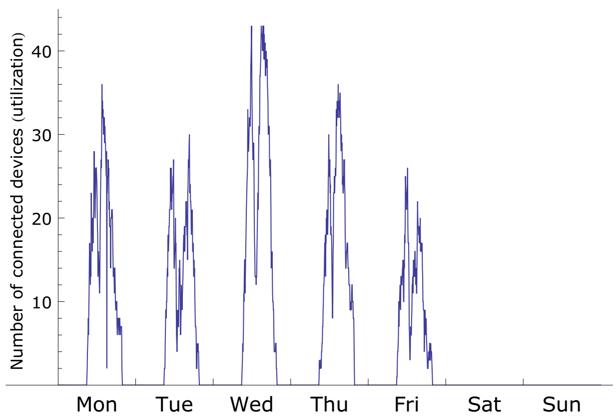


Figure 6.3: One week raw data of an access point

However, due to imperfections of the acquisition process, the real number of samples per day/week may vary. These imperfections are mainly due to server's clock inaccuracy and also to network or energy failures. Figure 6.3 shows an example of an access point raw data during one week. For a considerable number of access points, from Monday to Friday the utilization is more intense during the day and very weak or completely non-existent during the night and the weekend.

At the beginning of our analysis (2007 data), the utilization rate (as defined in Section 4.5.1) of the WiFi network was fairly low, 25% on average. Between 2007 and 2011 the growth has not been very significant, but with the increase of the number of WiFi enabled smartphones at the campus, it is expected to be more so in the near future. There still seems to exist a considerable number of underutilized access points. Table 6.2 shows the numbers for 2011 and Figure 6.4 illustrates the evolution from 2007 to 2011.

Utilization rate	% of access points
More than 50%	18%
Between 40% and 50%	8%
Between 30% and 40%	15%
Less than 30%	58%
Average utilization rate	29%

Table 6.2: WiFi network utilization rate in 2011.

Figure 6.5 shows the daily variation of the average number of users of the

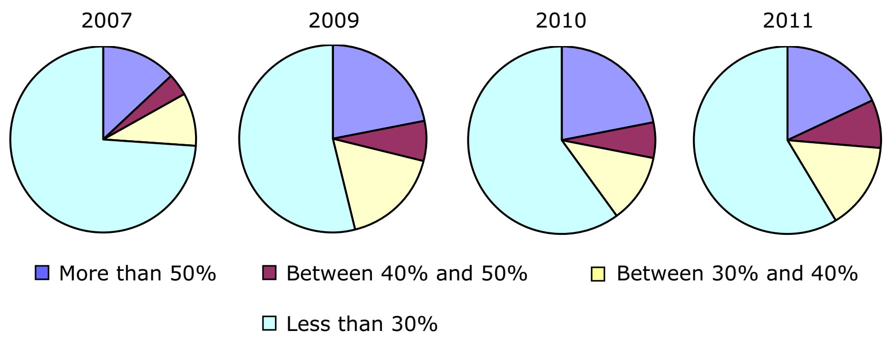


Figure 6.4: Percentage of access points with different utilization rates during the observed period from 2007 to 2011.

observed WiFi network. It shows data for both campuses in two different years. The figure illustrates how the average number of the WiFi network users grew between 2007 and 2010 in both campuses.

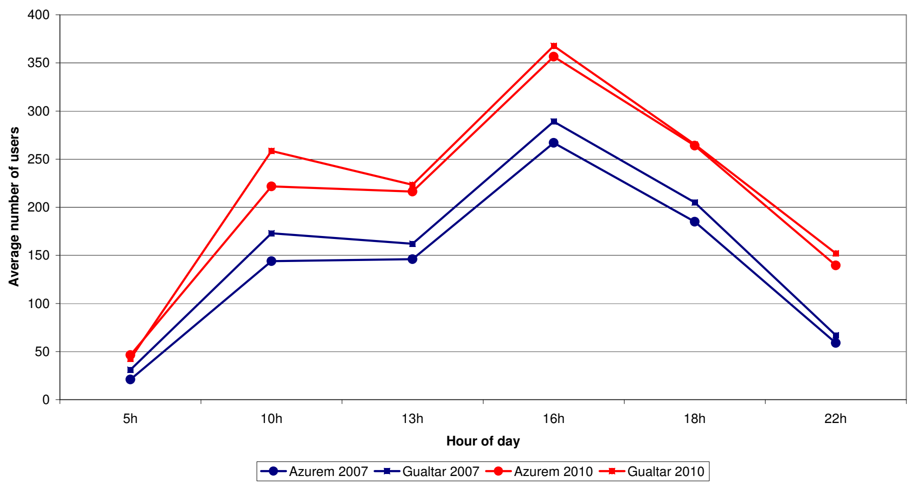
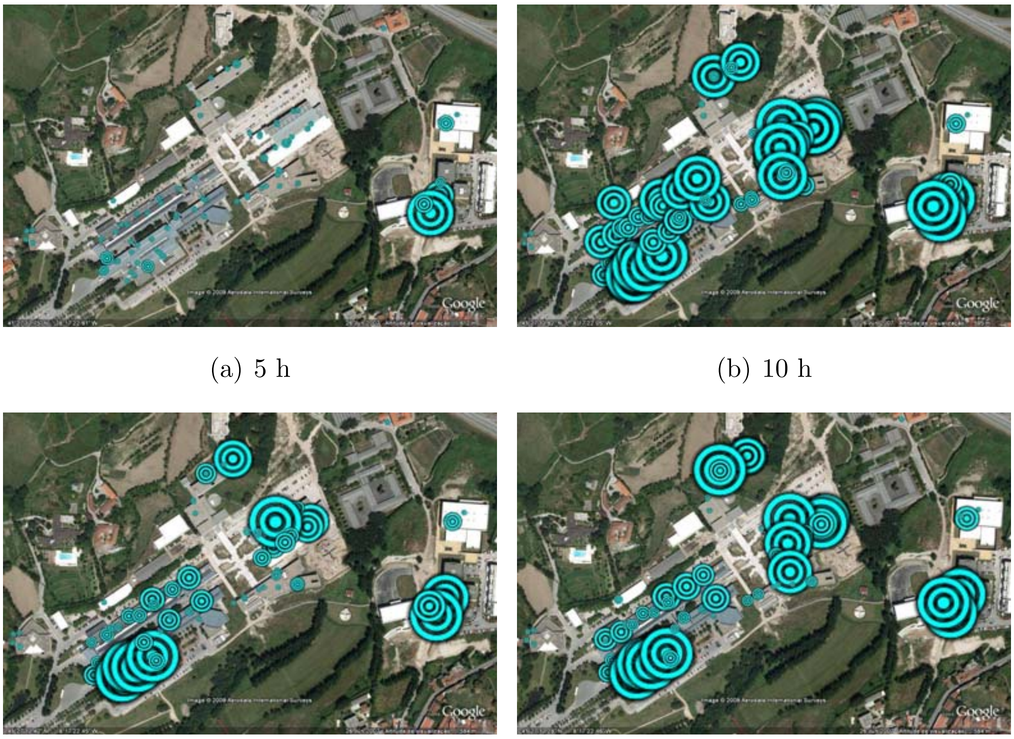


Figure 6.5: Average number of users daily variation in 2007 and 2010 for the Azurém campus and the Gualtar campus.

Figures 6.6 and 6.7 show another perspective of our data. The access points were observed at different hours of the day in order to visualize the most popular areas in the campus. All the calculations were made in Mathematica and the results were exported to KML files for visualization in Google Earth. Based on these results the class of an access point was defined as a function of its utilization rate and the average number of users.



(a) 5 h (b) 10 h

(c) 13 h (d) 18 h

Figure 6.6: Average number of users at the Azurém campus

We identified typical usage patterns based on the number of users (classification of access points). We were also able to identify places with similar characteristics (classification and correlation of access points). Table 6.3 shows the percentage of access points in each class for data ranging from May to July 2011. An extensive set of 67 additional access points was installed in the Residences at the Azurém campus during the academic year 2010/2011. Although their data was not accessible to the Location Server, this reflected in a significantly lower utilization level of the four existing access points. Due to this fact, no R-Type access points were detected in the observed period. This suggests that the criteria used for the access point classification could include more details besides the utilization rate and the average number of users. It also shows that the identifi-

(a) 5 h

(b) 10 h

(c) 13 h

(d) 18 h

Figure 6.7: Utilization rate daily changes at the Azurém campus

cation of a type tied to a location, like residential type, is not the most suitable solution.

AP Type	%
C-Type	5%
N-Type	48%
R-Type	0%
Z-Type	47%

Table 6.3: Percentage of access points in each class during 2011.

By combining the results of the classification of the access points and the correlation between them, we defined a way of creating new objects in our model – the hotspots. These objects may comprise the coverage areas of several access points and represent the most popular places at the campus. Another way of characterizing spaces was achieved through the definition of tenant levels based on the frequent visitor monitoring. Finally, we were able to detect groups of users and to identify places where these groups emerge which also contributes to the space characterization.

Several anomaly detection techniques were studied as described in detail in Section 5.2. The main motivation for this type of analysis was the idea of creating a visualization application that would allow to see in real time what was the state of each area in the campus. Some type of alert should be given when the present characteristics of a space exhibit significant differences from its usual or normal state. The implementation of this application was not achieved, but the proposed anomaly detection techniques may serve as a basis for a future development.

As a result of the data analysis conducted in order to achieve this objective we learned that a great variety of context data can be extracted from a single source which, in our case, is a WiFi network. We gained a better insight into the dynamics of a University campus and its wireless network utilization. We found that the University campuses are highly dynamic environments and that this characteristic is strongly reflected in the usage of the WiFi network.

A step towards our second objective was also achieved during the development of the above described actions as they resulted in a set of processing modules, each of which processes a different aspect of the WiFi network data and formats it adequately for the insertion in the space model, acting, thus, as the main source of the model dynamics.

6.1.2 Objective 2: the development of a dynamic symbolic world model

Researchers in the area of context-awareness do not agree on the role and the relevance of the world models. On one hand, the members of the Nexus research team argue that the location models are crucial for the context-aware applications (Lange *et al.*, 2009) and defend a global platform for context models – the World Wide Space. On the other hand, Waller & Johnston (2009) argue that world models distract the user from his current task and that the context data should be available directly in the space and that, the user should be aware of the available actions in each space without having to read the world model directly. As for us, we believe that the user does not necessarily have to interact directly with the world model and still can take advantage of its existence while it runs discreetly in the background.

We defend that the world models have an important role to play in the area of context-awareness as they provide localization services and descriptions of spaces which are useful for context-aware applications. Also, the existence of a world model allows for keeping a history record of the changes that occur in a space. Consequently, the space characteristics may be further analyzed based on these data.

The main achievement concerning this objective is a functional dynamic symbolic model, fed by a set of processing modules, capable of keeping a history record of the changes that occur in the modeled objects and their relations.

With the idea of a dynamic symbolic space model in mind, we performed a set of tasks that guided us towards it. The first step was to, based on an existing local symbolic space model, implement a new one representing a different area. This allowed us to get a deeper knowledge of the internal structure of the model and to learn all the characteristics of its interface functions. From there we went a step forward and aimed for the development of the Route Finder application. The Route Finder application allowed us to identify the strengths and the weaknesses of the existing model and to make changes to improve it.

The first two improvements of our symbolic model were the development of an algorithm for handling the symmetric relations and the introduction of relation attributes. The symmetric relations are relations representing the existence of a path between two objects, A and B, that is passable in both directions, from A to B and from B to A. With the symmetry of relations, only one relation needs to be created in the model when a two way path is available between two places. An example of this relations is the relation *Is_Accessible_From* which is, precisely, the one used by the routing application to find the shortest path between the origin and the destination objects. Other examples are: *Is_Next_To*, *Is_Near* and *Is_Similar_To*.

As a result of the analysis of WiFi data, a set of processing modules, which feed our model, emerged. In parallel, we worked on the development of a simple user interface for the manual model editing. As the processing modules were being developed and started to produce new data for the model, and for the purpose of keeping a history of our data, time stamps were introduced in the structure of object attributes and relation attributes. If history data was not kept, all the data generated by the processing modules would be wasted. There is an alternative solution to this in which applications are those that keep the historical data. However, while these applications are not developed, history records integrated in the model seemed to be a better option.

During the process of development of our model, new functions were added to the original Symbolic Contextualizer's interface. An additional set of queries was implemented: search by an object type (f2_4), search for all the existing types of objects and their frequencies (f2_5), search by a relation name (f2_6) and search by an object's author (f2_7). The insertion functions were also added in order to allow manual and automatic creation of the elements in the model: insert new object (f3_1), insert new object attribute (f3_2), insert new relation (f3_3) and insert new relation attribute (f3_4). Functions for the removal of a single relation (f5_3a) and the entire set of its attributes (f5_4b) were also implemented. More details about all of these functions are provided in Appendix A.

The running processing modules insert automatically new object attributes, relations and relation attributes. A processing module that inserts new objects is still awaiting to be implemented effectively. Two of a total of six projected processing modules were developed and tested. The remaining four exist as prototypes in Mathematica notebooks.

The SC Space Editor application was developed for the manual insertion of data in the model. Although it was developed for demonstration purposes, it allows for a simple insertion of new elements in the model, except for relation attributes. However, no social network using and feeding the model was created, because we invested more time and effort in the automatic insertion and updating than the manual one. As such, the model was not publicized and the creation of a social network was not initiated.

6.1.3 Objective 3: the development of the inference algorithms

The space dynamics as envisioned through the data acquired from the wireless network is closely related to people and their activities in the network's coverage area. All the parameters that are used for an access point usage pattern definition are related to people that visit a particular space and connect to the wireless network.

The main achievement in the scope of this objective was the finding of so many different details about a space that can be extracted from data as simple as the number of devices connected to an access point and the corresponding lists of MAC addresses.

All the proposed processing modules perform inference algorithms based, either on raw sensor data, or data from the model. Two types of raw data were used: the number of users connected to an access point and the list of MAC addresses of the detected users.

Based on raw data containing the number of users connected to each access point, we infer the access point classes and insert the results in the model – three attributes (containing values of average number of users, utilization rate and class) describing objects of type *access point*. Based on the same data we also infer the level of correlation between access points and insert the results in the model – a relation *Is_Similar_To* and its attribute containing the correlation coefficient value. These data containing the number of users per access point are also used by the proposed anomaly detection processing modules. The objective of these modules is to infer how different from normal is the current state of a space.

The second type of raw sensor data we use to infer characteristics of a space, consists of lists containing user MAC addresses. These data are used to obtain the tenant levels of a space and to detect the presence of groups. The tenant level processing module inserts two attributes that further characterize objects of type *access point* or *hotspot*, one for the long term and one for the short term tenant level. The groups processing module is responsible for the creation of new objects of type *group* and relations *Is_In_Range_Of* between groups and access points or hotspots. Additionally, it may insert object attributes that store the group dimension, a list of visited access points, etc.

Based on the already existing data from the model, the hotspots processing module infers the existence of popular areas in the campus based on the results inserted into the model by the classification and the correlation module. It inserts a new object in the model of type *hotspot* and a set of relations *Is_Served_By* with all the access points that participated in the creation of this object.

6.2 General world model evaluation

In the literature, location models are identified as sources of context data about located objects, like positions, distances and ranges. These models may be of different types as described in Chapter 2. Symbolic, geometric and hybrid are the main categories. Whichever the type of the location model, there is a set of requirements that it has to fulfil in order to be useful for the context-aware applications.

