Intelligent Parking Management Infrastructure Design

By

Mesbah Haque

Submitted to the Department of Civil and Environmental Engineering on May 9, 2003 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Civil and Environmental Engineering

ABSTRACT

This thesis discusses the different components required to build and design a prototype for the Intelligent Parking Management (IPM) infrastructure. Different aspects of the hardware and software components used to build the prototype is also discussed in the thesis along with discussion of business and marketing strategies for the viability of the solution for commercial use. A prototype solution was built using hardware and software components. The prototype was tested for real-time parking meter availability information for both general web customers and smart device users such as PDA with wireless access and Smart Phone mobile devices using 3G technologies. The tests were conducted in a controlled environment with simulation data for real parking meters.

The IPM prototype solution infrastructure built was able to provide real-time parking information. The information was accessible via the Internet through standard browsers and 3G enable Smart Phones. The integration between the hardware and software components and the IPM infrastructure design prototype was enabled through various technologies such as Microsoft .NET platform, Microsoft SQL Server 2000, Microsoft Mobile Internet toolkit, DTS, ASP.NET and ADO.NET. The thesis discusses these technologies and their interconnectivity within the IPM infrastructure.

Thesis Supervisor: Dr. John R. Williams
Title: Associate Professor of Civil and Environmental Engineering
ACKNOWLEDGEMENTS

I dedicate this thesis to my family for providing me with inspiration and encouragement while at MIT.

Firstly, I would like to thank my parents, Mozharul Haque and Zulekha Begum for all their sacrifice, support and hard work to get me to where I am today. I am thankful to my brothers: Manny, Monty and Roma for their encouragement and support. My thanks also goes out to my sister-in-laws. My special thanks to my endearing nephew, Zeshan, for letting me borrow his toy CLK Convertible to add reality to our IPM project prototype demonstration.

Secondly, I would like to thank my thesis advisor Dr. John R. Williams for his tutelage, support and friendship throughout this learning process. I would like to thank Dr. Eric Adams for his support. I would also like to thank Prof. George Kocur, Prof. Kevin Amaratunga, Prof. Jerry Connor, Prof Stuart Madnick, Prof. Brian Gibbs, Prof. Benjamin Grosof and Pat Vargas. My special thanks to Joan McCusker for all her help, Donna Hudson for helping me put together the MEng. IT trip to Tokyo, Pat Dixon for her support, Cynthia Stewart for her guidance, Patricia Maguire for her friendship and James Riefstahl for his friendship and humor. I am also thankful of all other MIT faculty, staff and students I had the pleasure of knowing during my studies here. I am very thankful to the people of Kajima Corporation – Tokyo and EDS – Boston for helping me with our trip to Tokyo.

Thirdly, I would like to give special thanks to my dearest friend, Sharon E. Walcott for her friendship, guidance, support, encouragement and help. I am also thankful to my friends Omar Hoda and Marco Ambrosoli for their help and support.

Lastly, but most importantly, I would like to thank God for blessing me with the opportunity to pursue my education this far.
# Table of Contents

TABLE OF CONTENTS ............................................................................................................................ 4
LIST OF FIGURES...................................................................................................................................... 6
LIST OF TABLES........................................................................................................................................ 7
CHAPTER 1 INTRODUCTION TO INTELLIGENT PARKING MANAGEMENT ........................................... 8

CHAPTER 2 THE PARKING PROBLEM AND THE BUSINESS OPPORTUNITY ....................................... 11
   2.1 INTRODUCTION TO THE PARKING PROBLEM AND THE BUSINESS OPPORTUNITIES ............. 11
   2.2 MARKET SIZE AND GROWTH OPPORTUNITIES: ....................................................................... 12
   2.3 COMPETITIVE LANDSCAPE: ....................................................................................................... 13
   2.4 COMPETITION AND COMPETITOR PROFILES: ......................................................................... 14
   2.5 TARGET MARKET SEGMENT: ....................................................................................................... 15
   2.6 MARKETING STRATEGY: ............................................................................................................... 18
      2.8 EXIT STRATEGIES .................................................................................................................... 19
      2.9 TEAM RESOURCES .................................................................................................................... 20

CHAPTER 3 INTELLIGENT PARKING MANAGEMENT SOLUTION CONCEPT .................................... 21
   3.1 INTRODUCTION TO INTELLIGENT PARKING MANAGEMENT SOLUTION CONCEPT .............. 21
   3.2 HARDWARE COMPONENTS: ....................................................................................................... 22
   3.3 SOFTWARE COMPONENTS: ....................................................................................................... 23
   3.4 SOFTWARE PROCESS: .................................................................................................................. 24
   3.5 IPM SOLUTION DEVELOPMENT PLAN: ....................................................................................... 25

CHAPTER 4 IPM SOLUTION INFRASTRUCTURE DESIGN REQUIREMENTS ..................................... 27
   4.1 INTRODUCTION TO IPM SOLUTION INFRASTRUCTURE DESIGN REQUIREMENTS ................ 27
   4.2 SOFTWARE REQUIREMENTS: ....................................................................................................... 27
      4.2.1 Users ..................................................................................................................................... 27
      4.2.2 System Features .................................................................................................................... 28
      4.2.3 Client Web Interaction .......................................................................................................... 28
      4.2.4 Database Requirements ....................................................................................................... 33
   4.3 HARDWARE REQUIREMENTS ....................................................................................................... 34
      4.3.1 Data Movement ..................................................................................................................... 34
      4.3.2 Collected Data Specifications ............................................................................................... 35
   4.4 EFFORT ANALYSIS ..................................................................................................................... 35
CHAPTER 5 IPM PROTOTYPE INFRASTRUCTURE .............................................................. 39

5.1 INTRODUCTION TO INTELLIGENT PARKING MANAGEMENT PROTOTYPE INFRASTRUCTURE ......... 39
5.2 IPM PROTOTYPE HARDWARE ...................................................................................... 39
  5.2.1 Transceivers: DLM-433-AT Master Logger & DLS-433-AT Remote Logger .................. 41
  5.2.2 HMC2003 Magnetic Hybrid: .................................................................................. 43
  5.2.3 Communication Sequence ................................................................................. 44
  5.2.4 Customization with the setup ............................................................................ 45
5.3 IPM PROTOTYPE SOFTWARE COMPONENTS: .............................................................. 46
  5.3.1 IPM Software Component Technology Overview: .............................................. 46
  5.3.2 Hardware Data acquisition and transformation ................................................... 54
  5.3.3 IPM Web Application Prototype ....................................................................... 59

CHAPTER 6 DISCUSSIONS AND CONCLUSIONS ............................................................ 69

6.1 DISCUSSIONS ........................................................................................................... 69
6.2 CONCLUSIONS ........................................................................................................ 70

APPENDIX A .............................................................................................................................................. 72

APPENDIX B .............................................................................................................................................. 76
List of Figures

Figure 1.1: IPM Prototype setup in the lab ................................................................. 9
Figure 1.2 IPM Infrastructure prototype vision ......................................................... 10
Figure 2.1: Ten most expensive market for parking per month ............................. 16
Figure 3.1 Intersections of Massachusetts Ave and Amherst Street, Cambridge, MA... 22
Figure 3.2: Conceptualized view of hardware components .................................... 23
Figure 3.3: Conceptualized view of the software components ............................... 24
Figure 3.4 Spiral Model as a basis for development ............................................... 25
Figure 3.5 IPM Product Development plan ............................................................... 26
Figure 4.1 IPM Client Web Interaction Illustration .................................................. 31
Figure 4.2 IPM Client Web Interaction UML Activity Diagram ............................... 32
Figure 4.3 Schema design for IPM database ............................................................ 33
Figure 5.1 Hardware Setup illustrations ................................................................. 40
Figure 5.2a DLM-433-AT Master Logger ................................................................. 42
Figure 5.2b DLS-433-AT Remote Logger ................................................................. 42
Figure 5.3 LoggerMaster Data Acquisition software showing triggered alarm ........ 43
Figure 5.4 Honeywell HMC 2003 Magnetic Hybrid Sensors ................................. 44
Figure 5.5 Microsoft .NET Architecture ................................................................. 49
Figure 5.6: ADO.NET Architecture ........................................................................ 50
Figure 5.7 (a) MMIT Architecture (b) MMIT Runtime Auto generate code ............. 52
Figure 5.8: Illustration of DTS architecture ............................................................. 53
Figure 5.9 Multi-step IPM SQL Server job ............................................................... 54
Figure 5.10: SensorDataX DTS package for IPM data transformation and updates .... 55
Figure 5.11 SensorStuff job multi-scheduling options ............................................ 58
Figure 5.12 SensorStuff job detailed scheduling options ......................................... 58
Figure 5.13: Illustration of IPM prototype web implementation .............................. 60
Figure 5.14 IPM Prototype General Web Portal Welcome page .............................. 61
Figure 5.15 IPM Prototype General Web Portal Search page ................................. 62
Figure 5.16: IPM Prototype General Web Portal parking availability result page .... 63
Figure 5.17 IPM Prototype Smart Device Web Portal Interaction .......................... 67
List of Tables

Table 2.1: Reasonable estimates of macroeconomic growth of the industry.................. 13
Table 2.2: Monthly Parking Rates in North America................................................... 17
Table 4.1: Estimated total Function Points and their complexities.............................. 37
Table 4.2: Estimated total Lines of Code................................................................. 37
CHAPTER 1

INTRODUCTION TO INTELLIGENT PARKING MANAGEMENT

Availability of free parking spots around major metro regions has become increasingly difficult. Parking has always been a growing problem in major cities like Boston. From a commuter’s perspective to be able to find a spot in the vicinity of their destination is further aggravated by the amount of time spent looking for a spot or an alternative. At the same time for a metropolitan city’s traffic department could benefit from the knowledge of proper maintenance and management of the parking meters. Therefore the information about parking is equally useful for both the commuter and the city managers. Intelligent Parking Management (IPM) project was developed here at Massachusetts Institute of Technology (MIT) in the Department of Civil and Environmental Engineering, Cambridge, Massachusetts, to alleviate the commuter’s issues and to better manage the parking meters using Information technology and the Internet as means for this solution. IPM project embodies latest sensor, wireless and emerging Internet technologies to support this IPM by delivering with real-time data to online clients. The premise of this thesis is to allow consumers of this information to make intelligent decisions and choices based on our IPM infrastructure implementation. These decisions are further facilitated through the application of popular and evolving technologies using PDA with wireless access and Smart Phones using 3G technology and service.

