Masters Program in Geospatial Technologies

Agent-based Parking Occupancy Simulation

Germán Martín Mendoza Silva

Dissertation submitted in partial fulfilment of the requirements for the Degree of *Master of Science in Geospatial Technologies*

UNIVERSIDADE NOVA ISEGI
UNIVERSITAT JAUME·I
WESTFÄLISCHE WILHELMS-UNIVERSITÄT MÜNSTER

Agent-based Parking Occupancy Simulation

Dissertation supervised by

Ph.D. Michael Gould
Universitat Jaume I,
Castellón de la Plana, Spain

Ph.D. Ana Cristina Costa
Universidade NOVA de Lisboa,
Lisbon, Portugal

Ph.D. Jorge Mateu Mahiques
Universitat Jaume I,
Castellón de la Plana, Spain

March 2015

Acknowledgements

I thank God for placing in my way wonderful people whom I owe much of this thesis's results. They have my most sincere gratitude:

My supervisors Ph.D. Michael Gould, Ph.D. Ana Cristina Costa and Ph.D. Jorge Mateu Mahiques, for their valuable advice.

Ph.D. Raúl Montoliu Colás, who has been another supervisor for me.

My family, who encouraged me to pursue my dreams, and has endured the rigors of distance.

My good friends who fought alongside me in Cuba for a better future.

Ph.D. Joaquín Huerta Guijarro, for his advice and support before and during the whole Master.

My Master's classmates, who made the living in Europe a much more rewarding experience.

Abstract

The existing parking simulations, as most simulations, are intended to gain insights of a system or to make predictions. The knowledge they have provided has built up over the years, and several research works have devised detailed parking system models. This thesis work describes the use of an agent-based parking simulation in the context of a bigger parking system development. It focuses more on flexibility than on fidelity, showing the case where it is relevant for a parking simulation to consume dynamically changing GIS data from external, online sources and how to address this case. The simulation generates the parking occupancy information that sensing technologies should eventually produce and supplies it to the bigger parking system. It is built as a Java application based on the MASON toolkit and consumes GIS data from an ArcGis Server. The application context of the implemented parking simulation is a university campus with free, on-street parking places.

Keywords

Intelligent parking systems

Agent-based modeling and simulation

Integrated simulations

GIS data and services

Software design

Acronyms

GIS – Geospatial Information System

ABMS – Agent-based modeling and simulation

ITS – Intelligent Transportation System

PGI – Parking guidance and information

UJI – Universitat Jaume I

2D – Bi-dimensional

3D – Three-dimensional

GPL – Gnu Public License

Content

List of Tables

List of Figures

1. Introduction

World's growing urbanization is one of the key challenges we face today. Half of the world's population was already living in cities in 2013, and 6.3 billion people are expected to live in cities by 2050 – almost twice the current amount[1]. Furthermore, it is likely that the number of people living in cities by the end of this century worldwide will peak and then stabilize to an 80% (Harrison & Donnelly, 2011).

Despite the positive aspects this trend has – like higher economic growth – it comes with drawbacks such as higher levels of pollution, crime and traffic congestion. This situation has led to a higher awareness about the importance of understanding the cities and about that the efforts we devote in the following decades to improve our growing cities will shape our future society.

Around these ideas, the concept of a Smart City has emerged. A Smart City has its systems, services and operations improved to deal with the aforementioned challenges, thus benefiting its inhabitants and businesses, thanks to the use of digital and telecommunication technologies and the data collection and analysis they offer. Literature describes several characteristics of a Smart City; Figure 1 shows the ones most often described.

The technologies needed for letting the Smart Cities become real are increasingly available. New smaller and cheaper sensing and computing devices of diverse nature are making possible to add new capabilities to equipment and tools we use in our daily lives: from industrial environments to cars to household equipment. Their numbers are now large, and steadily growing, as well as their ability to connect to the Internet. Also, techniques for analyzing those data are now more powerful: Big data analytics (Gandomi & Haider, 2015).

In this context, the concept of "Internet of Things" (IoT) is being favored with greater attention as it show promise to become a key foundation of the Smart Cities'

[1] IEEE Smart Cities Initiative Working Group (http://smartcities.ieee.org/about.html, accessed on February 8, 2015)

technologies (Markovic et al., 2012). The information provided by IoT devices will allow for new applications to improve our current technologies. The "Carbon War Room Research Report" from 2013[2], based on recent studies, states that by 2020, the IoT devices will indirectly lessen the CO2 emissions in 9.1 giga-tons, cutting off a 15% of current world emissions.

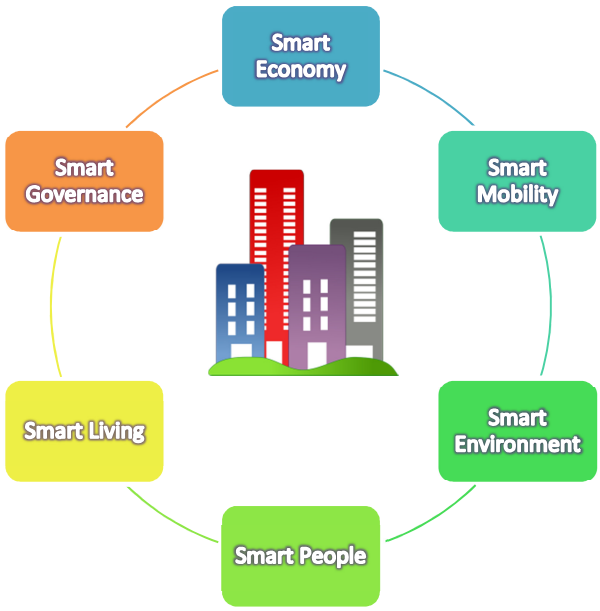


Figure 1. Most often mentioned characteristics of a Smart City.

A significant portion of those improvements will be a direct result of the Smart Mobility, one of the Smart City's characteristics (Figure 1). It is not only that more people are going to live in our cities, but the number of cars, public transportation demand, transportation infrastructure usage, and level of pollution will notably grow.

The growing number of cars has already caused problems for several years. Car owners in big metropolises are no strangers to traffic jams or to wasting a significant amount of time finding a suitable place to park. The Intelligent Transportation Systems (ITS) have emerged to provide solutions to those problems. These systems monitor the status of roads – and other traffic infrastructure elements – and act accordingly, e.g., by adjusting the traffic lights or providing guidance information (Shin & Jun, 2014). The Intelligent Parking Services are a part of the ITS that is

[2] http://www.carbonwarroom.com/sites/default/files/reports/CWR_SGP_Download_singless.pdf, [accessed on January 26, 2015]

becoming increasingly important and needed: Searching for parking does not only waste time, fuel, and increase pollution, but also notably contributes to traffic congestion. (Shoup, 2006) refers to studies of cruising in congested downtowns between 1927 and 2001, and states that between 8 and 74 percent of their traffic was a result of searching for parking.

This context has led several efforts to address those problems. Nowadays, there are systems – like the Streetline Smart Parking Platform[3] – running and providing suitable solutions. However, a notable amount of research is being done on Smart Parking and new, better solutions are needed.

1.1. The Smart Ways Project

The "Smart Ways" project mainly focuses on the Smart Mobility characteristic of Smart Cities. Specifically, it will create an efficient platform to address data collection, storage and analysis, and communication between sensors, car and people in order to foster a smart and sustainable mobility. This project is the collaboration between two Spanish universities – Universitat Jaume I (UJI)[4] and Universitat Politècnica de València (UPV)[5], and a company – Zed Worldwide (Zed WW)[6], each one with its own particular responsibilities. Smart Ways is both a research and development project.

The project also includes the development of a pilot system, applied to the UJI's campus, to test this platform and other required technologies and techniques. This pilot will provide the following functionalities:

- Locate available parking spots and guide the drivers to them using a suitable interaction, taking into account policies about efficient mobility and optimal occupancy.
- Dim public lights, depending on the presence of cars or people.

[3] http://www.streetline.com/manage-parking/product-portfolio/ [accessed on Febraury 9, 2015]

[4] http://www.uji.es/ [accessed on Janaury 26, 2015]

[5] http://www.upv.es/index-en.html [accessed on Janaury 26, 2015]

[6] http://www.zed.com [accessed on Janaury 26, 2015]

- Gather data about air quality.

In building this pilot, the Geospatial Technologies Research Group (GEOTEC)[7] from UJI is the major player. In particular, the parking-related part of the pilot system is the most relevant and has been called SmartParking. It will need to research on and develop several parts of a Smart Parking System, including interaction with the drivers, and the parking availability data acquisition, being the later relevant for other SmartParking subsystems.

By the time this document is written, the context of SmartParking is as shown in Figure 2. SmartParking stores the information of parking places and provides the required services for querying or updating that information. Sensing devices – or a software layer accessing them – will use those services to inform about changes in the status of each parking place. In turn, client applications can consume those services. Despite sensors represent the primary data input for SmartParking, they are not the only data source. SmartCampus' services provide both GIS data and processing capabilities.

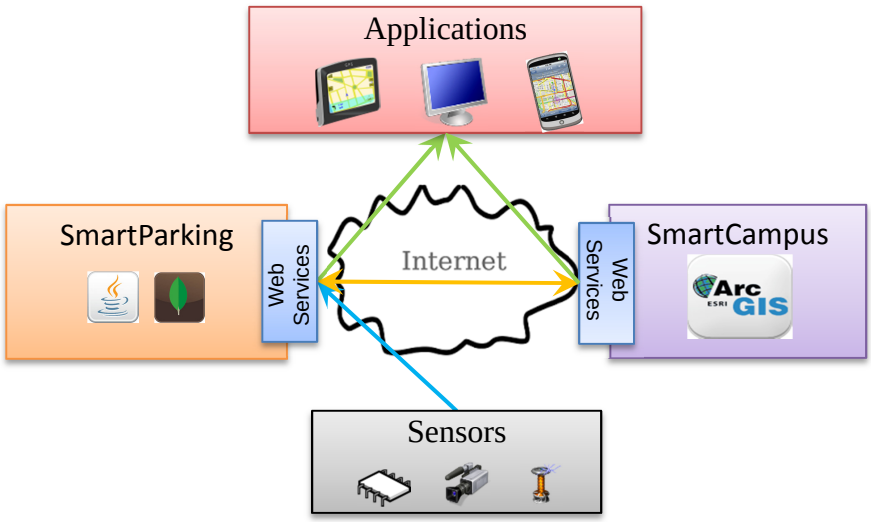


Figure 2. SmartParking's context.

SmartCampus[8] is known as a map-based web viewer for integrating and consulting in a unified and homogeneous way the university information of UJI (Benedito-

[7] http://www.geotec.uji.es, [accessed on January 26, 2015]

[8] http://smart.uji.es/, [accessed on February 19, 2015]

Bordonau et al., 2013). However, it is not just a web viewer. Based on an ArcGis Server©[9], it offers services that provide SmartParking and any related applications with the required geographic data and processing. SmartCampus' services provide much information about the university. For SmartParking, particularly important services (needed or already implemented) are:

- Road network as well as car routes.
- Pedestrian routes.
- Building and campus entrances.
- Visualization (Maps), including information about parking places.

As explained before, SmartParking is still in development. The programming language used on it is Java, which also defines the available frameworks used in its development. There are already implemented components – like the services for storing and querying parking availability data or for the communication with SmartCampus. Other components, like the one responsible for choosing a parking spot for a driver or guide him/her there, are not implemented yet. More importantly, the parking data acquisition is at an early stage.

The work of (Idris et al., 2009) reviews some smart parking systems and their technologies and shows 14 different technologies for vehicle detection. SmartParking will consider some of them. Electrical and magnetic sensors will eventually be deployed in the campus and cameras could be used and their output processed using computer vision techniques. Furthermore, other alternative parking data availability methods can notably improve results, or even make them possible in early development stages.

However, these options for capturing parking availability data are still being evaluated. Additionally, once the choice is made, its deployment could take considerable time. The characteristics of parking slots in UJI's campus make the technologies for vehicle detection particularly important: Parking places are on-street

[9] ESRI solution for geo-information servers, http://www.esri.com/software/arcgis/arcgisserver, [accessed on January 26, 2015]

slots; anyone can park in any of them for free; and there are no divisions into lots or barriers controlling access.

In order to momentarily provide the data that vehicle detection technologies will eventually supply, a parking occupancy simulator was suggested, thus emerging the primary goal of this thesis work. For the produced data to be valuable, it cannot be randomly generated, as it would be too far from the actual occupancy of parking spots found anywhere. Instead, the decision was to use a modeling approach.

In general, models are simplified abstractions of real systems. In this particular case, it was decided that modeling and simulating all properties of sensors and their communication specifics or a very realistic driver behavior are not requirements for the current status of SmartParking. The primary requirement is thus a software program able to produce “simulated” parking availability data for the UJI’s campus that may resemble the actual parking occupancy seen in UJI’s parking spaces.

