



GMIT

GALWAY-MAYO INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EÓ

The Development of Mobile Electronic Business Applications

In One Volume

Cristian Radu Radulescu



August 2002

Submitted for the Degree of
Master of Engineering

Submitted to : Galway-Mayo Institute of Technology, Ireland.
Research carried out at : Galway-Mayo Institute of Technology.
Research Supervisor : Mr. Paul O'Dowd.

Dedicated to my wife and my parents

Statement of Confidentiality

The work carried out in this thesis is a contribution to the MOBILE Tools and Technologies for customer care (MOTTO) project.

The material contained in this thesis should not be used, sold, assigned or disclosed to any other person, organisation or corporation without the express permission of:

Galway-Mayo Institute of Technology

Contact: Paul O'Dowd

Tel: 091-742205

Email: paul.odowd@gmit.ie

Nortel Networks Ltd, Galway

Contact: Eamon Walsh

Tel: 091-733287

Email: walshe@nortelnetworks.com

University of Limerick

Contact: Mark Southern

Tel: 061-202374

Email: mark.southern@ul.ie

National University of Ireland, Galway

Contact: Ingrid Hunt

Tel: 091-750414

Email: ingrid.hunt@nuigalway.ie

AMT Ireland, Holland Road, Limerick

Contact: Martin Dooner

Tel: 061-338601

Email: amt@ul.ie

Declaration

I hereby declare that the work presented in this thesis is my own and that it has not been previously used to obtain a degree in this institution or elsewhere.

A handwritten signature in black ink, appearing to read 'Cristian Radu', is written above a horizontal line.

Cristian Radu Radulescu

Prologue

The research conducted for this thesis has been carried out over a two-year period as part of the Mobile Tools and Technologies for customer care (MOTTO) project. The project was funded under the Applied Grant scheme administered by Enterprise Ireland and Nortel Networks. It was a partnership project between Galway-Mayo Institute of Technology, University of Limerick, National University of Ireland Galway and a global Internet and communications company, Nortel Networks. The project aimed to investigate the enabling mobile communications technologies in eBusiness and mobile communications in the area of Business-to-Business (B2B) customer care.

The development process for the applications discussed in this thesis was conducted at the Galway-Mayo Institute of Technology in conjunction with University of Limerick, National University of Ireland, Galway and AMT Ireland. The decision to develop the application in the Electronics Company of AMT Ireland came about as a result of the contact established by Mark Southern from the University of Limerick. Mark was closely involved in the development process by providing liaison with AMT Ireland and assisting in the process of designing the user requirements.

Abstract

Mobile devices offer interesting possibilities for electronic commerce business. This thesis tries to analyse how WAP and wireless technologies could help redefine the notion of electronic business applications. In order to determine the possibilities offered by current mobile technologies and their impact in the business sector, the thesis uses the following two scenarios:

- a prototype application in the business-to-business market with a view for possible applications in the business-to-consumer market
- a mobile application implementation in the specific area of Small-to-Medium sized Enterprise, Business-to-Business Customer Relationship Management.

The development of the prototype application tries to determine the present state of mobile technology and whether the functionality it provides is suited for the development of business applications. The development process provided valuable experience in the areas of WAP and related Internet technologies such as server-side technologies and their integration with WAP. The development of the second application for the purpose of implementation into an Irish SME (AMT Ireland) proved that there is great potential in developing mobile applications for the business area especially for the areas of B2B and B2C Customer Relationship Management. However, present WAP technology does not offer sufficient functionality necessary for designing complex and easy-to-use applications. In addition to that, the technology is also limited from the perspective of data transfer speed, usability and reliability.

Even when taking into account the fact that some consider WAP a failed technology, it must still be acknowledged as the de-facto standard in Europe for delivering content to small handheld devices such as mobile phones. It might be expected that WAP will soon improve and possibly gain advantage over other technologies (such as NTT DoCoMo's iMode) through the design of better specifications and the integration with new emerging technologies such as GPRS.

Acknowledgements

There are many people I would like to thank for helping and encouraging me during the course of the project and the preparation of this thesis.

Paul O'Dowd, my supervisor and project manager, for giving me the opportunity to study at GMIT, for the supervision and for help with the realisation of this thesis.

Nortel Networks Ltd, Galway for providing the funding for this project, and especially to Eamon Walsh for providing support and for being an integral part of our project group. I would also like to thank the staff of Nortel and especially to John Lavelle for help during the course of the project.

Valerie Butler, my good friend and colleague, for making the past two years an enjoyable experience. I wish you all the best and a happy life (Sláinte!).

The MOTTO project team members: Mark Southern from the University of Limerick, Laurentiu Vasiliu and Ingrid Hunt in Prof. Jim Browne's CIMRU group at NUIG. It has been a pleasure working with you.

Prof. Eamonn Murphy and Prof. Eamonn McQuade from the University of Limerick.

Dr Jos Evertsen and Enterprise Ireland for the funding.

The GMIT staff, especially to: Gerard Mac Michael, Patrick Delassus, Ann Murphy and Tom Conlon of the Industrial Liaison Office and Anita Mahony of the International Office.

I am sincerely grateful to Tom Roche for advice and for correcting this thesis.

The staff of AMT Ireland, Limerick, especially to Claire Ryan and Martin Dooner.

Padraig Hernon and Marisa Ocon, thank you both for helping me out with the initial steps.

A special acknowledgement goes to all my postgraduate friends, especially to Vlad Soare and Vlad Teleanca for helping me overcome daily stress with "creative activities".

Finally, I would like to express my personal gratitude to my wife and parents to whom this thesis is dedicated.

Sincere best wishes to all of you in your personal and professional lives.

Table of Contents

Statement of Confidentiality.....	III
Declaration.....	IV
Prologue.....	V
Abstract.....	VI
Acknowledgements.....	VII
List of Figures.....	XI
List of Tables.....	XIII
Chapter 1: Introduction and Objectives.....	1
1.1 Background.....	1
1.2 Thesis Motivation.....	2
1.3 Objectives.....	3
1.4 Approach to Work.....	4
1.5 Thesis layout.....	5
Chapter 2: Review of Business Environment and Technology	9
2.1 Introduction.....	9
2.2 Business Overview.....	9
2.2.1 The E-Commerce Solution.....	9
2.2.2 E-commerce Business Models	11
2.2.3 Mobile Commerce	13
2.2.4 Portals	16
2.2.5 Customer Relationship Management	17
2.2.6 Small to Medium-sized Enterprises	18
2.3 Internet and Mobile Technology	19
2.3.1 Internet as an environment for deploying applications	19
2.3.2 Protocols.....	21
2.3.3 Wireless Technologies	23
2.3.4 WAP and mobile applications	24
2.4 Concepts and Technologies used for designing WAP Applications	30
2.4.1 The concept of 3-tier applications	30

