# SIAPAS: A Case Study on the Use of a GPS-Based Parking System

Gonzalo Mendez[1], Pilar Herrero[2], and Ramon Valladares[2]

[1] Facultad de Informatica - Universidad Complutense de Madrid
C/ Prof. Jose Garcia Santesmases s/n, 28040 Madrid, Spain
gmendez@fdi.ucm.es
[2] Facultad de Informatica - Universidad Politecnica de Madrid
Campus de Montegancedo s/n, 28660 Boadilla del Monte (Madrid), Spain
pherrero@fi.upm.es

**Abstract.** GPS-based applications have become very popular during the last years, specially among drivers, who use them to find the best way to their destination. However, their use is still far from taking advantage of the wide range of possibilities that GPS offers. The SIAPAS application goes one step further by adding new functionality to the typical GPS-based map. SIAPAS runs on a PDA and it allows drivers to find a parking space that suits their needs inside a parking lot. This paper describes how the system has been designed and implemented, and shows the results of some experiments that have been carried out to test its utility and usability.

## 1 Introduction

Finding a parking space is a common challenge faced by millions of citizens every day. Let's imagine a driver who arrives to a shopping center looking for the place to park his car. Let's also imagine that the shopping center is on sale and therefore it is bursting with people. If the user needs to buy something quickly, something that he forgot the previous day when he did his weekly shopping, and he is also in a hurry because he just quit from his job for a few minutes, he would need extra help to find the best parking-position. The driver is not concerned with the shopping center entrances that are far away from his current location, rather he wants to choose one from several entrances near his current location and, if possible, closer to the requested shop.

A location-based application could help to this user with this problem as it would guide him depending on his current location. A crucial part of this location-based application is locating users' current location. Global Positioning System (GPS) is a widely used technology for this purpose and it is constantly being improved. With the advances in GPS and wireless communications technology and the growing popularity of mobile devices, such as PDA, the need for location-based applications has gained significant attentions.

In the last few years some similar projects have been developed in many different places with many different purposes. In fact, an overview of ad-hoc routing

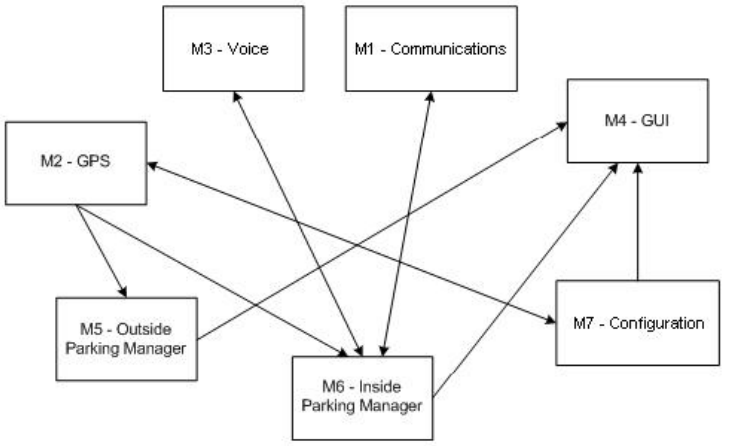


**Fig. 1.** Module-level system architecture

protocols that make forwarding decisions based on the geographical position of a packet's destination is presented in [1].

One of these projects is the one developed as a part of the digital campus project at the University of California, NAPA (Nearest Available Parking lot Application). This application, which finds an empty space from multiple parking lots scattered around a campus or some area like a city or an airport, is intended to reduce the bottleneck at the campus entrance, which is often a time consuming process in itself [2]. Another one is PMNET, a multi-hop wireless parking meter network that allows users to locate and navigate to an available parking space by equipping existing parking meters with wireless radio frequency (RF) transceivers and auxiliary hardware and software [3]. However, although both projects are similar in intentions, SIAPAS goes one step forward as it offers not just a location for the driver's car, but the best one: closer to the closest entrance of the shopping center.