The motivation of this thesis therefore stems from the information passed to customers that empower them to make intelligent decisions; in this case, intelligent decision that deal with parking. Intelligent parking therefore is a term that describes an aspect that will be part and parcel of the next generation city. Drivers will be able to intelligently find parking owing to the availability of parking information over the web.
This thesis describes how to design the infrastructure needed to build the Intelligent Parking Management prototype (Figure 1.1). It discusses the parking problem that currently exists and then the business strategy that can be used to commercialize the IPM product. The thesis also discusses in detail about the various hardware and software components that were used to design the IPM prototype. It also discusses the various new technologies implemented in order to support the IPM infrastructure.

Figure 1.1: IPM Prototype setup in the lab
Figure 1.2 IPM Infrastructure prototype vision
CHAPTER 2

THE PARKING PROBLEM AND THE BUSINESS OPPORTUNITY

2.1 Introduction to the parking problem and the business opportunities

Parking is an extremely scarce and poorly managed resource in most major metropolitan cities. Since 1990, Boston’s population has grown about 3% while auto registrations grew 36% over that same period [1]. Additionally, there are significant barriers between supply and demand. Foremost, parking is government managed (net outsourcing) which creates a lack of efficient price discrimination. The primary cause, however, of inefficient parking management is the root cause in most economics problems – information deficiency or lack of information transparency. This is compounded by limited enforcement capability.

Specifically, the city of Boston faces increasing traffic congestion and commuters face increasing costs for private parking and decreasing availability of on-street parking. According to Boston-based Colliers International, a global partnership of independently-owned commercial real estate firms, Boston is the second most expensive market for both private and public parking in the U.S. The study, published in June 2002, indicated Boston drivers pay $463 a month for reserved parking or a daily rate of $31 [2]. Additionally, Boston mayor Thomas Menino recently announced new parking plans to reduce traffic congestion after a child died because the arrival of firefighters was delayed by illegally parked automobiles. Since the number of automobiles in Boston has increased by 36% since 1990, parking has become a significant problem and there is only a finite amount of real estate for new parking. There are already 550 garaged parking facilities in the city of Boston [3].

According to the Boston Department of Transportation, there are 7118 meters within municipal borders and each meter generates five dollars per day in revenue. On average,
parking meters are only utilized 50% of a typical ten hour day. This is deceiving since it averages meters in CBD’s which have utilization rate of close to 99% and more remote areas with utilization rates close to 0%. We can derive these averages from total revenue and number of operational meters. However, the DOT reports that on average, 29.2% of all meters are inoperable or about 2078 meters in a given day. The opportunity cost alone adds up quickly. 2078 inoperable meters multiplied by $5 is equal to $10,390 of lost revenue a day. That’s $3,241,680 a year in lost revenue, assuming a pay-for-parking policy of six days a week [4].

2.2 Market Size and Growth Opportunities:

In major metropolitan areas like New York, Boston, etc… garages can, and typically do, charge fees of $250 to $500 to $1,000 a month per motorist or per automobile. And prices are still escalating [5]. However, governments are not able to price discriminate as well as private parking garages because they have the much more difficult task of monitoring numerous on-street spaces. While a parking garage can closely monitor and manage a few hundred space within a single building, cities must monitor fluid traffic with thousands of spaces and not enough resources to enforce parking regulations. Although the demand for parking is increasing relative to the number of automobiles, space itself is a constraining factor. Therefore, the improved operational management of current resources requires better tools for acquiring relevant data and implementing policy.

Based on industry estimates, growth has a high degree of variance, mostly a function of metro area population and automobile growth. However, there are approximately five million commercial parking spaces in the U.S. and Central Parking operates approximately one and a half million or about 30%. Six large parking management companies control one million or about 20% and the remaining two and a half million are operated by roughly 1000 individual firms. Given these figures, it is hard to track overall growth for 50% of the industry. We can make approximations based on historical trends
and reasonably state that the industry grows at the five percent over the rate of inflation and Gross Domestic Product. This means prices generally increase with the rate of eight percent per year and capacity grows at seven percent per year. The table below represents reasonable estimates of the macroeconomic growth of the industry [6].

<table>
<thead>
<tr>
<th>Year</th>
<th>2004</th>
<th>2007</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monthly Unreserved Parking Rate - Average ($)</td>
<td>$171.83</td>
<td>$216.46</td>
<td>$272.68</td>
</tr>
<tr>
<td>Monthly Reserved Parking Rate - Average ($)</td>
<td>$224.21</td>
<td>$282.43</td>
<td>$365.79</td>
</tr>
<tr>
<td>Daily Parking Rate - Average ($)</td>
<td>$15.10</td>
<td>$19.03</td>
<td>$23.97</td>
</tr>
<tr>
<td>Total Dollars Spent on Parking by Commuter in North America (in billions $)</td>
<td>$35.09</td>
<td>$44.20</td>
<td>$55.68</td>
</tr>
<tr>
<td>Total Capacity (in millions of parking spaces)</td>
<td>5.72</td>
<td>7.01</td>
<td>8.59</td>
</tr>
</tbody>
</table>

Table 2.1: Reasonable estimates of macroeconomic growth of the industry

2.3 Competitive Landscape:

There are hundreds of firms in the parking management/services industry, filling various niches. There are several firms providing advanced technology solutions for parking management companies and local government organizations. Furthermore, some governments have produced internal solution initiatives. However, most of these firms have not delivered their products on significant scale. Most of the industry competitors are unproven with no successful implementations and have a very short business lifecycles. The best-known key to success in this industry is getting the first customer and delivering what you promised. This is easier said than done. We are developing and implementing a prototype solution here at MIT that is a proof of concept and validates our basic assumptions and our value proposition. However, the ultimate goal is a large scale system implementation in a Boston neighborhood.
2.4 Competition and Competitor Profiles:

Our competition could include local government initiatives, parking management companies and parking meter companies. Given our strategy of value creation and partnership, we will look at parking meter companies specifically. Although the mechanical parking meter industry has been around since 1935, almost all new parking meters sold today are electronic models. Whereas the mechanical parking meter market was dominated by two companies—Duncan Industries and P.O.M. Incorporated—the more recent electronic parking meter market is shared by Duncan, POM, newcomer SchlumbergerSema, and a host of smaller companies. Figure F shows the current market shares held in both of these markets.

SchlumbergerSema was formed in April 2001, and currently offers a broad portfolio of technology solutions: enterprise software systems, smart cards, payphones, parking meters, and mass transit terminals. As a leader in smart card technology, their parking systems are mainly designed to leverage their smart card innovations to provide flexible payment solutions to the parking industry. They are the first company to attempt to integrate wireless networking capabilities into their high-end, multi-space parking meters. Sema are experienced business and engineering operators, leaders in the smart card industry, pioneers in applying wireless technology to parking meters and already have established customers (sales in New York, Oregon, Texas, etc.). Most importantly they are backed by a financially stable parent company – Schlumberger Co. Ltd. However, they have very expensive products ($5000 – $20,000 per unit) and specializes in multi-space meters, not on-street parking.

POM has been the self-described leader in parking meter innovation since its inception when it was founded by the inventor of the parking meter. POM is aggressive in electronic parking meter systems, smart card meter systems, security solutions, and partnered technologies. POM is a privately owned corporation that manufactures parking meters in Russellville, Arkansas. The United States Patent Office has granted POM 17 patents with several still pending. POM has strong experience in the parking
meter industry, a strong portfolio of parking meter related patents, a very strong capital position ($18.4M in revenue in 2001) and well-established distribution partners. However POM customers must purchase the latest meter to get the advanced digital functionality. Additionally, POM does not offer networking capabilities or remote management and the functionality of meters cannot be significantly upgraded once installed.

Duncan Industries is a global provider of parking meters and related management systems. Based in Arkansas, Duncan continues to expand its domestic markets via its large network of distributors. It has recently begun to aggressively pursue next-generation parking meters that are smart-card enabled and which locally track revenue and usage patterns. Like POM, Duncan has strong experience in the parking meter industry and well-established distribution partners. But also like POM, Duncan customers must purchase the latest meter to get the advanced digital functionality and Duncan offers no networking capabilities or remote management.

Additionally, we can include parking management firms and other solution vendors that service the parking industry (net parking meter vendors). The main parking management firms have been previously discussed in this report in the Industry Analysis. VehicleSense, a Cambridge, MA based firm, with roots in MIT’s Media Lab and Urban Planning department, is a solution provider with local customers. Given the short product lead time, its few customers and the large unserved market, VehicleSense is considered our direct competitor but possible customer as well.