2.4.2	Server-Side Technologies	35
2.4.3	Scripting Languages	38
2.4.4	The Wireless Markup Language (WML)	41
2.4.5	The WEB Server	44
2.4.6	The WAP Toolkit	47
2.5	Summary	53
 Chapter 3: Requirements for the Development of B2B Mobile CRM Applications		55
3.1	Introduction	55
3.2	Requirements for the area of B2B E-commerce	56
3.3	Requirements for SMEs	57
3.4	Requirements from a CRM perspective	57
3.5	Requirements and Characteristics of Web Versus WAP Applications	58
3.6	Summary	60
 Chapter 4: A Prototype Solution for Mobile CRM		62
4.1	Introduction	62
4.2	The Context Data Flow Diagram	63
4.3	Application design and implementation	64
4.3.1	Functional Data Flow Diagram	65
4.3.2	The Screen Flow	71
4.4	Coding details	85
4.4.1	The WML code	86
4.4.2	Working with databases using ADODB	88
4.4.3	The VBscript code	92
4.4.4	The WMLscript code	94
4.5	Summary	97
 Chapter 5: Implementation of a WEB and Wireless Application for an SME ...		98
5.1	Introduction	98
5.2	Company overview	99
5.3	User specifications	100
5.4	Functional requirements	105
5.5	The content type-based redirection	110

5.6	Web application design and implementation	112
5.6.1	Authentication and security issues	112
5.6.2	The user view	118
5.6.3	The administrator view	125
5.7	Mobile implementation	134
5.8	Summary	137
 Chapter 6: Testing and Validation		139
6.1	Introduction	139
6.2	Development and Testing Methodology	139
6.2.1	Development methodology	140
6.2.2	Testing methodologies	142
6.3	Prototype Testing	145
6.4	User Testing and Validation for the AMT Application	150
6.4.1	Application development details	150
6.4.2	Application testing	154
6.5	Outcome	156
6.5.1	The prototype application	156
6.5.2	The AMT application	157
6.6	Summary	161
 Chapter 7: Conclusions and Recommendations		162
7.1	Introduction	162
7.2	Conclusions	162
7.2.1	The mobile prototype application	162
7.2.2	The AMT Implementation	164
7.3	Recommendations for Future Work	167
7.4	Summary	168
 Bibliography		170
 References		171
 Appendix 1: The Prototype Mobile Application Code		188
 Appendix 2: The AMT Implementation Mobile Application Code		221

List of Figures

Chapter 1: Introduction and Objectives

Figure 1.1	Thesis layout.....	5
------------	--------------------	---

Chapter 2: Review of Business Environment and Technology

Figure 2.1	The WAP Architecture.....	29
Figure 2.2	The 2-tier application concept.....	32
Figure 2.3	The 3-tier architecture model.....	33
Figure 2.4	Web server architecture.....	36
Figure 2.5	ADO hierarchy.....	37
Figure 2.6	The structure of the WML deck.....	43
Figure 2.7	The Internet Information Services management console.....	46
Figure 2.8	Setting the MIME types.....	46
Figure 2.9	Nokia 6210, Nokia 7110 and the Blueprint phone simulations.....	50
Figure 2.10	Nokia WAP Toolkit 2.0 network configurations.....	51
Figure 2.11	The text editor of the Blueprint phone simulator.....	53

Chapter 3: Requirements for the Development of B2B Mobile CRM Applications

Figure 3.1	The convergence of SME, B2B and CRM areas.....	55
Figure 3.2	The wireless application hierarchy in the context of e-Business.....	56
Figure 3.3	The integration of an Automated Order processing System.....	58
Figure 3.4	Technical requirements for integrating a wireless application.....	58

Chapter 4: A Prototype solution for Mobile CRM

Figure 4.1	The context data flow diagram.....	63
Figure 4.2	The login and authentication data flow diagram.....	65
Figure 4.3	Prototype application functional data flow diagram.....	69
Figure 4.4	Inputting the user ID number.....	71
Figure 4.5	The welcome screens.....	72
Figure 4.6	Authenticating with the system.....	73
Figure 4.7	Successful authentication.....	73
Figure 4.8	Starting the registration process.....	74
Figure 4.9	The registration process.....	74

Figure 4.10	Editing user details.....	75
Figure 4.11	Using the edit function.....	75
Figure 4.12	Inputting special characters on the mobile device.....	76
Figure 4.13	Editing password fields on the mobile device.....	76
Figure 4.14	Authentication failure.....	77
Figure 4.15	Successful authentication.....	77
Figure 4.16	The main application page.....	78
Figure 4.17	The <i>Settings</i> form.....	79
Figure 4.18	Editing user settings.....	79
Figure 4.19	The <i>News</i> card.....	79
Figure 4.20	The <i>Info</i> card.....	80
Figure 4.21	The <i>Help</i> card.....	80
Figure 4.22	The <i>List all products</i> page.....	81
Figure 4.23	Selecting a product from the products list.....	81
Figure 4.24	Using the search function.....	81
Figure 4.25	Editing the search field.....	82
Figure 4.26	Selecting a product from the search results list.....	82
Figure 4.27	The product details card.....	83
Figure 4.28	The order form.....	83
Figure 4.29	The <i>Submission Confirmation</i> card.....	84
Figure 4.30	The <i>Order Status</i> section.....	84
Figure 4.31	The order details card.....	85

Chapter 5: Implementation of a WEB and Wireless Application for an SME

Figure 5.1	The AMT application database structure.....	107
Figure 5.2	Using the cookie mechanism to maintain session data.....	113
Figure 5.3	The login page.....	115
Figure 5.4	The authentication failure page.....	116
Figure 5.5	The authentication data flow diagram.....	117
Figure 5.6	The main page of the user view.....	118
Figure 5.7	Authentication failure due to unauthorised request.....	119
Figure 5.8	The <i>Submit Request For Quotation</i> form.....	120
Figure 5.9	The Request For Quotation submission confirmation page.....	121
Figure 5.10	The <i>Order Status</i> page.....	121
Figure 5.11	The <i>Order Details</i> page.....	122

Figure 5.12	The user view screen flow diagram.....	123
Figure 5.13	The successful administrative logon page.....	125
Figure 5.14	The administrator main page.....	126
Figure 5.15	The administrator view screen flow diagram.....	127
Figure 5.16	Active order detail.....	129
Figure 5.17	The <i>Add Order</i> page.....	130
Figure 5.18	Selecting a user from the customised list.....	131
Figure 5.19	The AMT WAP application greeting card.....	135
Figure 5.20	The authentication and main page.....	135
Figure 5.21	The <i>Order Status</i> section.....	136
Figure 5.22	The <i>Info</i> card.....	137