In this paper, we will show how the system has been designed (Section 2) and we will give a few details about the implementation (Section 3). Then, we will describe how the system has been tested and evaluated (Section 4) and we will end with the conclusions we have obtained (Section 5).

## 2 System Design

The SIAPAS system has been designed as a set of independent modules that communicate with each other through the use of web services (see Fig. 1).

- M1 - Communications: this module keeps track of the state of the parking spaces.

- M2 - GPS: it keeps track of the car's current position, minimizing the GPS position error.
- M3 - Voice: speech-based driver's assistance.
- M4 - GUI: it manages user interaction.
- M5 - Outside Parking Manager: this module controls the global parking state.
- M6 - Inside Parking Manager: it keeps track of the parking state: parking spaces, routes, entrance and the like.
- M7 - Configuration: it manages the GPS device configuration.

This division in modules is based not only on functionality reasons for each device, but also in the global functionality. Thus, there may be parts of the same module running in different devices.

### 2.1 M1 - Communications

This module consists of two parts (a client and a server) that communicate with each other through a WiFi network using SOAP.

The server side is based on an agent who is in charge of managing the clients' petitions to block and release parking spaces, keeping the ontology that is used to represent the parking state up to date. The clients can block the parking space they want to use so that no other driver can use it. To avoid the parking spaces being blocked without a car occupying them for too long, the agent is also in charge of releasing the ones that have been blocked for more than a predefined time (currently 30 minutes).

The client runs in the drivers's PDA, and it starts working as soon as the GPS detects that the car is inside the parking lot. It first requests the parking state, and then it blocks the parking space to be used to park the car.

### 2.2 M2 - GPS

This module is in charge of receiving data from the GPS system and preparing them to be used by the rest of the SIAPAS application. It is structured as a conventional compiler, with a component to read data and detect lexical errors, another one to parse the sentences and a third one to obtain position and precision data and prepare them to be used by other parts of the application, usually to update the state of close parking lots or to update the drivers' position inside the parking.

### 2.3 M3 - Voice

The Voice module was originally part of the Inside Parking Manager, but it was separated from it due to its complexity and different nature.

It has been divided in two submodules. The first one is in charge of analyzing the route that the driver must follow and create a list of events where some instruction must be told to the driver. These events include turning left and

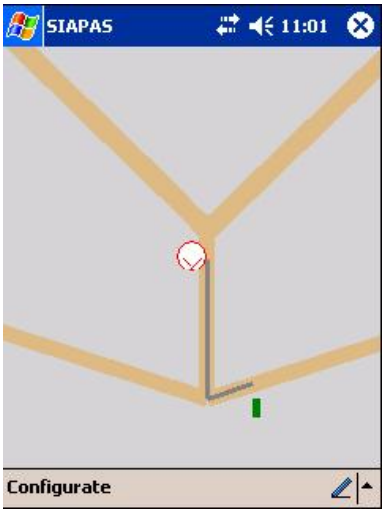


**Fig. 2.** Inside Parking Manager GUI

right and parking. The second submodule is in charge of checking, every time a GPS signal is received, whether the car is close to a mark where some instruction must be given to the driver and activate the speech synthesizer.

### 2.4 M4 - GUI

This module provides a graphical user interface for three of the modules that form the SIAPAS application: Inside and Outside Parking Managers and Configuration.

For the Configuration Module, the GUI offers the possibility to change the communication port with the GPS device, the communication speed and the protocol to be used to communicate with the GPS.

The GUI of the Outside Parking Manager is the default screen that users see when they are not in a parking that is controlled by SIAPAS. In this screen the user can see which are the closest parkings, how far they are from the user and in what direction.

As for the Inside Parking Manager, the necessary data to draw the parkings are stored in an ontology in this module. These data are related to the parking itself (lanes, entrances and exits and parking spaces) and to the vehicle's position. Fig. 2 shows the aspect of this GUI, where the user can typically see: the parking lanes painted in brown; the parking spaces in green (free), red (occupied) or maroon (blocked); the vehicle, as a white circle with an arrow point inside if it is moving or a dot if it is stopped; and the route to the closest free parking space painted with a grey line.