### 2.5 Target Market Segment:

With so many players in the value chain and such a fragmented and localized industry it is critical to identify the customer segment with the best strategic fit for advanced technology. Our initial target segment is the Parking Management organization within the Public Parking with Fee value chain. More specifically, we are targeting local
parking management companies/government organization within major metro area cities. Since we are most familiar with the business dynamics in the city of Boston, we will first target the Parking Management organization within the Public Parking with Fee value chain in Boston.

Cities with the highest concentration of commuters and CBD’s stand to benefit the most from better parking management. As an initial target segment, this will allow us to deliver proof of concept and validate our business proposition before approaching larger companies like Central Parking Corporation as an integrated vendor and partner. According to a Colliers International study of 50 CBD’s across the United States, parking rates in the northeast and west coast registered considerably more than the rest of the nation [7]. It makes sense for us to target the cities with the greatest need in order to deliver the greatest value. The chart below lists the top ten cities and our initial target segment:

![North America Monthly Unreserved Parking (June 2002)](image)

Figure 2.1: Ten most expensive market for parking per month
Among the largest cities, Atlanta ($100), Dallas ($100), Miami ($94) and Phoenix ($50) offer the lowest unreserved monthly parking. These cities would most likely not benefit dramatically from technology enabled parking systems and do not fall into our initial target segment. However, only 32% of major cities said parking was “limited” while 66% said parking was “fair” or “abundant.” Reserved parking rates fell 3% to an average of $192.22 and daily rates fell to 2.5% to an average of $12.95. Taken as whole, parking is becoming more constrained in the very largest metro areas and experiencing sufficient or ample capacity in the mid to small cities. This further helps to identify relevant target segments.

The Colliers study lists the major CBD’s with high, low and average figures for Monthly Unreserved, Monthly Reserved, Daily Parking and Additional Garage Information. Below is a summary table of the relevant data and can be used as a proxy to determine initial customer viability:

<table>
<thead>
<tr>
<th>North America Monthly Parking Rates June 2002</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monthly Unreserved Parking Rate - High ($)</td>
</tr>
<tr>
<td>Monthly Unreserved Parking Rate - Low ($)</td>
</tr>
<tr>
<td>Monthly Unreserved Parking Rate - Average ($)</td>
</tr>
<tr>
<td>Monthly Reserved Parking Rate - High ($)</td>
</tr>
<tr>
<td>Monthly Reserved Parking Rate - Low ($)</td>
</tr>
<tr>
<td>Monthly Reserved Parking Rate - Average ($)</td>
</tr>
<tr>
<td>Daily Parking Rate - High ($)</td>
</tr>
<tr>
<td>Daily Parking Rate - Low ($)</td>
</tr>
<tr>
<td>Daily Parking Rate - Average ($)</td>
</tr>
<tr>
<td>Garages Offering Additional Services (%)</td>
</tr>
<tr>
<td>Garages with Waiting Lists (%)</td>
</tr>
<tr>
<td>Typical Wait Period (months)</td>
</tr>
</tbody>
</table>

Table 2.2: Monthly Parking Rates in North America

---

1 Unreserved parking – The customer is guaranteed a space upon entry.
2 Reserved parking – The customer is guaranteed the same space for every entry.
3 Daily parking – The customer is permitted to park for a full day and is not effected by ‘early bird’ restrictions.
Specifically, we will use Monthly Unreserved Parking, Monthly Preserved Parking, Daily Parking and Garages Offering Additional Services as a proxy to assess customer viability.

**2.6 Marketing Strategy:**

We use the 5Ps of Marketing to develop our marketing strategy. We addressed Position, Promotion, Product, Placement, and Price. The most difficult to determine was price. In order to accurately price our products we used the 4Cs of Pricing. Our marketing philosophy follows the Geoffrey Moore “Crossing the Chasm” approach to marketing. Moore uses the analogy of “taking the beachhead,” dominating your initial target segment and then moving into adjacent, and logical, segments. For us, this requires capturing the early adopters and early majority of our initial segment of the top ten CBD. Additionally, this requires whole product delivery with follow on service, solidifying the customer relationship. The components of our marketing strategy can be defined using the 5Ps:

- Positioning – resource transformation
- Promotion – testimonial sales (1st customer)
- Product – technology solution for parking
- Placement – high end CBDs in top ten cities
- Price – industry leader in cost

Using the value-added pricing methodology, we price each transaction individually but adhering to a general baseline pricing structure. Since every city is unique, we must anticipate each customer will require a different price. Using the value-added pricing
method allows us to retain flexibility in our pricing lists and constantly yields a market value for our products, which have a minimum cost structure.

5.8 Exit Strategies

**Initial Public Offering.** The companies that do best in going public have strong and sustainable earnings growth or an explosive potential. Our focus on strong internal controls and systems will handle increased growth and timely financial reporting. The new funds will be used to finance growing operations, pay off debt, and provide liquidity for the company's early investors and employees by creating a market for the company's stock. It also provides a public image for venture capitalists, other investors and the general public, as an IPO can be one of the biggest marketing opportunities for a company. We know that by selling part of the company equity on the stock market, we will relinquish further control over operations and must submit to the rules for publicly listed companies. This late stage of financing is generally known as mezzanine financing [8].

**Acquisition.** As an alternative to going public, we are aware that many start-up companies choose to sell the company to another company. We will present almost the same information to a prospective acquirer as we would for an IPO. The buyer will be looking for depth of management, a history of audited financial statements, and consistent reinvestment of earnings into operations and regular business plans and projections. The negotiation of a company sale is complex and needs the assistance of qualified accountants, bankers and lawyers to make sure that you are getting the best deal possible. We will identify possible purchasers and go through the sales process [8].
5.9 Team Resources

The management team is a lean combination of engineering expertise and frugal business strategy. We will increase employees as function of sales in order to minimize operating expense and intellectual inventory and to maximize the throughput of our software sales. Our endgame organizational chart, once fully realized, will follow a functional organizational matrix. Key individuals from each department will be directly responsible for customer service. Appendix A contains information about each of the four IPM project team members.
CHAPTER 3

INTELLIGENT PARKING MANAGEMENT SOLUTION CONCEPT

3.1 Introduction to Intelligent Parking Management Solution Concept

The IPM project concept involved building a prototype to test the feasibility of a viable working demo model. To limit the test scope only three meters at the intersection on Massachusetts Avenue and Amherst Street in front of MIT building, Cambridge, Massachusetts was used as a primary setup venue. The initial prototype was to include hardware components such as magnetic sensors that could detect the presence of a vehicle and then relay that information wirelessly to a database that stores the information real-time. Once the data reaches the database it is to be analyzed and made available to users to be accessed. The end users both commuters and city staff will be able to access the IPM web server through the internet via PC or other smart devices such as PDA and mobile phones. The output to the user would be in both graphical and textual format. Power users will have further additional options to view various statistical analysis of parking meter utilization.
3.2 Hardware Components:

As mentioned above the general requirement includes detecting change in magnetic flux to determine the presence of a car and then forwarding that information through a transceiver to a master logger via wireless radio frequency (RF) communication. More detailed information of the hardware used is provided in Chapter 5 of this thesis. Figure 3.2 provides a conceptualized view of the hardware components.

Figure 3.1 Intersections of Massachusetts Ave and Amherst Street, Cambridge, MA
3.3 Software Components:

The software components include the movement of data from the sensor logs to the database where the information is analyzed. A web interface is also to be built to support user logins and access to the real-time parking information. This information is to be disseminated to various internet clients such as PCs, mobile phones with 3G support and also PDAs. The detail of each of these components is available in Chapter IV of this thesis. Figure 3.3 provides a conceptualized view of the software components.
3.4 Software Process:

The software development process chosen for the IPM project was the Iterative or Spiral model based on our total team membership of four and the time frame allocated to complete the project. Within the short timeline this method would allow the IPM team to go through at least two iterations before project deadline. This would allow building a working demo with minimum features yet demonstrating the main objective. Benefits in choosing this model include [9]:

- Coping well when requirements or schedule/resources are shifting or unpredictable or partially thought out
  - user interfaces, decision systems
  - fast moving markets or environments
- Enabling quicker start
- Encouraging involvement by users, marketing, biz partners
- Increasing concurrency in development process/labor
IPM team members’ diverse backgrounds would allow us to take multiple different roles within the Iterative cycles. Figure 3.4 shows the development cycles involved in a software project.

![Figure 3.4 Spiral Model as a basis for development [9]](image)

3.5 IPM Solution Development Plan:

Following the Iterative model, IPM will introduce new features with every version of product release. Figure 3.5 shows the product development plan that IPM will implement in the future product versions.
Figure 3.5 IPM Product Development plan

- **IPM v 1.0**
  - Web centric
  - Occupancy
  - Real Time Information

- **IPM v 2.0**
  - Web service
  - Meter Feeding Detection
  - "Rich" Content

- **IPM v 3.0**
  - Smart cards
  - v 1.0 features

- **IPM v 4.0 / 5.0 / 6.0 / etc.**
  - Compact Edition
  - v 2.0 features

- **Timeline:**
  - 2003
  - 2004
  - 2005
  - 2006
  - 2007
4.1 Introduction to IPM Solution Infrastructure Design Requirements

The IPM Solution Infrastructure is divided primarily into two categories namely software and hardware requirements. This chapter will cover the software and hardware requirements in detail and Chapter 5 will detail individual components of the IPM prototype built.

4.2 Software Requirements:

The software process used for the design of the IPM solution was the Iterative model (Section 3.4) using UML activity diagrams and web interaction illustrations to define requirements.