Chapter 6: Testing and Validation

Figure 6.1	The Waterfall model.....	140
Figure 6.2	Application usage.....	160

Appendix 1: The Prototype Mobile Application Code

Figure A1	The Orders table in the prototype application database	189
Figure A2	The Products table in the prototype application database	189
Figure A3	The Users table in the prototype application database	190

List of Tables

Chapter 2: Review of Business Environment and Technology

Table 2.1	Advantages and disadvantages of WAP and WML.....	28
Table 2.2	The WAP MIME types.....	46
Table 2.3	Simulators included with the Nokia WAP Toolkit 2.0.....	48

Chapter 3: Requirements for the Development of B2B Mobile CRM Applications

Table 3.1	Comparison between WAP devices and Web devices (browsers).....	59
-----------	--	----

Appendix 1: The Prototype Mobile Application Code

Table A1	The prototype application images	188
----------	--	-----

CHAPTER 1

Introduction and Objectives

- 1.1 Background**
- 1.2 Thesis Motivation**
- 1.3 Objectives**
- 1.4 Approach to Work**
- 1.5 Thesis Layout**

1.1 Background

Computing devices are becoming smaller, faster and more mobile as technology evolves. The first step of putting computer-like devices into truly portable cases was made with the introduction of the Personal Digital Assistant (PDA) around 1995. At about the same time, the Internet and the market penetration of mobile cellular phones started to grow dramatically.

Soon after the Internet was born, the World Wide Web emerged and businesses all over the world suddenly were focusing on *electronic commerce* (e-commerce) offering the possibility of buying real goods by shopping in a virtual world. Telecommunications companies began to offer Internet access through mobile phones for notebooks, and thus it was suddenly possible to access the Internet and its broad range of possibilities at any place, anytime. A new era of communication was born as people are getting more and more used to the idea of being able to do anything, anytime, anywhere. Although small wireless mobile devices promise an interesting range of opportunities in the electronic commerce business area, people are still restricted by the current technology because mobile Internet access is still slow, rather expensive and devices generally offer poor functionality. But despite the fact that current mobile solutions do not offer the flexibility and power to provide effective front-end interfaces for e-commerce applications on the Internet, they can still support relatively simple applications. Even if the technology is still in its early stages, much enthusiasm has been invested into the development and implementation of new business models taking advantage of the wide range of possibilities that the Internet and its offspring, mobile Internet offers. And although some of this enthusiasm diminished with the crash of many Internet *dotcoms* in the recent years, people

are still optimistic about the opportunities offered as analysts predict an explosive development especially with the advent of the new mobile technologies.

In this context, the Mobile Tools and Technologies for Customer Care (MOTTO) project strives to investigate trends in tools and technologies in the area of electronic business (eBusiness) and mobile communications. It attempts to investigate how these tools and technologies can be used to develop innovative solutions in Business-to-Business (B2B) Customer Relationship Management (CRM) by researching and modelling B2B communication. This two-year project is concerned with the application of eBusiness and mobile communications technology to the area of B2B customer care. It addresses technical challenges in designing robust solutions with emerging technologies including Wireless Application Protocol (WAP) and Wireless Markup Language (WML).

The research carried out in this project is performed by postgraduate research students from Galway-Mayo Institute of Technology (GMIT), the Computer Integrated Manufacturing Research Unit (CIMRU) which is part of National University of Ireland, Galway (NUIG), and University of Limerick (UL). The teamwork is coordinated by a project manager from GMIT, while the project funding is provided by both Enterprise Ireland and Nortel Networks.

1.2 Thesis Motivation

The recent convergence of enabling technologies in areas such as the Internet, mobile phones and customer relationship management gives business new data handling and communication capabilities. For instance, the recently developed WAP-enabled mobile phones allow users to browse the Internet via a wireless connection from their handset. These technical capabilities, coupled with innovative new electronic business models offer businesses the possibility of gaining a competitive edge by taking customer care to a new level.

Because of the widespread use of the Internet and the proliferation of mobile phone technology, Europe is in a position to take the lead over the USA in the integration of mobile communications technology and the Internet. Market-leading mobile phone manufacturers including Ericsson, Motorola and Nokia are already investing heavily in third generation mobile technologies, such as WAP and mobile Internet Protocol (IP) for the European market, in an attempt to be at the forefront of the recently termed *New Telecoms World*. A new report from Forrester Research shows that European eCommerce web sites are preparing for a surge in cell phone subscribers, rising from the current one

hundred and seventeen million users, to two hundred nineteen million by 2004 [Nordan et.al. 1999].

In parallel with these developments, the application of e-business is rapidly expanding. Increasingly, multi-national and Small to Medium-sized Enterprise (SME) organisations are creating Virtual Private Networks (VPN), which use the Internet as the transport backbone to establish secure links with business partners, regional offices, and an increasingly mobile work force. The global market for inter site, remote access and inter corporate VPN services is forecast to have an increase of one hundred seventeen percent, from twenty one point three billion dollars to forty six billion dollars between 2002 and 2006 [Infonetics 2002a]. In addition to that, worldwide application-layer VPN gateway annual revenues will grow to eight hundred seventy one million dollars by 2005 [Infonetics 2002b]. This will allow the adoption of eBusiness customer supplier interfaces as tools to communicate critical information through the supply chain. Companies are moving towards a total customer care model, aimed at producing not just a product but a service, linking customers and suppliers and generating closer relationships and ongoing revenue streams. The convergence of e-business and mobile technology is enabling a new paradigm of mobile business (m-business). The aim of this thesis is to determine how this convergence of technological and business developments can benefit industry in this time of increasing customer demands and global competition.

1.3 Objectives

This thesis tries to analyse how mobile applications could help shape the processes of electronic business in the areas of business-to-business applications and small to medium enterprises for both e-purchasing and value-added services.

The objectives of this thesis can be summarised as follows:

- To investigate what kind of mobile applications current mobile communications technology allows at present.
- To investigate how such state-of-the-art mobile communications technology can be used to develop innovative eCommerce and customer care solutions.
- To highlight the advantages of using mobile technologies over the older, more conventional ones already used (such as web applications).
- To develop a conceptual application model by integrating the requirements and specifications for the areas of SME, B2B and CRM.

- To develop a pilot customer care application in the eBusiness area in order to prove the concept. The application must be integrated into a Microsoft Windows platform.
- To determine the requirements and characteristics of such applications in order to integrate them into SMEs
- To use the lessons learned from the proof-of-concept prototype in order to develop and implement a mobile application into an SME.
- To test the prototype and the implementation in order to draw conclusions.
- To record the work done so that it will be available for different types of organisations, including SMEs.

1.4 Approach to Work

In order to achieve these objectives the thesis first looks at current mobile technologies. It analyses the integration of such technologies into applications developed during this project and then extrapolates the conclusions to give an overview of the future of the development for mobile applications.