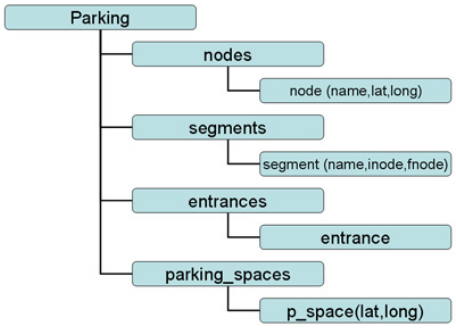


**Fig. 3.** Inside Parking Manager Ontology Structure

### 2.5 M5 - Outside Parking Manager

This module is in charge of maintaining the ontology that stores the information about the parkings that can work with the SIAPAS system. The ontology stores the name, location and size of the parking, and it is used to infer where the closest parking is or how to get there.

The ontology is updated every time the driver is close to a parking that is managed by the SIAPAS system. In addition, inferences are carried out after a predefined time (currently, 3 seconds) when a GPS signal is received, so the current distance or location of all parkings can be updated. The format to represent latitude and longitude is dd.mmmmmm, using a WGS84 datum. The distance is calculated using Vicenty's formula [4].

### 2.6 M6 - Inside Parking Manager

The Inside Parking Manager is in charge of controlling what happens inside a parking, so it manages information about parking spaces, lanes and routes from one place inside the parking to another. We have used an ontology to represent this information, the structure of which can be seen in Fig. 3.

This module starts its execution when the car is close to a parking. It retrieves all the information related to the state of the parking spaces and lanes inside the parking, and it uses a slight variation of Dijkstra's algorithm [5] to figure out the most suitable parking space. Once it has been found, the Inside Parking Manager checks periodically that the driver is following the right route. If this is not the case, it calculates a new route to the parking space.

### 2.7 M7 - Configuration

This module is in charge of managing information about the GPS hardware that is used by the system. Currently, the information that is used is the communication port (COM1 to COM6), the port speed (from 2400 bauds to 115200 bauds) and the communication protocol (only NMEA, for the moment).

When the application is launched, it reads the configuration file and tries to establish a connection with the GPS hardware. If the configuration is wrong or the PDA is using the chosen port for some other purpose, all the user will see is a message saying that there is no GPS signal available. If the user changes the configuration, the application will try to open the selected port. If it is successful, it writes the new configuration data in the configuration file; otherwise, no changes are made.

## 3 Implementation

The client side of the SIAPAS system runs on a Pocket PC that uses Microsoft Windows CE as Operating System. Among the different options that exist to develop software for this platform, Microsoft Visual Studio .NET 2003 has been chosen as the development environment, especially due to its good integration with the execution environment.

There are several options to implement a sockets-based communication between the clients and the server. The C++ Sockets Library has been chosen because it is object oriented and internally it makes use of POSIX libraries.

Finally, to develop the GUI there is the possibility to make use of the libraries provided by the .NET Compact Framework, a reduced set of GDI (Graphics Device Interface). Although this was the first choice, it soon became obvious that the possibilities it offers are quite low. Therefore, after analyzing different alternatives, OpenNETCF was chosen to substitute GDI, mainly because it is quite similar to GDI and easy to use (although the performance is not as good as, for example, that of GAPI).

## 4 Evaluation

One of the main objectives of this project is to develop an application that can be used in a short period of time, so a lot of stress has been imposed over the evaluation of SIAPAS to make sure it will be useful for the final user. The evaluation method that has been used is described in [6], and it basically consists of a theoretical validation, scenario validation and user validation.

### 4.1 Theoretical Validation

For the theoretical validation, the objective has been to test that the mathematical basis used in the application is accurate enough for the application to be useful. Three different experiments have been run.