According to Becker & Dürr (2005), the basic functional requirements for location models are:

1. Provide and track objects positions – geometric or symbolic coordinates have to be associated to each object;

2. Provide distance function – distance can be the direct physical length between the two positions or the length of the road to travel between them, but it also may be the number of hops in a graph location model;

3. Implement topological relations – spatial containment relation allows for range queries while spatial connection provides data needed for the navigation services;

4. Provide object's orientation (optional) – in order to complement the position of the object, horizontal or vertical orientation may be needed.

The same authors also alert to the fact that these requirements need to be fulfilled without exceeding the minimal modeling effort. According to them, the modeling effort is influenced by the accuracy, the level of detail and the scope.

Given the above requirements, we will now evaluate our model in the light of each one of them. Also, we shall refer to the characteristics highlighted by the authors for the graph-based symbolic models in order to asses the characteristics of our model.

1. Provide and track objects positions – In our model, symbolic names are given to objects and relations are established to localize them relative to other objects in the model (e.g. *Is_In*, *Is_Next_To*). The self-location is enabled by giving the objects their common names. Besides these symbolic coordinates it is possible to insert geographic or geometric coordinates as object attributes;

2. Provide distance function – In our model, there are relations which may give us a notion of distance: *Is_Near*, *Is_Next_To*, *Is_Accessible_From*. Additionally, the exact value for a distance between two objects may be added as an attribute of any of these relations. Also, if the geometric coordinates are available as object attributes, the Euclidean distance may be calculated;

3. Implement topological relations – Spatial containment relation in our model is implemented through the relation *Is_In* and the spatial connection by the relation *Is_Accessible_From*. As explained before, there are inference algorithms associated to their types (transitive and symmetric, respectively) which allow for obtaining the related objects. Besides these relations, it is possible to add other relations as needed;

4. Object's orientation (optional) – In our model no special attention was given to this, but it can also be defined as an object attribute.

Now we compare the properties of graph-based symbolic models according to Becker & Dürr (2005) with those of the model we propose. As illustrated in Table 6.4, our model brings several improvements to the basic graph-based symbolic world model. This is due mainly to the different types of relations between objects that may be defined. As there may be several relations connecting the objects in the model, our model may be seen as a set of superposed graphs. As such, we can divide it into a set of graphs based on the relation names. For instance, a graph containing objects connected by the *Is_In* relation shows the model from the containment perspective. A graph containing the objects connected by the *Is_Accessible_From* relations shows the connection perspective and so on.

The proposed model provides a solid support for the connection and containment relations. The first one is based on the *Is_Accessible_From* relation which represents the existence of a path between two objects. The second one relies on the *Is_In* relation. This relation also allows the hierarchy of the model to be observed. In the present implementation there is no explicit support for range queries. However, one may be implemented based on the relations such as *Is_Next_To* where the range is calculated by the number of hops.

The modeling effort, as mentioned before, depends on various factors (accuracy, level of detail and scope). We consider that our model has the medium modeling effort as it showed to be quite fast and easy to create a local model using our SC Space Editor. However, for large areas with a huge number of sensors feeding the model in real time and with an increased level of detail, the modeling effort becomes higher. In that case, the best way to deal with it is the "divide to conquer" approach, this is, to create a federation of smaller local models as proposed in Section 3.1.2.

From the above mentioned characteristics of our model, we may consider that it is a good alternative to the basic graph-based symbolic model. It is also dynamic as it allows for real time updating and creation of its elements and relations. Being symbolic makes it closer to our human mental models of space as studied in psychology. Similarly to Nexus, but in a smaller scale, we may provide a platform that supports several local models by interconnecting their

Properties	Graph Based	Our model
Distance support	Good to very good	Good
Connected to relation support	Yes	Good to very good
Containment relation support	Limited	Good to very good
Position Query	Good	Good
Range Query	Basic	Basic
Nearest Neighbor Query	Good	Good to very good
Modeling effort	Medium	Medium to large

Table 6.4: Comparison of the properties of the graph based symbolic world models according to (Becker & Dürr, 2005) and the model we have developed.

Symbolic Contextualizers and providing a mechanism for global queries. Also we may consider the development of a compatibility layer to connect our model to the World Wide Space.

6.2.1 User Study

A small user study was conducted aiming to get feedback from users about the world model concept, its creation and edition with the SC Space Editor, its utilization for pedestrian navigation assistance provided by the Route Finder application and browsing with the SC Browser. Here we describe the process and highlight the main results. The details are given in Appendix B.

A group of 14 volunteers received a list of tasks to execute during one week time. A user's manual was provided describing the world model concept and structure and the basics about the usage of applications – SC Browser, SC Space Editor and Route Finder. A database containing around 100 objects was created to store the base model upon which the users were invited to build. The applications were deployed on a server and their links were provided in the user's manual.

First, the users were invited to read the manual in order to get familiarized with the model and the applications they were about to use. After that, they were asked to browse through the base model with the SC Browser application in order to get the idea about the already existing objects and relations. Then, they were encouraged to register in the SC Space Editor, create their own model step by step and visualize it at the end. There was an established minimum for the number of elements to be created: five objects, two object attributes and at least one *Is_In* and one *Is_Accessible_From* relation for each object. Finally, the users were asked to experiment with the Route Finder application by searching for available paths between objects created by them or by other users. After finishing all the tasks, the users were invited to fill in an on-line questionnaire related to the performed tasks.

The participants of this survey can be grouped in three main categories. There were 4 basic users, 3 intermediate and 7 advanced users. By basic users we mean that they only use specific IT applications needed for their jobs or for entertainment. The intermediate users are more experienced and more at ease with the technology. The advanced users are professionals that work with IT on a daily basis and/or teach IT in school or at the University.

The comprehension of the world model concept described in the manual was evaluated from 1 to 5 where 1 meant very difficult and 5 meant very easy. The average score was 3.9, which means that for more than 7 people it was easy to understand the described concept. When asked about the similarity between the presented model and our mental models of space, most of the participants replied that it was very similar (average 3.8).

The SC Space Editor application works as a wizard guiding the user step by step during the creation of his/her local model. Most of the participants found the steps very clear. The existence of suggestions for object types and attribute names was found to be useful by 93% of the users.

In general the SC Space Editor was qualified as an easy to use tool for object creation where the user can freely create his/her own objects or accept suggestions when appropriate. The set of available relations that may be created between objects was found to be one of the most interesting features, as pointed out by several users. However, the visualization of the created model did not show to be satisfactory for many users as the results were spread between 1 and 5. The average score is 3.4, but it is clearly one of the weaknesses of this application. The major weakness, which is the non-possibility of editing or removing the created objects, was pointed out by all the participants when they were asked to describe what they liked less or to make comments or give suggestions.

One of the participants suggested that there could be also suggestions for the relations, making it compulsory to create some relations, *Is_Accessible_From*, for example, in order to facilitate the path searching in the Route Finder.

The minimum number of objects users were asked to create was five. Three people created less than five objects, five people created exactly five objects while all the other participants created more than five objects. The fact that the number of created objects reported in the questionnaire does not match the number of created objects in the database confirms what we said before, that many users were not completely aware of the exact model they had created. Also, it is related to the fact that the users could not edit previously created objects, and as such, they had to insert a new one in order to add more attributes or relations.

The Route Finder application was found useful by most of the participants. There was one user, though, that found it useless. This was due to the user thinking that the Route Finder would use any existing relation (*Is_In*, *Is_Next_To*, etc.) and not just one (*Is_Accessible_From*) for path searching. Most of the users replied that the Route Finder was easy to use.

In general, the Route Finder was described as a simple, useful, easy to use tool for finding paths between two objects in the model. The fact that several options are shown when the user inserts only a part of the object name, was also found interesting and helpful. The main weaknesses showed to be the lack of a model visualization tool to assist the search for objects, their location and the path viewing, the impossibility of easy and integrated browsing for more information about the objects that make up the path and not giving alternative or related paths when a direct one does not exist between the requested objects. One of the users commented that the shortest path is not always the one we look for and suggested that several possible paths could be shown and that the user should decide which one is the best for him/her.

In sum, the results of this small user study are mostly interesting and motivating, as the main objectives of the sample applications we developed during our research were fulfilled. One of the important aspects emerged in many user comments: the fact that the existence of relations is crucial for better object description and path finding. The freedom of creating one's own object was also praised as well as the diversity of relations that may be created between objects. All the participants pointed out that an integration with a geographic map would be mostly desirable and that it would attract more users to the system. Integrations with a social network, an interactive application or a game were also suggested as being useful for attracting users to the system.

6.3 Current status of development

Now we revisit the overall system architecture described earlier in Chapter 4 and shown again in Figure 6.8. We report about what are the developed components and which is their current state and version. Table 6.5 sums up the details about the current state of the system components.

In terms of data acquisition modules, we have the Location Server and the Sensor Data Service. The Location Server already existed before this project begun. It was updated to meet new requirements, as writing the acquired data to a database instead of a text file. Since May 2010, there are two machines doing this task, one collecting data from the Azurém campus and the other collecting data from the Gualtar campus. They both send their data to the same database ("wifi_data"). The Sensor Data Service was implemented to act as a mediator between the Location Server and the processing modules. It is a web service that allows the processing modules to require sets of an arbitrary length of raw data from the "wifi_data" database. The request is made by stating the initial and the final time stamp of the required data.

The access point classification and correlation processing modules are Java applications. They run periodically and use one week of data for the calculations.

The anomaly detection, the hotspots, the tenant level and the groups processing modules were all developed in Mathematica for preliminary testing, visu-

Applications	Current Version	Version Date	State
SC Space Editor	0.3	15.09.2010	Running
SC Browser	0.21	05.12.2009	Running
SC Visual Browser	0.3	05.02.2010	Not running
Route Finder	0.4	15.09.2010	Running
World Model Services	**Current Version**	**Version Date**	**State**
Symbolic Contextualizer	0.9.2	12.09.2010	Running
Symbolic Contextualizer	0.9	06.09.2010	Running
Processing modules	**Current Version**	**Version Date**	**State**
Classification	0.2	2010	Running
Correlation	0.1	2010	Running
Hotspots	n/a	2010	Mathematica notebook
Tenants	n/a	2011	Mathematica notebook
Groups	n/a	2011	Mathematica notebook
Anomaly detection	n/a	2009	Mathematica notebook
Data acquisition modules	**Current Version**	**Version Date**	**State**
Sensor Data Service	0.1	26.03.2010	Running
Location Server Azurém	3.0	03.05.2011	Running
Location Server Gualtar	3.0	03.05.2011	Running

Table 6.5: Current state of system components

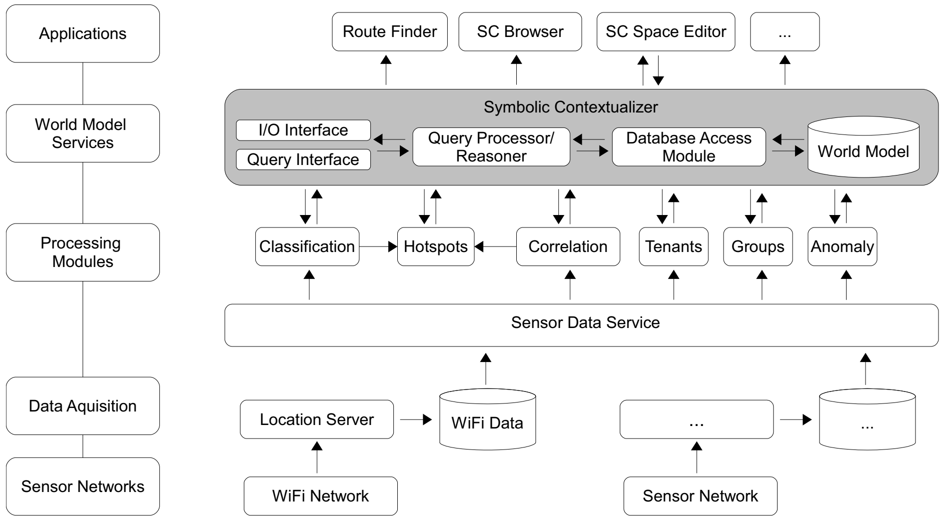


Figure 6.8: General system architecture

alization and obtaining of the first results.

Table 6.6 shows the configuration of the two servers that accommodate the existing applications, world models, processing and data acquisition modules.

	Servers	
Characteristics	**scontext.uma.pt**	**local.dsi.uminho.pt**
CPU	Pentium 4 3.2 GHz	Pentium 4 2.8 GHz
RAM	1 GB	1 GB
Operating System	Windows XP Professional, SP3	Windows XP Professional, SP3
Servlet container	Apache Tomcat 6.0	Apache Tomcat 6.0
Database Management System	MySQL 5.1.11	MySQL 5.1.48
World model database	uma_db_5	dsi_db_5
Java Runtime Environment	jdk1.6.0_20	jdk1.6.0_26
Running servlets	route, sc, scb, scse, sends	route, sc, scb, scse
Running processing modules	ClassAP, CorrelateAP	none

Table 6.6: Servers and their configurations as of October 2011

6.3.1 Performance issues

The two databases in our system and the processing modules are the critical components as far as performance is concerned.

The database that stores the model grows at the pace of the periodicity of the new data insertion by the processing modules. For example, the classification module inserts three object attributes for each object of the type *access point* each time it runs. If its period is one hour, a number of object attributes, equal to 3 times the number of access points, are added to the attributes' table (currently, $3 * 411 = 1233$ new attributes per hour). The correlation module inserts a set of relations and relation attributes for each access point each time it runs. As not all the access points are correlated with a correlation coefficient higher than 0.65, which is the established threshold, the number of inserted relations and relation attributes may theoretically vary between 0 and C_2^n, where n is the number of active access points.

The database that stores the world model was optimized by the creation of the B-tree type indexes based on the queries used by the Symbolic Contextualizer:

- Objects table: obj_name, obj_author, obj_type;
- Attributes table: obj_id, att_value, att_time;
- Relations table: rel_name, obj_dom, obj_cdom;
- Relation attributes table: rel_id, relatt_time.

After the creation of an index based on the time stamp field of the ap_data table in the "wifi_data" database storing the raw sensor data, the processing time needed for the initial acquisition of one week of data (168 hours, 2016 samples for each access point) in the classification and correlation processing modules decreased significantly (from hours to seconds). This first acquisition of one week data is the most time-consuming. However, when nothing goes wrong, it is done once and after that only small 1 hour (12 samples) long updates of data are retrieved.

6.4 Chapter summary

In this chapter we made an overview of the established objectives of this research project. The obtained results were discussed and the developed model evaluated according to the requirements announced in related research papers. The current state of the system's development and implementation was described. Although not all components were successfully implemented, we expect to have contributed for the creation of a dynamic symbolic world model capable of acquiring data from sensor networks and keeping a history of context data that helps to characterize spaces and infer new knowledge about them.

Chapter 7

Conclusions and outlook

In this final chapter we present the main conclusions for this research project. We provide a list of publications that were delivered during the unfolding of this work. Finally, we propose a list of possible future developments and challenges.

At the beginning of our research we did not have a clear view of the work that was to be done. Location models, ubiquitous computing, context-awareness were all completely new terms. However, it seems to have always been and will continue to be like this. As Albert Einstein once said: "If we knew what it was we were doing, it would not be called research, would it?".

The area of context-awareness and ubiquitous computing has been evolving at the pace of the technological development that has been extremely fast. The ever more sophisticated gadgets have been emerging, the utilization of social network applications has been highly contagious. Probably we missed some of the "fast moving trains" that might have led us to a different destination.

However, now is the time to stop, look back and recognize all the work done. We have to admit that it was not little. Although we did not achieve the perfect fulfilment of our initial objectives, we have learned a lot during the process and gained new insights, new perspectives and new skills which are already being useful in our present work and they will, certainly, continue to be so in future projects.