The following methodology will be employed in the development of this thesis:

- **Literature Review**

The main literature review for this thesis focuses on concepts like eBusiness, eCommerce and mCommerce and also into the area of WAP and mobile technologies. Additional research will also be performed into the areas of CRM and SME.

- **Software Development**

A prototype mobile application will be developed using new innovative technologies and based on modern business practices. The developed wireless Internet-based mobile order entry system will be used to demonstrate the capabilities of the existing technologies and to assess the impact of mobile communication technologies in the CRM area.

- **Software Implementation**

Using the lessons learned in the previous development phase a software implementation will be developed, specifically designed to meet the needs of an

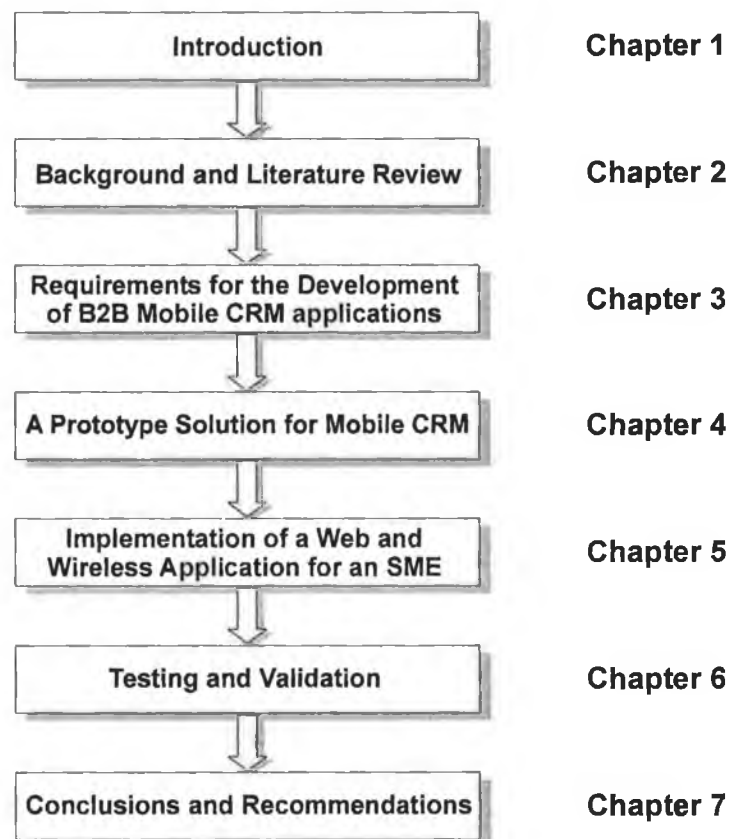
SME. The application will have its functionality and design refined to accommodate the needs of a specific company.

- **Software Testing and Evaluation**

The results of the implementation will be assessed in order to obtain useful information through feedback. Additional information will also be obtained in the development phase. The information obtained will be subsequently used in the dissemination phase of the Motto project.

1.5 Thesis layout

The structure of the thesis is as follows:



Chapter 1 – Introduction

This chapter provides the introduction for the thesis. The first section provides a brief overview of the area covered by this thesis. It then outlines the thesis motivation in the second section, after which it presents the thesis objectives in the third section. The fourth section presents the approach to work. Finally, the last section of the chapter outlines the thesis structure by presenting its layout.

Chapter 2 - Background and Literature Review

This chapter is composed of three separate sections. The first one, called *Business Overview*, analyses Internet-related business notions such as eBusiness, eCommerce and mCommerce. It also provides a brief overview of concepts such as Customer Relationship Management, Business-to-Business and Business-to-Consumer.

The second section, titled *Internet and Mobile Technology*, provides a brief overview of the Internet and highlights some of the concepts behind it. It also provides a description of the wireless technologies, WAP and other related concepts.

The third section is a review of the existing technologies in the area of Web applications. Some of these technologies have been used in the development of the applications described in this thesis. It describes notions like the 3-tier applications concept employed for developing web applications, server-side technologies such as ActiveX Data Objects (ADO^{*}) and Active Server Pages (ASP[†]). It also provides a brief overview of the scripting languages commonly used for designing web applications such as VBscript and JavaScript and also for SQL[‡]. The section concludes by presenting some of the tools used in the development of the applications presented in this thesis, such as the Internet Information Server (IIS) from Microsoft and the Nokia WAP Toolkit.

Chapter 3 – Requirements for the Development of B2B Mobile CRM applications

The third chapter of this thesis summarises the features and requirements of an application specifically designed for the targeted area that this thesis addresses: B2B Ecommerce for SMEs. It is intended as a guideline that underlines the conclusions drawn from the business and technology review performed in chapter two. There are three areas with the requirements of which such an application must comply: the area of B2B eCommerce, the area of applications designed to accommodate the needs of SMEs, and finally the area of Customer Relationship Management and value added services. In addition to that, the application that needs to be developed and implemented must also deal with the specifics of the WAP and wireless commerce applications, which means it has to adapt its design to accommodate the limited capability of offered by WML (Wireless Markup Language, a language similar to HTML but specifically designed for mobile devices such as mobile phones) and other specifics of ubiquitous commerce (mCommerce).

* a technology developed by Microsoft Corporation.

† a technology also developed by Microsoft Corporation.

‡ Structured Query Language, a language used for querying transactional database systems.

As a result, the chapter is structured into four sections, each of which underlines the specific requirements for an area. These areas are:

- B2B eCommerce
- SMEs
- CRM
- WAP versus WEB

The latter outlines the differences between mobile devices as opposed to WEB or WEB-based devices and browsers.

Chapter 4 - A Prototype Solution for Mobile CRM

This chapter provides a detailed description of the prototype wireless application developed during the course of this project. The application was developed by the author of this thesis as part of the MOTTO project, in conjunction with team members from GMIT and CIMRU. Further details including Data Flow Diagrams are available [Gannon 2001].

The first part of this chapter provides a description of the application structure by presenting the Context Data Flow Diagram. The Functional Data Flow Diagram is presented in the second part to describe both the authentication process and the functionality provided by the application itself once the authentication took place and the user has logged-on to the application. The next part will provide a suggestive description by presenting the screen flow for the main functions of the application as seen from a user's perspective. The last section of this chapter outlines the coding details for the application by providing a brief overview of some relevant portions of code that are either typically used in the application for performing common tasks or they perform the most complicated functions within the application. It will cover details about the WML, VBscript and the WMLscript code used, and also the use of ADO in the process of achieving database connectivity.

Chapter 5 - Implementation of a WEB and Wireless Application for an SME

This chapter describes the development process that led to the design of a real life application and its implementation.