**Experiment 1 - Distance accuracy** Both Vicenty's formula and the Haversine formula have been used to measure distance accuracy. Vicenty's method shows that, using a WGS84 datum, longitude and latitude precision should be of 0.00005".

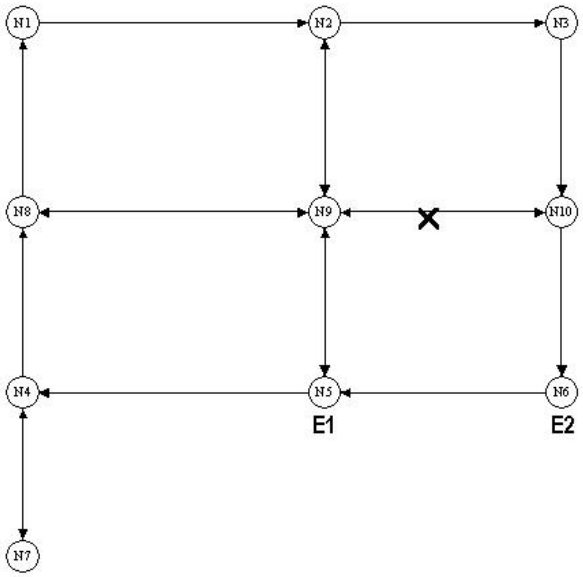


**Fig. 4.** Experiment 2 - Parking Structure

We have tested the algorithms running 27 tests with different data. For two points situated less than 100 meters far from each other, both methods showed a difference in measure that ranged between 1 centimeter and 51 centimeters, which can be considered a very good precision for this kind of application.

**Experiment 2 - Closest entrance selection** The objective of this experiment has been to determine whether Dijkstra's algorithm always selects the closest entrance to the building or not. Fig. 4 shows the elements of the experiment, where the graph shows the parking structure, E1 and E2 are the different entrances to the building and the X shows the position of the car.

We have run 25 different tests, changing the position of the car, E1 and E2. In all the tests, the distance of the route selected by the algorithm was slightly shorter than the real distance travelled by the car, being the mean deviation of 5.02%.

**Experiment 3 - Closest parking space selection** In this experiment, the main objective is to test that the chosen parking space is the closest one to the building and that it is the one that most people would choose. The structure os the parking is the same as in the previous experiment (see Fig. 4), and the location of the parking spaces can be seen in Fig. 5. Each one of the 25 tests that have been run in this experiment is a continuation of the corresponding test run for Experiment 2.

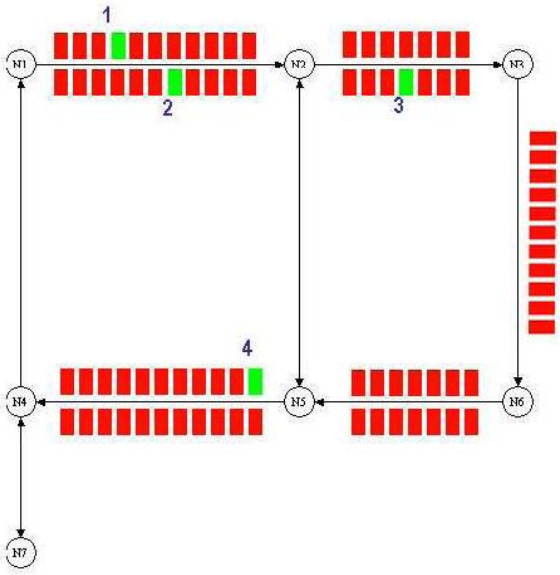


**Fig. 5.** Experiment 3 - Parking Spaces Location

After running the tests, each of the solutions have been evaluated according to what a user would think of it (this has been done by the same person who ran the tests; the results of the user validation will be shown later in this paper). The values used have been: totally disagree, slightly disagree, agree, quite agree, totally agree. Then, a numerical value ranging from 1 to 5 has been given to each of the options, and the resulting average was 4.65. This average shows that, most of the times, we believe a human user would think he would have chosen the same parking space.