However, due to the current state of development of SmartParking, other requirements were also attractive. Implementing the simulation software presented several opportunities. It would enable to test available services of SmartParking, to try different implementations of the component for selecting and assigning available parking places, and to develop and test the usage of GIS data and services from SmartCampus. Thus, the basic model requirements, the restrictions that the project imposes and general software development considerations shaped the goals of this thesis work.

1.2. Thesis Objectives

As explained before, the desired simulation does not only require a parking model: It is also intended to stay running and supplying data to SmartParking. As such, it needs to be implemented as an application able to communicate to both SmartParking and SmartCampus and to allow testing the SmartParking’s responsible for searching for and assigning available parking places.

The previous requirements describe the need for a very precise parking simulation. However, a more general issue arises from this context: *How to create a parking occupancy simulation that uses GIS data and services from an external, online*

source? It is common that frameworks, libraries, or toolkits used for creating parking simulations, or the simulations themselves, use predefined GIS data files formats, and that all data are loaded at simulation initialization and do not change during the simulation run. The simulation implementation proposed in this work is different because of its ability, to some extent, to work with dynamically changing data provided by an external, online source.

Therefore, the **primary goal** of this thesis work is *to create an application able to simulate the parking occupancy in the UJI campus, while consuming GIS data and services from SmartCampus, feeding the simulation output data to SmartParking, and allowing testing the SmartParking component for searching available places*. The development team of SmartParking will use this simulation software for testing purposes in these early development stages, while sensors' input is not available and SmartParking is evolving.

Additionally, the primary goal is subdivided into minor goals that are listed as follows:

- Study the parking modeling's state of the art.
- Define the parking model to be used in the intended simulation.
- Create or prepare the needed GIS data and services.
- Study modeling and simulation tools suitable for this work and choose one for implementing the model.
- Design and build the simulation.
- Build an application that wraps the simulation.

Those activities can be, in turn, divided into smaller steps, but for the sake of simplicity those smaller activities are addressed later in other sections of this work.

1.3. Document Organization

This Introduction chapter has already presented the context in which this work was born, its motivation and goals. The following chapters are Background, Parking Model, Parking Simulation, Discussion and Future Work, and Conclusions, in that order.

The Background chapter makes a literature review, addressing key concepts needed for the rest of the document. From Intelligent Transportation System to Parking Modeling, definitions and previous works provide foundations for understanding the vocabulary and the decisions explained in later chapters.

The Parking Model chapter explains the model devised for parking at UJI. It provides details on the represented system's dynamics, entities and attributes. Also, data collection is briefly described.

The Parking Simulation chapter presents the implementation of the model described in the previous chapter. The implementation choices are evaluated, and the final design is presented. Also, the output data evaluation and the web control application are described.

The Discussion and Future Work chapter describes limitations; difficulties found during the work and suggested improvements. Finally, the Conclusions chapter resumes the most relevant achievement of this work.

2. Background

The world economics has fostered an increase in the number of cars for decades. On the other hand, cities, which have limited space resources, have not grown as fast as car production. These facts have made parking a concern source for drivers and environmentalists, as finding a vacant parking spot in a city or urban context often leads to unnecessary time and energy consumption, as well as greater CO2 emissions.

Applications that help drivers searching for parking can provide solutions to the aforementioned problems, by, for example, offering predictions about future occupancy, adjusting parking pricing or providing drivers with valuable suggestions on available parking choices (Idris et al., 2009). Systems like the Streetline Smart Parking Platform[10], the PARKO products[11] or the Smart Parking Ltd products[12] are notable examples of what parking applications have achieved so far.

Research on parking systems and searching behavior is already four decades old. During this time, it has explored several aspects of the parking search, reservation, politics and pricing, among others. In particular, significant efforts have focused on trying to discover rules for describing how drivers search for parking (Teodorović & Lučić, 2006).

Computer simulations are increasingly used in science. It is, in part, a result of the benefits they provide – they help in evaluating the feasibility and correctness of scientific or engineering models and designs – and the high computational power of today's computers – which let run very complex models and get the results in a reasonable time. In particular, the research on transportation and parking, as well as the parking applications, have obtained significant benefits from modeling and simulation, since experimenting with traffic in real environments is not usually practical (Kotusevski & Hawick, 2009).

[10] http://www.streetline.com/manage-parking/product-portfolio/

[11] http://www.parko.co.il/product/

[12] http://smartparking.com/solutions/index.html

Agent-based modeling and simulation (ABMS) is increasingly getting more interest in a broad range of applications from different areas and disciplines (Macal & North, 2010), including parking applications. Parking models can help simulate possible scenarios and explore different design and implementation choices, thus helping in decision-making. In these models, individual agents (drivers) are at the core of the simulation, a high temporal and spatial resolution can be achieved, and changes can be made from the agent's perspective – e.g., search time or walking distance – or from the system perspective – e.g., parking pricing – (Waraich et al., 2012).

This chapter follows by explaining the concepts addressed before, especially those related to modeling and simulation. It will also make a brief review of traffic simulation systems, and parking models and simulation software, thus setting the necessary ground for the following chapters.

2.1. Intelligent Parking Systems

As previously stated in this chapter, the number of cars worldwide has significantly grown – and it keeps growing. However, it is not just more cars: Others transportation means and their related infrastructures have grown too. Nowadays, transportation systems are more complex than they used to be. These facts, coupled with technological developments, have led to the emergence of Intelligent Transportation System (ITS), which provides solutions to new and traditional problems of transportation.

The ITSs combine technologies – software, sensors, communication techniques and other tools – to improve transportation safety and efficiency, thus reducing adverse effects on the environment. They can achieve those benefits by providing the users with updated and accurate information, and predicting hazards and possible delays. ITSs aids to decrease congestion problems, air pollution and travel time delays (Sunmonu, 2011). Despite they can be grouped into several categories, as presented in (Sunmonu, 2011), the most relevant ones are the Advanced Traveler Information Systems, Transportation Management Systems and Vehicle Control Systems.

The Traveler Information Systems focus on providing real-time information to users. The Transportation Management Systems enable traffic managers to guide vehicles

based on real time information of roads and highways. Finally, Vehicle Control Systems provide improved vehicle information, control and handling.

Intelligent Parking Services are part of the ITSs. They provide several services like parking availability information, parking spot reservation and mobile parking payment (Faheem et al., 2013). (Revathi & Dhulipala, 2012) provides a comprehensive classification of parking systems.

There are several aspects to take into account for developing intelligent parking platforms. At least, the system should determine, or estimate, the parking occupancy, and provide the drivers with suggestions on the parking availability. Along the years, several studies have devised different approaches for implementing the functionalities of intelligent parking platforms. Some of the techniques used for intelligent car parking systems are expert systems (which uses agent-based technologies), fussy logic, sensors (including GPS) and communication with and among vehicles (Faheem et al., 2013).

The usual approach to determining the parking availability is to use (a network of) sensors. (Yoo et al., 2008) and (Shin & Jun, 2014) are good examples: They use sensors to collect the real-time status information of a parking lot and then transfer the data using a wire/wireless telecommunication network. However, a large sensor deployment for monitoring each single parking space is not feasible in every scenario. As stated in (Klappenecker et al., 2014), it could be viable to monitor the flows of entering and exiting vehicles in parking lots and to estimate the current occupancy based on those flows and the capacity.

The most known parking service is the parking guidance and information (PGI) systems, which provide drivers with dynamic parking information displayed in public ways or disseminated through the Internet. PGI systems rely on vehicle detection and parking space monitoring (Cheung, 2007). These systems have proven to provide several benefits, including increasing the chances of parking near the destination and decreasing the average trip time, thus aiding drivers while they are approaching their destinations (Teodorović & Lučić, 2006).

The work of (Geng & Cassandras, 2013) exposes three drawbacks of PGI systems: The utilization increase of monitored parking spaces may cause higher traffic

congestion nearby; drivers may miss better parking opportunities if they just pay attention to the PGI system suggestions; and PGI systems do not remove the competition for the available spots, so that several drivers may try to get the same spots. There are approaches for dealing with those drawbacks: Buffered Parking Information Sharing (BPIS), e-parking, dynamical control of parking pricing – in case of paid parking, and others like in (Klappenecker et al., 2014), which proposes on-the-car calculations for estimating the parking availability probability.

BPIS systems communicate an intentionally reduced amount of available parking spaces to create a buffer able to relieve the congestions caused by several drivers trying to get to the same spots. Depending on the buffer size, congestions may still appear or the parking utilization may be low (Shin & Jun, 2014).

The ideas of parking spot reservation and dynamically controlling the parking pricing have been around for several years (Teodorović & Lučić, 2006). The e-parking platforms extend PGI systems by enabling the reservation of parking spots (Rodier & Shaheen, 2010). Several e-parking platforms has already been implemented, as described in (Wang & He, 2011). A recent proposal for a parking reservation system is seen in (Shin & Jun, 2014). The dynamic control of parking pricing typically involves a negotiation between drivers and parking suppliers. (Chou et al., 2008) and (Li et al., 2009) go deeper on this subject using agent technologies.

2.2. Modeling and Simulation

Modeling has been used as a tool for research activity during several decades. As stated in (Wainwright & Mulligan, 2005), modeling can prove to be quite useful for understanding observations and developing and testing the theories underlying them. Additionally, a model can have several goals, including being a tool for simulation and prediction and a means of communicating science and its results. (Sokolowski & Banks, 2008) provides several advantages of using modeling and simulation, as well as applications and application fields.

Models are abstractions of reality. In these abstractions, only components of the system that the modeler considers to be significant are represented. Thus, a model

highly responds to its goals and the modeler perceptions. (Railsback & Grimm, 2011) states that a model is a “purposeful simplification of a system”.

Models are highly appreciated when they are used for system simulation and/or prediction. If it is not possible to study the system via analytical methods, a simulation approach is required (Law & Kelton, 2000). A simulation is the run of a model – likely repeatedly over time and perhaps needing some adaptation, to see how the system states change. Thus, a simulation is intended to mimic the (dynamic) behavior of the modeled system (Ingalls, 2002).

When doing simulations of a model, model parameters are crucial. Parameters are values that may vary between different simulation runs, but they stay constant for one run, enabling to try different flavors of the model. Some parameters are measured in the field and others result from model calibration (tuning), which is based on the comparison of model output with actual measurements. Good models are those that have a high realism with lower model and model parameters complexity. Calibration and other parameter related processes are well explained in (Wainwright & Mulligan, 2005).

Simulations are useful for obtaining an understanding of the outcome of the modeled processes over time and space. Thus, models can be used to make extrapolations, serving as means of prediction, retrodiction or near-term casting (Wainwright & Mulligan, 2005). Also, simulations are useful for testing several alternatives and see how a model’s components or subsystems interact (Carson, 2005).

There are several classifications of simulations and their models. (Sulistio et al., 2004) presents a comprehensive taxonomy of simulation tools. Figure 3 shows what they called the simulations’ taxonomy, which reflects the basic properties of any simulation. Other authors consider other simulations types, including the distributed simulations.

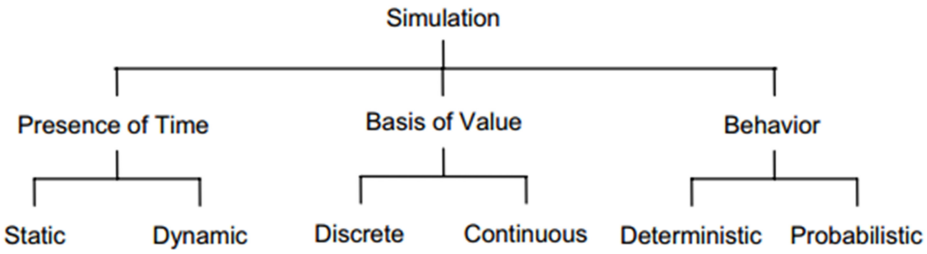


Figure 3 Simulations' taxonomy. Source: (Sulistio et al., 2004)**.**

(Sokolowski & Banks, 2010) presents a more comprehensive classification regarding prominent model types and simulation paradigms. For model types, it considers physics-based, finite element-based and agent-based models, among others. As for simulation paradigms, it shows the Monte Carlo, continuous and discrete-event simulation paradigms. The model type and simulation paradigm chosen for creating a simulation of a system depends on its nature.

The physics-based models rely on mathematical equations that are derived from physics principles, while the finite element modeling is based on the decomposition of the system or large components into smaller and easier to model pieces. Agent-based models consist of agents defined as autonomous entities that interact with their environment or other agents in order to achieve specific goals.

The Monte Carlo simulation paradigm repeatedly uses random input values sampling to see how the output varies, in order to find the relation between the input and output and thus model the system behavior. In the continuous paradigm, the system is modeled using differential equations of continuous functions of time, while the discrete-event paradigm uses discrete functions in time to represent system states.

(Sokolowski & Banks, 2008) also makes a distinction between simulation programs depending on whether they run independently of the modeled system: Stand-alone and integrated simulations, being the later the ones used to "enrich and support real systems".

Despite models and simulations particularities, they share the basic iterative process of creating a model and its related simulation. (Sokolowski & Banks, 2010) also describes four phases of this process: modeling, coding, execution and analysis.