7.1 Concluding remarks

The initial hypothesis of this research project was stated in the following terms:

> Context-aware environments and the applications they offer to users can benefit from the existence and availability of dynamic models of the world, as they introduce new sources of knowledge about spaces where they are deployed.

The main result of this research project is a fully functional dynamic symbolic world model which consists of objects, object attributes, relations and relation attributes and which tries to mimic human mental models of space. There is a mechanism for keeping the history record in the model which allows recognizing utilization patterns and predicting trends. The model can be fed manually by the users and automatically by the processing modules.

The architecture of our system is based on the five layered architecture described by Ailisto *et al.* (2002).

The Symbolic Contextualizer is the component that supports technologically the developed world model. It provides a request-response mechanism based interface that provides access to the model.

Strong foundations for the dynamic symbolic model needed to be built. So we did not go as far as we thought we would at the beginning. The development of applications was not our priority because of the need to establish the structure of the world model and the mechanisms of WiFi data processing for feeding the model so it became dynamic at the first place. We have certainly found a wealth of new sources of context that could be extracted from the WiFi network usage data and used for different purposes such as anomaly detection, frequent visitors and groups detection and the identification of the main areas of gathering at the campus.

We used WiFi network traces as our prime source of context data as the WiFi network was the most widespread, freely available and easily accessible sensor network at our campus and also because other researchers, e.g. (Sevtsuk *et al.*, 2008), have proved that WiFi data is an interesting source of knowledge about space usage.

The existence of a set of processing modules which feed new data into the model in real time make it change according to the changes in the physical world. Six processing modules were defined: classification of access points, correlation between access points, detection of popular areas, place visitor profile, group detection and anomaly detection. The first two were implemented and are running on one of our servers.

The idea of creating an application that monitors the state of the campus areas led us to exploit different anomaly detection techniques and see how adequate they were for the type of data we had in hands. The article by Nucci & Bannerman (2007) motivated us to experiment with the entropy calculations and it turned to be the most promising technique for anomaly detection in the context of our research.

The manual creation of local world models by the users showed to be easy but not as straightforward as desirable. The lack of control over objects and of an adequate way of visualization of the model were the main reasons. However, the existing features of the SC Space Editor and SC Browser were found simple and easy to use. The way the objects are created and related was mostly praised by the users, especially due to the diversity of relations and the possibility of creating the objects and attributes without restrictions to predefined types and names. The development of the Route Finder application for pedestrian navigation based on the relations *Is_Accessible_From* allowed us to improve the model we had initially. Besides this benefit and based on the results of the user study, it showed to be an application of interest to the users. As such, it may be further explored and may have new features added in future in order to meet the requirements of the potential users.

7.2 Publications

During our research, the published papers represented important milestones as they acted as a confirmation for the path we were taking.

The following papers were published:

- K. Baras and A. Moreira, "Anomaly Detection in University Campus WiFi Zones", in *Proceedings of the 7th Workshop on Managing Ubiquitous Communications and Services*, Mannheim, 2010.
- K. Baras and A. Moreira, "Symbolic Space Modeling Based on WiFi Network Data Analysis", in 7^{th} *International Conference on Networked Sensing Systems (INSS'10)*, Kassel, 2010.
- K. Baras and A. Moreira and F. Meneses, "Navigation Based on Symbolic Space Models", *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, 2010.
- K. Baras and A. Moreira, "Groups and frequent visitors shaping the space dynamics", *The* 4^{th} *Conference on Smart Spaces ruSMART 2011*, S. Petersburg, 2011.

A demonstration of the manual creation process of our symbolic world model and its utilization for the navigation in indoor spaces was also presented at IPIN 2010 conference:

- A. Moreira and F. Meneses and K. Baras, "Self-location and navigation without maps", *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, 2010.

The following paper is under revision at the time of writing:

- K. Baras and A. Moreira, "Human Inspired Modeling of Space Dynamics", re-submitted to the *Journal of Location Based Services* on 27^{th} October 2011.

7.3 Future research and challenges

When doing research, the more we do it, the more improvements we find to do. The following are some of the possible future paths that we would like to follow in order to improve our system and deepen our research in this area.

In the domain of the world model services, there is a need for a strategy to manage the availability of objects (*private, public*) and their relation with the object's author. Also, the management of the current status of objects (*active, inactive*) needs to be implemented adequately. The integration with a geometric model is another feature that would bring new capabilities to the system. The creation of a compatibility layer in order to connect our model to the Nexus platform would bring a completely new dimension to the user experience and much more visibility to our system. After the integration in the World Wide Space platform the user should be able to access data from the federated models around the world.

For the issue of the distribution of the model there are several possible paths to follow. One is to adopt an existing strategy, the publish/subscribe mechanism used in projects that use several context providers. Another option is to wait and see what the Diaspora team is going to propose for this purpose. Or, we can come up with a solution ourselves. The management of global queries is closely related to the federation issue and needs to be handled simultaneously.

In order to enhance the user experience during the manual edition of their local model, the SC Space Editor needs to allow more control over the model elements (insert, update, delete), insertion of new relation names, insertion of relation attributes, and a more intuitive interface. An integration with a social network may also be pondered. There is already a prototype of the SC Browser application that shows the model as graph-looking mesh of objects with their names and types and the relation names, but it needs further development.

A study similar to the one we performed at the university campus could be conducted in different urban areas and additional sensor networks could be exploited. Based on the work described in (Kostakos, 2010), space syntax methodologies could be applied in order to better understand how the urban areas operate and integrate these data with our model.

The users need strong motivation to use and share their models. So a strategy for this needs to be defined probably by conducting a more thorough user study and by providing a better user experience with the editor application. Adapting our system to hand-held devices, taking advantage of their sensors and developing applications for these would also open new possibilities and attract more users.

The Route Finder application can be improved by exploiting additional relations other than *Is_ Accessible_From*. It would be also interesting if it had a connection to a positioning system like GPS or used some other smartphone resources, like reading the labels from the environment through the smartphone camera instead of requiring the user to type in the origin name.

The four proposed processing modules for the WiFi data source need to be implemented and put to work. This includes the implementation of the new object creation based on data acquired from the sensors and also based on the existing data in the model. However, these would make more sense if a set of applications that will use the available data from the model were developed.

An application that allows for real time visualization of the state of the campus or other urban space could take advantage of the data produced by the processing modules. Another application that relies on data about frequent visitors and groups and, eventually, connects with some geo-located social networking applications such as Gowalla or foursquare would also bring interesting feedback to the developed system.

Appendix A

Symbolic Contextualizer v0.9: Service interface specification

A.1 Internal architecture of the Symbolic Contextualizer

The complete system diagram in which the Symbolic Contextualizer is integrated is shown in Figure A.1. Context information about spaces is acquired by the deployed sensor networks and/or inserted manually by the user. The role of the data acquisition and processing modules is that of making the sensor networks data adequate for the model input. As well, these modules share with the users the responsibility for maintaining the model up to date. The *Sensor Data Service* is an interface through which the processing modules can obtain the datasets of variable length from the sensor networks. Each processing module makes specific calculations that can be based on present and past data of the variable length. The results produced by the processing modules are injected in the world model through its interface in real time. Like this, the model is maintained up to date according to the sensed data from the covered areas. Context-aware applications may use the space model as a source of the context information.

The Symbolic Contextualizer consists of three components that are described next.

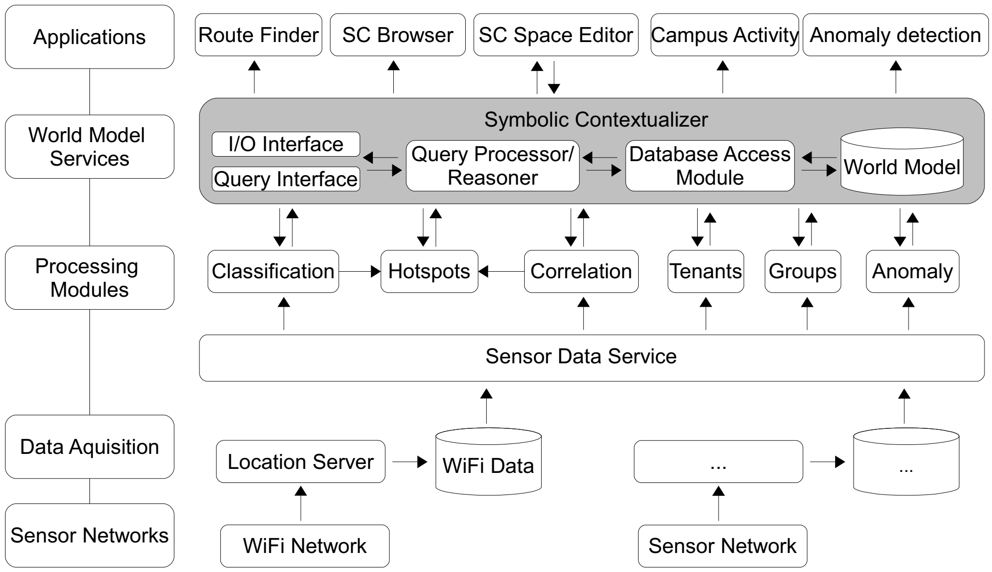


Figure A.1: System Architecture

I/O and Query Interface. The first module of the SC consists of two components, the I/O Interface and the Query Interface. The I/O interface handles insert and remove operations and the edition of the data in the model. It invokes appropriate methods from the Query Processor/Reasoner module in order to insert, remove or edit the descriptions of the symbolic objects. It replies with a XML description of the inserted or edited objects. The Query Interface accepts and validates requests for query operations on the data stored in the model. According to the received request, it calls the corresponding functions implemented in the Query Processor/Reasoner and replies with a received XML description of the found objects.

Query Processor/Reasoner. The Query Processor/Reasoner receives the requests from the I/O and Query Interface module and generates XML formatted responses. This module invokes appropriate methods from the Database Access module in order to insert, remove or obtain the descriptions of the symbolic objects needed for the response. Based on semantic rules established for each relation name and type, the Query Processor/Reasoner uses inference algorithms to decide which objects should be included in the response. All the inference algorithms are implemented in this module. It also implements the methods responsible for retrieving information from the URLs stored in the object attribute values.

Database Access module. The Database Access module abstracts the access to the relational database where the model is stored. It implements SQL statements allowing for querying and editing the model. A relational database provides an efficient way of storing and querying the model as well as its editing in runtime. More details about the database itself and its implementation are described in Section 4.1.

A.2 Service functional specification

The current version of the SC supports a set of query functions and a set of data manipulation functions. The requests for these functions may be issued by an application or by a processing module. The supported functions are organized in two categories, query functions and editing functions. To issue a request the established syntax should be followed. There exists also an optional parameter, `infer`, which can take one of four possible values (0, 1, 2 and 3). Their meaning is the following:

- `infer=0` – no inference;
- `infer=1` – transitive only;
- `infer=2` – symmetric only;
- `infer=3` – both, transitive and symmetric.

The response consists of a XML description of the requested objects if they are found or an error message, otherwise.

Function f2: Free query. Given a String (the name or the type of an object or the value of an attribute), the service returns a description of the matching object(s) if existent in the model. Partial names are allowed, such as, *dep* instead of *department* for the query parameters. This function is particularly useful when a new model is about to be explored. The response to this query, in XML format, includes a complete description of the found object(s), the names and the values of the object attributes and the set of relations with other objects. If a value of an object attribute is a URL, the service returns the response from the server as a value of the attribute.

Function f2 HTML request:

`http://host:port/sc/SCTX?function=f2&param=house&infer=0`

where `function` defines the f2 function, `param` defines the name or the type of an object or a value of one of its attributes and `infer` indicates that no inference of the relations should be performed. If objects matching the requested conditions are found, then the server replies with the full XML description of the found objects, similar to the following:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="39" type="house" name="myPlace">
<attribute-set>
<attribute name="Wall_Color" value="White"/>
<attribute name="Door_Number" value="106"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
```

```
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<error>
  <code>QP0f2</code>
  <description>Requested object not found</description>
</error>
```

Function f2_1: Get an object by its name. Given a String (the name of an object), the service returns a description of the object(s), if they exist in the model. This query is useful when we know at least a part of the name of the object. This function is used in the path finding application described in Section 5.1 to identify the origin and the destination objects. The format of the response is the same as in f2.

Function f2_1 HTML request:

```
http://host:port/sc/SCTX?function=f2_1& objname=myPlace&infer=2
```

where `function` defines the function f2_1, `objname` defines the name of the object and `infer` indicates that the inference for the symmetric relations is to be used in this example. If objects are found then a XML stream similar to the following is returned:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="39" type="house" name="myPlace">
<attribute-set>
<attribute name="Wall_Color" value="White"/>
<attribute name="Door_Number" value="106"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
```

```
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Next_To" objid="123" objhost="neigh.bor"
objport="8080" objtype="House" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Division type" value="garden fence"
time="2011-05-13 11:12:13.0"/>
</relation-attribute-set>
</relation-set>
</object>
</context>
```


In case of failure, the response is given by the following XML code:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<error>
  <code>QPf21</code>
  <description>Requested object not found</description>
</error>
```


Function f2_2: Get an object by the value of one of its attributes. Given a String (the value of an object attribute), the service returns the description of all the objects that have an attribute that matches the requested value. This query may be used to search for an object with specific characteristics. It may be used for self-location as labels may exist attached to some objects, like a door label. It allows to get the description of the object in question, e.g., a room by searching for its attribute value – the door label. The format of the response is the same as in f2.

Function f2_2 HTML request:

```
http://host:port/sc/SCTX?function=f2_2&attvalue=White&infer=0
```

where `function` defines the function f2_2, `attvalue` defines the value of an object attribute and `infer` indicates that the symmetric relations inference should be used. If objects are found then a XML stream similar to the following is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="39" type="house" name="myPlace">
<attribute-set>
<attribute name="Wall_Color" value="White"/>
<attribute name="Door_Number" value="106"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```


In case of failure, the response is given by the following XML code:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<error>
  <code>QPf22</code>
  <description>Requested object not found</description>
</error>
```

Function f2_3: Get an object by its id. Given a String (the id of an object), the service returns the description of the object, if it exists in the model. This query may be used for the expansion of the results obtained from other queries that contain references to other objects. This function is particularly useful because it allows to retrieve an object hosted on are remote server by its URI, as it happens for the co-domain object in a relation. The format of the response is the same as in f2.