4.2.1 Users

The following shall be the primary users of the system:

- Drivers of automobiles in Boston
- Those looking for immediate parking in a specified location
- Boston Transportation Department (BTD) and other municipalities
- BTD is interested monitoring parking use in the city
- Internal Staff
- The IPM team shall require log in into the server from remote locations for maintenance purposes
4.2.2 System Features

The current system features incorporates the web interaction of users including commuters, power users and internal staff. The interaction is based on web access to IPM service web site. The system is logically represented as an n-tier environment. This includes client access through the Internet using standard web browsers and smart mobile devices, presentation layer through the IPM web server, the logic implemented through algorithms in the Microsoft SQL Server 2000 and lastly the data layer is represented within the Microsoft SQL Server 2000 IPM database.

4.2.3 Client Web Interaction

The overall client web interaction is illustrated in Figure 4.1 and Figure 4.2.

Client Website

The client web site will allow the client to interact with the IPM system and take advantage of the various services and offering that will be made available. The client’s web interaction will consists of various interactive pages.

Homepage

The homepage will include the following functionalities:

- IPM service offering information will be displayed.
- Existing clients can log in on this page to access the parking information input form. Login information (Login and password) will be verified against the information contained in the database before the customer can proceed. All existing clients will be registered in our database.
- Client will be redirected to either Free Parking Search (FPS) or Power User Tools (PUT) page based on their Roles membership.
- New clients will be redirected to the registration page where they will provide necessary information to be registered.
New Client Registration

New end users will be able to register with the IPM system via the web. The following information will be required from the users for registering including login, password, name, address, email and credit card information. Figure 4.3 provides the schema for this table. A confirmation will be sent to the user after successful registration.

Confirmation Page

This page contains the LoginID and a message confirming that the information has been successfully entered into the database. It also has links to the homepage in order for the user to retrieve specific parking information.

Error Page

This page will contain information that informs the user that the transaction has not been successful and directs the user to enter the information again. This page also has links that direct the customer either back to the registration page or the homepage.

Free Parking Search

Users will be able to at minimum provide a location name or a zip code to search for free parking meter availability. Initially the following features will be incorporated:

- Limited number of neighborhoods will be available through a dropdown list. The current list of possible neighborhoods will include:
  - MIT
  - Harvard Square
  - BackBay
  - Theater District
  - Fenway
  - North End
  - South End
• Clients will be given a checkbox option of whether to remember their neighborhood preference for future searches.
• Clients will submit their selection and the information will be sent to the database as they are redirected to the Free Parking Availability (FPA) page.

**Free Parking Availability**
This web page will provide the client with a map of the neighborhood they had selected with red or green points indicating the availability of the parking spots. Additionally the following features will be available in this page:

• Client will have the option to navigate and zoom into different parts of the map
• Client will also receive textual information of the map representation.
• This page will automatically render based on the type of user agents in different devices accessing the information i.e. PCs, PDA’s and Mobile Phones.
• Driving direction link will be provided for client’s benefit either via MapQuest or YahooMaps.

**Power User Tools**
This page will be only available to city staff and other metro city staff. On this page statistical information will be available for reporting purposes based on current and historical data stored in the IPM database. A link to Free Parking Search will be also available.

**Internal Staff Page**
The website will possess a simple interface for the maintenance staff. The interface will assist the internal staff do the following:

• View the database in its entirety
• Make changes to the database where necessary
Figure 4.1 IPM Client Web Interaction Illustration
Figure 4.2 IPM Client Web Interaction UML Activity Diagram
4.2.4 Database Requirements

The IPM system solution will use a relational database management system (RDBMS) to store information from the sensors as well information needed for the client web interaction as mentioned above. Microsoft SQL Server 2000 is selected for this purpose. All the information will be stored in tables in the IPM database server. Figure 4.3 shows the schema design for the IPM database.

![Schema design for IPM database](image-url)

Figure 4.3 Schema design for IPM database
4.3 Hardware Requirements

The sensor and transceiver used must conform to available power requirements i.e. low energy use will be important because sensor nodes will be placed in remote areas where power may not be immediately available. In this case, the lifetime of a node may be determined by the battery life, thereby requiring the minimization of energy expenditure. Sensors must be located in areas where they can be easily serviced and also in locations where they are not easily disturbed. Information pertaining to parking that is gathered via the sensor network is collected and fed into a database server. The sensors should meet the following minimum requirements criteria:

- The sensors of choice must be able to sense vehicle presence.
- Sensors unit must emit an analog signal.
- The transceivers used must be compatible with the sensors i.e. receive an analog signal from the sensor and convert it to a digital signal.
- The choice of hardware used i.e. sensors, transceivers and wireless access points (WAP) must be able to relay to the IPM server.

Chapter 5 goes into the details of the hardware selected and its setup for the IPM prototype. Figure 5.1 summarizes the various hardware sensor components connectivity.

4.3.1 Data Movement

IPM system implementation requires that parking data from parking sensor network must be delivered to the IPM server (a server that is controlled by the IPM team). The following model illustrates the infrastructure that we propose to deliver the data to our server. Communication between the master logger and the sensors are facilitated by RF transceivers fitted to each onboard sensor. The master logger will then pass the data to the attached computer. The data then will be delivered to the appropriate server through the local area network (LAN). Once the data is delivered to the IPM server, it can be cleaned, analyzed and distributed in any what that suits the end user.
4.3.2 Collected Data Specifications

Data collected from the parking sensors must be available via the internet. Data collected by the sensors should be edited by relevant system and must be easy to maintain. The data transferred via the wireless network must not exceed the networks capacity and data rates must be within appropriate range. Figure 4.3 illustrates the data storage schema in the IPM database.

4.4 Effort Analysis

Clients of the IPM Project will utilize our services through an ASP.NET application. This application will be supported by a single database. This document details the effort estimation necessary to build both of these items.

We will use a Function-Point approach for estimating the size of our system. Steve McConnell describes this approach in Rapid Development [9]. To work through this algorithmic approach, we must first identify how many items we have that match the following:

**Definitions:**

**Inputs:** Screens, forms, dialog boxes, controls or messages through which an end-user adds or changes a program's data. For our purposes, these are the web pages that users can enter simple information into (Login, Search Parameters, etc) to achieve a desired result.

**Outputs:** Screens, reports, graphs that the program generates for use by an end-user. For our purposes, these are the web pages that contain neighborhood selection and parking information.
**Inquiries:** Input/output combinations in which an input results in an immediate, simple output. Queries retrieve data directly from a database and provide only rudimentary formatting. For our purposes, these are the web pages that provide the Power User Tools.

**Logical Internal Files:** Single table in relational database.

Here is a listing of our total set of web pages: Home Page, New Client Registration, Free Parking Search, Free Parking Availability, and Power User Tools.

These are categorized as follows:

**Inputs:** Home Page, New Client Registration, Free Parking Search. All of these are of medium complexity.

**Outputs:** Free Parking Availability, Hard.

**Inquiries:** 3 Power User Searches: Total Revenue Based on Availability, Frequency of Vehicle Occupancy, Overnight Occupancy. All of these are of medium complexity.

Here is a listing of our total set of database tables: ShortTermSensorData, LongTermSensorData, Sensor, StreetCode, Neighborhood, User, Roles. Of these, all are low complexity with the exception of Sensor and User. Sensor and User are medium complexity.
<table>
<thead>
<tr>
<th>Program Unit</th>
<th>Low Complexity</th>
<th>Medium Complexity</th>
<th>High Complexity</th>
<th>FP / PU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Inputs</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Web Outputs</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Web Queries</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Database Tables</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td><strong>Total Function Points</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>86</strong></td>
</tr>
</tbody>
</table>

Table 4.1: Estimated total Function Points and their complexities

Our total function points is <100. This reflects the minimal number of web pages and database entities that we have elected to implement.

<table>
<thead>
<tr>
<th>Program Unit</th>
<th>Tool</th>
<th>Lines / Function Point</th>
<th>Lines / Program Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>ASP.NET</td>
<td>45</td>
<td>1395</td>
</tr>
<tr>
<td>Database</td>
<td>SQL Server</td>
<td>40</td>
<td>2200</td>
</tr>
<tr>
<td><strong>Total Lines Of Code</strong></td>
<td></td>
<td><strong>3595</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Estimated total Lines of Code

We are making the assumption that C# will require approximately 45 lines of code (LOC) per function point (FP). The basis for this assumption is that C# will require slightly less lines of code per function point than C++ (50). We are assuming that SQL Server will require the same number of LOC per FP as Oracle (40).

**Person Months**

IPM is in the McConnell category of a “Business Product.” Assuming that we are operating under a nominal schedule, we could build a 10,000-line product in 6 schedule months and 9 person months. Our system is approximately 36% of the size so it will take slightly less than 36% of 9 person months to build. 3 months is a good, approximate estimate.
**Schedule Months**

Assuming that we could build a 10,000-line project in 6 schedule months, we use the Capers Jones “first-order estimation” to estimate that 3,600 lines could be completed in about \((3,600/10,000)^{0.4}\) time, or about 0.66 the time. Where 0.4 is the exponent associated with building best in class Business Product. So, 6 schedule months is reduced to 4 months.

**Ranges**

We are in the initial project definition stage. This means that the true amount of effort needed to complete the project could be as low as 0.25 or 4.0 times as much as we have currently projected. This means that our true range will lie somewhere between 0.8 months and 13 months. Finally, our schedule may range between 0.6 and 1.6 times what we have currently projected, meaning the true range is between 2.4 and 6.4 schedule months.
CHAPTER 5
INTELLIGENT PARKING MANAGEMENT PROTOTYPE INFRASTRUCTURE

5.1 Introduction to Intelligent Parking Management Prototype Infrastructure

This chapter describes in detail the actual infrastructure implemented for the IPM project. It describes the different components of the hardware and software and their functionalities. The first part of this chapter covers the hardware aspect of the infrastructure to build the prototype followed by the software. Hardware section will cover sensors, transceivers and loggers. Whereas, software section will cover data transfer from sensor to IPM database, intelligent parking detection algorithm using Microsoft SQL Server 2000 components such as Data transformation Service (DTS) and Jobs, and code developed for the web interaction for both PC and smart mobile devices on the Microsoft .NET platform.