These phases are intended to be used in iterations (cyclic) and using the specific technologies associated to each.

Regarding analysis, (Sokolowski & Banks, 2010) and (Wainwright & Mulligan, 2005) describe important steps in the development of models and simulations: Verification and validation. Verification is the process of checking whether the simulation code accurately represents the model specifications. Validation is the process of evaluated the extent to which the model represent the real system, usually by comparing the simulation outputs to real system observations. There are several approaches for conducting verification and validation, as it can be seen in (Wainwright & Mulligan, 2005) and (Sargent, 2005).

2.3. Agent-Based Models and Simulation

The term "agent" in computer sciences is widely used. Many computer science fields have used software agents: networks, control systems, artificial intelligence, human-computer interaction, distributed and concurrent systems, mobile systems, and transportation applications are some examples. According to (Chen & Cheng, 2010), behind this growing success lies the fact that its inherent distribution eases the decomposition of the system into multiple interacting agents, especially if some criteria are met – like when the problem domain is geographically distributed.

An agent, in this context, is a computer program capable of acting autonomously in a dynamic environment. An agent has a position and interacts with other agents (Luck et al., 2005; Zambonelli & Parunak, 2003). Among agent technologies, the intelligent agents (with artificial intelligence included) are the most relevant. Since 2005, there is an IEEE Computer Society standards organization for agents and multi-agent systems: FIPA[13].

When compared to the previously described technology of software agents, the agent-based modeling and simulation (ABMS) is relatively new. In recent years, however, it has attracted substantial attention from the modeling and simulation community. When searching for papers on simulation or modeling, it appears evident

[13] The Foundation for Intelligent Physical Agents (http://www.fipa.org).

the impact of agent-based modeling on this field. (Macal & North, 2009) supports these ideas by stating “the systems that we need to analyze and model are becoming more complex in terms of their interdependencies.”

The agent-based modeling is a computational modeling technique where individual system components (agents) and their behaviors are described (coded) as entities that are both autonomous and unique. Those agents may interact with each other and the environment locally, and exhibit adaptive behavior. This approach allows for studying individual behavior (its diversity) and the emergent group behavior resulting from aggregations of individual behaviors (Railsback & Grimm, 2011).

(Macal & North, 2010) describes, from a practical point of view, that an agent has several properties: Self-contained, modular, uniquely identifiable, autonomous, self-directed, and with a state that varies over time. It also describes additional properties agents may have: social, adaptive, goal-directed (rational) and heterogeneous. These other features allow for more complex models with more intelligent agents.

It is worth to mention the relation between agent modeling and geographic information systems (GIS) before going on parking specifics. Models have been used for environmental studies for many years, where the spatial component is usually relevant and needs a representation, such as using cellular automata. (Sarkar, 2000) describes the concept and history of cellular automata and (Wainwright & Mulligan, 2005) provides insights into environmental modeling.

The space representation and the GIS techniques are strongly related to ABMS. Agents are located in an environment, usually a two-dimensional or three-dimensional space, and the agent acts on its surroundings and those surroundings influence the agent behavior. In turn, as (Johnston, 2013) states, a GIS can be seen as “a spatial modeling tool that stores, display and analyzes data on spatial relation”. As a result of this relation, several modeling tools and systems have emerged to support the creation of models having both flavors. (Crooks & Castle, 2012) provides key criteria of some toolkits (Swarm, MASON, Repast, StarLogo, NetLogo, AgentSheets, and AnyLogic) that help finding the proper one for building a model. However, they do not describe another important option: the Agent Analyst

extension of ArcGis, which is focused on agent modeling as ArcGis applications (Johnston, 2013).

2.4. Parking Modeling

Modeling and simulation have aided transportation systems development for decades, and it has become more apparent as they have become more complex due to infrastructure and interdependencies issues. It is far more convenient to see the effects of a change in a model than, for example, in an actual subway network. It is notorious the growing use and development of traffic simulations and traffic simulator software in recent years.

Traffic simulations, and the software to create them, can be divided into macroscopic simulations, microscopic simulations, or hybrids of those two. Macroscopic simulations cover large areas and are useful for exploring changes in networks whose complexity is high. They employ techniques like statistical dispersion or freeway traffic models. On the other hand, microscopic simulations are used to study smaller areas with greater details. They employ techniques like cellular automata, multi-agent simulation, and particle system simulation (Kotusevski & Hawick, 2009; Sokolowski & Banks, 2008).

(Maciejewski, 2010) makes a comparison of three prominent microscopic traffic simulators (TRANSIMS, SUMO, and VISSIM). TRANSIMS was recently used for a parking work in (Guo et al., 2013). For macroscopic simulators, though used for microscopic ones too, one of the most relevant is MATSim, which will be briefly described in Chapter 4.

There has been numerous research works regarding parking behaviors of drivers and parking management's efficiency improvement. They initially focused on understanding and replicating the parking choice behavior (Geng & Cassandras, 2013), but they are also used to study the effects of changes and properties like reliability, to predict parking availability, and to implement price negotiation between drivers and parking.

As for parking management improvement, (Benenson et al., 2008) created a model to simulate parking in a city and used it to study the impact of adding more parking lots

in a residential area. They implemented the model, agent-based and spatially explicit, as an ArcGis application and named it PARKAGENT.

(Klappenecker et al., 2014), based on (Caliskan et al., 2007) and others, focused on improving the prediction of available parking lots, using a homogeneous Markov model to cope with problems of PGI systems. Instead of monitoring each parking slot, they gathered other information, like the parking lot in-flow and out-flow of cars.

An important aspect of parking model is parking selection and the factors related to it, either from the viewpoint of the parking guidance system or the driver. For example, (Leephakpreeda, 2007) proposed a car-parking guidance model that considered several factors – like distance to building's entrances – for finding the best parking facility in a building. It used a fuzzy knowledge-based decision-making. (Geng & Cassandras, 2013) also proposes a parking guidance from the viewpoint of the car parking allocation, combining proximity to destination and parking cost, while ensuring an efficient overall parking utilization.

Drivers usually pursue clear goals while searching for parking. However, before they actually park, they evaluate different parking choices. This temporal sequence of alternatives strongly influences drivers' final decision (Teodorović & Lučić, 2006). That is why some research works have explored the driver's behavior in choosing parking place, providing parking choice behavior models. Various kinds of factors are considered in the parking choice under different environments, and the objectives of parking management are diverse (Shin & Jun, 2014). Usually, these works propose a utility function involving several factors that affect parking choice, like in (Jonkers et al., 2011).

2.5. Agent-based parking modeling

Agent-based modeling is a natural choice for modeling parking. Agents represent drivers that pursue goals while moving along a network of roads. Those goals usually include parking as near as possible to their destination, reducing search time and taking into account other parking characteristics like parking fees and type. Some of the research works on parking mentioned in previous sections have used agent-based

modeling. This section will briefly review some of them: PARKAGENT, SUSTAPARK, the TRANSIMS-based work of (Guo et al., 2013) and MATSim-based work of (Waraich et al., 2012).

PARKAGENT (Benenson et al., 2008) is an agent-based parking model for residential parking (applied to a neighborhood in Tel Aviv). Each agent tries to get to its destination – that is set at the beginning of the simulation – and follows fixed rules to search for parking. While searching for parking, an agent evaluates at each step the local parking availability and decides whether to park or continue driving, also taking into account factors like search time, walking distance, and parking costs. Using their simulations, the researchers showed the effects of additional parking supply on extreme values and average search time.

SUSTAPARK (Dieussaert & Aerts, 2009) focused on creating a simulation to study the effects of parking search behavior in traffic. They used an agent-based model where each agent (driver) has a day schedule (the required trips) and interacts with other agents in the traffic. Agents are spatially located using a cellular automaton approach. The searching strategy is particular to each agent and specified by rules. Specific parameters determine the initial strategy selection: access time, search time, egress time and expected parking fee. They used a discrete choice structure (Hess & Polak, 2005) and considered both on-street parking and off-street parking as alternatives. The model loads the roads and parking data from GIS layers. They obtained good preliminary results in their tests.

(Guo et al., 2013) describes an agent-based transportation model that improves a previous work about a methodology for the demand estimation and microscopic traffic simulation of university campuses. The parking search process is modeled using a sequential game-theoretic, neo-additive capacity model that takes into account drivers' psychological characteristics (attitudes towards parking availability in their most desirable lot). They integrated the model with the Transportation Analysis and Simulation System (TRANSIMS), using it as a traffic micro-simulation engine.

The work presented in (Waraich et al., 2012) is based on a previous one – made by two of the authors – that proposed a new parking search model intended to be built

on top of the Multi-Agent Transport Simulation (MATSim). One of the main advantages of that work over previous ones is that it considers several plans for each agent, i.e., a simulation run is used to improve other runs – one of the benefits they got from using MATSim, and thus can find a more realistic characterization. Their work deals with several parking search strategies and addresses factors of each agent that may impact the utility evaluation for each agent.

2.6. Chapter Conclusions

Apart from laying down concepts and work previously done in each section, the intention of this chapter has been to show the connections between smart transportation systems, smart parking systems, and agent-based modeling. Also, though briefly, this chapter has shown alternatives and relevant literature advice for choices about creating parking models.

3. Parking Model

The simulation required for this work has to be able to mimic the availability data of parking spots that sensing devices will eventually provide to SmartParking through its services. For simplification's sake, it is considered that there is only one input per parking spot, and they all have the same nature.

This chapter describes the definition of an agent-based parking model suitable for the needs of this work. This conceptual model defines goals, system elements, rules of agent's behavior, and the environment.

3.1. Drivers' Representation

The review of parking systems, modeling and simulations led to choose ABM for creating the required model. In this model, an agent represents a person driving within the UJI's campus. The agent starts at a UJI's entrance point and, at least, goes to its destination facility. More precisely, it wants to get to a particular entrance of that facility. The decision of representing facility's entrances is a result of buildings' characteristics in UJI: They are large, surrounded by roads and usually with entrances facing all those roads.

The simulation is meant to generate data for a "virtual week" period. For each day of that week, agents arrive at the environment and follow their activity plan for that day. After an agent has fulfilled its activity plan, it goes to an entrance point of the campus for leaving the simulation.

Model's agents are rational, meaning this that they pursue an explicit primary goal: An agent wants to park as near to its destination point as possible. The criterion for measuring proximity is walking time. This model takes inspiration from the work of (Dieussaert & Aerts, 2009) and (Benenson et al., 2008), mainly regarding model parameters and search behavior, with some simplifications and additions. The idea of activity plans resembles the one from (Rieser et al., 2015).

Each agent looks for a parking place following the rules defined in one of the two possible behaviors: "Guided" and "Explorer". An agent with a "Guided" behavior

relies on an external component for the parking place selection. If an agent has the "Explorer" behavior, the agent itself decides where to park. Search behaviors will be addressed later in more detail.

3.1.1. Agent attributes

Every agent has a daily activity plan. Those plans define when an agent arrives at the simulation, which its destinations are and the time it stays parked at each of them. Those data allows to consider situations where, for example, a person first goes to her office and then to a sports facility. In this work, these activity plans are also called agent profiles.

Attribute	Description
Maximum Walking Distance[14]	Maximum distance an agent is willing to walk from the place it parks to its destination.
Activity Plan	List of timings and destinations of an agent.
Entrance Point	Place where an agent starts in the simulation.

Table 1. Agents' attributes.

Agent profiles are meant to describe the most typical behaviors found in people from UJI, and define the types of agents present in the model. It means that agents that belong to the same profile have the same destinations and similar arrival and departure times. Those times are not strictly the same to allow for variation (e.g., not everybody from an office arrives at the same time). Agent profiles will be addressed later in more detail.

All agents' attributes are shown in Table 1. Apart from the ones mentioned before in this section, agents may also have the Maximum Walking Time, which is meant to vary randomly in a range, to reflect people willingness to walk.

3.1.2. Agents' global parameters

The model also defines global parameters that affect agents (see Table 2). For simplicity, it is considered that all cars move inside the campus at the same speed, and the parking action is instantaneous (anyway, if a car is next to a parking spot, it

[14] This is only used if the agent is doing the search by itself.

can usually take that place without problems). This simplification is possible because modeling physical properties is not one of the goals of intended simulation. Additionally, the model includes a time unit parameter to allow controlling the simulation's speed that is later implemented. Next section addresses the other three parameters described in Table 2.

Parameter	Description
Cars Speed	Velocity, expressed as meters/time unit, of all cars inside the campus.
Time Unit	It states how many units (seconds) of real time that the simulation's unit of time represents.
Maximum Walking Distance Range[14]	Value range where the Maximum Walking Distance value of an agent is chosen.
Availability Reduction[14]	Value of the availability statistic below which the agent decides to park even though it has not reached the closest point to its destination.
Critical Availability[14]	Value of the availability statistic below which the agent decides to park even though it has not reached the closest point to its destination.
Visibility Distance[14]	Distance at which the agent can "see" parking places.

Table 2. Agents' global parameters.

