Sensorless parking area management system

Dan Adrian Marior*, Constantin Cîrciumaru**

** Department of Automation in the Faculty of Automation, Computers and Electronics, Craiova (e-mail: marior@ automation.ucv.ro).*
*** IT Department of the Ford Motor Company, Craiova (e-mail: ccircium@ford.com)*

Abstract: The paper deals with the problem of managing the utility vehicle park of the Ford Motor Company plant in Craiova. In order to obtain the minimal cost solution, we propose a sensorless parking management system, which handles the parking spots, the tracking of the vehicles (whether they are inside the parking area or not), and reports to management in case there are situations that go beyond normal workflow rules.

Keywords: sensorless, management, Ford, parking, ASP.NET.

1. INTRODUCTION

Utility vehicles from companies around the world are an issue to manage and synchronize with the workflows that take place in the daily activity of their plants. Most companies rely on their own vehicle fleet (whether they are land, sea or air vehicles) for the incoming and outgoing objects necessary for the factory to work. Analogous to the control systems theory approach, the plant as a whole can be regarded as a system that accepts input (raw materials, semi-assembled goods, etc.) processes them and produces finite goods, which can be regarded as an output.

The problem of parking space management started as soon as the number of cars in a given area has increased over the limit to which human or automatic supervising intervention is not necessary. Of course, in this case, we are talking about industrial platforms, or their adjacent employee parking area. In the usual public city car park situation, the car owners apply the greedy type law of “first come, first served”, which seems to work just fine until various conflicts arise. The approach we propose however depends on whether the managers decide it is useful to implement it or not.

In this paper we intended to approach the already investigated problem of optimizing parking spots allocation due to another motivation: cost optimization by eliminating the sensors needed to appropriately track every movement of the cars pertaining to a large company such as Ford. The optimization problem is mainly a theoretical one (www.stackoverflow.com) and has been approached from a lot of angles, for example in (Zoghi et al, 2006), (Space Utilization Optimization, 2009), (Chen et al, 2011) and (Meiping et al, 2008) only to mention a few, and our goal was to implement a practical approach to the subject in the real case of the Ford plant here in Craiova. In the various other approaches authors use sensors and dedicated hardware to keep track of the cars and trucks on routes that are usually either predefined but can also be partially fixed. Sensors are placed on cars and/or in every parking spot or in the immediate vicinity in order to locate the car precisely or just to detect its presence somewhere in the parking spot. In some approaches data about the cars is stored in databases. Ford plants across the globe have internal tracking systems for the cars, with the help of sensors and cameras, for the route the car has to take before it leaves the factory – a rather complicated one.

Our approach was to design a software application that would be used to take care of every aspect in the usual parking workflow. The application is an ASP.NET web application, the latest and best Microsoft alternative to build client/server web applications from the ground up.

The first stage in the automation of the parking lot, in our opinion, is the implementation of the software we built in the case of the employee car park and then, after the time period the drivers need to accommodate to it, move it on to the next stage where it will have to manage the inbound and outbound flow of vehicles serving the plant. The second stage would be most useful when the company will go to mass production according to the initial plans dating back when the factory was placed in the possession of Ford. However, the application is not yet able to withstand that amount of processing, given the fact that the hardware requirements will be most certainly different which will in turn should trigger the usage of a different database management system, for example SQL Server Developer or Standard edition, for example, as opposed to the free Express version we used.

Section I prepares the reader for the main purpose of this paper, explains the foundation for it, mentions and explains other papers in the field, and mentions the first stage in the lifecycle of the .NET application discussed. Section II describes certain aspects (advantages mainly) of the platform we chose to implement the solution with, and section III presents the application with its features and usage patterns only to conclude with its possible future evolutions.

2. THE MEANS TO COMPLETE THE TASK

The goal we had set was to design and eventually implement an application that was supposed to keep track of the activity of the parking area of the company, given the fact that there are a lot of employees coming in with their personal car but even though they do not particularly appreciate being tracked, we saw in this the perfect opportunity to test the application for this stage of the development; we would then be able to collect any feedback from the persons involved and modify or eventually improve it to meet other requirements too.