The first part of the chapter provides a brief overview of AMT Ireland, the company that benefited from the implementation. The second part outlines the user requirements that helped shape the application and adapt it to the company's needs. The chapter continues by detailing the functional requirements for the application, requirements that defined the application from a technical perspective. It is in this section that the arguments supporting

the decision for splitting the application into two parts are presented. The application is divided into a WEB part and a WAP part. The WEB view is designed for customers using personal computers to connect to the Internet and also for the application administrator to enable him to manage the integrated Data Management System. The WAP view is designed for customers using mobile devices connected to the Internet mainly for the reason of obtaining order status updates. Both applications interact with the same database.

The following section begins to describe the structure of the application by first describing the content type-based automatic redirection system incorporated into the application. The description continues into the next section as the authentication process and its associated issues are explained. The detailed descriptions of the web application's user view and administrator view are also presented in this section. The last section of this chapter describes the WAP view of the application by presenting its screen flow.

Chapter 6 - Testing and Validation

This chapter describes the development and testing procedures employed in order to validate the work done for both applications (the prototype application and the actual implementation).

The first section of the chapter presents a brief overview of the development and testing methodologies commonly employed for software development. The second section describes the development story and some of the testing procedures for the prototype application described in chapter four. The third section details the development process and the testing process for the implemented application described in chapter five. The last section concludes the chapter by presenting the outcome of the development process.

Chapter 7 - Conclusions and Recommendations

This chapter outlines the conclusions resulted from the development process, both in terms of development issues and user feedback.

The first section presents the conclusions obtained from the development of the prototype mobile application described in chapter four. The second section discusses the conclusions resulted from the development and implementation of the second application, which was presented in chapter five. The third section constitutes a overall summary of conclusions for all the development work done. Finally, the last section of the chapter discusses the recommendations for future work.

CHAPTER 2

Review of Business Environment and Technology

2.1 Introduction

2.2 Business Overview

2.3 Internet and Mobile Technology

2.4 Concepts and Technologies Used for Designing WAP Applications

2.5 Summary

2.1 Introduction

This chapter will describe the business issues that this project tries to address and the technologies employed to address these issues. The first part of this chapter provides a brief overview of notions like e-Commerce, Small to Medium-sized Enterprise (SME), the notions of B2B and B2C, and Customer Relationship Management (CRM). The second part of the chapter provides a brief description of the Internet and its underlying technologies and also shows how mobile applications should address the business issues discussed in the first part of the chapter. After describing the WAP architecture and explaining the basics of WML (Wireless Markup Language), the last section will provide a brief description for some of the concepts, programming languages and technologies involved in designing mobile applications.

2.2 Business Overview

This section will provide an overview of the business concepts such as electronic business, electronic commerce, mobile commerce, CRM and SME, concepts approached during the course of this project.

2.2.1 The E-Commerce Solution

E-commerce, ecommerce, or electronic commerce is defined as the conduct of financial transactions by electronic means [B2Bdiversity 2000]. With the growth of commerce on the Internet and the World Wide Web, ecommerce often refers to purchases from online stores on the Web, otherwise known as e-commerce Web sites. They may also be referred

to as "virtual-stores" or Cyber stores. Since the transaction goes through the Internet and the Web, some have suggested the term I-commerce (Internet commerce) or icrosoft [Goldmann 2000]. Few have referred to it as Web Commerce [Scheier 2001]. For online retail selling, the term e-tailing is sometimes used [Perman 2000]. The terms "electronic trading", "electronic purchasing", "electronic marketing" or "electronic procurement" are also synonymous to E-commerce and are sometimes used instead [Clarke 1999b, ECP-NL 2000, Booth 2000, Commonwealth 2000].

E-commerce is the main component of E-business, which is defined as the conduct of business with the assistance of telecommunications and telecommunications-based tools [Clarke 1999a]. E-business is essentially a collection of tools and procedures involving Internet-related technologies used for the purpose of optimising the business relations with consumers or other businesses.

The main advantages that E-business has to offer are in the areas of sales and distribution. It provides an opportunity for companies to reduce costs, thus increasing profits, and gain a competitive and strategic advantage by utilizing the technological developments of the Internet to generate the primary exchange of information.

Some of the segments that Electronic Business is comprised of are:

- EDI (Electronic Data Interchange)
- Electronic Publishing
- Electronic Advertising
- Electronic Retailing
- Electronic Customer Channels
- Data Warehousing and Mining

EDI is defined as "the exchange of documents in standardised electronic form, between organisations, in an automated manner, directly from a computer application in one organisation to an application in another" [Clarke 1998]. It is basically a form of data transaction between businesses (a B2B application) but performed using digital technology, a transaction that ensures the data transfers are performed in an efficient and reliable fashion between the two participating entities.

The essential elements that EDI is based on are:

- The existence of an electronic transmission medium.

This is the physical layer at which communication takes place. It's usually achieved either through direct point-to-point connection (in the early days using dial-up

connections on phone lines), either using private subscription-based networks from specialised providers or (lately) using the Internet [Lankford 2000].

- A structured syntax based on agreed standards.

The information is processed, formatted and is in accordance to a specific set of rules. The standards for formatting the messages insure application independence at each end of the connection between the two participating entities.

- An application layer

This is the specific application that each entity uses in order to handle the data at its end. After processing the data, the application either displays it, stores it in relational databases or resubmits it.

Two important features of the EDI transfers are:

- Relatively fast delivery of the messages
- The delivery of the data is guaranteed when using dedicated networks.

As collaborative business processes evolve, it is likely that in the near future the EDI framework and its underlying standards will migrate towards implementing other standards such as Standard Interchange Language (SIL) and eXtensible Markup Language (XML) which may provide other flexible data standards that enable tighter supply chain integration [VICS 1997, Williams 2000a].

Another component of the E-Business is the Electronic Publishing, defined as “electronic commerce in digital goods and services that are intended for consumption by the human senses” [Clarke 1998].

The term is used to denote the fact that conventional publishers adapt their existing forms of hard-copy publishing to take advantage of the new opportunities offered by the information infrastructure [Clarke 1997].

2.2.2 E-commerce Business Models

As e-commerce has evolved, two major types of store models have emerged: Business-to-Consumer (B2C), and Business-to-Business (B2B). These two aspects of E-commerce are distinguished by big differences concerning fundamental aspects like how transactions take place and what customer needs they address. Another popular e-commerce model is the auction model.

The Business-to-Consumer (B2C) e-commerce model is a publicly accessible Web site offering products for sale. It is analogous to a store on the street, where any member of the

public can walk in and make a purchase. A new, unknown customer is called a guest shopper. The guest shopper has the option of making purchases, after providing some general information about themselves to fulfil the transaction (name, address, credit card, etc.).

Most B2C sites encourage users to register and become members. In doing so, the business can establish a relationship with the customer, provide better service, and build customer loyalty.

B2C sites are designed to provide customers with a pleasant, enjoyable experience by providing personalised content and a well designed, easy to navigate interface. But besides the appearance and functionality of these sites, they have to tackle a very important issue, which is handling of payments.