3.1.3. Agents' Search Behaviors

As mentioned before, the model considers two types of parking search behaviors. The "Guided" behavior is a direct result of one of the purposes of this work: To test the component from SmartParking responsible for finding available parking places and assigning them to drivers who want to park. Every time an agent needs to go to a destination in UJI campus, it relies on an external component that chooses the best available parking place for the agent destination. The agent then just needs to head to that parking place. As the agent always accepts the suggested parking place, that place is considered to be reserved, and it will not be offered to any other agent until the occupying agent explicitly release it.

On the other hand, an agent with the "Explorer" search behavior mainly depends on what it "sees" in its local environment. When the agent starts in the system, it drives to its destination following the shortest path along the road network. After it reaches

its maximum walking distance, it starts looking at the available parking places closest to it and evaluates whether to park in one of them.

At a given step, an agent decides whether to park following these rules: If it has just reached the maximum walking distance, it calculates the ratio of available to occupied parking places (availability statistic). For the following steps along its route, it tries to get to the final point of its route (the closest to its destination), using the *Critical Availability* and *Visibility Distance* parameter to decide whether to move forward or to park. When it decides to park, it does it at the parking place closest to its destination that it can "see".

If an agent reaches the last point of the route to its destination and it has not been able to park, it decides to move to another entrance to the target building. While moving, it keeps trying to find a place to park, but without considering anymore the maximum walking distance. If it is unsuccessful while searching for parking around a building, it starts searching around another building, and so on. If it is not finally able to park, it just heads to a campus entrance for leaving.

Before a "Guided" agent gets to its assigned parking place, an "Explorer" agent might get to and occupy that spot. When the "Guided" agent realizes of that situation, it asks for a new spot and heads for it. The situation described here might indeed happen for SmartParking's users once that it is operational.

3.1.4. Agents' Profiles

The main profiles of people coming to UJI's campus are current and former students, PAS, PDI, and enterprises' workers[15]. PAS is the way to call the university's administrative and service staff while PDI is for the academic and research staff. Regular students, PAS, and PDI represent the bulk of people coming to UJI on a regular basis, as expected from a university.

[15] UJI's main web page (http://ujiapps.uji.es/) provides links to some data and description about them. Accessed on February 10, 2015.

To base agents' profile on main people profiles is a natural choice: The PAS staff has typical working hours[16]; students usually have classes either in the morning or the afternoon; and, in general, the university's daily life goes from 08:00 a.m. to 10 p.m. on weekdays. Additionally, the profile a person belongs to may define the places that he/she is likely to go. For example, a student may go to a sports facility or the library before or after going to class.

Though difficult, it is possible to obtain the data regarding amounts and locations of people from each profile[17]. Using that information, it is feasible to describe most of the important groups of people that come to UJI's campus on a regular basis, in order to define agents' profiles for the model. Each group will belong to a people profile, and its members will usually follow a weekly activity plan. Annex A shows a suggestion for the basic profiles from which more detailed ones can be derived.

The final agent profiles must be templates for creating agents, but they should not define the actual numbers of each kind of agent. Not every person comes to the university in his/her car: Some come by bicycle, public transport, or just share the car with a classmate or colleague. The percentage of each people profile coming to UJI in their cars (and thus needing parking) should be estimated, e.g., using the survey suggested in Annex B.

3.2. Environment's Representation

Parking places, campus entrances, buildings entrances, and roads represent the environment in the model. Both campus and buildings entrances are points with some basic information, e.g., a description. The building entrances also have the building they belong to as an attribute. Annex C provides a picture of entrances and roads in the UJI's campus.

[16] A general description is found in: http://www.uji.es/CA/serveis/rec-hum/pas/callab.html. Accessed on February 10, 2015.

[17] Some online data is available at http://www.uji.es/bin/serveis/rec-hum/pas/rlt.pdf and http://www.uji.es/bin/serveis/rec-hum/pdi/rlt2010.pdf [Accessed on February 10, 2015]. Despite updated information was requested, that process of getting it takes long time.

The roads have the type of road and direction as attributes. They were defined as an ArcGis network dataset. However, their details are mostly hidden by the fact that they are not directly used in the simulation: Instead, the routing service provided by an ArcGis server is consumed. That implies that the model works with the representation of routes along those roads.

The GIS data for entrances and roads was done using ArcGIS[18] 10.1's ArcMap, in a laborious work. Entrance points were created in a completely manual way and for roads it was possible to import some information from OpenStreetMap[19] using the ArcGIS Editor for OpenStreetMap[20].

The data of parking spaces (also called parking slots) is the same as in SmartParking. Their data relevant for the model are position (they are points) and state (free, occupied or reserved). They also have other attributes (like id and type) that are not interesting for the model. The GIS data collection for parking places was not part of this work.

3.3. Chapter Conclusions

This chapter has described the proposed parking model. It has shown the required data, the elements' representation and how the agents, the core elements of the model, behave. Next chapter addresses the implementation of these descriptions into a simulation application.

[18] http://www.esri.com/software/arcgis [accessed on January 31, 2015]

[19] http://www.openstreetmap.org/ [accessed on January 31, 2015]

[20] https://github.com/Esri/arcgis-osm-editor [accessed on January 31, 2015]

4. Parking Simulation

When creating a computer simulation from a model, the model needs to be formalized into a specification that can be developed into a computer program. There are many simulation/modeling systems available to assist in the development stage of a model's simulation, and specifically for an ABMS. However, in some cases, the model implementation is also tied to restrictions imposed by the project that it is part of.

As previously mentioned in this document, the simulation is intended to temporarily replace the input that sensors will later provide through a software layer. This input is delivered to SmartParking using its already implemented web services. SmartParking provides the parking spot data (including its GIS data), and the SmartCampus (web) services supply the remaining GIS data needed for the simulation. Additionally, the simulation must serve for testing the component of SmartParking responsible for assigning parking places.

The characteristics mentioned above shape the requirements for selecting a simulation/modeling system for building the simulation intended for this work. The basic requirements are:

- Agent-based modeling support.
- Implemented in the Java programming language.
- Light, flexible and easy to integration into a bigger application.
- Basic GIS support.
- Free usage.

This chapter will explore the implementation requirements, options and decisions for building the simulation of the model described in the previous chapter. It will also address other details related to software design and the simulation usage for the SmartParking project.

4.1. Explored ABMS Toolkit Choices

In general, there are two kinds of simulation/modeling systems that assist in the development of agent-based simulations: toolkits or software (Crooks & Castle, 2012). Toolkits provide much more flexibility than fixed software. Also, they reduce the burden that modelers face when programming, compared to creating a simulation from scratch, thus allowing the modeler to devote more efforts on model's definition. (Crooks & Castle, 2012) also states that there are more than 100 toolkits available for ABMS and provides some guidelines for choosing a simulation/modeling system.

For this work, the restriction of a Java-based implementation drastically reduced the list of potential toolkits. Furthermore, when considering basic GIS support, the toolkits Java Agent-Based Modeling (JABM)[21] and Ascape[22] were ruled out. Finally, the studied Java ABMS toolkits were MASON and Repast. Also, a traffic simulator implemented in Java was studied: MATSim. Next, a brief description of each is provided.

4.1.1. MATSim

The Multi-Agent Transport Simulation (MATSim)[23] Toolkit focuses on large-scale agent-based transport simulations. It is developed in Java and distributed under the GPL[24] license. MATSim allows for modeling demand, simulating traffic flows, iteratively controlling simulation runs and analyzing the generated output. It comes with the required tools to be used as is, but it can be integrated with custom applications. (Rieser et al., 2015) describes the key features of MATSim, which are summarized here:

- Large Scale, Fast, Agent-Based, Multi-Modal Simulation of Daily Mobility Behavior. MATSim can simultaneously simulate millions of agents that travel

[21] http://jabm.sourceforge.net/ [accessed on January 26, 2015]

[22] http://ascape.sourceforge.net/ [accessed on January 26, 2015]

[23] http://www.matsim.org/ [accessed on January 26, 2015]

[24] http://www.gnu.org/copyleft/gpl.html [accessed on February 15, 2015]

using different modalities and the represented road networks can be as large as hundreds of thousands of road segments.

- Modular Approach, Versatile Analyses and Simulation Output. MATSim provides a framework consisting of several modules that can be combined and replaced by custom implementations. It provides a detailed output from the traffic simulation and aid tools for data visualization.

The MATSim's simulation process tries to optimize the daily plan of each agent. Once those plans are executed in a microsimulation, they are scored and assigned a utility value, which usually is related to travel time. That value, which agents try to maximize, is used for creating new plans after each iteration. Thus, each agent has several plans to consider, discarding those with lower utility.

In MATSim, the input data is supplied with a particular format in individual XML files. Typically, there are files for representing configuration options, the network, agents, destinations, activity plans and transportation means.

MATSim also has several extensions, as can be seen on its website[25]. One of those extensions is for parking, but no documentation (apart for the plain mainly poor-explained Javadoc) is found, nor its maintainer mentioned.

4.1.2. MASON

The Multi-Agent Simulation Of Neighborhood (Mason)[26] Toolkit is a Java, fast, discrete-event, multi-agent simulation library, designed to build custom-purpose lightweight and large simulations. It is free, open-source, and provided under the Academic Free License[27] version 3.

Apart from a model library, it provides an optional suite for visually controlling the simulation run, for 2D and 3D model output visualization and for creating dynamic charts, like histograms and line graphs.

[25] http://matsim.org/extensions, [accessed on January 26, 2015]

[26] http://cs.gmu.edu/~eclab/projects/mason/, [accessed on January 26, 2015]

[27] http://opensource.org/licenses/AFL-3.0, [accessed on January 28, 2015]

MASON includes a GIS-support extension called GeoMASON[28], which enables to use, import and export GIS vector data and allows the use of raster data in the creation of geospatial agent-based models. It helps modelers by providing a good, growing set of technical documents, demonstration models, and a well- explained Javadoc. Also, there are several publications (mainly from the time window 2010-2012) detailing the implementation and applications of MASON. Other features of MASON are:

- Models are fully separated from visualization, entirely serializable to disk and entirely encapsulated (which allows to run multiple models in parallel).
- MASON is modular and consistent, carefully written looking for efficiency.
- MASON has a high-quality random number generator (Mersenne Twister) and uses the fastest known implementation of it.
- Models are largely duplicable (provided that the same parameters and random number seed are used).

4.1.3. Repast

The Recursive Porous Agent Simulation Toolkit (Repast) Repast Simphony 2.2[29], released in 2014, is a tightly integrated, richly interactive, cross-platform Java-based modeling system. It is distributed under the "New BSD" license[30].

Repast supports the development of flexible agent-based models, which could be targeted for both workstations and computing clusters. Different forms are available for developing Repast Simphony models, including Java, Groovy[31], the ReLogo dialect of Logo (Ozik et al., 2013), or point-and-click statecharts, all of which can be fluidly interleaved.

Repast Simphony provides all the core functionality of previous Repast implementations (done in different languages, and still provided) but programmed in

[28] http://cs.gmu.edu/~eclab/projects/mason/extensions/geomason/, [accessed on January 26, 2015]

[29] http://repast.sourceforge.net/repast_simphony.php, [accessed on January 26, 2015]

[30] http://repast.sourceforge.net/license.php, [accessed on January 29, 2015]

[31] http://groovy.codehaus.org/, [accessed on February 16, 2015]

Java. There are useful papers on models done with Repast, as well as proper documentation for creating simple models. It also has several options for space representation, including GIS support. There have been models that have explored these aspects and provided examples on importing shapefiles, and creating and agents and moving them around a road network.

Among other features[32] of Repast Simphony, it is worth to mention:

- A concurrent, multithreaded, discrete event scheduler.
- Libraries for genetic algorithms, neural networks, regression, random number generation, and specialized mathematics.
- Model serialization and restoration.
- Built-in systems dynamics and social network modeling tools.

4.2. Selected Simulation Toolkit

When studying the main characteristics of each option, MATSim seemed a likely winner, mainly because it directly provides traffic simulation and activity plans for agents. However, it was disregarded because of the need for substituting a component of MATSim itself in order to cope with GIS data in a format different from the predefined one and the difficulties for creating an integrated simulation – an application incorporating a simulation made with the toolkit.

For Repast Simphony and MASON, simple test models were built. Repast surpasses MASON regarding aid tools for visual design, visualization, and data analysis. However, MASON makes the process of creating an integrated simulation easier. Also, MASON is more light-weighted, both in demanded resources and application size. Quoting (Luke, 2014), *"MASON still remains very much the high-performance, high-hackability leader among systems in this area"*. The referred area is the "ultra-light" multi-agent simulations. These facts, along with its support for scalability, led to choose MASON for the model implementation.

[32] https://repastsimphony.wordpress.com/tag/what-is-repast-simphony/

4.2.1. Building Simulations with MASON

MASON separates the model (the simulation itself) and the 2D or 3D visualization, which allows for running the model without visualization. Figure 4 shows a general overview of the MASON architecture.