Given the facts that the application is to run in an industrial context, it has to rise up to corresponding standards, concerning mainly the functionality and pay less attention to having a visually appealing graphical interface, thus making the use of web application technology more appropriate than the traditional desktop approach. The platform of choice was easy: the .NET Framework can provide classes and features for almost any type of application an engineer can imagine. This really is just a matter of preference and experience, actually, as the alternatives (Java and PHP) can be used to reach the same goal. The reasons we chose .NET are based on the fact that Microsoft have placed considerable efforts into Visual Studio development as easy as possible (having taken the Rapid Application Development they invented to perfection), thus allowing the developer more chances to concentrate on the actual task to complete instead of other aspects and last, but not least, the prior experience of the authors.

Another feature we appreciated about developing with Visual Studio is the possibility to see almost immediately how the changes to the code are expressed in the interface. In order to properly present the .NET Framework we have to mention from the beginning that it is an alternative to create managed code, and not binary executables - .exe files – and that means code which cannot run unless we have the .NET Runtime installed on the computer the application is run. In the case of web applications all we need is a regular browser to access it, and we can also run it directly from Visual Studio, hosted by the built-in development server (MacDonald et al, 2007).

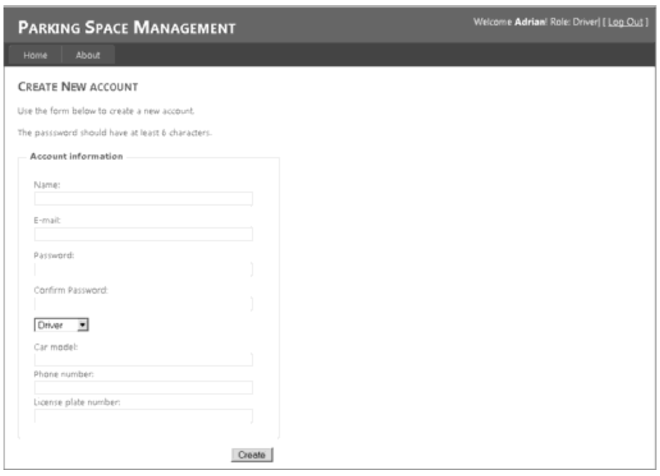
A set of helpful features come from the C# language, which has reached version 4.0, the same as the framework (LINQ, introduced with the 3.0 version, generic types, the “partial” keyword, setting of properties at runtime, delegates, and others). Of course, using the latest platform available does have its advantages, regarding the language improvements and general programming paradigm advances.

Like all the modern preferred programming languages, C# can interact with relational database management systems, operate on files and ports, compress data, facilitate threaded programming, security and working with XML. The execution speed, considering the fact that it is managed and not binary code is quite sufficient for most applications. Of course, programming is an art and optimizations are still welcome in some cases, in spite of contemporary hardware resources (MacDonald et al, 2010).

3. THE APPLICATION ARCHITECTURE AND FEATURES

Following programming best practices we designed the application to be a typical 3 tier one. As it is already known, this paradigm stands for separating the architecture into three layers (hence the name): the data layer, the presentation and the application layer. As the layers do not need explaining we shall simply state that for the data layer we used the Microsoft SQL Server Express version to host our data, and that the code for the presentation and the actual business logic were separated. We used most of the security features of the ASP.NET framework, for example roles, a membership provider and username/password login authentification.

We designed the application to be used by the gatekeeper at the entry point of the parking area, who has the right credentials to login and operate on the application (manager accounts). After the authentication, which takes place in the welcoming page of the application, the user sees the page in Fig. 1, which gives him the possibility to insert the data about the drivers and their cars in the database. We built a database schema and the corresponding tables which can be seen in figure 7. When a car comes to the front gate, the gatekeeper must retain data about the make and model of the car, its driver, phone number and its licence plate number (additional data can be stored, of course, but is sufficient for this stage).


PARKING SPACE MANAGEMENT

Welcome **Adrian**! Role: Driver| [Log Out]

Home | About

CREATE NEW ACCOUNT

Use the form below to create a new account.