5.2 IPM Prototype Hardware

The following equipment has been chosen for our prototype purposes because it conforms to the specifications and constraints listed in the hardware design parameters:

- Transceiver Model DLM-433-AT Master Logger
- Transceiver Model DLS-433-AT Remote Logger
- Honeywell Sensor (Integrated with circuit board) HMC2003
- The above equipment will be wired according to the following illustration

39
The HMC2003 Honeywell sensors are directly connected to the DLS-433-AT remote transceiver via copper wire. The transceiver has the capacity to host 8 sensors and therefore is operating below its capacity for prototype purposes. The DLS-433-AT Remote transceiver communicated with the DLM-433-AT Mater transceiver wirelessly via a master slave architecture.
5.2.1 Transceivers: DLM-433-AT Master Logger & DLS-433-AT Remote Logger

The system comprises of a master DLM-433 data logger radio transceiver (Figure 5.2a) which connects via an RS232 port to a PC running the included data logger software program. The DLM-433 communicates with up to 4 remote DLS-433 radio data loggers (Figure 5.2b), each of which contains an eight input, 12 bit resolution A/D converter.

The full configuration of one master DLM and four remote DLS modules allows for up to 32 analog voltages to be monitored. With the minimum configuration of one master DLM module and one remote DLS module up to eight analog voltages may be monitored. The measuring range for each analog input is 0-2.5Vdc. Appendix A shows the connection details of the DLM and the DLS modules.

The data acquisition software program allows individual addressing and reading of each remote DLS module (Figure 5.3). The results of each DLS debriefing is updated in the display window for each active channel. The program also features an AutoLog mode which is free running and will collect the data from the selected DLS continuously and update the display accordingly.

Additional features include [10]:

- Up to 400ft range
- 5.5V-12Vdc operation, 30mA
- DLM dimensions: 63mm x 87mm x 30mm
- DLS dimensions: 64mm x 81mm x 30mm
- 418MHz or 433MHz FM operation
- DLS addressing via jumper setting
- Compatibility with a wide range of sensors
- 3 wire RS232 interface
Figure 5.2a DLM-433-AT Master Logger

Figure 5.2b DLS-433-AT Remote Logger
Figure 5.3 LoggerMaster Data Acquisition software showing triggered alarm

5.2.2 HMC2003 Magnetic Hybrid:

Honeywell’s three-axis magnetic sensor hybrid uses three permalloy magnetoresistive transducers and custom inter-face electronics to measure the strength and direction of a magnetic field [11]. These transducers are sensitive to magnetic fields along the length, width, and height (x, y, z axis) of the 20-pin dual-in-line hybrid. Fields can be detected less than 40 microgauss and up to ±2 gauss. Analog outputs are available for each x, y, z, axis from the hybrid. With the sensitivity and linearity of this hybrid, changes can be detected in the earth’s magnetic field to provide compass headings or attitude sensing. The high bandwidth of this hybrid allows anomaly detection of vehicles, planes and other ferrous objects at high speeds. Appendix B shows the HMC2003 specifications.
5.2.3 Communication Sequence

The remote logger (DLS) initially sends a two byte command to the DLM (Master Logger). The first byte is sent and followed by a byte that specifies the address of the logger. The DLM Master logger then broadcasts a RF formatted data request to a specific DLS remote logger address to retrieve their information. After broadcasting the data request, it reverts to receive mode and awaits data from the remote logger address. Once the DLS logger receives the request from the Master, and the address in the request matches its own, it reads 8 analog voltages at the A/D converter to give a 12 bit voltage reading. The DLS then converts the digital signal to RF and transmits it to the DLM where the data is transmitted to the computer where it is interpreted, displayed and stored by the LoggerMaster Software.

It is important to note that the DLM is directly connected to a computer that sits on a MIT network. The LoggerMaster Software has been installed on this machine so that it communicates seamlessly with the DLM to interpret, display and store data. Once the data is on this computer, it can be imported to a web-server on the network where it is accessible via the web; Section 5.3 discusses the software components involved.
5.2.4 Customization with the setup

The sensor equipment purchased requires some signal processing before it can be directly connected to the transceiver for readings to be taken. The output from the HMC2003 unit is in the range of 0.5V to 4.5 V while the input specification for the transceiver is 0V to 2.5 V. To condition this signal, a rather crude approach can adopted to minimize sensor output voltage. A 100 ohm resistor can be inserted between the remote logger and the sensor to drop the maximum voltage from 4.5 V to 2.5 V. The there may be significant signal loss due to this but we anticipate that the signal loss will not affect the interpretation of data at a later stage.
5.3 IPM Prototype Software components:

Software components used for supporting the IPM prototype Internet based services includes:

- Microsoft Windows 2000 Advanced Server operating system
- Internet Information Services 5.0 (IIS)
- FrontPage Server extensions
- Microsoft .NET Platform with support for Web Services and ADO.NET
- Microsoft Internet Mobile Toolkit
- Microsoft SQL Server 2000 Enterprise Edition

The IPM server for the prototype was installed with all of the above components. The name of the server was verrix.mit.edu. The following subsection provides technology overview of the components used.

5.3.1 IPM Software Component Technology Overview:

The software technologies used to implement the IPM prototype infrastructure are as follows [12]:

**Microsoft Windows 2000 Advanced Server operating system**

Microsoft Windows 2000 Advanced Server is the server operating system for line-of-business applications and e-commerce. Windows 2000 Advanced Server includes all the features and application availability of Windows 2000 Server, with additional scalability and reliability features, such as clustering, designed to keep business-critical applications up and running in the most demanding scenarios. To increase availability, Advanced Server includes two clustering technologies: Cluster service and Network Load Balancing. Cluster service is used to connect two servers together so that one can pick up the load if the other fails. This is appropriate for any application that you need to have running
without interruption. NLB lets you spread processing across as many as 32 servers, for example to handle incoming traffic for your Web site or to support Terminal Services.

Building on the solid Internet technologies delivered in Windows NT® Server 4.0, Windows 2000 Advanced Server provides a well-integrated package containing the application development environment, security, and scalability needed to get more out of existing applications.

**Internet Information Services 5.0**

The features in Internet Information Services (IIS), a part of Microsoft Windows 2000 Advanced Server, make it easy to share documents and information across a company intranet or the Internet. With IIS, one can deploy scalable and reliable Web-based applications, and can bring existing data and applications to the Web. IIS is a set of services that that supports web site creation, configuration, and management, along with other Internet functions. Microsoft Internet Information Services include Network News Transfer Protocol (NNTP), File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP). Internet Information Services is also called IIS.

**FrontPage Server Extensions**

The FrontPage Server Extensions make it easy for web authors to include sophisticated functionality in their web sites and for web administrators to support that functionality. The server extensions:

- Enable authors to collaborate on creating and maintaining a web site, edit a web site directly on the server computer (saving downloading time), and add functions to their Web sites easily, without doing any programming.
- Support hit counters, full-text searches, e-mail form-handling, and other functions that an author can add to a Web site by using FrontPage. One doesn’t have to download, buy, or install a separate -compatible program to enable each function to work.
- Automatically update hyperlinks after a page in a web site is moved, deleted, or renamed (only the page's file name, rather than the entire file, needs to be transmitted to the web server).
- Support integration with Microsoft Office, Visual Studio.NET, and Index Server.

**Microsoft .NET Platform with support for Web Services and ADO.NET**

The .NET Framework (Figure 5.5) is an integral Windows component for building and running the next generation of software applications and Web services. The .NET Framework:

- Supports over 20 different programming languages.
- Manages much of the plumbing involved in developing software, enabling developers to focus on the core business logic code.
- Makes it easier than ever before to build, deploy, and administer secure, robust, and high-performing applications.
- The .NET Framework also offers significant performance and scalability benefits over the previous Active Server Pages (ASP) technology, thanks to its just-in-time (JIT) compilation and caching technologies.

ASP.NET is a set of technologies in the Microsoft .NET Framework for building Web applications and XML Web services (Figure 5.5). ASP.NET pages execute on the server and generate markup such as HTML, WML, or XML that is sent to a desktop or mobile browser. ASP.NET pages use a compiled, event-driven programming model that improves performance and enables the separation of application logic and user interface. ASP.NET pages and ASP.NET XML Web services files contain server-side logic (as opposed to client-side logic) written in Visual Basic .NET, C# .NET, or any .NET-compatible language. Web applications and XML Web services take advantage of the features of the common language runtime, such as type safety, inheritance, language interoperability, versioning, and integrated security.
ADO.NET is a set of classes that expose data access services to the .NET programmer. ADO.NET provides a rich set of components for creating distributed, data-sharing applications. It is an integral part of the .NET Framework, providing access to relational data, XML, and application data. ADO.NET supports a variety of development needs, including the creation of front-end database clients and middle-tier business objects used by applications, tools, languages, or Internet browsers.

The .NET Framework ships with two .NET Framework data providers: the .NET Framework Data Provider for SQL Server and the .NET Framework Data Provider for OLE DB. Since the IPM prototype was using Microsoft SQL Server 2000, Data Provider for the SQL Server was used. The following diagram illustrates the components of ADO.NET architecture.
The Microsoft Mobile Internet Toolkit (MMIT) contains server-side technology that extends the Microsoft ASP.NET programming model to deliver content to a wide variety of mobile devices. Because each device can have a unique combination of capabilities, the Mobile Internet Toolkit provides an abstraction layer so developers can write applications without worrying about the specific details of each device.