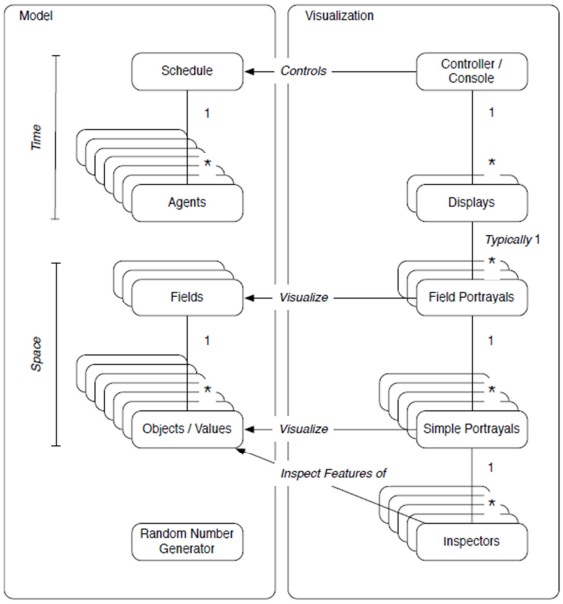


Figure 4. MASON architecture. Source: (Luke, 2014).

Each MASON model is entirely encapsulated in a *SimState* object. MASON uses a discrete event schedule to handle the representation of time. With this schedule, which is in turn managed by the *SimState* object, it is possible to schedule various agents that will be called at some time in the simulation. Agents are represented by objects that implement the *Steppable* interface.

For space representation, a MASON model contains one or more fields to which agents are added. MASON provides a number of built-in fields, such as networks, continuous space, and grids. Additionally, the GeoMASON extension (Coletti, 2013), offers particular fields for representing GIS data, also encouraging the use of JTS Topology Suite[33] for simple GIS operations.

[33] http://www.vividsolutions.com/jts/main.htm, [accessed on February 16, 2015]

The visualization in MASON is achieved using Field Portrayals and Displays, based on the information held in the fields. Again, the GeoMASON extension provides Field Portrayals and other supporting classes for visualizing GIS information.

Summing up, for creating a basic simulation with MASON, a modeler needs to create an instance of a *SimState* subclass, initializing its random number generator with a seed; create objects (agents) implementing the *Steppable* interface; and add such agents to fields and schedule them.

4.3. Model Implementation Design

MASON provides a solid ground and useful utilities to build the required simulation. However, MASON is meant for creating many types of agent-based simulation. The development of the intended simulation required to create a software program that harnessed the MASON's benefits – e.g., model representation, including time and space, and visual utilities – while incorporating the requirements imposed by the needs of the SmartParking project.

The design of the model implementation applies an adaptation of the well-known multilayered or n-tier software architecture, where concerns are separated into stacked layers, being the most popular the three-layered one (Fowler, 2002). As presented in Figure 5, the simulation is organized into three layers. The *Data* layer takes care of obtaining the necessary information for running the simulation. The *Model* layer represents the actual MASON-implemented model along with the parking places assigner component. Finally, the *Consumers* layer has the components that handle model output and simulation control. *Data* and *Model* layers have facades (Gamma et al., 1994) that consume the *Model* and *Consumer* layers respectively.

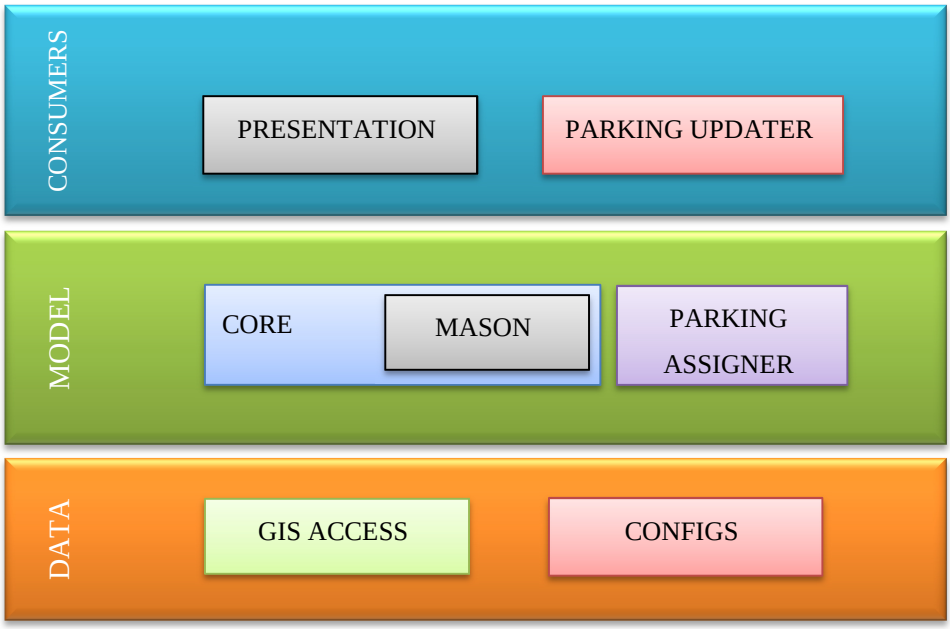


Figure 5. Simulation layered organization.

4.3.1. Data layer

The simulation requires knowing the model parameters, the agents' plans and the environment description. Agents' plans are set in an XML file that describes the characteristics of each agent profile, which include the timing and destinations of an agent's plan and the amount of agents of each profile. Agents' plans are loaded at simulation start. Annex D provides details on the agents' profiles configuration file.

GIS data and services are, mainly, consumed from an ArcGis Server. The server provides them as Web Feature Services[34] and Route (Network Analysis) Services[35], accessible through the ArcGIS REST API[36]. The obtained data are the building entrances and the campus entrances. The consumed services are the car and pedestrian route calculation services.

[34] http://resources.arcgis.com/en/help/arcgis-rest-api/#/Feature_Service/02r3000000z2000000/, [accessed on January 29, 2015]

[35] http://resources.arcgis.com/en/help/arcgis-rest-api/#/Solve_Route/02r3000000q3000000/, [accessed on January 29, 2015]

[36] http://resources.arcgis.com/en/help/arcgis-rest-api/#/The_ArcGIS_REST_API/02r300000054000000/, [accessed on January 29, 2015]

This layer is also responsible for communicating with SmartParking for getting the information regarding the parking places. Those data are obtained from a web service and contain all relevant information for spatially locating the parking places and for later communicating changes in their statuses to SmartParking.

4.3.2. Model Layer

This layer contains the "heart" of the simulation. Figure 6 provides an overall view of its design. As expected from a MASON-based simulation, the core classes are based on MASON classes. For simplicity, the classes of MASON (and GeoMASON) that represent fields are not shown in Figure 6, though they are important and used for locating the agents in the space.

The updaters are the mechanism chosen for notifying changes in the model. They follow the Observer software design pattern (Gamma et al., 1994). If a component is interested in receiving notifications every time the model changes, it must subscribe to the appropriate updater.

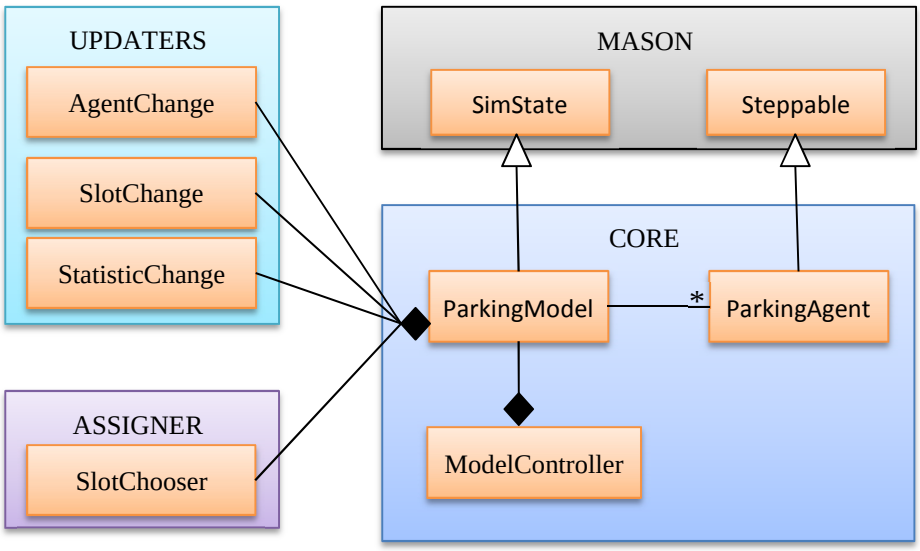


Figure 6. Main classes of the Model layer and their relation.

The model treats all agents in a similar way. Though they share similar data, their behavior makes them different. This differentiation is achieved through the basic parking agent specialization, as shown in Figure 7, which uses the Abstract Factory software design pattern (Gamma et al., 1994).

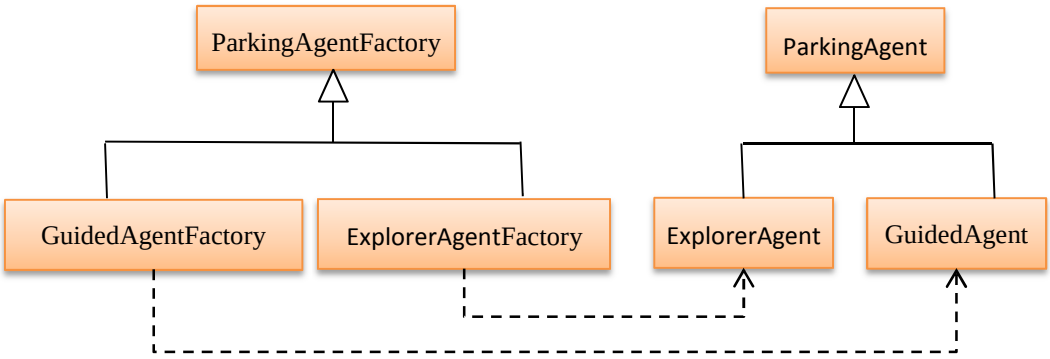


Figure 7. Giving behavior to agents.

The other two relevant classes in this layer are the one responsible for choosing the best available parking space for an agent – if the agent requests it – and the one responsible for controlling model creation and execution. The implementation allows for doing the parking spot selection using walking distance via the aforementioned pedestrian route service or via geometric calculations. The model controller is the responsible for setting up the necessary relations (e.g., updaters) according to the environment in which the model is to be run. Figure 8 shows examples of model controllers.

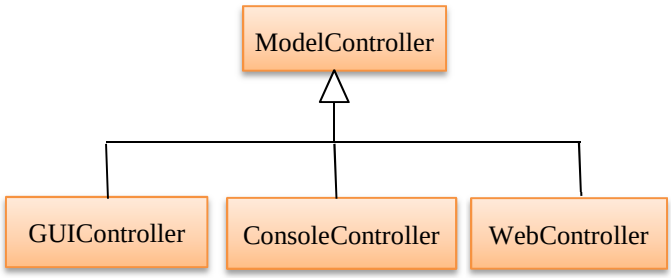


Figure 8. Controlling the model in different contexts.

4.3.3. Consumers Layer

As indicated by its name, this layer has components that consume services from the Model layer. The parking updaters are the ones responsible for communicating the changes in availability state of parking places to SmartParking. On the other hand, the presentation shows simulation output (parking, agents and statistics changes) and controls the simulation. So far, three implementations for presentation have been made: A desktop version that uses MASON visual utilities, a console version

intended to rapidly get results, and another one for controlling the simulation through a web interface. Those interfaces are described later in this chapter.

4.4. Simulation Output

The model's primary output is the parking availability data needed for SmartParking. However, for verifying the model simulation or studying the parking characteristics, it is not enough. It was required to explore agents' behaviors, to see whether they were moving along the roads and parking at the right places. Furthermore, the ability to access data useful for statistical study – like the traveled distance or the occurrence of certain situations – is a desirable feature.

For verifying the model, a desktop application that provides visual feedback on model' changes was created. This application wraps the simulation and displays changes in both agents and parking spots. Figure 9 shows agents that are parked (black), cruising for parking (blue) or leaving the campus (magenta). Additionally, it shows the available parking places (green) and the roads. It also allows controlling simulation speed and scaling, showing/hiding visual elements and videos creation.

Figure 9. Model output as seen in the desktop application.

The desktop application is suitable for visually analyzing the simulation behavior. However, visualizing the model's elements notably slows the simulation speed. Besides, in the desktop application the simulation's runs are controlled by the MASON's visual utilities. For avoiding those inconveniences, a console application that wraps the simulation was also created. This application focuses on collecting data useful for statistical analysis, and it can run the simulation several times.