Function f2_3 HTML request:

`http://host:port/sc/SCTX?function=f2_3&objid=12&infer=3`

where `function` defines the function f2_3, `objid` defines the id of an object and `infer` indicates that both inference algorithms should be used. If objects are found then a XML stream similar to this one is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="12" type="Street" name="Rua da Carreira">
<attribute-set>
<attribute name="extension" value="200m"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="13" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<error>
  <code>QPf23</code>
  <description>Requested object not found</description>
</error>
```


Function f2_4: Get an object by its type. Given a String (the type of an object), the service returns the description of all the objects, that match the requested type. This function allows to look for an object which name is not known, but we know what type of object it is, whether it is a room or a building, for instance. The format of the response is the same as in f2.

Function f2_4 HTML request:

```
http://host:port/sc/SCTX?function=f2_4&objtype=house&infer=0
```

where `function` defines the function f2_4, `objtype` defines the type of an object and `infer` indicates that no inference should be done. If objects are found then a XML stream similar to the following is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="39" type="house" name="myPlace">
<attribute-set>
<attribute name="Wall_Color" value="White"/>
<attribute name="Door_Number" value="106"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
```

```
<relation-attribute name="Entrance description" value="staircase"
time="2010-05-13 19:17:53.0"/>
</relation-attribute-set>
</relation>
<object>
<object objid="123" type="house" name="JonsPlace">
<attribute-set>
<attribute name="Wall_Color" value="Yellow"/>
<attribute name="Door_Number" value="108"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2011-02-11 12:11:56.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Next_To" objid="39" objhost="myhost"
objport="8080" objtype="House" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Division type" value="garden fence"
time="2011-05-13 11:12:13.0"/>
</relation-attribute-set>
</relation-set>
</object>
</context>
```


In case of failure, the response is given by the following XML code:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<error>
  <code>QPf24</code>
  <description>Requested object not found</description>
```

```
</error>
```

Function f2_5: Get all the existing types of objects and their frequencies. As a response to this request, the service returns a String containing a list of all existing types of objects and their frequencies. This function is used for the purpose of giving suggestions to the user during the manual insertion of new objects in the model. It does not support the "infer" parameter. If objects are found, then the response is formatted as follows:

```
["type_1", 12; "type_2", 123; ...; "type_n", 43]
```

where "type_i" indicates the type of object and the integer value indicates the number of found objects of that type.

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf25</code>
   <description>No object types found</description>
 </error>
```

Function f2_6: Get all objects having a given relation. Given a String (the name of a relation), the service returns the description of all the objects, that have the requested relation. This function is useful if we want to retrieve only one of the sub-graphs representing our model. There may be more than one relation between pairs of objects. So if we query for the *Is_In* relation, we will have a sub-graph containing a set of objects which are related among them with the *Is_In* relation. The number of vertices and edges in each sub-graph vary according to the existing relations among objects. This function is presently used by the processing module described in Section 4.5.3. The format of the response is the same as in f2.

Function f2_6 HTML request:

```
http://host:port/sc/SCTX?function=f2_6&relname=Is_Next_To&infer=0
```

where `function` defines the function f2_6, `relname` defines the name of a relation and `infer` indicates that no inference should be done. If objects are found then a XML stream similar to the following is returned:

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="123" type="house" name="JonsPlace">
<attribute-set>
<attribute name="Wall_Color" value="Yellow"/>
<attribute name="Door_Number" value="108"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2011-02-11 12:11:56.0"/>
</relation-attribute-set>
</relation>
<relation relname="Is_Next_To" objid="39" objhost="myhost"
objport="8080" objtype="House" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Division type" value="garden fence"
time="2011-05-13 11:12:13.0"/>
</relation-attribute-set>
</relation-set>
</object>
</context>
```


In case of failure, the response is given by the following XML code:


```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<error>
  <code>QPf26</code>
  <description>No objects found with the given relation name</description>
</error>
```

Function f2_7: Get all objects having the same author. Given the name of an author, the service returns the description of all the objects that have been created by the requested author. This function allows for an author to browse his/her own local model (see Section 4.2). The format of the response is the same as in f2.

Function f2_7 HTML request:

```
http://host:port/sc/SCTX?function=f2_7&objauthor=meMyself&infer=0
```

where `function` defines the function f2_7, `objauthor` defines the name of the author and `infer` indicates that no inference should be performed. If objects are found then a XML stream similar to the following is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="39" type="house" name="myPlace">
<attribute-set>
<attribute name="Wall_Color" value="White"/>
<attribute name="Door_Number" value="108"/>
</attribute-set>
<relation-set>
<relation relname="Is_In" objid="10" objhost="cityhall.cti" objport="8080"
objtype="City" reltype="transitive">
<relation-attribute-set>
</relation-attribute-set>
</relation>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2011-02-11 12:11:56.0"/>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<error>
  <code>QPf27</code>
  <description>Requested object not found</description>
</error>
```

Function f3_1: Insert an object. This function allows for creating new objects in the model. It receives a set of variables describing the object and inserts it in the database through an SQL statement. The SC SpaceEditor uses this function for manual insertion of objects. Automatic insertion of new objects is supported as well.

Function f3_1 HTML request:

```
http://host:port/sc/SCTX?function=f3_1&objname=MalinxPlace&objtype=house&
objauthor=malinx&objcdate=2011-09-22 12:02:23
```

where `function` defines the function f3_1, `objname` defines the name of the object, `objtype` defines the type of the object, `objauthor` defines the author of the object and `objcdate` indicates the creation date of the object. In the manual insertion of the objects, the object name and type are defined by the user while all the rest is filled in automatically. If objects are successfully inserted in the model then a XML stream similar to the following is returned:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="MalinxPlace">
<attribute-set>
</attribute-set>
<relation-set>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf31</code>
   <description>Insert object failed.</description>
 </error>
```

Function f3_2: Insert an object attribute. This function allows for creating new attributes describing an existing object in the model. It receives a set of variables identifying the existing object and its new attributes and inserts it in the database through an SQL statement. The SC SpaceEditor uses this function for attribute insertion during the manual edition of the model. Automatic insertion of new attributes is supported as well.

Function f3_2 HTML request:

```
http://host:port/sc/SCTX?function=f3_2&objid=123&attname=area&
attvalue=120sq.m&atttime=2011-09-22 12:03:13
```

where `function` defines the function f3_2, `objid` defines the id of the object which is going to be described by this attribute, `attname` defines the name of the attribute, `attvalue` defines the value of the attribute at the time given by `atttime` . If attributes are successfully inserted in the model then a XML stream similar to the following is returned:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace" time="2011-09-22 12:03:13">
<attribute-set>
<attribute name="area" value="120sq.m"/>
</attribute-set>
<relation-set>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf32</code>
   <description>Insert attribute failed.</description>
 </error>
```


Function f3_3: Insert a relation. This function allows for creating new relations between two existing objects in the model. It receives a set of variables identifying the involved objects and describing the new relation and inserts it in the database through an SQL statement. The SC SpaceEditor uses this function for relation insertion during the manual edition of the model. Automatic insertion of new relations is supported as well.

Function f3_3 HTML request:

```
http://host:port/sc/SCTX?function=f3_3&relname=Is_Accessible_From&
reltype=symmetric&objdom=34&objcdom=12&hostcdom=cityhall.cti&portcdom=8080
```

where `function` defines the function f3_3, `relname` defines the name of the relation, `reltype` defines the type of the relation, `objdom` defines the id of the object which owns this relation, `objcdom`, `hostcdom` and `portcdom` define the id, the host and the port of the related object.

If relations are successfully inserted in the model then a XML stream similar to the following is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
<attribute-set>
<attribute name="area" value="120sq.m" time="2011-09-22 12:03:13"/>
</attribute-set>
<relation-set>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
</relation-attribute-set>
```

```
</relation>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf33</code>
   <description>Insert relation failed.</description>
 </error>
```

Function f3_4: Insert a relation attribute. This function allows for creating new relation attributes describing the existing relations in the model. It receives a set of variables describing the relation and its attribute and inserts it in the database through an SQL statement. The SC SpaceEditor does not support yet the insertion of relation attributes. Automatic insertion of new relation attributes is supported by the processing modules.

Function f3_4 HTML request:

```
http://host:port/sc/SCTX?function=f3_4&objdom=34&relname=Is_Accessible_From
&objcdom=12&hostcdom=cityhall.cti&portcdom=8080&
relattname=Entrance description&relattvalue=staircase&
relatttime=2011-09-22 12:11:56
```

where `function` defines the function f3_4, `objdom, relname, objcdom, hostcdom` and `portcdom` define the relation which will be described by this relation attribute, `relattname` defines the name of the relation attribute, `relattvalue` contains the value of the attribute and `relatttime` the time at which it was inserted. If relation attributes are successfully inserted in the model then a XML stream similar to the following is returned:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
```

```
<attribute-set>
<attribute name="area" value="120sq.m" time="2011-09-22 12:03:13"/>
</attribute-set>
<relation-set>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="staircase"
time="2011-02-11 12:11:56.0"/>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf34</code>
   <description>Insert relation attribute failed.</description>
 </error>
```

Function f4_2: Update an object attribute. This function allows for editing the already created attributes describing an object during the manual edition of the model. It receives a set of variables identifying the object and the attribute that is to be changed and updates it in the database through an SQL statement. Future versions of the SC Space Editor will use this function for editing the already existent object attributes.

Function f4_2 HTML request:

```
http://host:port/sc/SCTX?function=f4_2&objid=123&attname=area&
attvalue=220sq.m&atttime=2011-09-23 16:03:13
```

where `function` defines the function f4_2, `objid` defines the id of the object to which the attribute belongs, `attname` defines the name of the attribute to be changed, `attvalue` defines the value of the attribute at the time given by `atttime` .

If attributes are successfully updated then a XML stream similar to the following is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
<attribute-set>
<attribute name="area" value="220sq.m" time="2011-09-23 16:03:13"/>
</attribute-set>
<relation-set>
</relation-set>
</object>
</context>
```


In case of failure, the response is given by the following XML code:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf42</code>
   <description>Insert attribute failed.</description>
 </error>
```


Function f4_3: Update a relation. This function allows for updating the already existing relations in the model. It receives a set of variables identifying the involved objects and describing the relation and updates it in the database through an SQL statement. Future versions of the SC Space Editor will use this function for editing the already existent object relations.

Function f4_3 HTML request:

```
http://host:port/sc/SCTX?function=f4_3&relname=Is_Accessible_From&
reltype=symmetric&objdom=34&objcdom=12&hostcdom=cityhall.cti&portcdom=8080
```

where `function` defines the function f4_3, `relname` defines the name of the relation, `reltype` defines the type of the relation, `objdom` defines the id of the object which owns this relation, `objcdom,` `hostcdom` and `portcdom` define the id, the host and the port of the related object.

If relations are successfully inserted in the model then a XML stream similar to the following is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
<attribute-set>
<attribute name="area" value="220sq.m" time="2011-09-23 16:03:13/>
</attribute-set>
<relation-set>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```


In case of failure, the response is given by the following XML code:


```
 <?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
  <error>
    <code>QPf43</code>
    <description>Insert relation failed.</description>
  </error>
```


Function f4_4: Update a relation attribute. This function allows for updating the already existing relation attributes in the model. It receives a set of variables describing the relation and its attribute and updates it in the database through an SQL statement. As the SC Space Editor does not support yet the insertion of relation attributes, the update is not available either.

Function f4_4 HTML request:

```
http://host:port/sc/SCTX?function=f4_4&objdom=34&relname=Is_Accessible_From
&objcdom=12&hostcdom=cityhall.cti&portcdom=8080&
relattname=Entrance description&relattvalue=approach ramp&
```

```
relatttime=2011-09-22 15:11:56
```

where `function` defines the function f4_4, `objdom`, `relname`, `objcdom`, `hostcdom` and `portcdom` define the relation which is described by this relation attribute, `relattname` defines the name of the relation attribute, `relattvalue` contains the value of the attribute and `relatttime` the time at which it was updated. If relation attributes are successfully updated in the model then a XML stream similar to the following is returned:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
<attribute-set>
<attribute name="area" value="120sq.m" time="2011-09-23 16:03:13/>
</attribute-set>
<relation-set>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="approach ramp"
time="2011-02-11 15:11:56.0"/>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf44</code>
   <description>Insert relation attribute failed.</description>
 </error>
```

Function f5_1: Delete an object. This function allows to remove an object from the model. It receives a set of variables describing the object and removes it from the database through an SQL statement. The removal of an object implies that all its attributes, relations and relation attributes are deleted as well. As such, it calls the functions f5_2b and f5_3b which will be described next. Future versions of the SC Space Editor will use this function for removing a created object from the model.

Function f5_1 HTML request:

```
http://host:port/sc/SCTX?function=f5_1&objid=34
```

where `function` defines the function f5_1 and `objid` defines the object which will be removed.

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf51</code>
   <description>Insert relation attribute failed.</description>
 </error>
```

Function f5_2a: Delete one attribute. This function allows to remove an object from the model. Future versions of the SC Space Editor will use this function for removing an existing object attribute from the model.

Function f5_2a HTML request:

```
http://host:port/sc/SCTX?function=f5_2a&attid=123
```

where `function` defines the function f5_2a and `attid` defines the object which will be removed. The removal of an object implies that all its attributes, relations and relation attributes are deleted as well.

If an object attribute is successfully removed from the model then a XML stream similar to the following is returned:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
<attribute-set>
```

```
</attribute-set>
<relation-set>
<relation relname="Is_Accessible_From" objid="12" objhost="cityhall.cti"
objport="8080" objtype="Street" reltype="symmetric">
<relation-attribute-set>
<relation-attribute name="Entrance description" value="approach ramp"
time="2011-02-11 15:11:56.0"/>
</relation-attribute-set>
</relation>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf52a</code>
   <description>Insert relation attribute failed.</description>
 </error>
```

Function f5_2b: Delete all attributes of an object. This function is used to remove all the attributes of an object that is about to be deleted. Future versions of the SC Space Editor will use this function for removing all the attributes of an object that is going to be removed from the model.

Function f5_2b HTML request:

`http://host:port/sc/SCTX?function=f5_2b&objid=34`

where `function` defines the function f5_2b and `objid` defines the object which will have all its attributes removed.

Function f5_3a: Delete one relation. This function allows to remove one relation of an object. The removal of a relation implies that all its attributes are deleted as well. As such, it calls the function f5_4b which will be described next. Future versions of the SC SpaceEditor will use this function for removing all the relations of an object that is going to be removed from the model.

Function f5_3a HTML request:

http://host:port/sc/SCTX?function=f5_3a&objdom=34&relname=Is_Accessible_From name&objcdom=12&hostcdom=cityhall.cti&portcdom=8080

where function defines the function f5_3a, objdom, relname, objcdom, hostcdom and portcdom define the relation which will be removed.

If relation attributes are successfully inserted in the model then a XML stream similar to the following is returned:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
<attribute-set>
<attribute name="area" value="120sq.m"/>
</attribute-set>
<relation-set>
</relation-set>
</object>
</context>
```

In case of failure, the response is given by the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf53a</code>
   <description>Insert relation attribute failed.</description>
 </error>
```

Function f5_4a: Delete one relation attribute. This function allows to remove one relation attribute. Future versions of the SC Space Editor will use this function for removing a single relation attribute from the model.

Function f5_4a HTML request:

http://host:port/sc/SCTX?function=f5_4a&objdom=34&relname=Is_Accessible_From name&objcdom=12&hostcdom=cityhall.cti&portcdom=8080

where function defines the function f5_4a, objdom, relname, objcdom, hostcdom and portcdom define the relation which will be removed.

If a relation attribute is successfully removed from the model then a XML stream similar to the following is returned:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<context>
<object>
<object objid="34" type="house" name="malinxPlace">
<attribute-set>
<attribute name="area" value="120sq.m"/>
</attribute-set>
<relation-set>
</relation-set>
</object>
</context>
```


In case of failure, the response is given by the following XML code:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
   <code>QPf54a</code>
   <description>Insert relation attribute failed.</description>
 </error>
```


Function f5_4b: Delete all relation attributes of a relation. This function is used to remove all the attributes of a relation that is about to me deleted. Future versions of the SC Space Editor will use this function for removing all the relation attributes of relation that is going to be removed from the model.

Function f5_4b HTML request:

```
http://host:port/sc/SCTX?function=f5_4b&objdom=34&relname=Is_Accessible_From
&objcdom=12&hostcdom=cityhall.cti&portcdom=8080
```

where `function` defines the function f5_4b, `objdom, relname, objcdom, hostcdom` and `portcdom` define the relation which will have all its attributes removed.

In case of failure, the response is given by the following XML code:


```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
 <error>
```

```
    <code>QPf54b</code>
    <description>Insert relation attribute failed.</description>
  </error>
```

Appendix B

User study

This appendix contains the documents that were provided to the participants of the user study. They are all in Portuguese. First we list the tasks, then we include the user manual and finally the questionnaire and the summary of its results. For the questions that required descriptive answers from the users, we transcribe only those answers that are not repeated. We also include an example of a local model created by one of the participants.

B.1 Tarefas

1. Ler o manual de utilizador (B.2) de modo a familiarizar-se com o modelo de espaços e as aplicações SC Space Editor, SC Browser e Route Finder;

2. Pesquisar o modelo existente com SC Browser para saber que objectos já existem no modelo;

3. Registar-se em SC Space Editor e usando a aplicação criar o vosso modelo de espaços com os locais que mais lhes interessam e tendo em conta os seguintes aspectos:

   (a) pelo menos 5 objectos, pelo menos 2 atributos por objecto, pelo menos 2 relações para cada objecto (das 4 do conjunto principal indicado no manual), das quais obrigatoriamente 1 com o nome *Is_In* e 1 com o nome *Is_Accessible_From*(bidireccional), caso se aplique ao tipo de objecto criado;

(b) os nomes dos objectos não devem incluir espaços, acentos, nem outros símbolos. Para nomes compostos usar _ para separar as palavras ou alternar letras maiúsculas/minúsculas (ex. Nome_Objecto, NomeObjecto);

(c) começar por criar primeiro os objectos maiores, que contêm outros ou que servirão de referência para outros objectos: p.ex. criar primeiro o objecto "edifício_da_UMa" e só depois o objecto "Piso0" de modo a poder relacionar o Piso0 com o edifício através de uma relação *Is_In*.

4. Visualizar o modelo criado e navegar pelos objectos;

5. Experimentar a procura de caminho mais curto entre dois objectos do seu modelo utilizando o Route Finder;

6. Responder ao questionário (Secção B.3).

B.2 Manual de utilização do modelo de espaços

B.2.1 Modelo de espaços

Trata-se de um modelo de espaços simbólico, baseado em modelos mentais, que assume a forma de um grafo. Cada vértice do grafo representa um objecto e cada aresta representa uma relação entre dois objectos. Pode haver mais do que uma relação entre dois objectos. Cada objecto e cada relação podem ter um conjunto de atributos que os descrevem com mais detalhe. Um exemplo é dado na fig.B.1.

B.2.1.1 Definição 1: Objecto

Um objecto é um lugar ou um objecto físico (uma sala, um corredor, uma praça, uma mesa, um veículo, um computador, etc.), um evento (uma conferência, uma reunião, etc.) ou uma entidade (grupo de pessoas, uma instituição, etc.). Cada

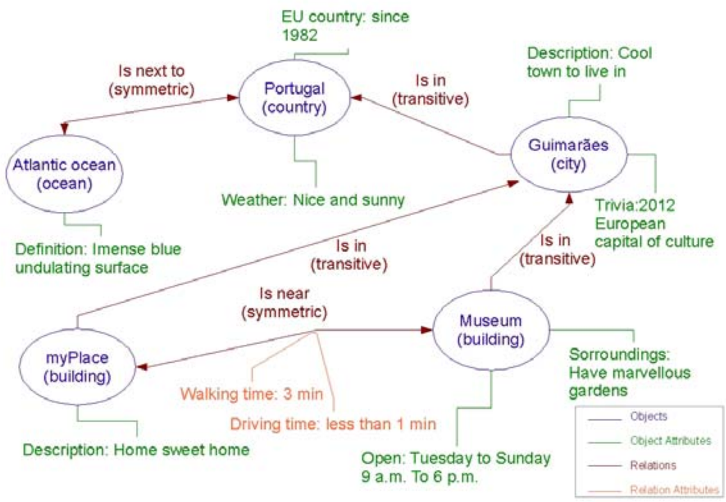


Figura B.1: Exemplo de modelo de espaços

objecto deve ter um nome pelo qual é facilmente identificado (ex. Funchal) e um tipo (ex. cidade). Os objectos criados por cada autor ficam identificados com o seu nome.

Na fig. B.1 os objectos e os respectivos tipos são os seguintes:

- myPlace (building)
- museum (building)
- Guimarães (city)
- Atlantic ocean (ocean)
- Portugal (country)

B.2.1.2 Definição 2: Atributo de objecto

Os atributos de um objecto podem ser características constantes ou variáveis (cor, área de um espaço, temperatura, número de pessoas presentes numa sala, etc.). Cada atributo tem um nome (cor do chão) e o respectivo valor (verde). O instante de tempo em que o valor é registado também é guardado. Assim, poderá haver diferentes valores para o mesmo atributo se for uma característica que varia no tempo. Os nomes atribuídos aos atributos devem ser facilmente compreensíveis e devem ser nomes usuais.

Na fig.B.1 cada objecto possui alguns atributos que podem ser descritivos ou numéricos. O objecto museum possui um atributo que descreve o seu espaço exterior e outro que guarda o horário de funcionamento.

B.2.1.3 Definição 3: Relação

Uma relação estabelece-se entre dois objectos e representa a relação que existe no mundo real entre eles. Cada relação tem um nome (*Is_Near*, significa proximidade entre dois objectos) e um tipo (*symmetric*, pois é válida nos dois sentidos).

Na fig.B.1 existe uma relação *Is_In* entre os objectos myPlace e Guimarães e entre Guimarães e Portugal. A existência destas relações e o seu tipo (transitive) permitem inferir que o objecto myPlace também fica em Portugal.

B.2.1.4 Definição 4: Atributo de relação

Os atributos das relações permitem descrevê-las melhor. Podem ser características constantes ou variáveis tal como acontece nos atributos dos objectos. Cada atributo tem um nome (distância, pode descrever a relação acima mencionada, *Is_Near*) e um valor (50m, quantifica a proximidade entre dois objectos).

Na fig.B.1, a relação *Is_Near* entre os objectos *myPlace* e *museum* tem dois atributos, um que indica o tempo que se leva a percorrer a distância entre os dois a pé e outro de carro.

B.2.1.5 Implementação

O modelo acima descrito é implementado em cima de uma base de dados relacional em que cada elemento do modelo é representado por uma tabela. O DER da base de dados apresenta-se na fig.B.2. O modelo é suportado por um serviço web, cujo nome é Symbolic Contextualizer, e que fornece a interface para o modelo.

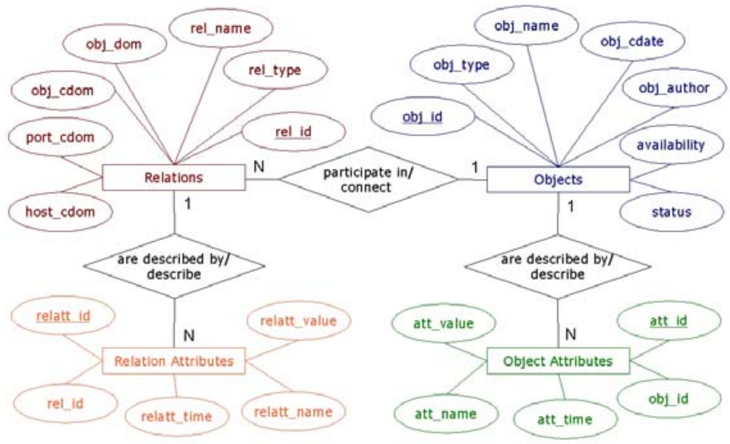


Figura B.2: DER da base de dados que implementa o modelo de espaços.

B.2.2 SC Browser (v0.2)

É uma ferramenta que permite a pesquisa de objectos no modelo de espaços existente. Actualmente só funciona no Internet Explorer.

Está disponível em: http://scontext.uma.pt:8080/scb

Na página de entrada, o utilizador deve inserir o nome do objecto procurado, completo ou parcial e escolher o tipo de inferência que pretende (Inference level). O resultado da pesquisa apresenta o nome do(s) objecto(s) encontrado(s), o tipo e o conjunto de atributos que o(s) descrevem (Fig.B.3).

Figura B.3: Pesquisa por um objecto devolve um único objecto ou uma lista de objectos.

Ao seleccionar um objecto da lista, o tipo de inferência vai definir quais as relações a apresentar. A primeira opção só mostra as relações básicas do objecto procurado (Fig.B.4).

Figura B.4: O resultado da pesquisa depois de especificar o objecto. Não havendo inferência só é mostrado o conjunto de relações definidas para o objecto (neste caso, uma relação *Is_In* com o objecto Lisboa (district)).

A segunda opção para Inference level (Fig.B.5) faz devolver todos os objectos que contêm o objecto procurado apresentando desta forma toda a hierarquia existente no modelo que termina com o objecto world.

Figura B.5: A segunda opção (Transitivity) permite visualizar toda a hierarquia de objectos que contêm o objecto escolhido.

A terceira opção (Fig.B.6) permite ver a lista de todos os objectos que possuem relações simétricas com o objecto escolhido.

Figura B.6: A terceira opção (Symmetry) permite visualizar as relações definidas para outros objectos, mas válidas nos dois sentidos (p.ex. Lisboa *Is_Accessible_From* Funchal (city)).

Finalmente, a última opção (Fig.B.7) mostra o conjunto de resultados das opções 2 e 3.

Figura B.7: A opção Both mostra todas as relações existentes em que o objecto escolhido participa.

Carregando em Expand relations é possível visualizar as relações de todos os objectos relacionados com o objecto inicial.

B.2.3 SC Space Editor (v0.3)

É uma aplicação web que permite criar modelos de espaço locais. A versão actual só funciona no Internet Explorer.

Encontra-se disponível em: http://scontext.uma.pt:8080/scse

Depois de efectuar o registo, o utilizador deve inserir os seus dados de acesso (user name e password) para começar a criar o seu modelo.

Na página inicial são apresentadas duas opções (fig.B.8). A primeira opção (Insert object) permite iniciar a criação de um novo modelo ou acrescentar novos objectos a um modelo já existente. A segunda opção (My Space Model) permite visualizar o modelo já existente criado pelo utilizador que iniciou a sessão.

Figura B.8: Página inicial do SCSE.

Escolhendo a primeira opção (Insert object), o utilizador inicia a criação de um novo objecto tal como mostra a fig. B.9. Ao inserir uma letra no campo de tipo de objecto e quando já existem objectos no modelo local, é apresentada uma lista de tipos usados ordenados por frequência. O utilizador pode escolher um dos tipos sugeridos ou escrever a segunda letra do tipo pretendido para continuar. Ao fazer isso, surge uma nova lista com os tipos de objectos existentes que são iniciados pelo par de letras inserido, tal como mostra a fig. B.10. Caso não apareça o tipo pretendido, o utilizador pode inseri-lo escrevendo as restantes letras.

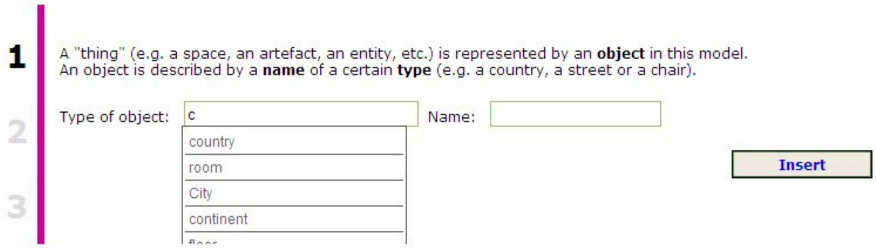


Figura B.9: Sugestões para os tipos de objectos com base nos objectos já existentes no modelo local.

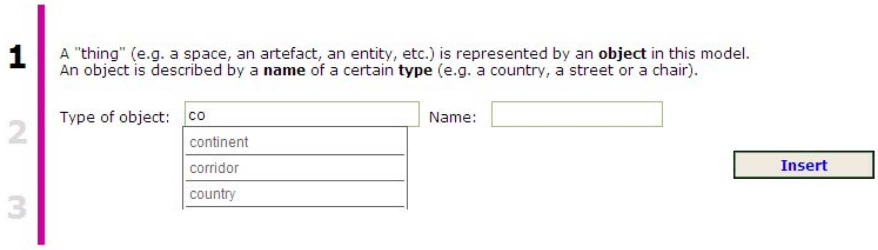


Figura B.10: Sugestões de tipos de objectos depois de inserir a segunda letra para o tipo pretendido.