The password should have at least 6 characters.

Account information

Name:

E-mail:

Password:

Confirm Password:

Driver

Car model:

Phone number:

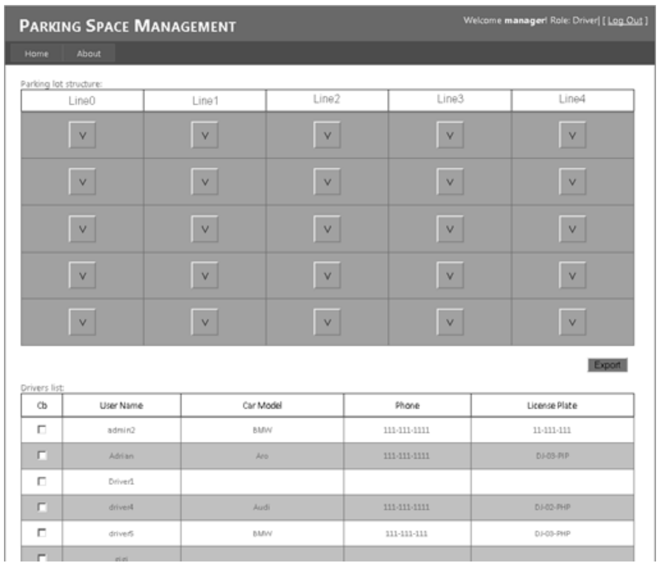
License plate number:

Create

Fig. 1. Inserting information about a car and its driver

The data is inserted into the database and then the gatekeeper proceeds to the next page of the application, which can be seen in Fig. 2. Here we can see an “all spaces free” view of the application immediately after the login (the light green colour has another significance that of optimal choice zones for parking the next car). Of course, it will become similar to what we see in Fig. 3 as soon as the gatekeeper clicks on any spot that he chooses from the area. It is important to mention that the first one can always be randomly chosen, and the following ones can be chosen in accordance with the recommendations made by the algorithm which presents a realtime view of the parking lot to the user of the application.

We denoted with the dark green colour the suboptimal choice zones, as calculated by the algorithm and with light green the optimal ones. Of course a red spot (cell) means an occupied space. It is obvious that the interface is very easy to use, thus reducing the probability for human error in the process of choosing free parking spots. In order to have a parking spot occupied we first have to introduce the drivers in the database dedicate to them (practically representing a drivers list) and only then we can check the checkbox corresponding to the one we currently assign a parking space to. The application will not occupy the space unless the line corresponding to the driver is checked and the driver has been inserted in the database before, which is a common sense security feature.



PARKING SPACE MANAGEMENT — Welcome **manager**! Role: Driver| [Log Out]

Home | About

Parking lot structure:

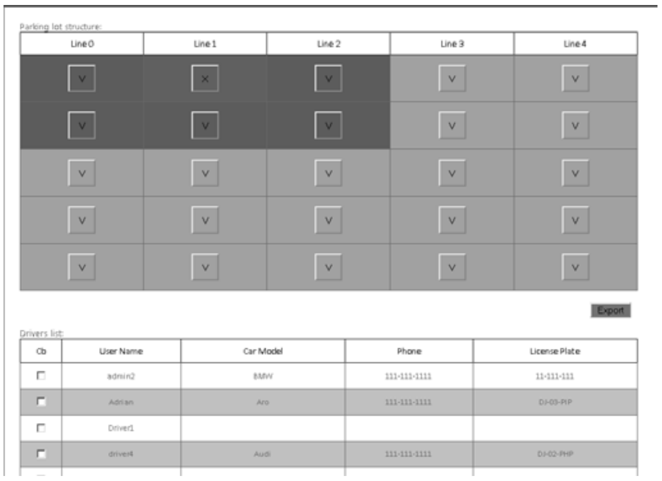
Line0	Line1	Line2	Line3	Line4
V	V	V	V	V
V	V	V	V	V
V	V	V	V	V
V	V	V	V	V
V	V	V	V	V

Export

Drivers list:

Cb	User Name	Car Model	Phone	License Plate
☐	admin2	BMW	111-111-1111	11-111-111
☐	Adrian	Aro	111-111-1111	DJ-03-PIP
☐	Driver1			
☐	driver4	Audi	111-111-1111	DJ-02-PHP
☐	driver5	BMW	111-111-111	DJ-03-PHP

Fig. 2. An initial view of the interface, with all spaces free



Parking lot structure:

Line0	Line1	Line2	Line3	Line4
V	X	V	V	V
V	V	V	V	V
V	V	V	V	V
V	V	V	V	V
V	V	V	V	V

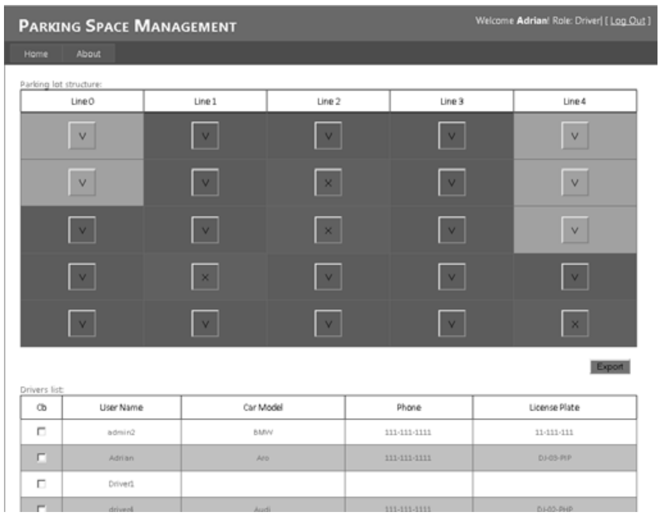
Export

Drivers list:

Cb	User Name	Car Model	Phone	License Plate
☐	admin2	BMW	111-111-1111	11-111-111
☐	Adrian	Aro	111-111-1111	DJ-03-PIP
☐	Driver1			
☐	driver4	Audi	111-111-1111	DJ-02-PHP

Fig. 3. Optimal and suboptimal areas for a single spot occupied

In Fig. 4 we can see a more realistic view of the situation, as the parking area is free only in the early hours of morning; the flow of cars is mainly inbound in the time interval of 7 and 8 in the morning and mainly outbound after 5 o'clock in the afternoon. The application was built with the ability to report the situation to the management by exporting an Excel file, which is easy to do, as can be seen in Fig. 5, by pressing the "Export" button.



PARKING SPACE MANAGEMENT — Welcome **Adrian**! Role: Driver| [Log Out]

Home | About

Parking lot structure:

Line0	Line1	Line2	Line3	Line4
V	V	V	V	V
V	V	X	V	V
V	V	X	V	V
V	X	V	V	V
V	V	V	V	X

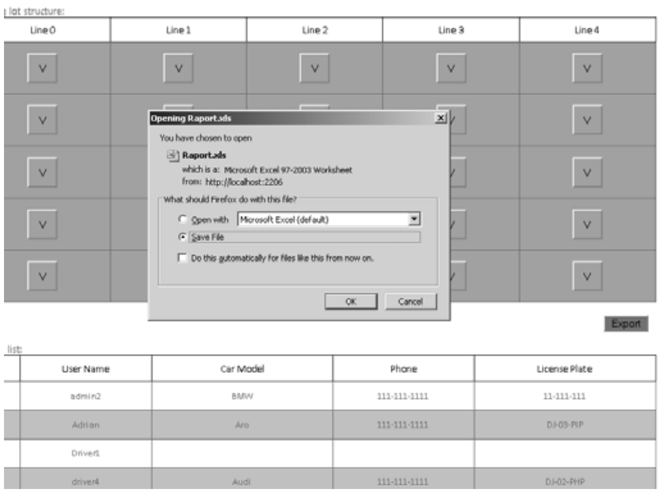
Export

Drivers list:

Cb	User Name	Car Model	Phone	License Plate
☐	admin2	BMW	111-111-1111	11-111-111
☐	Adrian	Aro	111-111-1111	DJ-03-PIP
☐	Driver1			
☐	driver4	Audi	111-111-1111	DJ-02-PHP

Fig. 4. View of the parking lot with more spaces occupied

Other types of reports containing statistics can also be created, in case there is a problem; these modifications will have to stand the test of time, however, as it will tell if they are necessary or not, given the fact that the Excel file is a 100 % accurate view of what the gatekeeper sees, with the additional calculation power of the Excel environment. Every space, free or occupied, is represented as a cell in the exported spreadsheet, as can be seen in Figure 6.