The Microsoft Mobile Internet Toolkit addresses these following challenges by isolating them from the details of wireless development:

- Different markup languages necessity, including HTML for PDAs, wireless markup language (WML) for wireless application protocol (WAP) cell phones, and compact HTML (cHTML) for Japanese i-mode phones.
- Devices having different form factors. For example, devices have varying numbers of display lines, horizontal or vertical screen orientation, and color or black and white displays.
- Devices having different network connectivity, ranging from 9.6 KB cellular connections to 11 MB Wireless LANs.
- Devices having different capabilities. Some devices can display images, some can make phone calls, and some can receive notification messages.

Thus, developers can quickly and easily build a single, mobile Web application that delivers appropriate markup for a wide variety of mobile devices.

The mobile Web Forms controls are ASP.NET server-side controls that provide user interface elements such as list, command, call, calendar, and so on. At execution time, the mobile controls generate the correct markup for the device that makes the request (Figure 5.7b). As a result, a mobile application can be written once and accessed from multiple devices. Figure 5.7a shows the MMIT architecture.

MMIT also allows reuse of the same business logic and data access code that is used in desktop applications. Mobile and desktop Web Forms can reside in the same Visual Studio .NET project. This would enable faster development of IPM web applications for the mobile devices.
Microsoft SQL Server 2000 Enterprise Edition

Microsoft SQL Server is a family of products and technologies that meets the data storage requirements of OLTP and OLAP environments. Microsoft SQL Server is a relational database management and analysis system for e-commerce, line-of-business, and data warehousing solutions. SQL Server 2000, the latest version, includes support for XML and HTTP, performance and availability features to partition load and ensure uptime and advanced management and tuning functionality to automate routine tasks and lower total cost of ownership. SQL Server is fully integrated with Windows 2000 and takes advantage of many of its features. SQL Server is integrated with the security system in Windows 2000. This integration allows a single user name and password to access both SQL Server and Windows 2000. SQL Server also uses encryption features in Windows 2000 for network security, including Kerberos support. SQL Server provides
its own security for clients that need to access SQL Server without authentication by Windows 2000.

SQL server also provides a Data Transformation Services (DTS) utility which was used during the implementation of the IPM prototype.

DTS uses an OLE DB provider to transform and transfer data from one or more sources and export it to one or more destinations (Figure 5.8). When you use DTS, the data source and destination can be heterogeneous. Neither the data source nor destination has to be SQL Server. DTS can be the mechanism that transfers data between two data sources. OLE DB supports accessing any data storage format—relational or non-relational—for which an OLE DB provider is available. Each OLE DB provider is specific to a particular storage mechanism, such as SQL Server databases, Microsoft Access databases, or Microsoft Excel spreadsheets.

![Figure 5.8: Illustration of DTS architecture](image-url)
5.3.2 Hardware Data acquisition and transformation

Vehicle presence data collected through the sensors were continuously passed on to the transceiver, DLS-433-AT Remote Logger, which in turn passed the data to the DLM-433-AT Master Logger which was then monitored and captured by the LoggerMaster software to an attached PC via the serial port. The LoggerMaster software stored the real-time data onto a local text based log file. Since direct processing of the log file was not allowed by another process from the IPM server, a multi step SQL Server job called SensorStuff (Figure 5.9) was created. The integrity of the job was maintained by serializing the steps and quitting the job if any of the steps failed for any reason.

![Figure 5.9 Multi-step IPM SQL Server job](image)

The first step, *CopyLog*, was to copy the file over locally to the IPM server for data transformation and processing. In order to achieve this goal with flexible option to connect to the PC where the logs were generated, a Transact-SQL (T-SQL) job was defined. The following code was used to achieve above step:
The Run DTS SensorData Package step involved the running of a DTS package named SensorDataX that was stored in the IPM SQL Server (Figure 5.10). The package begins with a connection to the locally copied log file and then transforms the format to upload the data to the SensorData table in the IPM database by making a connection to the IPM local server. For the IPM prototype the SensorData table was always dropped first before repopulating with cumulative sensor data. This ensured guaranteed data consistency for the preliminary prototype testing. Appendix B shows details of the various steps within SensorDataX package.

Figure 5.10: SensorDataX DTS package for IPM data transformation and updates
The SensorDataX package was run from the SensorStuff job by using operating system command step that called the Dtsrun.exe executable to the run the package by proving parameters including the server name (/S), username (/U), password (/P) and package name (/N) as shown below:

```
dtsrun /SVerrix /UJedSQL /Ppipmjed /NSensorDataX
```

The GenerateShortData step’s function was to run a T-SQL job step to collect a subset of the latest real-time data that was updated to the SensorData table by the above step and store it into the ShortSensorData table in the IPM database. It was determined through trials that about 200 entries were logged by the sensors per minute. Therefore the logic implemented was to collect the latest 200 records from the SensorData table based on timestamp and copy only the relevant columns for Channel 0 only. Channel 0 was used by the sensor for the relevant axis for magnetic flux variation. The T-SQL used is shown below:

```
truncate table ShortSensorData
insert ShortSensorData
select top 200 [time], channel, emf, alarm
from sensordata

where channel = 'channel 0'
order by [time]desc
```

Finally, the UpdateShortTermSensorData job step was also defined as a T-SQL statement. This step defined the algorithm used to determine whether a vehicle was present at the parking meter or not based on the count of the alarms triggered stored in the ShortSensorData table. It was determined that if there were over 30 alarms in the ShortSensorData table rows then there was a vehicle present. Then the
ShortTermSensorData table would be updated with the availability information along with the timestamp of when the process was run. The following code below shows the algorithm used:

```sql
if (select count(alarm) from ShortSensorData where alarm='ALARM') > 30
update ShortTermSensorData
set available = 0, time = getdate()
where SensorID = 1
else
update ShortTermSensorData
set available = 1, time = getdate()
where SensorID = 1
```

The SensorStuff job provided the middle-tier component of the IPM prototype implementation. It allowed modularity in testing and designing the other components of the IPM prototype. The SensorStuff job can support flexible multi-scheduling options. The IPM prototype was tested with different timing intervals for data accuracy and data transformations. Figure 5.11 and Figure 5.12 shows the IPM SensorStuff job scheduling options.
Figure 5.11 SensorStuff job multi-scheduling options

Figure 5.12 SensorStuff job detailed scheduling options
5.3.3 IPM Web Application Prototype

The IPM web component consisted of two web applications, one for general Internet access methods and secondly for the smart mobile device based internet access. Each of the application portals was hosted at the IPM verrix server at:

- http://verrix.mit.edu/MobileIPM - IPM Prototype Smart Device Web Portal

Both the portal provides the users with a consistent interface. The general overview of the entire implemented IPM prototype is illustrated in Figure 5.13.

The verrix server used for hosting the above web portals along with all the server software mentioned above was equipped with a 600 MHz Pentium III processor with 768 MB of RAM and 100 GB hard disk space. The Internet connectivity was through a 10 Mb Ethernet LAN routed through the MIT backbone network.
Figure 5.13: Illustration of IPM prototype web implementation
IPM Prototype General Web Portal

The client interaction with the general IPM internet portal begins with the welcome page. Here the users have the option to either login or register as a new client. The prototype login was based on forms based authentication. Forms authentication refers to a system where unauthenticated requests are redirected to a Hypertext Markup Language (HTML) form (by using HTTP client-side redirection). The user provides credentials and submits the form. If the application validates credentials on the form, the system issues a cookie to the user. Subsequent requests are issued with the cookie in the request headers; therefore, they are authenticated. The authentication mode is defined in the application web configuration file (web.config) at the root of the IPM Web application.

![IPM Prototype General Web Portal Welcome page](http://www.example.com/ipm.png)

Figure 5.14 IPM Prototype General Web Portal Welcome page
Once the client is validated, the user is redirected to the search page (search.aspx) as shown in Figure 5.15. Here the user is given the choice to pick a location from a dropdown list. Currently only MIT-West location is active for the test of the IPM prototype implementation.