O nome do objecto é livre, mas deve ser o nome comum que se dá a esse objecto de modo a que seja facilmente reconhecível pelas pessoas. O segundo passo na criação de um objecto é acrescentar-lhe atributos que permitam descrevê-lo melhor. Cada objecto pode ter um número arbitrário de atributos. O campo de nome do atributo também oferece sugestões quando se insere a primeira letra (fig.B.11). As sugestões são dadas com base na frequência dos atributos usados para o tipo de objecto que se está a criar. Se não houver nenhum objecto do mesmo tipo, não haverá sugestões. Tal como no caso do tipo de objecto, as sugestões para nomes de atributos podem ser ignoradas e novos nomes podem ser introduzidos. Uma vez terminada a inserção de atributos, o utilizador deve carregar em Skip para passar para o terceiro passo: inserção de relações entre objectos.

O terceiro passo na criação de um objecto consiste em inserir relações com outros objectos do mesmo modelo (fig.B.12). Para que este passo seja realizável,

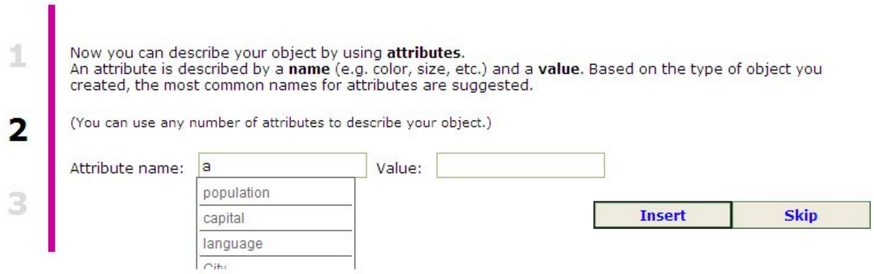


Figura B.11: Sugestões de atributos de objectos com base nos objectos já existentes que sejam do mesmo tipo que o objecto que se está a criar.

é necessário que o utilizador comece a criar o seu modelo de uma perspectiva top-down, i.e., primeiro os objectos mais gerais e depois os mais específicos. Por exemplo, se o utilizador quiser criar o modelo da sua casa, deverá começar por criar o país, a região, a cidade onde se situa a sua casa (se não existirem já no modelo), depois a casa como um todo, depois os pisos e as divisões existentes, e finalmente os objectos mais pequenos que se encontram dentro dessas divisões.

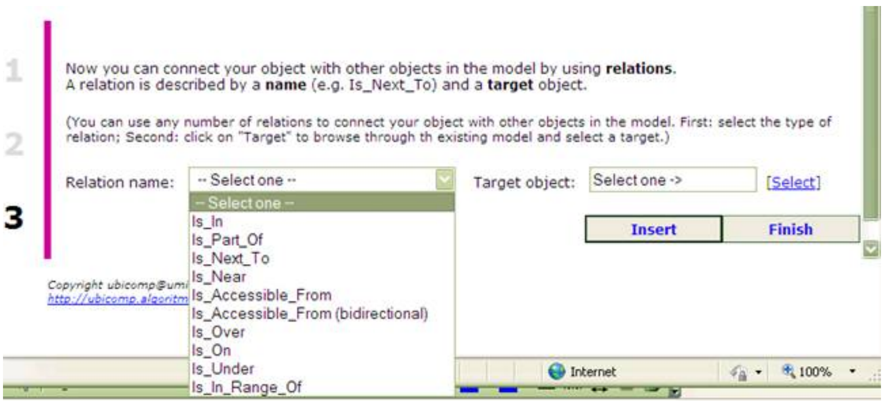


Figura B.12: Inserção de relações entre objectos.

Actualmente não é possível inserir novos nomes de relações. As relações suportadas são:

- Conjunto principal
  - *Is_In* - sempre do tipo *transitive*; usada quando um objecto se situa dentro de outro, física ou administrativamente (Funchal *Is_In* Madeira;

table *Is_In* kitchen)

- *Is_Accessible_From* - pode ser do tipo *symmetric* (bidireccional) ou *simple* (unidireccional); usada quando existe um caminho entre dois objectos (Funchal *Is_Accessible_From* Porto)
- *Is_Next_To* - sempre do tipo *symmetric*; usada para dois objectos adjacentes (CMF *Is_Next_To* Praca_do_Municipio)
- *Is_Near* - sempre do tipo *symmetric*; usada para objectos próximos (Quinta_das_Cruzes *Is_Near* Igreja_S_Pedro)

- Conjunto adicional
  - *Is_Part_Of* - sempre do tipo *transitive*; usada para objectos que criam conjuntos (myMouse *Is_Part_Of* myPC; paintingXYXY *Is_Part_Of* collectionxxx)
  - *Is_Over* - sempre transitive; usada para objectos que se encontram num nível superior relativamente a outros objectos (hisFlat *Is_Over* someShop)
  - *Is_On* - sempre transitive; usada para objectos que se encontram colocados em cima de outros objectos (myPc *Is_On* table)
  - *Is_Under* - sempre transitive; usada para objectos que se encontram colocados por baixo de outros objectos (myCat *Is_Unde*r table)
  - *Is_In_Range_Of* - usada entre as áreas cobertas por redes sem fios e as estações base ou access points (myPC *Is_In_Range_Of* myAP) Entre as relações suportadas, apenas as relações transitivas e simétricas têm algoritmos de inferência associados (ver explicação do SC Browser).

O objecto destino da relação é seleccionado no campo Target object e carregando sobre o link Select. Ao carregar neste link, é aberta uma nova janela e é iniciada a aplicação SC Browser que permite fazer pesquisas no modelo. O utilizador deve inserir o nome do objecto pretendido e quando este aparece na lista, selecciona-lo. Ao seleccionar um objecto, deve fechar a janela e regressar ao SCSE onde o campo de target object fica preenchido com o objecto escolhido. Carregando no botão Insert, é possível inserir mais relações, não existindo o limite para o número destas. Carregando em Finish, ao utilizador é apresentada a descrição do objecto criado e são dadas as opções de adicionar mais atributos e/ou relações (fig.B.13). Ao seleccionar as opções Insert object ou My space model, deixa de ser possível editar o objecto criado.

Na versão actual do SCSE não é possível inserir atributos de relações.

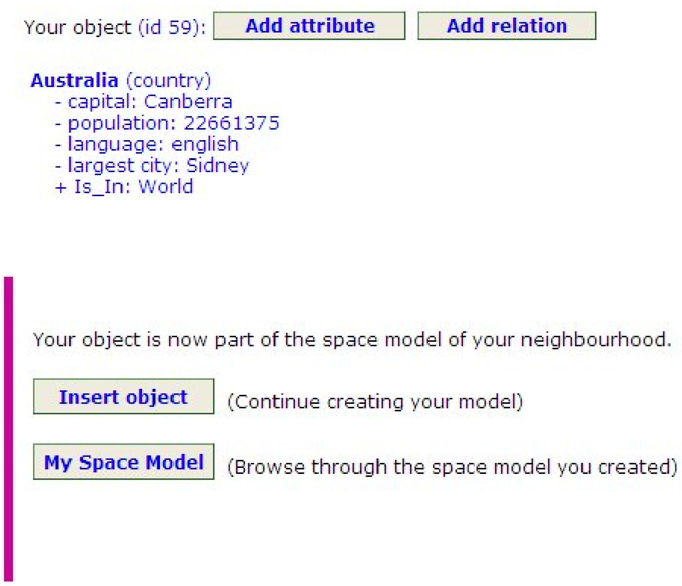


Figura B.13: A descrição do objecto criado e as opções para adicionar mais atributos e relações.

A visualização do modelo local pertencente ao utilizador pode ser feita carregando em My space model. O resultado é gerado pela aplicação SC Browser em forma de uma lista de objectos que contém o nome, o tipo e os atributos dos objectos (fig.B.14). Seleccionando um dos objectos, é possível ver e expandir as suas relações (fig.B.15).

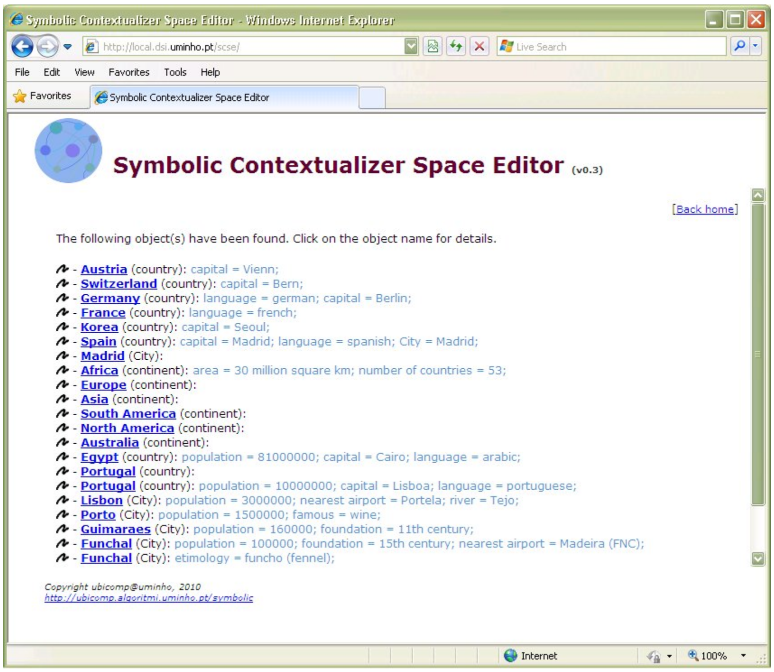


Figura B.14: Visualização dos objectos existentes no modelo local criado pelo utilizador.

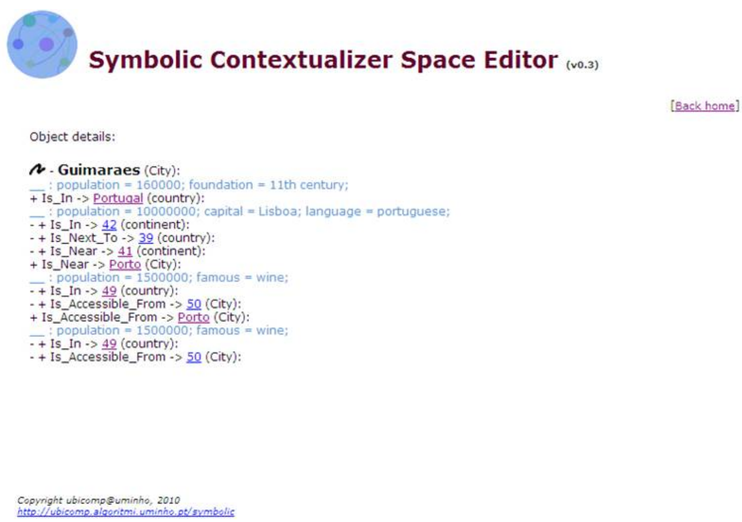


Figura B.15: Descrição completa de um objecto do modelo depois de expandir as suas relações.

B.2.4 SC Route Finder (v0.4)

Depois de criar um modelo local, o utilizador pode querer encontrar o caminho mais curto entre dois objectos do seu modelo. O Route Finder facilita isso, sobretudo quando no ambiente real existem etiquetas que indicam os nomes dos locais e permitem que o utilizador se auto-localize. As indicações dadas por esta aplicação são semelhantes às que normalmente as pessoas transmitem quando interrogadas por alguém sobre um caminho possível entre dois locais.

Encontra-se disponível em: http://scontext.uma.pt:8080/routeT

Esta aplicação simples de navegação pedestre baseia-se nos dados existentes no modelo de espaços. Utiliza a relação *Is_Accessible_From* para encontrar os caminhos entre objectos tendo em conta se esta é do tipo *symmetric* ou *simple*.

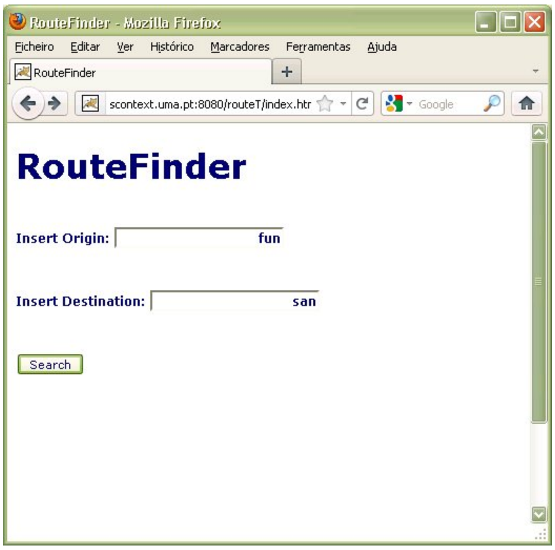


Figura B.16: Ecrã inicial do Route Finder. O utilizador pode inserir o nome completo ou parcial dos objectos.

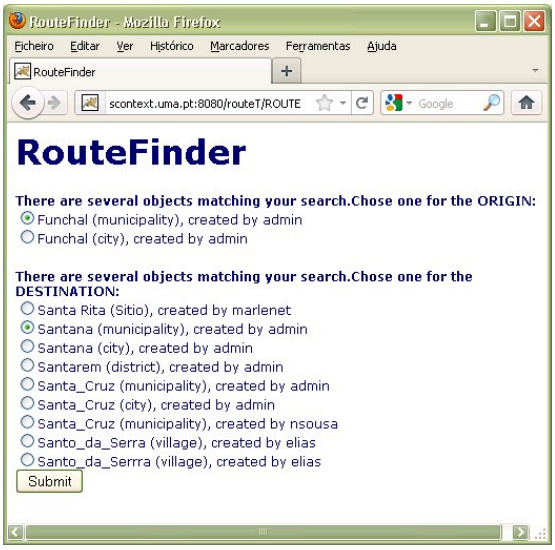


Figura B.17: Desambiguação de objectos de origem e destino. Quando são encontrados vários objectos, é apresentada uma lista onde o utilizador deve escolher os objectos que pretende com base no tipo (entre parêntesis) e no criador (created by...).

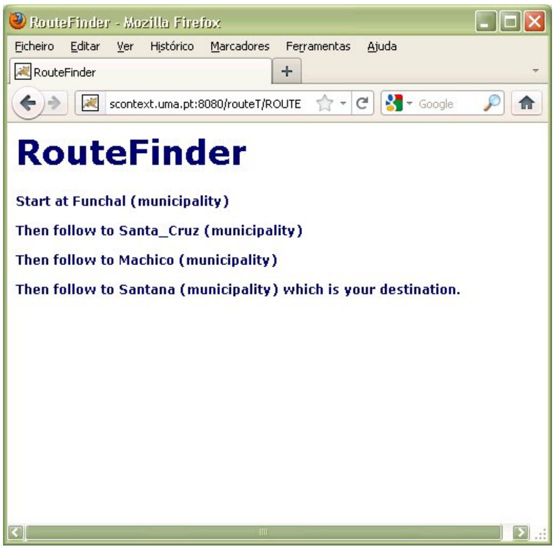


Figura B.18: Descrição do caminho a seguir para ir da origem ao destino.

Figura B.19: Quando não é encontrado nenhum caminho possível, o utilizador é informado e convidado a fazer uma nova pesquisa.

B.3 Questionário

B.3.1 Formulário

De modo a poder responder a este questionário é necessário que tenha efectuado as seguintes tarefas:

1. Ler o manual de utilização do modelo de espaços e das aplicações SC SpaceEditor, SC Browser e RouteFinder;

2. Visualizar o modelo existente com SC Browser para saber que objectos já existem no modelo;

3. Registar-se em SC SpaceEditor e usando a aplicação criar o seu modelo de espaços com os locais que mais lhe interessam;

4. Visualizar o modelo criado e navegar pelos objectos;

5. Experimentar a procura de caminho mais curto entre dois objectos do seu modelo utilizando o RouteFinder.

O questionário tem 21 questões, das quais 18 são obrigatórias. O tempo estimado de preenchimento é de 10 minutos.

1. Que idade tem?

☐ Menos de 20

☐ 21 a 30

☐ 31 a 40

☐ 41 a 50

☐ mais de 50

2. Indique o seu género. ☐ Feminino ☐ Masculino

3. Qual é a sua relação com as tecnologias de informação e comunicação?

☐ Utilizador básico

- □ Utilizador intermédio
- □ Utilizador avançado
- □ Formador
- □ Profissional

Modelo de espaços:

4. **A compreensão do conceito do modelo de espaços tal como descrito no manual foi:**

   Muito difícil □—□—□—□—□ Muito fácil

5. **Até que ponto acha que este modelo se assemelha aos modelos de espaços que criamos mentalmente?**

   Nada a ver □—□—□—□—□ Tal e qual

6. **O que é que o faria contribuir para um modelo destes? Múltiplas respostas possíveis**
   - □ Uma aplicação apelativa
   - □ Um jogo
   - □ Integração da aplicação numa rede social
   - □ Integração com um mapa geográfico
   - □ Outro: ____________________

7. **Escreva aqui sugestões ou comentários relacionados com o modelo de espaços**

   ____

   ____

   ____

SC Space Editor:

8. **Os passos para a criação do modelo estão definidos de forma clara?**

   Nada □—□—□—□—□ Completamente

9. **Indique quantos objectos criou:** ________________

10. **Indique quanto tempo demorou a criar os objectos do seu modelo**

    □ 5 a 10 minutos

    □ 10 a 20 minutos

    □ 20 a 30 minutos

    □ mais de 30 minutos

    □ Outro: ________________

11. **Na sua opinião a existência de sugestões para o tipo de objectos é útil?** □ Sim □ Não

12. **Na sua opinião a existência de sugestões para os nomes dos atributos é útil?** □ Sim □ Não

13. **Depois de criar o modelo conseguiu visualizar bem o modelo criado? Ficou com uma ideia clara do modelo que tinha criado?**

    Não, de todo. □—□—□—□—□ Sim, perfeitamente.

14. **O que gostou mais nesta aplicação?**

    ____

    ____

    ____

15. **O que gostou menos nesta aplicação?**

    ____

    ____

    ____

16. **Escreva aqui sugestões e comentários relacionados com a aplicação SC SpaceEditor**

    ____

    ____

    ____

Route Finder:

17. Utilidade da aplicação

Inútil □—□—□—□—□ Muito útil

18. Facilidade de utilização

Muito difícil □—□—□—□—□ Muito fácil

19. O que gostou mais nesta aplicação?

20. O que gostou menos nesta aplicação?