B2C organisations that employ E-purchasing as the main means of selling goods using the Internet are usually called *dotcoms* (derived from the domain type that is common for commercial sites: *.com*) [McGarvey 2000]. The entire business model of some companies revolves around on-line transactions and Internet. For example Amazon.com relies entirely on electronic means to advertise, sell and receive payments in a process that goes on twenty-four hours per day. These companies are perceived as giving the pulse of the Internet E-commerce business, and analysts tend to consider their evolution as one of the factors influencing the E-business forecasts [Asmussen 2002]. Since the interaction between the company and the buyer takes place at an individual level, systems that allow payments using credit cards are established. These systems allow users to enter credit card details and other sensitive information using a secure method (HTTPS*), an encryption protocol that allows data to be sent over the Internet using a challenge-response encryption algorithm. Besides setting up this system that insures user's privacy, companies often use private channels that connect to major credit card companies to verify the validity of the credit card information provided by users.

E-commerce at a B2C level can also be categorised as organisations that enable users to purchase goods over the Internet (E-purchasing or E-shopping) or organisations that provide value-added service. Some of these companies providing value-added services are, for instance, UPS, the global parcel post carrier, and, separately, the Dutch post office, which have recently launched document delivery services which allow them to deliver messages and documents securely over the Internet [Kueter 1999]. Kueter also emphasizes the fact that consumer trust in using these value-added services does not come from a

* HTTP Secure, a technology based on Secure Sockets Layer (SSL)

knowledge of the technology or procedures involved in secure information transfers over the Internet, but is rather due to the huge base of consumer recognition and trust enjoyed by both of these organisations.

The Business-to-Business (B2B) e-commerce model refers to an e-commerce shop specifically designed for organizations to conduct business over the Internet. The two entities are known to each other and all users are registered.

B2B applications can streamline operations between businesses. For example, a retailer can place orders from a supplier B2B Web site. This type of e-commerce model greatly increases the speed and efficiency of the buying process between businesses.

Despite the collapse of many “dotcoms” in the recent past, there is still an optimistic view about E-commerce’s current success and its future. E-commerce is penetrating more deeply into organisations, moving beyond sales and marketing and into the back-office areas of purchasing, logistics and human resources. Increasing number of organisations report that most of their E-commerce initiatives have been a success, and this trend is expected to continue as companies plan to increase investments in the future. While very few investments will be directed towards the B2C segment, the bulk of those investments will be oriented towards back-office areas, where E-commerce can strengthen businesses by increasing supply-chain efficiency [Accenture 2001].

Relatively new types of E-commerce sites are the auction sites. These sites are dedicated to auctions and act as brokers, facilitating relationships between buyers and sellers. Auctions can be incorporated into B2C or B2B models, and they are a well-known way of moving surplus merchandise.

One of the newly emerged components of E-commerce is Mobile Commerce (M-commerce). The next section will provide an overview of the M-commerce and will try to determine the trends in the development of this relatively new branch of E-commerce.

2.2.3 Mobile Commerce

Mobile E-commerce (M-commerce) is in its early stages. As the next level of E-commerce, it is defined in a similar fashion as the E-commerce itself. M-commerce is the electronic transaction or information interaction conducted using mobile devices and mobile networks that lead to transfer of value in exchange for information, services or goods [Mobilocity 2001].

As a new form of E-commerce, mobile commerce relies on the latest wireless technologies in order to provide customers with ubiquitous connectivity to perform

transactions. It enables commercial transactions of goods and services using wireless mobile devices, such as mobile telephones, pagers, personal digital assistants (PDA), and handheld computers.

The high penetration rate of mobiles, combined with new technologies that provides for fast transfer of data on mobile networks, standard protocols that deliver Internet-like services on smaller screens, and the personal nature of mobile telephones could drive the m-commerce revolution.

The evolution of m-commerce as an accepted business paradigm however has only become apparent with the emergence of latest technologies such as the Wireless Application Protocol (WAP) and iMode. The latter is an Interactive Internet-based wireless service developed by NTT DoCoMo, Japan's largest cell phone service company with 57% market share of 48 million cell phone users [Fujii 2002].

Since mobile hand-held terminals together with WAP are a new access technology to the Internet-based E-commerce world, E-commerce based on WAP services does not change the structures at the Internet side. This allows business processes to remain the same as when the access to them is performed in the conventional fashion by using PCs. Mobile devices are much smaller in size than PC's and their bandwidth is still scarce, and therefore there is a need to adapt the E-commerce services into the WAP world, and mobile world in general [Veijalainen 2000].

While E-commerce generally implies the use of personal computers (PCs) or laptops connected to the Internet usually through a hard-wire connection, M-commerce assumes the use of mobile phones, PDAs (Personal Digital Assistants) or other such devices connected to the Internet through a wireless network connection. Thus, the connection speeds are greatly reduced and the display and browsing capabilities are also more basic.

E-commerce usually implies only the use of HTML* as the markup language. Mobile devices, however, use not only HTML but also WML†, HDML‡ or (as for i-Mode) other proprietary languages.

One of the aspects that highlight the importance of M-commerce especially for Western Europe is the fact that in this region mobile penetration is 64%, which more than doubles fixed-line Internet penetration rate [Bell 2001]. In addition to that, it is estimated that mobile penetration in the same region will approach 100% by 2006. This leads to the

* HyperText Markup Language

† Wireless Markup Language

‡ Handheld Device Markup Language

conclusion that in contrast to the United States (US), in Western Europe mobile devices will constitute the preferred method for accessing online services.

Studies show that by the end of 2005, more than 23 million Europeans will use their mobile phones to buy travel products and services, believed to be the most representative products and services on the mobile market, since this type of channel is considered to be easier and more cost-effective than traditional channels [IDC].

According to the same study, 49 million users will also buy public transport tickets with their mobile phones by 2005, fact that is believed will have a positive effect on other mobile commerce segments such as bill payment, tickets for entertainment, parking meters and vending machines [Barnard 2001b].

It has become a habit to contrast the enthusiasm and rate at which the US has developed the Internet with Europe's relative conservatism. The leading role that the United States has taken thus far in the Internet revolution is not under dispute, but despite the fact that key e-commerce trends and business models usually derive from the US, Europe has adopted a clear lead in terms of usage and application development in the specific area of mobile communications [RTD 2000].

For the US market, analysts predict a 73% annual growth rate in the 2000-2005 interval for the wireless subscribers using wireless Internet services [Vyas et.al. 2001].

Although the predictions are optimistic, a large number of SMEs in Europe are slow to implement E-business due to its ever-increasing complexity. They have proved to be reluctant in engaging E-business beyond elementary email and simple web sites, as fear of investing time and funds into this rapidly changing environment is an important inhibitor [Mazzi 2001].