![Search page with location selection](image)

**Figure 5.15 IPM Prototype General Web Portal Search page**

Based on the selection of the location, availability of parking is checked against the IPM database and an appropriate image is displayed based on the parking availability page (availability.aspx). A red or green star is displayed to indicate availability of the parking spot (Figure 5.16). A textual based table is also displayed representing the same information as in the map image. The simulated parking meters are numbered according to the illustration in Figure 3.1.
The ASP.NET availability page is generated based on a search of the IPM database and the selection of an appropriate map image for display. The database lookup is done via ADO.NET calls made through the code behind pages using C# language. The database lookup is implemented using the following code:
// Determine which neighborhood to search
string nbhood = Request.QueryString["Nb"]; 

string cString = ConfigurationSettings.AppSettings["ParkingDB"];  
SqlConnection connection = new SqlConnection(cString);  
string sqlString = "SELECT S.SensorID, SI.Street, SI.Description, STSD.Available, STSD.Time "; 
sqlString += "FROM Sensor AS S, StreetInfo AS SI, ShortTermSensorData AS STSD "; 
sqlString += "WHERE S.StreetInfoID = SI.StreetInfoID AND S.SensorID = STSD.SensorID"; 

SqlDataAdapter adapter = new SqlDataAdapter(sqlString, connection);  
DataSet result = new DataSet(); 
adapter.Fill(result);  
DataTable sensorData = result.Tables[0]; 

// Select proper Image 
string imageName = "..\images\MassAve";  
int imageNumber = 0; 

foreach(DataRow row in sensorData.Rows) { 
    int sensorID = ((int)row.ItemArray[0]) - 1; 
    bool occupied = (((short)row.ItemArray[3]) == 1); 
    if (occupied) { 
        imageNumber += (int)Math.Pow(2, sensorID); 
    } 
} 

imageName += imageNumber + ".gif"; 
resultImage.ImageUrl = imageName;
The textual version of parking availability information is generated by cloning the above results table and then descriptive availability information is updated into the datagrid as follows:

```csharp
DataTable modTable = sensorData.Clone();
modTable.Columns.Remove("SensorID");
modTable.Columns["Available"].DataType = Type.GetType("System.String");
foreach(DataRow origRow in sensorData.Rows) {
    DataRow newRow = modTable.NewRow();

    short oldAvail = (short) origRow.ItemArray[3];

    newRow["Street"] = origRow["Street"];
    newRow["Description"] = origRow["Description"];
    if (oldAvail == 0) {
        newRow["Available"] = "No";
    } else {
        newRow["Available"] = "Yes";
    }
    newRow["Time"] = origRow["Time"]; 
    modTable.Rows.Add(newRow);
}
```

The web.config file defines application keys for the connection string parameters to the IPM database server. In the above codes the “ParkingDB” is the key variable with the IPM database connection string information.
IPM Prototype Smart Device Web Portal

The Smart Device Web Portal interaction and experience is very similar to the IPM Prototype General Web Portal interaction. There are some subtle differences taken into account due to variations in device specifications. It is assumed that clients are already registered with the IPM database and that the Smart Device browser at least supports WAP 2.0 technology to maximize the user experience. The entire Smart Device Web Portal interaction is illustrated in Figure 5.17.
The support for IPM Prototype Smart Device Web Portal is based on MMIT platform on top of .NET. The web portal automatically renders the pages based on the type of smart
devices interaction as discussed in Section 5.3.1. Appendix B shows the coding used to generate the web pages along with support for ASP.NET mobile web controls.

In order to display the map images in the smart device WAP browsers, the images had to be converted to Potable Network Graphics (PNG) format from the GIF format used in the general prototype web portal.
CHAPTER 6

DISCUSSIONS AND CONCLUSIONS

6.1 Discussions

The IPM project prototype implemented provided the feedback necessary to scale this project to actual production level implementations. There were several areas in which changes could be made. This includes both hardware and software aspects. The hardware used for this prototype in general served the purpose for the project but in order to commercialize the product it would be necessary to test the sensitivity of the change in the magnetic flux to accurately detect vehicle presence. The tests were conducted in a controlled environment to facilitate the integration with the software components. Also the attachment of the sensors along with parking meters beyond controlled environments needs to be investigated. The transceivers used worked very well for the prototype. The lack of source code for the LoggerMaster software did not allow us to customize the software to accommodate various magnetic flux sensitivity options. We were also limited by the detection range preset by the vendor software.

The software components used to build the IPM prototype integrated very well. The Microsoft .NET framework platform with support for Microsoft Mobile Internet Toolkit (MMIT) along with ASP.NET provided the foundation necessary to build the software infrastructure needed to support various types of web clients. Although the reporting tools for the power users were not implemented in this prototype, it is being currently tested for the next version of the IPM tools. The algorithms used to determine vehicle presence would need to be updated and optimized inline with any changes in hardware sensitivity. Stored procedures were considered for the optimized use of the IPM database but were used minimally in the current prototype version. The web pages were designed to meet minimal requirements to demonstrate the workings of the IPM infrastructure prototype and would need to be redesigned for commercial implementation. Further...
optimization could be achieved by leveraging .NET DLLs for the code behind calls to the IPM database server. MMIT allowed enhancements for the IPM prototype smart device web portal by automatically rendering web pages for different smart device clients’ thus increasing efficiency and reducing coding complexity in the IPM prototype infrastructure design implementation.

6.2 Conclusions

It is concluded from the implementation of the IPM infrastructure design prototype that a viable real-time parking solution can be commercially tested and marketed. The various hardware and software components used to build the prototype can be used to scale up for large implementations. It is also concluded that the software architecture design used was successful in relaying real-time parking information for both general web access methods and also through 3G enabled smart devices.
REFERENCES


APPENDIX A

IPM Team Members:

Mesbah Haque

Since 1998, Mesbah has been the president and founder of i.e.Networks, Inc., located in Medford, MA. The services provided at i.e.Networks involve providing Microsoft based solutions in Training, Systems, Network and Development; providing training on Microsoft .NET platform technologies; planning and configure messaging infrastructure solution using Microsoft Exchange Server; database administration and design solutions using Microsoft SQL Server, Access and integration with Oracle, Sybase, DB2 and Informix for analysis through Data Mart/Warehouse. Mesbah has been a freelance technical specialist since 1996 and a Microsoft Certified Partner since 1999. As a Certified Microsoft instructor, Mesbah has delivered Microsoft Official Curriculum (MOC) courses, and developed custom courses, in corporate classroom environment on state-of-the-art technologies.

Mesbah will receive his Master of Engineering in Information Technology from the Massachusetts Institute of Technology in June, 2003. In addition to his studies, Mesbah is a research and teaching assistant for the Malaysia University of Science & Technology (MUST) and MIT collaborative initiative for Professor John Williams, Director of Intelligent Engineering Systems Lab. He received his Bachelor of Science in Engineering, Civil Engineering from the University of Massachusetts – Lowell, MA in 2001, with honors: Cum Laude and Dean’s List.

Mesbah has numerous certifications to include Microsoft Certified Systems Administrator (MCSA) 2002, Microsoft Certified Professional + Site Building (MCP + Site Building) 2002, Microsoft Certified Solution Developer (MCSD) 2002 and CompTIA Certified Technical Trainer (CTT+) 2001.
James Muriithi

James has been a civil & environmental engineer General Motors Corporation from 2000 to 2002. Additionally, he developed software to integrate, analyze and report data reducing turnaround time for delivery of environmental reports by 90%; James designed a web-based document control system to reduce time spent on maintenance of document control centers by 80%; James implemented environmental management system based on International Organization of Standards (ISO 14001). James maintained the GM site’s environmental compliance (i.e. Implemented Clean Air Act (CAA), Title V Permit, Resource Conservation Recovery Act (RCRA), Department of Transportation DOT guidelines, Pollution Prevention PP, Resource Management and Regulatory Reporting). In two years, James conducted regular management reviews, trained environmental auditors, planned regular internal audits, and developed & implemented various preventative and corrective actions.

James will receive his Master of Engineering in Information Technology from the Massachusetts Institute of Technology this June. Relevant coursework includes Foundations of Software Engineering & Web Services, Database Internet & Telecom System Integration and Software Architecting. Relevant computer skills include a working knowledge of Java, C#, .NET Platform (Microsoft Visual Studio) SQL, XML, HTML, VB, VBScript, JavaScript, VBA and AUTOCAD. James graduated in 2000 from Tennessee State University with a Bachelor of Science in Civil & Environmental Engineering, Summa Cum Laude. James is also an All American Scholar, Presidential Scholar, and member of the Golden-Key National Honor Society.
Dion Edge

Dion was, most recently, an internal consultant for Bracom Corporation, a healthcare IT solutions provider in Dallas, Texas. Prior to that, Dion founded and sold his own company, ModernPopArt, an online retail art, clothing and merchandise retailer. He employed and managed a dozen fine and graphic artists, web designers and general laborers; managed company finances, developed marketing strategy and managed operations; and Dion financed his graduate education with sale of the company to W3Commerce in 2001. Finally, Dion served his country in the US Army for five years as a combat arms officer. He held numerous leadership positions as the Fort Sill Training Command Operations Officer, Battalion Logistics and Maintenance Officer, Executive Officer, Logistics and Maintenance Officer and Platoon Leader.

Dion is currently a second year Master of Business Administration candidate at the MIT Sloan School of Management and Master of Engineering in Information Technology candidate at the MIT School of Engineering. He is also a teaching assistant for Professors Stewart Myers and Robert Pindyck and the Zoe Capital Fellow, a research and due diligence position for a Boston Common Angel investor. Additionally, Dion is the recipient of the Thomas Vincent Fellowship Award, Alvin J. Siteman Fellowship Award and United Technologies Fellowship Award. Dion received his Bachelor of Science in Systems Engineering from the United States Military Academy at West Point, where he was the USMA Boxing Team Captain and recipient of the David Marcus Memorial Award for Leadership.

Dion is a dedicated husband and father of two young children, a voracious reader and muscle car enthusiast. He still finds time to box occasionally.
Jedidiah Northridge

Jed was a software developer for Lucent Technologies, Billing and Customer Care, and the Energy and Utilities Group (previously known as Kenan Systems). Jed created modular extensions for the Arbor/BP Billing Framework. These database-centric extensions were written in C, using PL/SQL to interact with Oracle databases. Additionally, he designed and authored associated tables, triggers, and stored procedures. As a Lucent Unix administrator, Jed played an integral role in a 3 person team supporting an international software development environment composed of $10^2$ servers and $10^3$ users.

Jed will receive his Master of Science in Information Technology in July of 2004 from the Massachusetts Institute of Technology. In addition to his coursework, Jed participated in the Technology Enabled Active Learning project (TEAL). This project utilizes Java applications and applets to simulate and teach the fundamentals of electricity and magnetism. These simulations are now being used in an introductory Physics course at MIT. Jed is also a research assistant at the Center for Educational Computing Initiatives at MIT, (CECI), working with MIT professors to create educational materials aimed at teaching core computer science concepts in an undergraduate course at MIT.