21. Escreva aqui sugestões e comentários relacionados com a aplicação Route Finder

B.3.2 Resumo dos resultados

B.3.2.1 Informação geral sobre o utilizador

Na primeira parte do questionário foram colocadas três perguntas aos participantes: a sua idade, género e relação com as tecnologias de informação e comunicação. O resumo das respostas encontra-se nas Figuras B.20, B.21 e B.22.

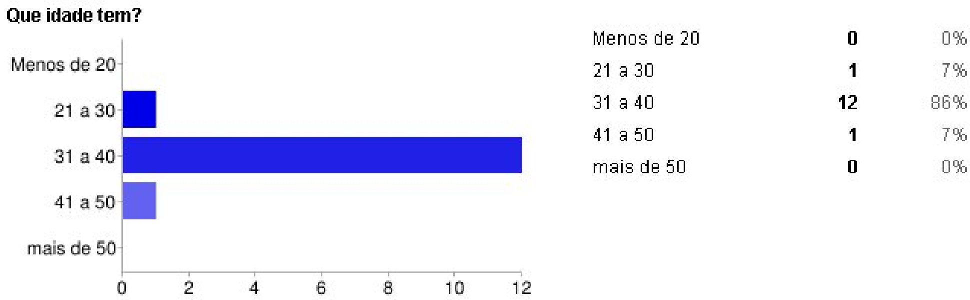


Figura B.20: Questão 1: Que idade tem?

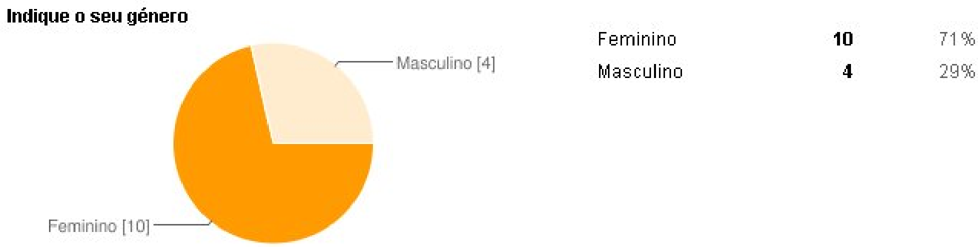


Figura B.21: Questão 2: Indique o seu género.

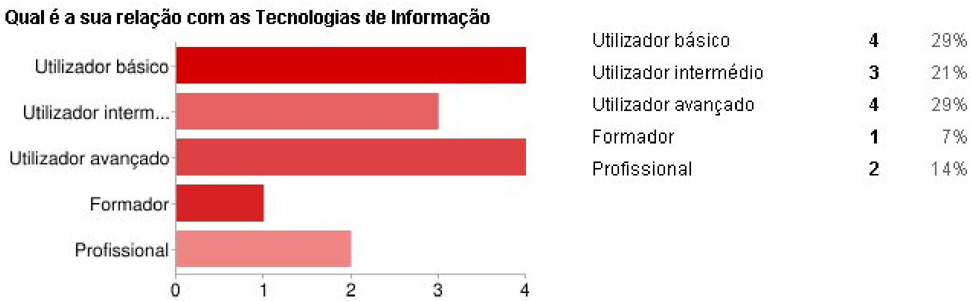


Figura B.22: Questão 3: Qual é a sua relação com as tecnologias de informação e comunicação?

B.3.2.2 Modelo de espaços

O grupo de questões que se segue é sobre o conceito do modelo de espaços. O resumo das respostas encontra-se nas Figuras B.23, B.24 e B.25.

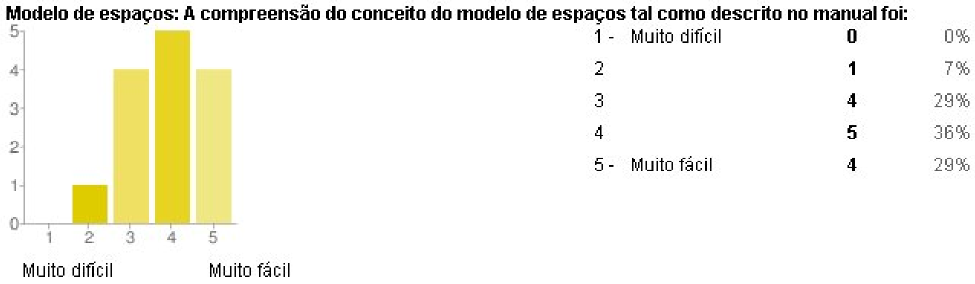


Figura B.23: Questão 4: A compreensão do conceito do modelo de espaços tal como descrito no manual

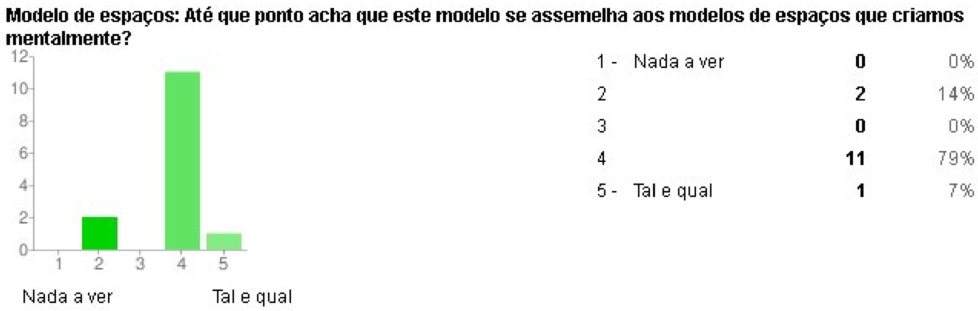


Figura B.24: Questão 5: Até que ponto acha que este modelo se assemelha aos modelos de espaços que criamos mentalmente?

Figura B.25: Questão 6: O que é que o faria contribuir para um modelo destes?

B.3.2.3 Sugestões e comentários sobre o modelo de espaços

- *Possibilidade de alterar o tipo de objecto e até eliminá-lo, para evitar duplicação.*

- *Penso que seria muito interessante ter a possibilidade de reeditar os com-*

pomentes tanto nos atributos como nas relações.

- *Representação gráfica das relações por tipo de relação (os 'Next_to', os 'Part_of', etc.).*
- *Talvez fosse útil usar uma medida para quantificar a proximidade dos objectos criados (distância, tempo e/ou custo) e usá-la para encontrar o melhor percurso.*
- *A representação gráfica (e não apenas textual) dos objectos e das relações entre os objectos do modelo criado*

B.3.2.4 SC Space Editor

O grupo de questões que se segue é sobre a aplicação SC Space Editor que permite a criação manual de objectos, atributos de objectos e relações entre objectos no modelo. O resumo das respostas encontra-se nas Figuras B.26, B.27, B.28 e B.29.

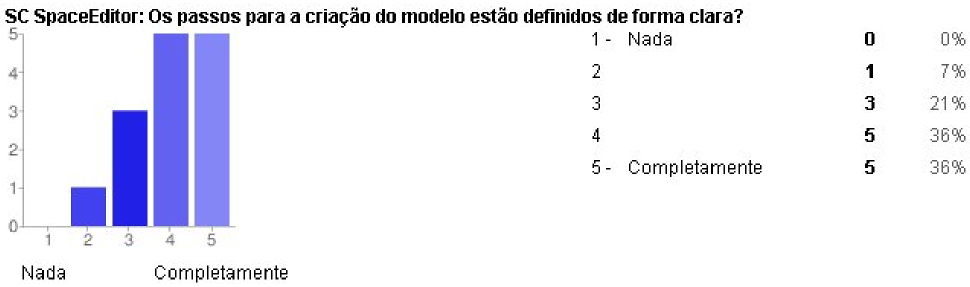


Figura B.26: Questão 8: Os passos para a criação do modelo estão definidos de forma clara?

B.3.2.5 SC Space Editor: o que gostou mais?

- *A facilidade e rapidez de criação de objetos*
- *A parte de ser o utilizador a criar os objectos do modelo de espaços.*
- *Das relações diversificadas que se podem criar.*

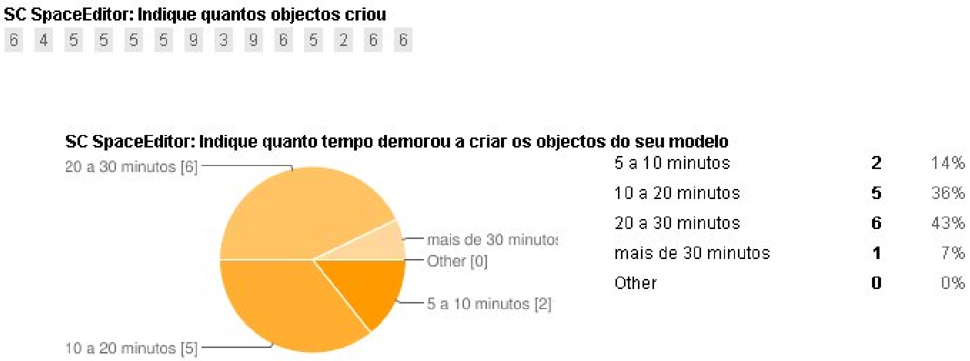


Figura B.27: Questão 9: Indique quantos objectos criou. Questão 10: Indique quanto tempo demorou a criar os objectos do seu modelo

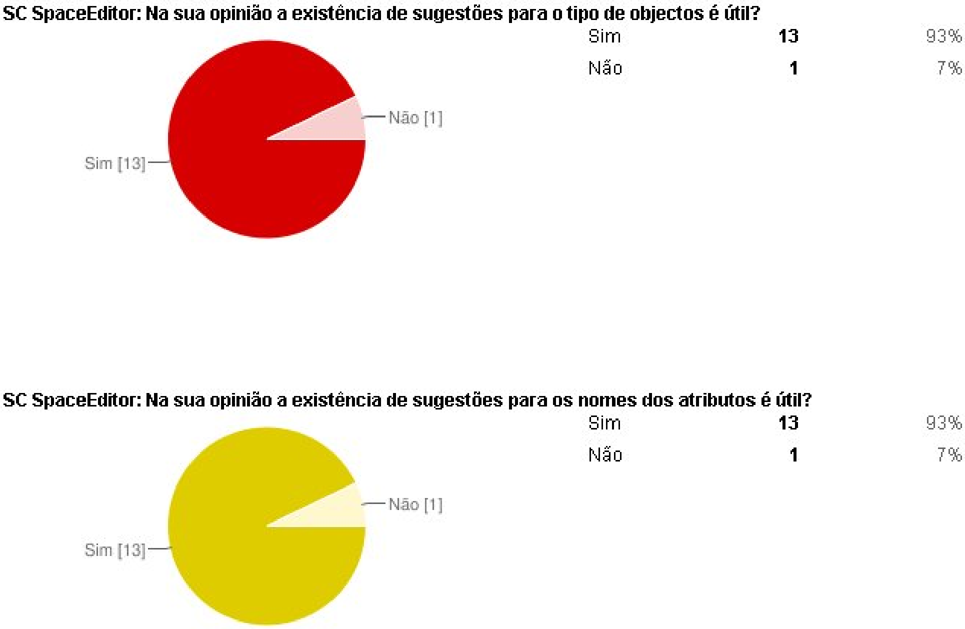


Figura B.28: Questão 11: Na sua opinião a existência de sugestões para o tipo de objectos é útil? Questão 12: Na sua opinião a existência de sugestões para os nomes dos atributos é útil?

- *A interligação entre os diferentes objectos criados. Os dados solicitados são concretos e a explicação de cada item é clara e sucinta. Os exemplos dados também facilitam a sua compreensão.*

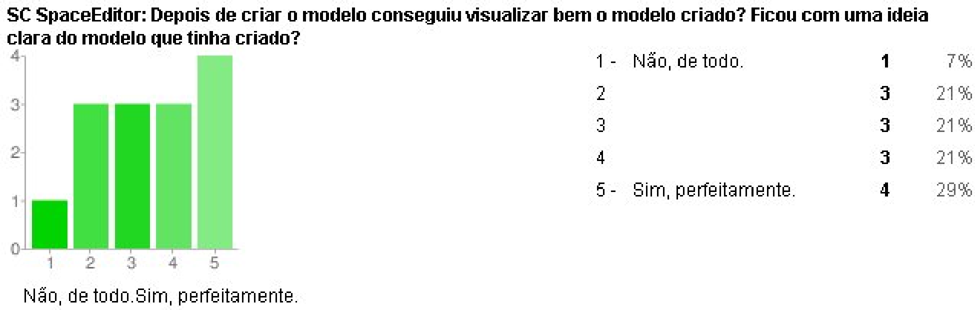


Figura B.29: Questão 13: Depois de criar o modelo conseguiu visualizar bem o modelo criado? Ficou com uma ideia clara do modelo que tinha criado?

- *Gostei bastande da grande liberdade em que tinhamos de escolher o tipo de objecto, não nos obrigando a tipos pré-definidos.*
- *Estrutura de relações rica e interessante, mas que necessita algum tempo a ser interiorizada.*
- *Sugestões para o tipo de objecto e para os atributos consoante o tipo de objecto, usando tipos e atributos já existentes, evitando assim novas entradas desnecessárias.*
- A *ideia de usar uma ferramenta para permitir e facilitar a criação do modelo de espaços.*

B.3.2.6 SC Space Editor: O que gostou menos?

- *Não poder corrigir os meus erros.*
- *O tempo limite muito curto para preenchimento de todos os campos. Ao fim de alguns minutos tinha de fazer novamente o login. O facto de não poder dar continuidade ao objecto que ficou inacabado, e não poder apagar ou corrigir nomes depois de submeter.*
- *Falta de capacidade de edição de objectos criados. A sequência necessária, do 'maior' para o 'menor', parece-me que não acompanha o processo mental natural, mais aleatório.*

- *Não poder visualizar os objectos criados e as relações entre eles graficamente*

B.3.2.7 SC Space Editor: Sugestões e comentários

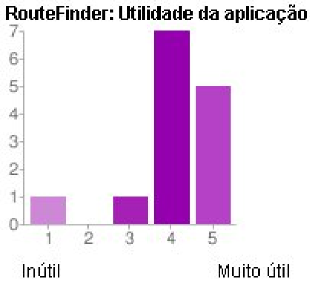
- *Deveriam existir sugestões de relações obrigatórias para os objetos. Criei 6 objetos e depois quando usei o Route Finder não encontrava relação entre eles. Provavelmente defini mal as relações dos objetos.*
- *Possibilidade de adicionar imagens*
- *Alargar o tempo de preenchimento dos campos, sem ser necessário voltar a fazer login. Poder editar os objectos em qualquer altura, mesmo após ter inserido.*
- *Ser possível alterar dados dos objectos criados. Durante a consulta das ramificações dos objectos ser possível voltar à consulta anterior em vez de ir para Home.*
- *Melhoria da ferramenta de edição, actualização de objectos e maior descrição das relações.*
- *Implementar atributos das relações (já previsto no projecto)*
- *Adicionar uma representação gráfica do modelo para melhorar a percepção por parte dos utilizadores relativamente ao modelo criado*

B.3.2.8 Route Finder

O grupo de questões que se segue é sobre a aplicação Route Finder. O resumo das respostas encontra-se nas Figuras B.30 e B.31.

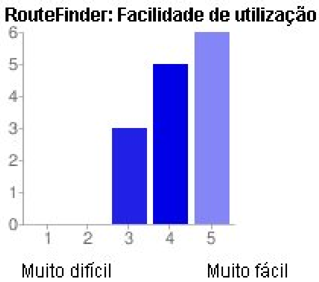
B.3.2.9 Route Finder: O que gostou mais?

- *A utilidade da aplicação, ou seja, ser possível pesquisar um objecto e nos ser fornecido a localização deste mesmo objecto, sendo que esta, pelo que me apercebi, é a menor possível.*



1 -	Inútil	**1**	7%
2		**0**	0%
3		**1**	7%
4		**7**	50%
5 -	Muito útil	**5**	36%

Figura B.30: Questão 17: Utilidade da aplicação



1 -	Muito difícil	**0**	0%
2		**0**	0%
3		**3**	21%
4		**5**	36%
5 -	Muito fácil	**6**	43%

Figura B.31: Questão 18: Facilidade de utilização

- *Simples de usar*
- *Descobrir outras ligações a outros espaços/locais*
- *Funcionar mesmo quando se escreve apenas a parte inicial dos pontos de partida e chegada. E o facto de podermos escolher de forma fácil entre as opções correspondentes à palavra que escrevemos.*
- *Possibilidade de detectar falhas na definição das relações.*

B.3.2.10 Route Finder: O que gostou menos?

- *Só consegui encontrar os caminhos entre nomes adicionados pelos administradores.*
- *Não ter acesso a mais informação dos objectos que fazem parte da rota.*
- *A forma da pesquisa.*

- *Quando as relações não estão bem inseridas, podem aparecer caminhos que não são os mais curtos.*

B.3.2.11 Route Finder: Sugestões e comentários

- *Talvez, futuramente, inserir um género de mapa ou algo visual do mundo criado, como também o caminho a percorrer. Considero que com isso a aplicação iria fornecer uma noção espacial mais apelativa ao utilizador.*

- *Na rota descrita deveria ser possível entrar nos objectos que fazem parte da rota de forma a poder ter informação sobre estes.*

- *Nesta, também deveriam existir sugestões, tal como nas aplicações anteriores.*

- *Talvez devesse aparecer mais do que um caminho, porque pode haver um com mais "nós" que seja melhor. Ou podemos querer ir a vários sítios próximos e RouteFinder podia ajudar a escolher a ordem pela qual vamos a esses sítios.*

B.4 Exemplo de um modelo local

Figura B.32 ilustra um exemplo de um modelo local criado por um dos participantes do estudo.

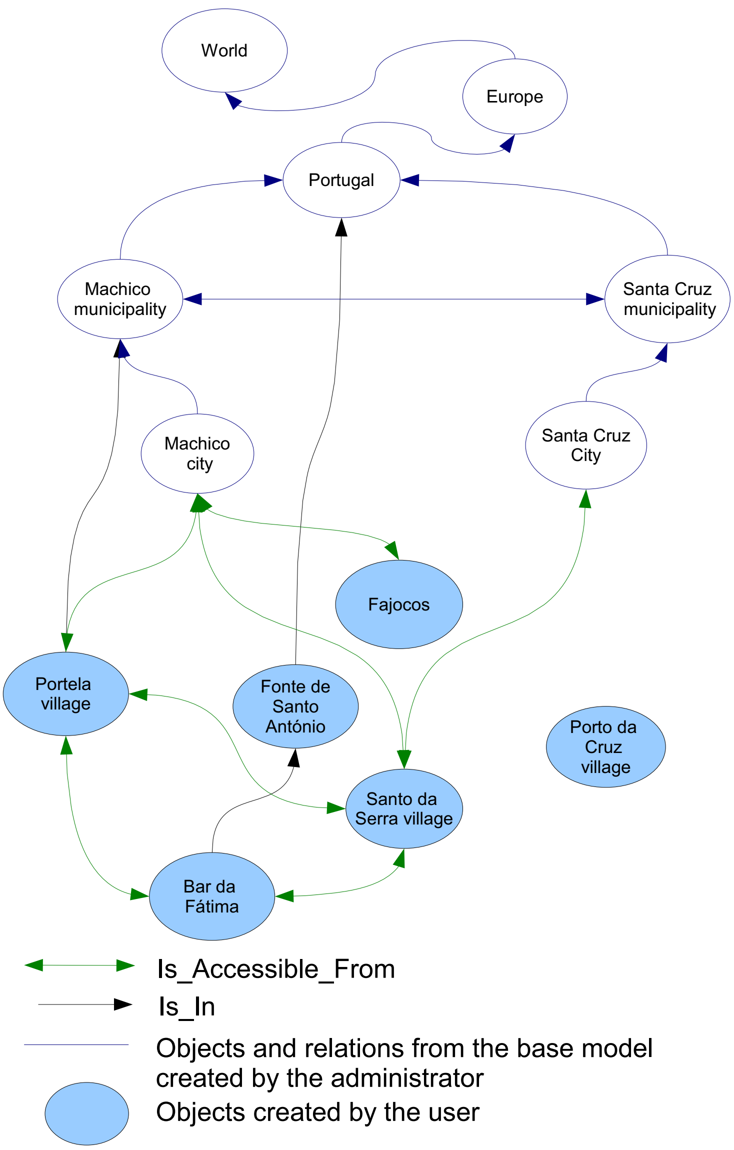


Figura B.32: Exemplo de um modelo local criado por um utilizador.

References

Ackerman, E. & Guizzo, E. (2011). 5 technologies that will shape the web. *Spectrum, IEEE*, **48**, 40 –45. 32

Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., Seitz, S.M. & Szeliski, R. (2010). Reconstructing rome. *Computer*, **43**, 40 –47. 18

Ailisto, H., Alahuhta, P., Haataja, V. & Vesa Kyllönen, M.L. (2002). Structuring context aware applications: Five-layer model and example case. In *In UbiComp 2002 Workshop on Concepts and Models for Ubiquitous Computing*. xiii, 55, 56, 88, 144

Baldauf, M., Dustdar, S. & Rosenberg, F. (2007). A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, **2**, 263–277. 38, 55

Barford, P., Kline, J., Plonka, D. & Ron, A. (2002). A signal analysis of network traffic anomalies. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 71–82, ACM, New York, NY, USA. 96

Bauer, M. & Rothermel, K. (2004). How to observe real-world events through a distributed world model. In *ICPADS '04: Proceedings of the Parallel and Distributed Systems, Tenth International Conference*, 467, IEEE Computer Society, Washington, DC, USA. 22

Bauer, M. & Rothermel, K. (2005). An architecture for observing physical world events. In *ICPADS '05: Proceedings of the 11th International Confe-*

rence on Parallel and Distributed Systems, 377–383, IEEE Computer Society, Washington, DC, USA. 22

Bauer, M., Becker, C. & Rothermel, K. (2002). Location models from the perspective of context-aware applications and mobile ad hoc networks. *Personal Ubiquitous Computing*, **6**, 322–328. 22