Opening Raport.xls

You have chosen to open

Raport.xls

which is a: Microsoft Excel 97-2003 Worksheet

from: http://localhost:2206

What should Firefox do with this file?

( ) Open with: Microsoft Excel (default)

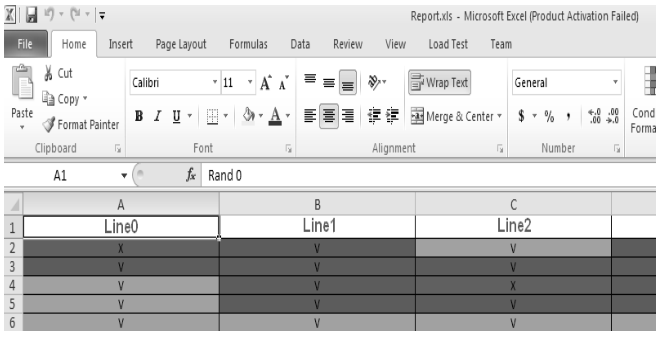
(•) Save File

☐ Do this automatically for files like this from now on.

OK | Cancel

	User Name	Car Model	Phone	License Plate
	admin2	BMW	111-111-1111	11-111-111
	Adrian	Aro	111-111-1111	DJ-03-PIP
	Driver1			
	driver4	Audi	111-111-1111	DJ-02-PHP

Fig. 5. The situation can be exported to Excel



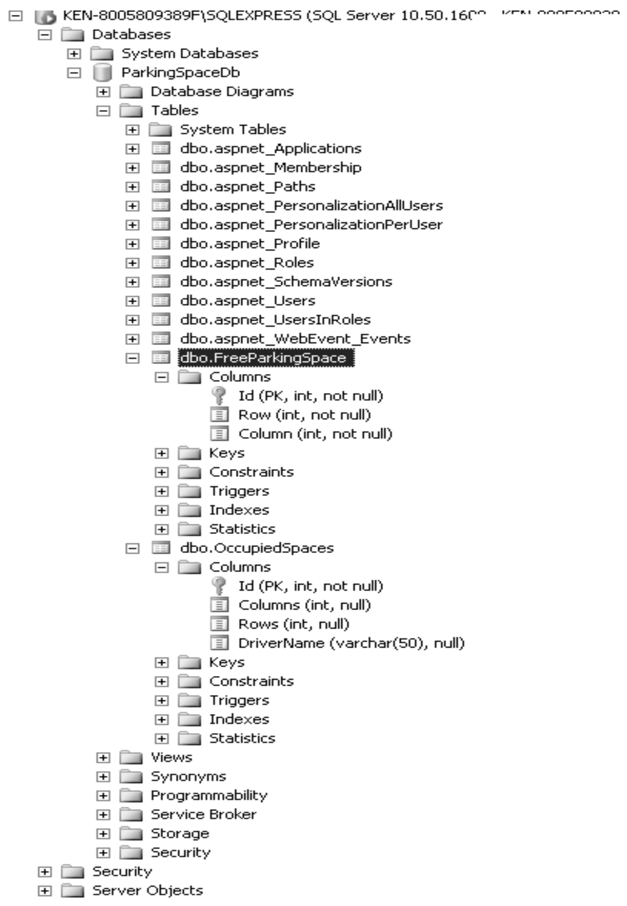
Report.xls - Microsoft Excel (Product Activation Failed)

A1 — f_x Rand 0

	A	B	C
1	Line0	Line1	Line2
2	X	V	V
3	V	V	V
4	V	V	X
5	V	V	V
6	V	V	V

Fig. 6. Aspect of the exported Excel 2003 file (.xls file)