Analysts predict that by the end of 2003 there will be in excess of 270 million mobile subscribers in Europe, opening tremendous opportunities for mobile financial services expected to have a 76% yearly growth rate [Barnard et.al. 2001].

Same article anticipates that mobile devices will become wireless wallets, not just for purchasing on line, but also for paying for goods in shops or through vending machines.

According to analysts, in the long term the M-commerce part of revenues for many mobile operators will begin to rival mobile data carriage revenue [Sheedy 2001].

This might be a valid point since the normal use of the Internet from a desktop computer is somewhat reserved to the computer-literate users, which are familiar with computer operating systems, web browsers and even a bit of Internet technology. On the

other hand, the use of mobile devices appears to be simpler, the amount of knowledge required to operate such devices is lower, thus bringing this technology within the grasp of a wider number of users.

According to the same article, one of the key inhibitor factors for the emerging M-commerce will be security. Privacy issues combined with the need to provide secure and reliable payment facilities make the security one of the key factors in promoting the usage of such technologies for mobile commerce purposes and gaining user's acceptance.

Because of its inherent nature, mobile Internet connectivity relies on data transfers across two different networks, the wireless network and the Internet. While the Internet itself is known to be vulnerable, the addition of the wireless network security issues might prove to be disastrous unless complex applications employing powerful encryption algorithms are implemented.

2.2.4 Portals

Since the introduction of WAP and mobile Internet access, a new notion is emerging: mobile or wireless portals.

Portals are specifically designed web sites that offer a number of services and functionalities to the site's visitors or subscribers, from basic content and web search to communication and commerce applications.

Mobile portals are portals tailored to provide content specifically designed for mobile devices. Compared to their fixed-line counterparts, mobile portals are shaped by two key factors:

- The physical limitations of the terminal devices and
- The low data transfer capacity of the wireless networks

Despite these limitations, mobile portals still have the ability to offer personalisation as the means to make the service more attractive to customers.

Some of the services offered by such portals, besides mobile commerce, are: news services, financial services, conferencing, messaging, entertainment services and location-based services [Bell 2001]. These types of services will enable consumers to use mobile phones as a familiar, convenient and inexpensive channel for access to goods and services. Banks, retailers and travel companies will make their brand and their offerings available to customers at the touch of a button – anywhere. It is widely anticipated that m-commerce will personalize the global digital economy. It is about consumer empowerment and

providing new ways to access information, as the true value of m-commerce lies in its ability to bring specially tailored services directly to the customer using portable devices.

2.2.5 Customer Relationship Management

Customer Relationship Management (CRM) is an information industry term for methodologies, software, and usually Internet capabilities that help an enterprise manage customer relationships in an organized way [Williams 2000b].

As competition in industry began to increase over a decade ago, it was no longer enough to allow users access to information via an automated system twenty-four hours a day and three hundred sixty-five days a year. Although services would continue to grow in sophistication, it was also crucial to make sure that the users' specific needs were addressed. This meant concentrating on how the user felt about the service. This concept has been termed Customer Relationship Management (CRM) or, as access to call centers, automated services and information services have proliferated on the Internet, electronic-CRM (e-CRM).

e-CRM implies customer management for e-Businesses, a concept that must confront the complexity of managing customers and business partners in a variety of media including: online and offline media, personal contact, and more automated and electronic forms of communication.

Customer Relationship Management is a comprehensive approach to doing business. It provides seamless integration of every area of business that touches the customer - namely marketing, sales, customer service and field support - through the integration of people, process and technology, taking advantage of the revolutionary impact of the Internet [DCI 1999]. One of the main issues that CRM tackles is defining new strategies and methodologies for preserving the customer base, as opposed to acquiring new ones. It can cost up to forty times more to attract a new customer than to keep a current one. So keeping just five percent more customers translates into savings of twenty-five to fifty-five percent in profitability [Kolsky 2001].

CRM can be split into three elements [Robinson 2000]:

- Sales force automation, providing facilities for call center telephone sales, E-commerce, field sales and other forms of product distribution
- Customer service / call center management providing support for call centers, web-based services and field support services

- Marketing automation providing marketing campaign management, content management, data analysis and business support tools

Many businesses are exposed to customers without geographical boundaries and customers may not have constant access to devices that are physically connected to a network. Wireless devices have saturated the market and have already gained popularity among the general public. Modern customers require, then, an increased degree of freedom when contacting call centers, possibly via wireless communications. It is believed that by the end of the year 2002 at least forty percent of Web access will be done from wireless devices [Barnard 2001a, Fooladi 2001, Vyas 2001]. For this reason, companies who want to stay competitive in the market need to adapt to the new business requirements. It is not expected that traditional methods of accessing information will be replaced, but rather that new ones will be added to increase flexibility.

As the new mobile technologies emerge, phones, personal digital assistants, and other devices will be connected by a variety of wireless technologies and will become mandatory for corporate communications with customers and employees. Mobile applications provide opportunities in the ability to interact with customers in new ways, new places, and at new times. Adding mobility and immediacy to an application offers the potential for new products and services, business process improvements, cost savings, and improved response times [Tornbohm 2002].

2.2.6 Small to Medium-sized Enterprises

According to the Commission of the European Communities, in order to fall into the category of Small and Medium-sized Enterprise, a company must meet the following conditions:

- It has to employ less than two hundred and fifty employees
- It must have either an annual turnover not exceeding forty million euro, or an annual balance sheet total not exceeding twenty seven million euro
- It must not belong to a group of linked enterprises unless that group meets the conditions specified above [EC 1996] [EC 2001].

There are nineteen million small and medium-sized enterprises in the European Union representing ninety-nine point eight percent of all EU enterprises and employing more than seventy-four million people.

As recognition of their important social and economic function, as well as of the particular difficulties they might face in their development, SMEs benefit from special

assistance in several areas such as national state-aid schemes, Community support programmes and financial aid mainly in the form of facilitating loans. [EC 2002]

SMEs are considered significant players in business-to-business (B2B) electronic commerce, which constitutes more than eighty percent of all e-commerce activity. As such, if they can demonstrate their capabilities to use electronic commerce they will have a competitive advantage in the B2B marketplace.

2.3 Internet and Mobile Technology

The following section provides a brief description of the Internet and some of its related technologies. It will also provide a description of WAP and its related technologies.

2.3.1 Internet as an environment for deploying applications

The Internet is defined as a global communications network consisting of thousands of networks typically interconnected by fibre optic cabling. It is a big network of computers scattered all round the world, linked to each other, a network of networks [FNC 1995]. It had two parent networks whose joining began the ongoing evolution;

- U. S. Military (tactical communication in the event of telephone downtime during wartime)
- Academics (shared information between researchers) [Leiner 2000].