Becker, C. & Dürr, F. (2005). On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, **9**, 20–31. xix, 2, 15, 16, 19, 20, 49, 119, 132, 134, 135

Beigl, M., Zimmer, T. & Decker, C. (2002). A location model for communicating and processing of context. *Personal Ubiquitous Comput.*, **6**, 341–357. 20

Bell, G. & Dourish, P. (2007). Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. *Personal and Ubiquitous Computing*, **11**, 133–143. 9

Bleicher, A. (2011). The anti-facebook. *Spectrum, IEEE*, **48**, 54 –82. 37

Brumitt, B. & Shafer, S. (2001). Topological world modeling using semantic spaces. In *In UbiComp 2001 Workshop on Location Modeling for Ubiquitous Computing*. 15

Brumitt, B., Meyers, B., Krumm, J., Kern, A. & Shafer, S.A. (2000). Easyliving: Technologies for intelligent environments. In P.J. Thomas & H.W. Gellersen, eds., *Handheld and Ubiquitous Computing*, vol. 1927 of *Lecture Notes in Computer Science*, 97–119, Springer-Verlag, London, UK, handheld and Ubiquitous Computing,Second International Symposium, HUC 2000, Bristol, UK, September 25-27, 2000, Proceedings. 19

Dey, A.K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, **5**, 4–7. 10

Dey, A.K., Abowd, G.D. & Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, **16**, 97–166. 57

Domnitcheva, S. (2001). Location modeling: State of the art and challenges. In *Proceedings of the Workshop on Location Modeling for Ubiquitous Computing*, 13–19, Atlanta, Georgia, United States. 15

Ejigu, D., Scuturici, M. & Brunie, L. (2008). Hybrid approach to collaborative context-aware service platform for pervasive computing. *Journal of Computers*, **3**, 40–50. 14

Erickson, T. (2002). Some problems with the notion of context-aware computing. *Commun. ACM*, **45**, 102–104. 11

Henderson, T., Kotz, D. & Abyzov, I. (2008). The changing usage of a mature campus-wide wireless network. *Computer Networks*, **52**, 2690–2712. 24

Henricksen, K., Indulska, J. & Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. In *Pervasive '02: Proceedings of the First International Conference on Pervasive Computing*, 167–180, Springer-Verlag, London, UK. 13, 14

Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K. & Schwehm, M. (1999). Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom '99)*, 249–255, not available, Seattle, WA, USA. 21

Hsieh, P.H. & Yuan, S.T. (2003). Dynamic semantic location modeling in mobile enterprise applications. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, 102–110, ACM, New York, NY, USA. 15

HSIEH, W.T., MILLER, N., YANG, Y.F., CHOU, S.C. & STEENKISTE, P. (2004). Calculating walking distance in a hybrid space model. In *First International Workshop on Advanced Context Modelling, Reasoning And Management*. 20

HU, H. & LEE, D.L. (2004). Semantic location modeling for location navigation in mobile environment. In *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)*, 52–61. 21

HUTCHINS, R. & ZEGURA, E. (2002). Measurements from a campus wireless network. In *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 5, 3161–3167. 24

JUDD, G. & STEENKISTE, P. (2003). Providing contextual information to pervasive computing applications. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*. 13, 14

KIM, M. & KOTZ, D. (2007). Periodic properties of user mobility and access-point popularity. *Personal Ubiquitous Computing*, **11**, 465–479. 24

KOLODZIEJ, K. & DANADO, J. (2004). In-building positioning: modeling location for indoor world. In *Database and Expert Systems Applications, 2004. Proceedings. 15th International Workshop on*, 830– 834. 17, 19

KOSTAKOS, V. (2010). Space syntax and pervasive systems. In B. Jiang, X. Yao & D.Z. Sui, eds., *Geospatial Analysis and Modelling of Urban Structure and Dynamics*, vol. 99 of *GeoJournal Library*, 31–52, Springer Netherlands. 147

KUMAR, S. (1995). *Classification and detection of computer intrusions*. Ph.D. thesis, Purdue University. 96

LANGE, R., CIPRIANI, N., GEIGER, L., GROSSMANN, M., WEINSCHROTT, H., BRODT, A., WIELAND, M., RIZOU, S. & ROTHERMEL, K. (2009). Making the world wide space happen: New challenges for the nexus context platform.

Pervasive Computing and Communications, IEEE International Conference on, 1–4. 22, 38, 128

Lehmann, O., Bauer, M., Becker, C. & Nicklas, D. (2004). From home to world - supporting context-aware applications through world models. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, 297–306. 14, 22

Leonhardt, U. (1998). *Supporting Location-Awareness in Open Distributed Systems*. Ph.D. thesis, Department of Computing Imperial College of Science, Technology and Medicine University of London. 2

McNett, M. & Voelker, G.M. (2005). Access and mobility of wireless pda users. *SIGMOBILE Mob. Comput. Commun. Rev.*, **9**, 40–55. 24

Miller, N. & Steenkiste, P. (2007). Design and evaluation of a hybrid physical space service for pervasive computing applications. In *Mobile and Ubiquitous Systems: Networking and Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, 1–8. 20

Moreira, A., Gonçalves, M. & Vilas Boas, N. (2006). *LOCAL: Location contexts for location-aware applications, Technical Report TR4, Estimation of enhanced user location*. 57, 58

Myers, J.L. & Well, A.D. (2003). *Research Design and Statistical Analysis*, 45,508. Lawrence Erlbaum Associates. 106

Nucci, A. & Bannerman, S. (2007). Controlled chaos [internet security]. *Spectrum, IEEE*, **44**, 42 –48. 96, 145

Ostermaier, B. & Bolliger, P. (2008). Creating location-based services by utilising a web of places. In *SAINT '08: Proceedings of the 2008 International Symposium on Applications and the Internet*, 161–164, IEEE Computer Society, Washington, DC, USA. 22

Peddemors, A. & Yoneki, E. (2009). Decentralized probabilistic world modeling with cooperative sensing. *Electronic Communications of the EASST*, **17**. 22

Pradhan, S. (2000). Semantic location. *Personal Ubiquitous Computing*, **4**, 213–216. 15

Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Lorenzo, J., Valla, M., Fra, C., Paspallis, N. & Papadopoulos, G.A. (2008). A comprehensive context modeling framework for pervasive computing systems. In *Proceedings of the 8th IFIP WG 6.1 international conference on Distributed applications and interoperable systems*, DAIS'08, 281–295, Springer-Verlag, Berlin, Heidelberg. 14, 40

Rogers, Y. (2006). Moving on from weiser's vision of calm computing: Engaging ubicomp experiences. In P. Dourish & A. Friday, eds., *UbiComp 2006: Ubiquitous Computing*, vol. 4206 of *Lecture Notes in Computer Science*, 404–421, Springer Berlin / Heidelberg, 10.1007/11853565_24. 11

Satoh, I. (2008). A spatial communication model for ubiquitous computing services. *Journal of Networks*, **3**, 10–20. 16

Schilit, B. & Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, **8(5)**, 22–32. 10, 11

Schilit, B., Adams, N. & Want, R. (1994). Context-aware computing applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and applications*. 11

Schmidt, A. (2010). Ubiquitous computing: Are we there yet? *Computer*, **43**, 95 –97. 9

Schmidt, A., Beigl, M. & Gellersen, H.W. (1999). There is more to context than location. *Computer and graphics journal*, **23(6)**, 893–902. 4, 10, 12, 13

SEVTSUK, A. & RATTI, C. (2008). Urban activity dynamics. In *http://web.mit.edu/asevtsuk/www/articles.html*. 25

SEVTSUK, A., HUANG, S., CALABRESE, F. & RATTI, C. (2008). Mapping the MIT campus in real time using WiFi. In M. Foth, ed., *Handbook of Research on Urban Informatics: The Practice and Promise of the Real-Time City*, 326–337, Hershey PA, IGI Global. 25, 65, 144

STAHL, C. & HAUPERT, J. (2006). Taking location modelling to new levels: A map modelling toolkit for intelligent environments. In *LoCA*, Lecture Notes in Computer Science, 74–85, Springer Berlin/Heidelberg. 64

STRANG, T. & LINNHOFF-POPIEN, C. (2004). A context modelling survey. *First International Workshop on Advanced Context Modelling, Reasoning and Management*. 13, 14, 40

TANG, D. & BAKER, M. (2000). Analysis of a local-area wireless network. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, 1–10, ACM, New York, NY, USA. 23

TAUB, H. & SCHILLING, D.L. (1986). *Principles of Communication Systems*, 514. McGraw-Hill. 101

THOTTAN, M. & JI, C. (2003). Anomaly detection in IP networks. *Signal Processing, IEEE Transactions on*, **51**, 2191–2204. 96

TVERSKY, B. (2000). Remembering spaces. In E. Tulving & F.I.M. Craik, eds., *The Oxford handbook of memory*, 363–378, New York, NY, US: Oxford University Press. 1, 28

VACCARI, A., SAMOUHOS, S., GLICKSMAN, L. & RATTI, C. (2009). Mit enernet: Correlating WiFi activity to human occupancy. In *Proceedings of Healthy Buildings 2009, the 9th International Conference and Exhibition*. 25

WALLER, V. & JOHNSTON, R.B. (2009). Making ubiquitous computing available. *Commun. ACM*, **52**, 127–130. 128

WEISER, M. (1991). The computer for the 21st century. *Scientific American*, **265**, 94–104. 2

WEISER, M. (1993). Some computer science issues on ubiquitous computing. *Communications of the ACM*, **36**, 75–84. 2

YE, J., COYLE, L., DOBSON, S. & NIXON, P. (2007). A unified semantics space model. In *Location- and Context-Awareness*, vol. 4718 of *Lecture Notes in Computer Science*, 103–120, Springer Berlin Heidelberg. 21