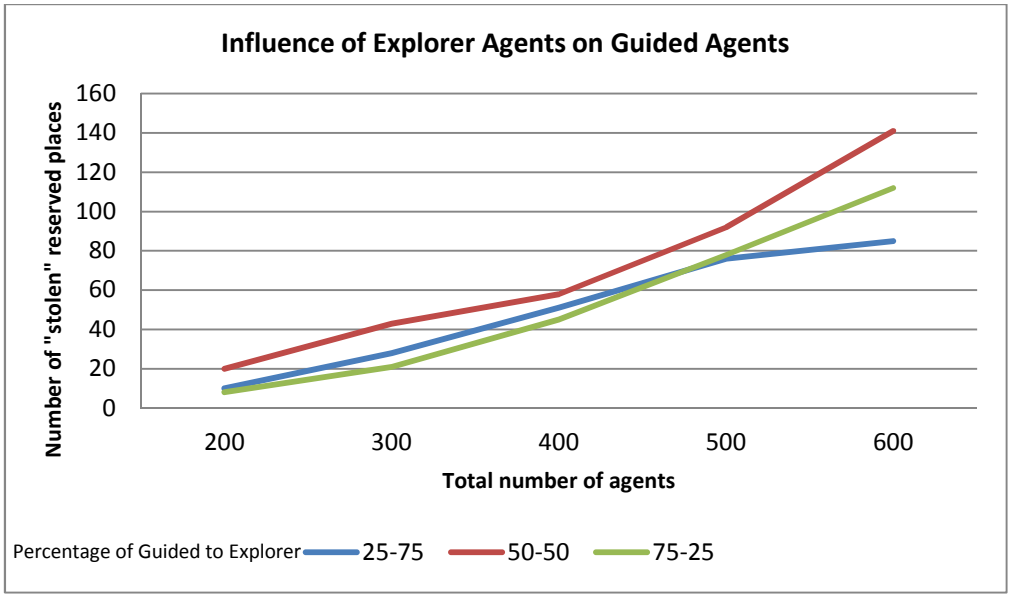


Figure 10. The "problem" of Explorer agents taking places reserved for Guided agents.

Figure 10 shows the results of exploring the situation when an Explorer agent occupies a parking place that is already reserved for a Guided agent. In this experiment, the agents try to park near a particular building. The chart shows that the occurrence of this situation grows with the total number of agents. However, it also indicates that this situation happens more when the amounts of Guided and Explorer agents are similar. This result highlights the idea that this situation needs both kinds of agents to happen: If the proportion of one of them is smaller, then it is less likely to happen. Annex E provides the results for the traveled distance for both kinds of agents and describes the situation for what these numbers were obtained.

In this section, the desktop and console wrapping applications has been addressed. Next section is devoted to the third option: the web interface that allows for remotely controlling the simulation that feeds parking occupancy data to SmartParking.

4.5. Web Interface to Control the simulation

Most of the simulations are run in a fast-paced way to rapidly have the output ready for performing analyzes. Despite that, the simulation toolkits usually provide visual tools for controlling them: They enable starting, pausing and stopping the simulation, speed control, output visualization and charts generation. While the simulation of parking availability data provides some of those elements, it was required to go beyond. The simulation is intended to be run, at least, for some days, continuously feeding in real-time data to SmartParking. In this regard, it is not used in the way most of the simulations are. Those facts led to the idea of creating a wrapper application that allows for remotely controlling the simulation. Figure 11 depicts the deployment organization.

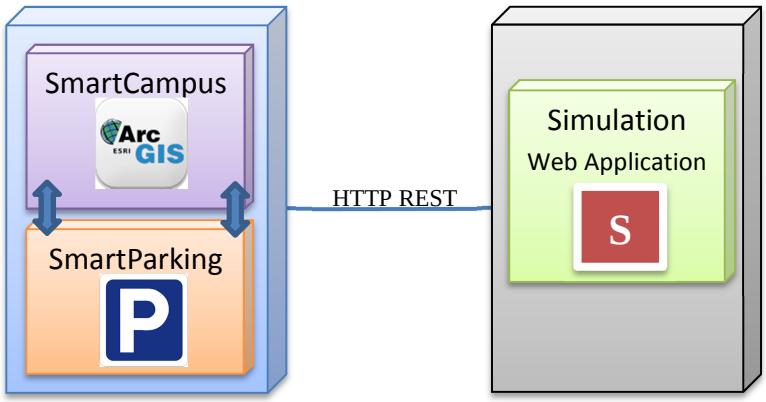


Figure 11. Deployment view of SmartParking-SmartCampus-Simulator.

For simplicity, the logical choice was to build the wrapper application as a web application. In a Java development environment, the one used for building the parking simulation, there are convenient platforms for web development.

Apache Tomcat[37] is a free web server chosen to host the application web application. It implements the Java Servlet and JavaServer Pages technologies[38] provided under the Apache License[39] version 2. It is very popular and despite its ease of use and

[37] http://tomcat.apache.org/ [accessed on January 28, 2015]

[38] http://jcp.org/en/introduction/overview [accessed on January 28, 2015]

[39] http://www.apache.org/licenses [accessed on January 28, 2015]

being light-weighted, it has been used to power many large-scale, robust web applications for different industries and organizations.

The wrapper web application was designed based on the well-known Model-View-Controller (MVC) architectural pattern for implementing user interfaces. The MVC pattern (Krasner & Pope, 1988) have been used for many years and current leading web application frameworks incorporate support to enforce it. Figure 12 shows how the pattern is applied to the wrapper application.

The simulation implementation is the MVC's model. The MVC's model and the MVC's controller interact via the simulation updaters and controllers (see section 4.3.2). The MVC's controller receives the user orders sent by the view and supplies updates to it. The communication between server and client sides (between the Controller and the View) is implemented using the REST (Richardson & Ruby, 2008) and JSON[40] standard.

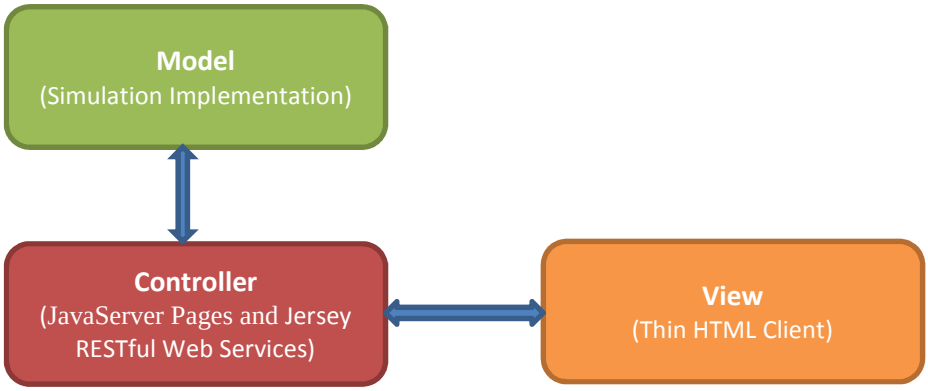


Figure 12. MVC applied to the simulation web wrapper application.

Java technologies used for implementing the web application include JavaServer Pages[41] and Jersey[42] RESTful Web Services for server-side; and Bootstrap[43] and

[40] http://www.json.org/ [accessed on January 28, 2015]

[41] http://www.oracle.com/technetwork/java/jsp-138432.html [accessed on January 28, 2015]

[42] https://jersey.java.net/ [accessed on January 28, 2015]

[43] http://getbootstrap.com/ [accessed on January 28, 2015]

ArcGIS JavaScript API[44] for client-side. JavaServer Pages technology was used as it provides a simplified, fast way to creating Java-based dynamic web content. It was decided to use REST web services to allow for flexibility in the View part and the Jersey framework is a recommended way to implement them. The aim of achieving a good look and responsiveness led to use Bootstrap, which is claimed to be a very popular HTML, CSS, and JS framework. The decision of using ArcGIS JavaScript API, apart from being a convenient, lightweight way to embed maps in web applications, is a result of consuming ArcGIS Web Feature Service (WFS)[45] provided by SmartCampus for parking places visualization.

The resulting web application has two interfaces: Simulation control interface and the comparison window interface. Figure 13 shows the control interface, where it is possible to start or stop the simulation, set simulation parameters and visualize the simulation output (parking places occupancy) while it is generated.

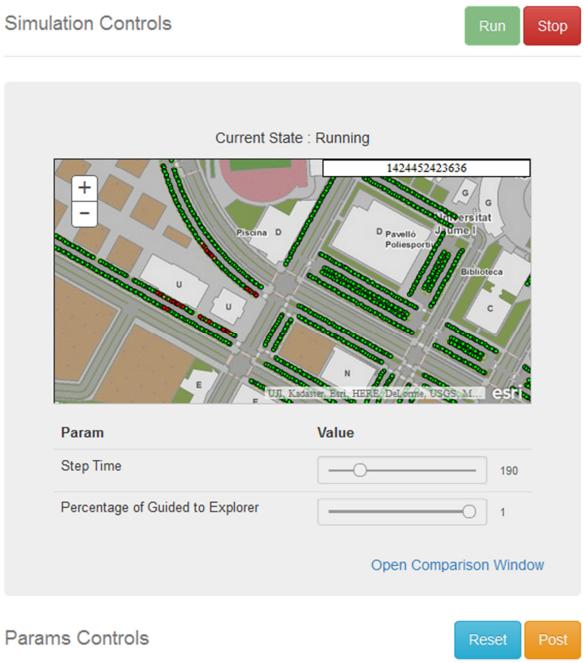


Figure 13. Simulation control web user interface.

[44] https://developers.arcgis.com/javascript/ [accessed on January 28, 2015]

[45] http://resources.arcgis.com/en/help/main/10.1/index.html#//0154000004mm000000 [accessed on January 28, 2015]

Figure 14 shows the comparison window interface, where it is both presented the parking occupancy as it is generated by the simulation and what SmartCampus actually has. This interface is required for the current, early development state of the SmartParking: Developed parts are still being tested, including communication between SmartParking and SmartCampus.

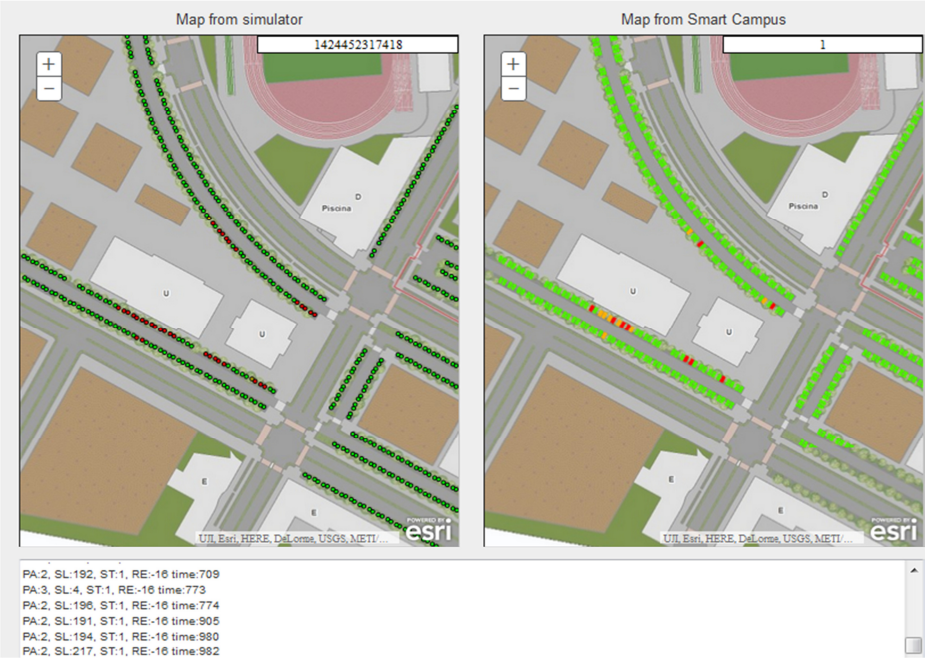


Figure 14. Comparison window of the web interface.

4.6. Chapter Conclusions

This chapter has presented the parking simulation based on the model defined in the previous chapter. It has described the implementation choices – toolkit needed to avoid building it from the scratch, and the reasons for selecting of one of them. The software design of the simulation application is presented, as well as simulation output, control, and visualization.

5. Discussion and Future Work

Along the years, parking simulations have focused on creating more and more detailed models to gain insights or make predictions. Those simulations (or toolkits for building them) use ad-hoc, format-specific data, and thus they are tied to them.

However, for situations where a high model fidelity is not a priority (the case explored in this work), it is possible to build simulations that focus more on flexibility. In this thesis, that flexibility is the ability to work with dynamically changing GIS data and easily customize some simulation elements. The ability to work with dynamically changing GIS data is achieved by consuming third-party (ArcGis Server), online services. It forced to implement the model under the restrictions imposed by the information provided by those services. The ease for customizing some simulation elements was achieved by using the proper design techniques. The achieved flexibility makes the simulation attractive for GIS-related software projects in an early stage of development, where their data and components are changing or evolving.

The current deployment context of the simulation applications creates lags in the simulation run. Every time a new car or pedestrian route is needed, the simulation makes a network request to the GIS server. It is affordable in the case of car routes because such requests are only made when the car starts moving, and, for some cases, they could be beforehand calculated, stored and later supplied to agents. For pedestrian routes, it was necessary to implement geometric calculations. Indeed, the pedestrian route service was mainly used for demonstration purposes due to its lags accumulation. A deployment configuration where the simulation application resides on the same machine as the GIS server might notably reduce those lags.