The database with two tables (FreeParkingspace and OccupiedSpaces) has been designed with the necessary constraints (primary and foreign keys) linking the tables which represent the entities involved in the process, such as users, roles, schemas, cars, drivers, free parking spaces and occupied parking spaces. All the database structure was captured in Fig. 7.



```
⊟ KEN-8005809389F\SQLEXPRESS (SQL Server 10.50.160
  ⊟ Databases
    ⊞ System Databases
    ⊟ ParkingSpaceDb
      ⊞ Database Diagrams
      ⊟ Tables
        ⊞ System Tables
        ⊞ dbo.aspnet_Applications
        ⊞ dbo.aspnet_Membership
        ⊞ dbo.aspnet_Paths
        ⊞ dbo.aspnet_PersonalizationAllUsers
        ⊞ dbo.aspnet_PersonalizationPerUser
        ⊞ dbo.aspnet_Profile
        ⊞ dbo.aspnet_Roles
        ⊞ dbo.aspnet_SchemaVersions
        ⊞ dbo.aspnet_Users
        ⊞ dbo.aspnet_UsersInRoles
        ⊞ dbo.aspnet_WebEvent_Events
        ⊟ dbo.FreeParkingSpace
          ⊟ Columns
              Id (PK, int, not null)
              Row (int, not null)
              Column (int, not null)
          ⊞ Keys
          ⊞ Constraints
          ⊞ Triggers
          ⊞ Indexes
          ⊞ Statistics
        ⊟ dbo.OccupiedSpaces
          ⊟ Columns
              Id (PK, int, not null)
              Columns (int, null)
              Rows (int, null)
              DriverName (varchar(50), null)
          ⊞ Keys
          ⊞ Constraints
          ⊞ Triggers
          ⊞ Indexes
          ⊞ Statistics
      ⊞ Views
      ⊞ Synonyms
      ⊞ Programmability
      ⊞ Service Broker
      ⊞ Storage
      ⊞ Security
  ⊞ Security
  ⊞ Server Objects
```

Fig. 7. The structure of the database

4. CONCLUSIONS

This paper deals with an alternative manner of managing inward and outward transports of the Ford Motor Company plant in Craiova, but may also be implemented in other companies; it can be modified to act as a decision support subsystem. It also reduces costs by eliminating the need of sensors to keep track of the cars; the management is done purely in software with the help of a database.

The space optimization problem can be expanded to the management of fixed and mobile assets, such as buildings and vehicles, with the help of Geographical Information Systems, as in (Space Utilization Optimization, 2009) but that is for the moment beyond the scope of the paper and the original goal.

In the future the application may require more elaborate architecting, such as using the MVC pattern, as an alternative to Webforms Development and it could also include workflow characteristics.

REFERENCES

http://stackoverflow.com/questions/2828954/optimizing-a-parking-lot-problem-what-algorithims-should-i-use-to-fit-the-most-a

Zoghi, B., Singhal, R., Fink, R., Jung, Y. (2006). RFID Solutions: Parking Optimization and Customer Satisfaction, *Proceedings of the 2006 IJME - INTERTECH Conference.*

Environmental Systems Research Institute, Inc., 2009, 380 New York St., Redlands, CA, “Space Utilization Optimization”, an ESRI white paper.

Chen, M., Hu, C., Chang, T., (2011), The research on optimal parking space choice model in parking lots, *The 3rd International Conference on Computer Research and Development (ICCRD)*, vol 2, page 93, 11-13 March.

Meiping, Y., Ruisong, Y., Xiaoguang, Y., (2008), On Modeling on Scale of Public Parking Lot Based on Parking Choice Behavior, *International Conf. On Intelligent Computation Technology and Automation (ICICTA)*, 20-22 oct., vol. 2, pag. 259.

MacDonald, M., Szpuszta, M., *Pro ASP.NET 3.5 in C# 2008*, second edition, Apress, 2007.

MacDonald, M., Freeman, A., Szpuszta, M., *Pro ASP.NET 4.0 in C# 2010*, fourth edition, Apress, 2010.