### 4.2 Scenario Validation

In this validation, the objective has been to test that the SIAPAS application offers a practical solution to the previous experiments. Fig. 6 shows the structure of the parking, where the stars mark the entrance to the building and the rectangles show the parking spaces.

We have run 26 tests, changing the position of the car and the free parking spaces, and the results have been evaluated the same way it was done in Experiment 3. This time, the obtained average has been 4.58, which is quite close to the previous result.

After running this test, we can see that the experimental results obtained using SIAPAS are the ones expected after running the theoretical validation, and the system has shown that most of the times it is able to find the most

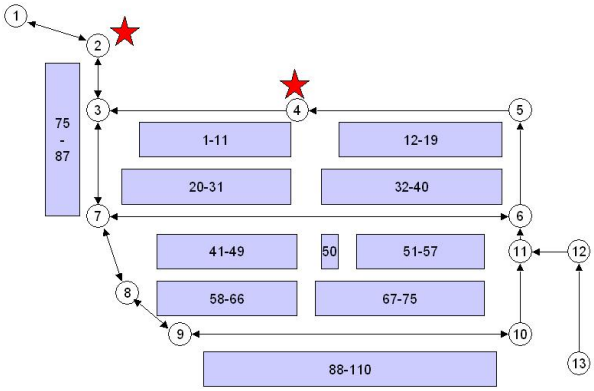


**Fig. 6.** Parking Structure for the Scenario Validation

suitable parking space for a driver. This point will be effectively evaluated in the next validation.

### 4.3 User Validation

This validation has been carried out using the *Performance Measure* technique [7]. Three groups of five people were made to be able to compare whether the results depended a lot on the kind of user or not:

- Users between 21 and 26 years old with low skills using hardware and software.
- Users between 24 and 25 years old with an average level experience using PCs.
- Users between 21 and 28 years old with high skills using PCs, PDAs and GPS applications.

Each user had to park his car twice, first without using SIAPAS and then using it, and we measured the time it took since they got into the car till they arrived walking to the building entrance. After that, the users had to fill in a questionnaire about their opinion of the system.

In most cases, it took shorter to park using SIAPAS, although the differences in time ranged between 6 and 112 seconds, being the average difference of 28 seconds.

The analysis of the results of the questionnaire showed that all the users thought the system was easy to use, even the ones without experience, but they also agreed in the need to provide a more user-friendly GUI, and specially in the level of detail of the parking representation.

## 5 Conclusions

Finding a parking space is a common challenge faced by thousands of people every day. Wireless ad-hoc networking technologies offer a new and efficient means to simplify the parking process. In this paper we have described a GPS-based application (SIAPAS) that allows a user to quickly locate and drive to an available parking space.

Our solution is achieved by equipping drivers with a PDA to navigate in the area. The results of the evaluation that has been carried out point out that the system is accurate enough to be useful, which has been confirmed by the people who have taken part in the experiments: in their opinion, the system is useful and easy to use, although some improvements need to be made in the GUI for the application to be a little more user-friendly.

## References


1. Mauve, M., Widner, J., , Hartenstein, H.: A survey on position-based routing in mobile ad-hoc networks. IEEE Network **15** (2001) 30–39
2. Chon, H.D., Agrawal, D., Abbadi, A.E.: Napa: Nearest available parking lot application. In: Proceedings of the 18th International Conference on Data Engineering (ICDE'02), IEEE (2002) 496–497
3. Basu, P., Little, T.: Networked parking spaces: Architecture and applications. In: Proceedings of the IEEE Vehicular Transpotation Conference, IEEE (2002) 1153–1157
4. Vicenty, T.: Direct and inverse solutions on the ellipsoid with application of nested equations. Survey Review **XXII** (1975) 88–93
5. Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms. MIT Press (1990)
6. Juristo, N., Moreno, A.: Basics of Software Engineering Experimentation. Kluwer Academic Publishers (2001)
7. Dumas, J.: A Practical Guide to Usability Testing. Intelect (1999)