Human nature is much richer than a small set of rules, and much work has been devoted to studying the decisions that drivers make while searching for parking. The actual parking occupancy cannot be obtained using only the “Explorer” behavior implemented in this work, even if proper data regarding estimates on the parking demand and diverse enough agent profiles are used. The search behaviors mainly represent drivers unfamiliar with the campus, which do not use previously acquired

knowledge for finding a parking place. With this simulation, however, it is possible to easily add new behaviors in order to achieve a higher realism.

The current simulation implementation has been verified: It produces the results expected according to the model definition. Once a rich agent profile set and an improved set of behaviors are available, it is suggested to perform the model validation, i.e., to see if the simulation output matches the actual parking occupancy. It will require a campaign for collecting typical occupancy data and the study of validation methodologies.

Currently, the simulation and the model only consider free, on-street parking places, which is a result of being built for the UJI's campus case. However, drivers' behavior changes when facing different pricing alternatives for parking. For using the model in such a context, the price for parking could be incorporated into it as an attribute of parking places and taken into account for searching behaviors.

Other possible improvements to the current simulation are:

Better characterize the usage of UJI's campus entrances. In the current model, when an agent arrives at the simulation, it is randomly placed in one of the UJI's entrances. Estimating the actual usage of each entrance might improve the model output because the entrances notably influence the parking choices and the traveled distance.

Create a check-pointing tool. MASON provides mechanisms that can be used for saving the simulation state and later restoring it. However, simulation's state also depends on the states of parking places stored in SmartParking. Then, the suggestion is to create a tool that allows saving and restoring parking occupancy states for SmartParking and the rest of data for the simulation.

Create a visual tool for editing agent profiles. The XML configuration file provides the necessary flexibility to create the rich profile set needed for the simulation output to resemble the actual parking occupancy. When the amount of profiles is high, editing this file is cumbersome and error prone. A tool with improved editing utilities and rules checking would be useful.

Improve simulation data flexibility. The current simulation implementation consumes ArcGis Server services. Other routing and feature-providing services could be explored in order to build an implementation that is able to work with different GIS service providers.

Consider multi-destination parking. The current model assumes that an agent always wants to park as near as possible to its current destination. The model would be richer if it also considers the situation where an agent wants to park equally close to two destinations.

6. Conclusions

It is usual that simulations, parking simulations, and toolkits for creating them use predefined GIS data files formats, load the data at simulation initialization and those data do not change during the simulation run. The simulation implementation proposed in this work differs from them in its ability, to some extent, to work with dynamically changing data provided by an external, online source. This thesis work has shown the case where that ability is attractive and has described how to achieve it through an implementation that considers and consumes only data provided by online services.

That implementation resulted in an application for simulating the parking occupancy in the UJI's campus. It generates the parking occupancy information that sensing technologies will eventually produce for the SmartParking system. This application consumes GIS data and services from a GIS server (SmartCampus), feeds the simulation's output data to SmartParking, and allows testing the SmartParking's component for searching and assigning available parking places.

The work of creating the simulation application involved several activities, and has adequately fulfilled the intended goals.

The *study of the parking modeling's state of the art* enabled setting the required theoretical grounds and exploring ways of building the model. It remains as a theoretical review that, to some extent, is also useful for the SmartParking project.

The *defined parking model* properly represents drivers that move along a road network – according to their defined behavior – and the parking occupancy they create. It helped to understand the characteristics of the modeled system and to define the data, operations and parameter requirements for the simulation.

The *created or prepared GIS data and services* are adequate for the intended simulation. This data are not only useful for the simulation, but also for SmartParking and other future applications related to mobility in UJI.

The *simulation implementation* follows proper software design principles, enables flexibility regarding the data needed for the simulation, agents' behaviors, and

development of applications that use the simulation. It also allows testing the parking places assigner. The implementation is based on MASON, which was chosen after analyzing the requirements of this work and correctly *studying the modeling and simulation tools suitable for it*.

The main built *wrapper application* is a web-based application, and it allows for remotely starting and stopping the simulation, as well as setting parameters, thus enabling to properly control the supply of parking occupancy data to SmartParking. It was also created a desktop wrapper that uses MASON's visual utilities and allowed for implementation verification. A console version was also built for quickly getting results in order to obtain statistical information. This information showed effects of the interactions between agents (drivers) who use a smart parking system and those who do not, and the advantage that the first ones have over the others.

Additionally, the simulation has served for testing the SmartParking services and further developments in that system can be made under the assumption that there is already a source of parking occupancy data.

7. References

BENEDITO-BORDONAU, M., GARGALLO, D., AVARIENTO, J., SANCHIS, A., GOULD, M., & HUERTA, J. (2013). UJI Smart Campus: Un ejemplo de integración de recursos en la Universitat Jaume I de Castelló. In *IV Jornadas Ibéricas de Infraestructura de Datos Espaciales* (pp. 417–426). Toledo, Spain.

BENENSON, I., MARTENS, K., & BIRFIR, S. (2008). PARKAGENT: An agent-based model of parking in the city. *Computers, Environment and Urban Systems*, *32*, 431–439.

CALISKAN, M., BARTHELS, A., SCHEUERMANN, B., & MAUVE, M. (2007). Predicting Parking Lot Occupancy in Vehicular Ad Hoc Networks. *2007 IEEE 65th Vehicular Technology Conference VTC2007Spring*, 277–281.

CARSON, J. S. (2005). Introduction to Modeling and Simulation. In *Proceedings of the 2005 Winter Simulation Conference* (pp. 1643–1649).

CHEN, B., & CHENG, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, *11*, 485–497.

CHEUNG, S. (2007). Traffic Surveillance by Wireless Sensor Networks : Final Report. *California Path Program Institute of Transportation Studies University of California,Berkeley*, *1*, 161.

CHOU, S.-Y., LIN, S.-W., & LI, C.-C. (2008). Dynamic parking negotiation and guidance using an agent-based platform. *Expert Systems with Applications*, *35*, 805–817.

COLETTI, M. (2013). *The GeoMason Cookbook.*

CROOKS, A. T., & CASTLE, C. J. E. (2012). *Agent-Based Models of Geographical Systems*. (A. J. Heppenstall, A. T. Crooks, L. M. See, & M. Batty, Eds.) (pp. 8–10). Dordrecht: Springer Netherlands.

DIEUSSAERT, K., & AERTS, K. (2009). SUSTAPARK: An Agent based Model for Simulating Parking Search. *AGILE International …*, 1–11.

FAHEEM, F., MAHMUD, S. A., KHAN, G. M., RAHMAN, M., & ZAFAR, H. (2013). *A Survey of Intelligent Car Parking System*. *Journal of Applied Research and Technology*.

FOWLER, M. (2002). *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc.

GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J. (1994). *Design patterns: elements of reusable object-oriented software*. Pearson Education.

GANDOMI, A., & HAIDER, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*, 137–144.

GENG, Y., & CASSANDRAS, C. G. (2013). New "smart parking" system based on resource allocation and reservations. *IEEE Transactions on Intelligent Transportation Systems*, *14*, 1129–1139.

GUO, L., HUANG, S., & SADEK, A. W. (2013). A novel agent-based transportation model of a university campus with application to quantifying the environmental cost of parking search. *Transportation Research Part A: Policy and Practice*, *50*, 86–104.

HARRISON, C., & DONNELLY, I. (2011). A theory of smart cities. In *Proceedings of the 55th Annual Meeting of the ISSS* (pp. 1–15).

HESS, S., & POLAK, J. W. (2005). *Mixed logit estimation of parking type choice*. *83rd Annual Meeting of the Transportation Research Board, Washington, DC*.

IDRIS, M. Y. I., LENG, Y. Y., TAMIL, E. M., NOOR, N. M., & RAZAK, Z. (2009). Car park system: A review of smart parking system and its technology. *Information Technology Journal*, *8*, 101–113.

INGALLS, R. G. (2002). Introduction to simulation. *Proceedings of the Winter Simulation Conference*, *1*, 7–16.

JOHNSTON, K. M. (2013). *Agent Analyst: Agent-Based Modeling in ArcGIS*.

JONKERS, E., NOORT, M. VAN, & VEEN, J. L. VAN DER. (2011). Parking guidance - Modelling, simulation and impact assessment. *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 858–864.

KLAPPENECKER, A., LEE, H., & WELCH, J. L. (2014). Finding available parking spaces made easy. *Ad Hoc Networks*, *12*, 243–249.

KOTUSEVSKI, G., & HAWICK, K. A. (2009). A Review of Traffic Simulation Software. *Research Letters in the Information and Mathematical Sciences*, *13*, 35–54.

KRASNER, G. E., & POPE, S. T. (1988). A Cookbook for Using the Model- View-Controller User Interface Paradigm in Smalltalk-80. *Joop Journal Of Object Oriented Programming*, *1*, 26–49.

LAW, A. M., & KELTON, W. D. (2000). *Simulation modeling and analysis*. (W. David Kelton, Ed.). McGraw-Hil.

LEEPHAKPREEDA, T. (2007). Car-parking guidance with fuzzy knowledge-based decision making. *Handbook of Environmental Chemistry, Volume 5: Water Pollution*, *42*, 803–809.

LI, Y., MA, R., & WANG, L. (2009). Intelligent parking negotiation based on agent technology. *2009 WASE International Conference on Information Engineering, ICIE 2009*, *2*, 265–268.

LUCK, M., MCBURNEY, P., SHEHORY, O., & WILLMOTT, S. (2005). Agent technology: computing as interaction (a roadmap for agent based computing).

LUKE, S. (2014). *Multiagent simulation and the MASON library*. *George Mason University*.

MACAL, C. M., & NORTH, M. J. (2009). Agent-based modeling and simulation. *Proceedings of the 2009 Winter Simulation Conference (WSC)*, 86–98.

MACAL, C. M., & NORTH, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, *4*, 151–162.

MACIEJEWSKI, M. (2010). A comparison of microscopic traffic flow simulation systems for an urban area. *Transport Problems*, *5*, 27–38.

MARKOVIC, D. S., ZIVKOVIC, D., CVETKOVIC, D., & POPOVIC, R. (2012). Impact of nanotechnology advances in ICT on sustainability and energy efficiency. *Renewable and Sustainable Energy Reviews*, *16*, 2966–2972.

OZIK, J., COLLIER, N. T., MURPHY, J. T., & NORTH, M. J. (2013). The relogo agent-based modeling language. In *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World* (pp. 1560–1568).

RAILSBACK, S. F., & GRIMM, V. (2011). *Agent-based and individual-based modeling: a practical introduction*. Princeton University Press.

REVATHI, G., & DHULIPALA, V. R. S. (2012). Smart parking systems and sensors: A survey. *2012 International Conference on Computing, Communication and Applications, ICCCA 2012*.

RICHARDSON, L., & RUBY, S. (2008). *RESTful Web Services*. O'Reilly Media.

RIESER, M., DOBLER, C., DUBERNET, T., GRETHER, D., HORNI, A., L¨AMMEL, G., … NAGEL, K. (2015). *MATSim User Guide*.

RODIER, C. J., & SHAHEEN, S. A. (2010). Transit-based smart parking: An evaluation of the San Francisco Bay area field test. *Transportation Research Part C: Emerging Technologies*, *18*, 225–233.

SARGENT, R. G. (2005). Verification and Validation of Simulation Models. *Proceedings of the 2005 Winter Simulation Conference M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, Eds.*, 130–143.

SARKAR, P. (2000). A brief history of cellular automata. *ACM Computing Surveys (CSUR)*, *32*, 80–107.

SHIN, J.-H., & JUN, H.-B. (2014). A study on smart parking guidance algorithm. *Transportation Research Part C: Emerging Technologies*, *44*, 299–317.

SHOUP, D. C. (2006). Cruising for parking. *Transport Policy*, *13*, 479–486.

SOKOLOWSKI, J. A., & BANKS, C. M. (2008). *Principles of Modeling and Simulation: A Multidisciplinary Approach*. *Principles of Modeling and Simulation: A Multidisciplinary Approach* (pp. 1–259).

SOKOLOWSKI, J. A., & BANKS, C. M. (2010). *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains*. John Wiley & Sons.

SULISTIO, A., YEO, C. S., & BUYYA, R. (2004). A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Software - Practice and Experience*, *34*, 653–673.

SUNMONU, O. (2011). Intelligent Transportation System. In *10 th Research Seminar Series Workshop*.

TEODOROVIĆ, D., & LUČIĆ, P. (2006). Intelligent parking systems. *European Journal of Operational Research*, *175*, 1666–1681.

WAINWRIGHT, J., & MULLIGAN, M. (2005). *Environmental Modelling: Finding Simplicity in Complexity* (John Wiley., p. 430).

WANG, H., & HE, W. (2011). A reservation-based smart parking system. *2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011*, 690–695.

WARAICH, R. A., ZURICH, E. T. H., DOBLER, C., & AXHAUSEN, K. W. (2012). Modelling Parking Search Behaviour with an Agent-Based Approach. *13th International Conference on Travel Behaviour Research (IATBR)*.