Jed received his Bachelor of Science in Mathematics from Trinity College in 1998, where he also received the Phi Gamma Delta Senior Prize and the Phi Gamma Delta Teaching Fellowship.
APPENDIX B

Connection Details of the DLM

Connection Detail of the DLS and Address setting methods
### SPEC SHEET HMC2003 [11]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage (3)</td>
<td></td>
<td>6.0</td>
<td>15</td>
<td>Volts</td>
<td></td>
</tr>
<tr>
<td>Supply Current</td>
<td></td>
<td></td>
<td>20</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Field Range</td>
<td></td>
<td>2</td>
<td>2</td>
<td>gauss</td>
<td></td>
</tr>
<tr>
<td>Output Range</td>
<td></td>
<td>0.5</td>
<td>4.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Resolution</td>
<td></td>
<td></td>
<td>40</td>
<td>ugauss</td>
<td></td>
</tr>
<tr>
<td>Bandwidth</td>
<td></td>
<td>1</td>
<td></td>
<td>KHz</td>
<td></td>
</tr>
<tr>
<td>Field Sensitivity</td>
<td></td>
<td>0.98</td>
<td>1</td>
<td>1.02</td>
<td>V/gauss</td>
</tr>
<tr>
<td>Null Field Output</td>
<td></td>
<td>2.3</td>
<td>2.5</td>
<td>2.7</td>
<td>V</td>
</tr>
<tr>
<td>Linearity Error</td>
<td>+/- 1 gauss Applied Field Sweep</td>
<td>0.5</td>
<td>2</td>
<td>%FS</td>
<td></td>
</tr>
<tr>
<td>Linearity Error</td>
<td>+/- 2 gauss Applied Field Sweep</td>
<td>1</td>
<td>2</td>
<td>%FS</td>
<td></td>
</tr>
<tr>
<td>Hysteresis Error</td>
<td>3 sweeps across +/- 2 gauss</td>
<td>0.05</td>
<td>.1</td>
<td>%FS</td>
<td></td>
</tr>
<tr>
<td>Repeatability Error</td>
<td>3 sweeps across +/- 2 gauss</td>
<td>0.05</td>
<td>.1</td>
<td>%FS</td>
<td></td>
</tr>
<tr>
<td>Offset Strap Resistance</td>
<td></td>
<td></td>
<td>10.5</td>
<td>ohms</td>
<td></td>
</tr>
<tr>
<td>Offset Strap Sensitivity</td>
<td></td>
<td>46.5</td>
<td>47.5</td>
<td>48.5</td>
<td>mA/gauss</td>
</tr>
<tr>
<td>Offset Strap Current</td>
<td></td>
<td></td>
<td>200</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Field Sensitivity Tempco</td>
<td></td>
<td>-600</td>
<td></td>
<td>ppm/C</td>
<td></td>
</tr>
<tr>
<td>Null Field Tempco</td>
<td>Set/Reset not used</td>
<td>+/-400</td>
<td>ppm/C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null Field Tempco</td>
<td>Set/Reset used</td>
<td>+/-100</td>
<td>ppm/C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Temperature</td>
<td></td>
<td>-55</td>
<td>125</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Operating Temperature</td>
<td></td>
<td>-40</td>
<td>85</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Shock</td>
<td></td>
<td>100</td>
<td></td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>Vibration</td>
<td></td>
<td>2.2</td>
<td></td>
<td>g rms</td>
<td></td>
</tr>
<tr>
<td>Power Supply Effect (shifts in field offset or sensitivity)</td>
<td>Power Supply varied from 6 to 15VDC with +/- 1 gauss Applied Field sweep</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>%FS</td>
</tr>
</tbody>
</table>

(1). Unless otherwise stated, test conditions are as follows: power supply= +12VDC, ambient temp=25°C, Set/Reset switching is active.
(2). 1 Gauss (G) = 1 Oersted (in air), 1G =79.58 A/m, 1G=10E-4 Tesla, 1G =10E5 gamma, ppm=parts per million
(3). Transient protection circuitry should be added across V+ and GND if an unregulated supply is used.
SensorDataX Package Details:

Sample Raw Data (report.txt)

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Logger</th>
<th>Channel</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel0</td>
<td>254.9377</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel1</td>
<td>185.2734</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel2</td>
<td>88.9014</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel3</td>
<td>69.9756</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel4</td>
<td>67.0496</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel5</td>
<td>65.9290</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel6</td>
<td>69.6643</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel7</td>
<td>77.8198</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel0</td>
<td>234.8914</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel1</td>
<td>144.3713</td>
</tr>
<tr>
<td>5/4/2003</td>
<td>10:03:45 PM</td>
<td>logger1</td>
<td>channel2</td>
<td>25.8362</td>
</tr>
</tbody>
</table>

Connection to local text log file (Connection 1)

![Connection Properties](image)
Raw data formatting

Select File Format
To copy data, you must first confirm the file format. Confirm that the file properties are correct before proceeding.

- Delimited: Columns are separated by character(s)
- Fixed field: Information is aligned into columns of equal width

File type: ANSI
Skip rows: 0
Row delimiter: (CR)(LF)
First row has column names: yes
Text qualifier: <none>

Preview:

```
"4/4/2003 3:38:20 PM" logger 1 channel 016.9958"
"4/4/2003 3:38:20 PM" logger 1 channel 118.3032"
"4/4/2003 3:38:20 PM" logger 1 channel 213.4473"
```

Data transformation for upload to SensorData table

Transform Data Task Properties
Define the transformations between the source and destination.

Name: DTSTransformation_ALARM
Type: ActiveX Script

Source: Col001, Col002, Col003, Col004, Col005
Destination: TIME, Channel, E, TIME char(18) NULL, ALARM

Select All, Delete All

OK, Cancel, Help
Data transformation sample code using VBScript

```
Function Main()
    If InStr(DTSSource("Col05"), "A") > 0 then
        DTSDestination("ALARM") = Mid(DTSSource("Col05"), InStr(DTSSource("Col05"), "A") + 5)
    Else
        DTSDestination("ALARM") = ""
    End If
    Main = DTSTransformStat_OK
End Function
```

Transformed updated data in IPM Database

![SQL Server Enterprise Manager - [2:Data in Table 'SensorData' in 'IPM']](image)
IPM Prototype Smart Device Web Portal Codes

Login page (default.aspx)

```csharp
private void cmdLogin_Click(object sender, System.EventArgs e)
{
    string login = txtUserID.Text;
    string pw = txtPassword.Text;
    string cString = ConfigurationSettings.AppSettings["UserDB"];
    MobilePage mp = new MobilePage();
    SqlConnection connection = new SqlConnection(cString);
    string sqlString = String.Format("SELECT LoginID FROM Users WHERE LoginID = '{0}' AND Password = '{1}'", login, pw);
    SqlDataAdapter adapter = new SqlDataAdapter(sqlString, connection);
    DataSet result = new DataSet();
    adapter.Fill(result);
    DataTable t = result.Tables[0];
    if (t.Rows.Count == 1)
    {
        mp.RedirectToMobilePage("Parking/Search.aspx");
    }
    else
    {
        errorLabel.Visible = true;
    }
}
```
Availability page (availability.aspx)
Database lookup:

```csharp
private void Page_Load(object sender, System.EventArgs e) {

    // Determine which neighborhood to search
    string nbhood = Request.QueryString["Nb"];

    string cString = ConfigurationSettings.AppSettings["ParkingDB"];
    SqlConnection connection = new SqlConnection(cString);
    string sqlString = "SELECT S.SensorID, SI.Street, SI.Description,
STSD.Available, STSD.Time ";
    sqlString += "FROM Sensor AS S, StreetInfo AS SI,
ShortTermSensorData AS STSD ";
    sqlString += "WHERE S.StreetInfoID = SI.StreetInfoID AND
S.SensorID = STSD.SensorID";

    SqlDataAdapter adapter = new SqlDataAdapter(sqlString, connection);
    DataSet result = new DataSet();
    adapter.Fill(result);
    DataTable sensorData = result.Tables[0];
```
// Select proper Image
String imageName = "..\images\MassAve";
int imageNumber = 0;

foreach(DataRow row in sensorData.Rows) {
    int sensorID = ((int)row.ItemArray[0])-1;
    bool occupied = (((short)row.ItemArray[3]) == 1);
    if (occupied) {
        imageNumber += (int)Math.Pow(2, sensorID);
    }
}

imageName += imageNumber + ".png";
resultImage.ImageUrl = imageName;
Populating textual based information onto datagrid (availability.aspx contd.)

    // Create new DataTable, modTable, that contains exactly the
    // information we want. Need to create new because you can't
    // modify contents
    // of a DataTable that has information in it.
    DataTable modTable = sensorData.Clone();
    modTable.Columns.Remove("SensorID");
    modTable.Columns["Available"].DataType =
    Type.GetType("System.String");
    foreach(DataRow origRow in sensorData.Rows) {
        DataRow newRow = modTable.NewRow();

        short oldAvail = (short) origRow.ItemArray[3];

        newRow["Street"] = origRow["Street"];
        newRow["Description"] = origRow["Description"];
        if ( oldAvail == 0 ) {
            newRow["Available"] = "No";
        } else {
            newRow["Available"] = "Yes";
        }
        newRow["Time"] = origRow["Time"];
        modTable.Rows.Add(newRow);
    }

    // Now that the modified table exists, show it via a DataGrid
    textResults.DataSource = modTable;
    textResults.DataBind();