YOO, S. E., CHONG, P. K., KIM, T., KANG, J., KIM, D., SHIN, C., … JANG, B. (2008). PGS: Parking guidance system based on wireless sensor network. In *3rd International Symposium on Wireless Pervasive Computing, ISWPC 2008, Proceedings* (pp. 218–222).

ZAMBONELLI, F., & PARUNAK, H. V. D. (2003). Signs of a Revolution in Computer Science and Software Engineering. *Engineering Societies in the Agents World III*, *2577*, 120–125.

Annex A Basic Agents' Profiles

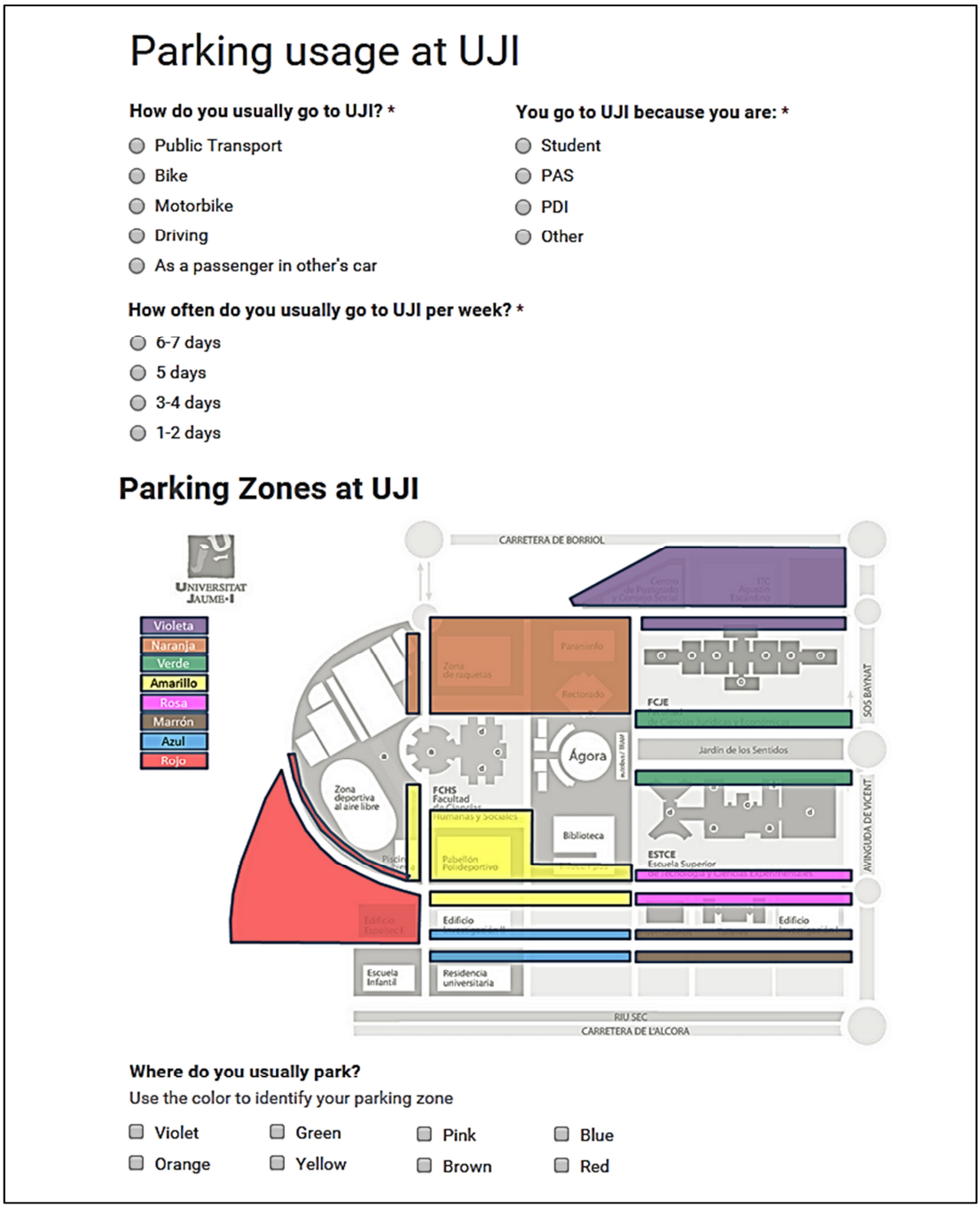
The following table shows some basic profiles from which more detailed ones can be derived. Profiles represent typical agent activity plans. The suggested plans have two one or two activities. Some of them include others (e.g., S2 includes S1, which means it represents a student that first goes to class and then to the library). A rich profile set should also considered "atypical" elements, like the guards or students living in the Students Residence.

Name	Includes	Destination	Arrival Time	Parked Time	Days
Students					
S1	-	A class building	Starting class time in morning or afternoon.	4 hours	Week Days
S2	S1	The library	If its classes are in the morning, then in the afternoon and vice versa.	2 hours	
S3	S1	A sport facility	If its classes are in the morning, then in the afternoon and vice versa.	2 hours	
PDI					
P1	-	An office building	Variable ranging from 09:00 to 14:00	Variable, 3 to 8 hours, not beyond 17:00.	Week Days
P2	P1	A class building	Variable ranging from 09:00 to 15:00	2 hours	
P3	P1	A sport facility	Variable ranging from 08:00 to 19:00	2 hours	
PAS					
O1	-	An office building	08:00	8 hours	Week Days and Saturdays
O2	O1	A sport facility	Just after work time	2 hours	
Other People					
G1	-	A sport facility	Variable ranging from 08:00 to 19:00	2 hours	Week Days
G2	-	The library	Variable ranging from 08:00 to 19:00	2 hours	

Annex B Parking Needs Survey

The following image presents a translation of the survey form that the SmartParking team is planning to apply. The original survey form can be seen at:

https://docs.google.com/forms/d/1xpY-nMni6DXnj8KaHRyv9eqA-eXdYOQOwNfYBwCB5ts/viewform



Parking usage at UJI

How do you usually go to UJI? *

- Public Transport
- Bike
- Motorbike
- Driving
- As a passenger in other's car

You go to UJI because you are: *

- Student
- PAS
- PDI
- Other

How often do you usually go to UJI per week? *

- 6-7 days
- 5 days
- 3-4 days
- 1-2 days

Parking Zones at UJI

Where do you usually park?

Use the color to identify your parking zone

- Violet
- Green
- Pink
- Blue
- Orange
- Yellow
- Brown
- Red

Annex C　Campus and Building Entrances

The locations of campus entrances are represented by blue dots. Building entrances are shown as violet dots.

Annex D Agent Profiles Configuration

The following image shows the elements that an agent profile should usually have. In this case, the agent profile defines two activities.

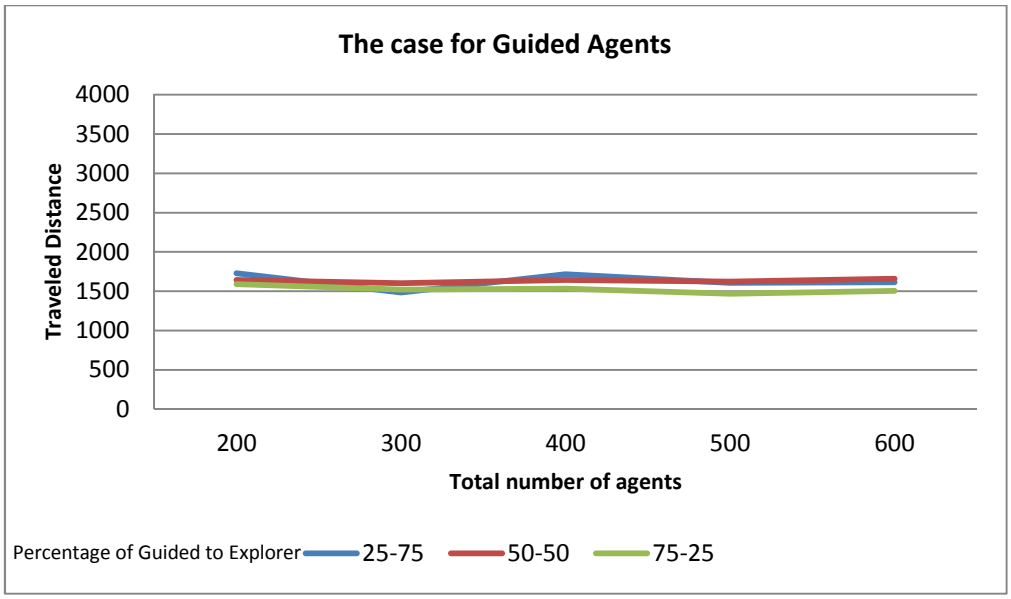
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ConfigsXML>
    <Profiles>
        <Type>Estudiante</Type>
        <Amount>100</Amount>
        <Repetition>Sunday</Repetition>
        <MaxWalkingDistance UpLimit="100" DownLimit="80"/>
        <StartTime>
            <Time weekDay="0" hour="8" minute="05" second="0"/>
            <TimeRandomizer>
                <UpLimit weekDay="0" hour="0" minute="30" second="0"/>
                <DownLimit weekDay="0" hour="0" minute="0" second="0"/>
            </TimeRandomizer>
        </StartTime>
        <Goals>
            <Order>1</Order>
            <Destination building="2" door="0" description="Docente"/>
            <DepartureTimeLap>
                <Time weekDay="0" hour="4" minute="0" second="0"/>
                <TimeRandomizer>
                    <UpLimit weekDay="0" hour="0" minute="30" second="0"/>
                    <DownLimit weekDay="0" hour="0" minute="0" second="0"/>
                </TimeRandomizer>
            </DepartureTimeLap>
        </Goals>
        <Goals>
            <Order>2</Order>
            <Destination building="3" door="5" description="Biblioteca"/>
            <DepartureTimeLap>
```

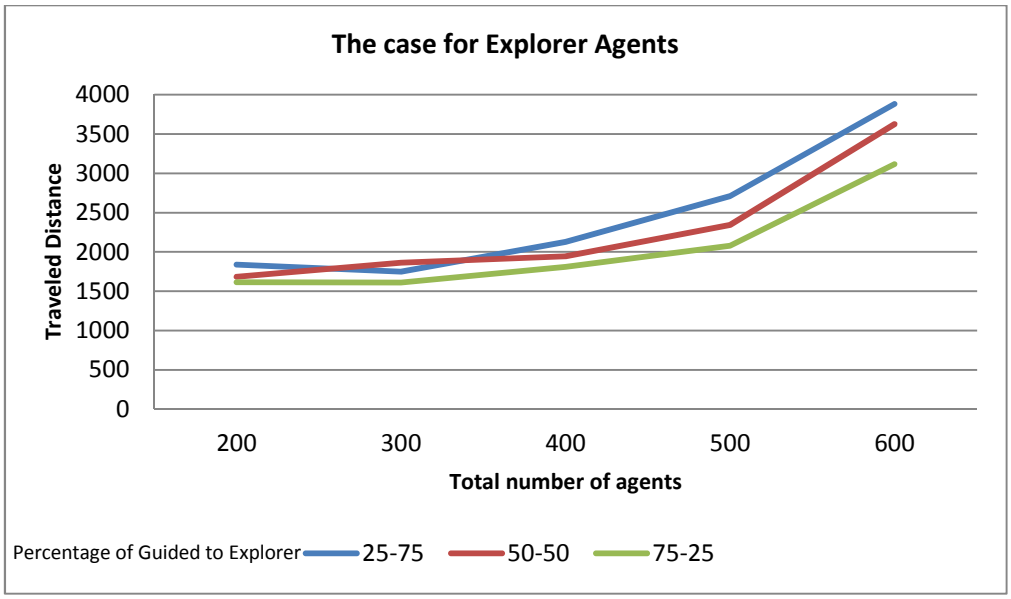
The following table summarizes the main XML configuration elements.

Element Name	Description
Profile	It contains all information needed for defining a profile. There should be one or more in the configuration file.
Type	Textual description of the profile.
Amount	Amount of agents that will be created using the profile as a template.
Repetition	The days that the agents from the profile get into the simulation. Apart from the usual weekdays, other values are *AllWeek*, *WeekDays*, *ThreeWeekDays* and *OneDay*.
Max WalkingDistance	Value for the maximum distance an agent is willing to walk, if it needs it. This value is chosen randomly in a range.
StartTime	The time at which the agents start in a simulation day. It includes a center value and a variation range. Based on that, the actual value is randomly chosen.
Goals	It holds the destinations that the agent has, as well as the time it will spend parked in each of them.

Annex E Traveled Distance Charts

This section presents two charts that explore the emergent characteristic of traveled distance regarding both types of agents. The first chart shows that, for Guided agents, a high demand does not necessarily create parking problems. However, for the Explorer agents, a high occupancy may lead to additional searches.





Masters Program in **Geospatial Technologies**

Supported by:
Education and Culture
ERASMUS MUNDUS