From its beginnings in 1969 the Internet was designed as a network over which many different applications could be run at the same time using a variety of different telecommunications protocols [Gromov 1995]. In contrast, the phone system was designed for a single application, that is, communication by voice. Today the phone system can be used for other purposes, such as sending faxes, or indeed connecting to the Internet, but the system wasn't designed specifically for those purposes, which is why dial-up connections to the Internet are so slow.

The most common applications that are run on the Internet are:

- Electronic mail, which allows sending and receiving electronic messages
- The World Wide Web, which facilitates easy "point and click" navigation of text and graphics from millions of computers worldwide.

The "World Wide Web" (WWW) is the most popular application of the Internet. The Web was born in 1991, developed at the CERN Physics Laboratory in Switzerland by Tim

Berners-Lee [Berners-Lee 1990b]. However, it took the launch in 1993 of Mosaic, the first graphical Web browser for Windows, to really boost the development of the Internet.

Web pages are documents displayed by the web browser, documents formatted in HTML code. HTML (HyperText Markup Language) is the standardized language of computer code, embedded in "source" documents behind all Web documents, containing the textual content, images, links to other documents (and possibly other applications such as sound or motion), and formatting instructions for display on the screen [Berners-Lee 1990a]. A user viewing a Web page is looking at the product of this code working behind the scenes in conjunction with a browser. Browsers are programmed to interpret HTML for display.

HTML often embeds within it other programming languages and applications such as SGML, XML, JavaScript and CGI-script. It is possible to deliver or access and execute virtually any program via the WWW.

In order to obtain the Web content, a browser has to make requests to Web servers. A Web server is a computer with special software that stores Web pages and allows them to be accessed by other computers. At the beginning of June 1993 there were one hundred and thirty Web servers in the world [December 1994]. By July 2000 there were seventeen million Web servers, serving up an estimated two point one billion unique Web pages, which reached over thirty-seven million active web servers by April 2002 [Netcraft 2002]. In fact, experts are divided about how many pages there are now, but it is a fairly big number and growing rapidly [Odlyzko-2001].

The growth of the Internet is reflected through its traffic and the traffic growth rate. Similar to Moore's law for semiconductors which states that the number of components (transistors) embedded into integrated circuits (the component density of the chips) is doubling every year, it has been stated that (despite some minor digressions) the Internet growth rate estimated through transmission volumes is remarkably close to doubling every year [Coffman et.al. 2001b]. Moore's law, which seems also true for the storage capacity of computers, although is not a natural law, reflects a complicated process, the interaction of technology and the speed at which new technologies are absorbed [Odlyzko 2000, Coffman et.al. 2001a].

Although the Internet was born in the United States, statistics are showing its global expansion with Western Europe forecasted to reach about the same Internet population percentage as the U.S. by the end of 2003 [Thompson 1999].

The functioning of the Internet is governed by standards, which are designed by a number of organisations like the Internet Society, the Internet Engineering Task Force (IETF), the European Telecommunications Standards Institute (ETSI) and World Wide Web Consortium (W3C) [ISOC, IETF, ETSI, W3C].

Ever since it was founded in 1992, the Internet Society (ISOC) has grown steadily, so that today government agencies, other non-profit organisations, private corporations and private individuals are included as members. Its objectives are:

- to maintain and develop Internet standards;
- to expand and develop Internet architecture;
- to develop effective administrative processes for the operation of the Internet;
- to promote the development of and accessibility to the Internet.

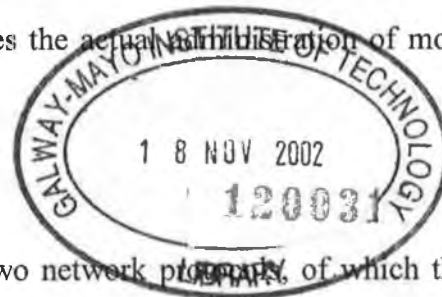
Some Internet Standards require administrative implementation in order to allow the Internet to be operational. These include, for example, Internet Protocol address or domain names at upper levels. The overall responsibility for this work is vested in the Internet Assigned Numbers Authority (IANA), which delegates the actual administration of most functions to other bodies.

2.3.2 Protocols

The functioning of the Internet revolves around two network protocols, of which the two best known are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). These are the world's most popular open-system (non-proprietary) protocol suite because they can be used to communicate across any set of interconnected networks.

Internet protocols were first developed in the mid-1970s, by the Defence Advanced Research Projects Agency. DARPA initiated the research aiming at establishing a packet-switched network that would facilitate communication between dissimilar computer systems mainly located in research institutions. The result of this development effort was the Internet protocol suite, completed in the late 1970s [Gromov 1995].

TCP/IP is a set of protocols developed to allow communication between computers and resource sharing across a network. They constitute the main protocol layers that enable the functioning of the Internet. Each host on a TCP/IP network is assigned a unique 32-bit logical address grouped eight bits at a time, separated by dots, and represented in decimal format (known as dotted decimal notation). At a logic level this 32-bit logical address is divided into two main parts: the network number and the host number. The network number identifies a network and must be assigned by the Internet Network Information



Center (InterNIC) if the network is to be part of the Internet. The host number identifies a host on a network and is assigned by the local network administrator.

The Internet Protocol (IP) is the primary network-layer protocol in the Internet protocol suite and contains addressing information and some control information that enables packets to be routed [RFC791]. The TCP provides reliable transmission of data in an IP environment. It corresponds to the transport layer (Layer 4) of the OSI reference model. Among the services TCP provides are stream data transfer, reliability, efficient flow control, full-duplex operation, and multiplexing. With stream data transfer, TCP delivers an unstructured stream of bytes identified by sequence numbers. This service benefits applications because they do not have to chop data into blocks before handing it off to TCP. Instead, TCP groups bytes into segments and passes them to IP for delivery. TCP offers reliability by providing connection-oriented, end-to-end reliable packet delivery through an internetwork. It does this by sequencing bytes with a forwarding acknowledgment number that indicates to the destination the next byte the source expects to receive. Bytes not acknowledged within a specified time period are retransmitted. The reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets. A time-out mechanism allows devices to detect lost packets and request retransmission [RFC793]. TCP offers efficient flow control, which means that, when sending acknowledgments back to the source, the receiving TCP process indicates the highest sequence number it can receive without overflowing its internal buffers. Full-duplex operation means that TCP processes can both send and receive at the same time. Finally, TCP's multiplexing means that numerous simultaneous upper-layer conversations can be multiplexed over a single connection.

Besides TCP, another fairly common protocol is User Datagram Protocol (UDP). UDP is a connectionless transport-layer protocol (Layer 4) that belongs to the Internet protocol family. It is basically an interface between IP and upper-layer processes. UDP protocol ports distinguish multiple applications running on a single device from one another.

Unlike the TCP, UDP adds no reliability, flow-control, or error-recovery functions to IP. Because of UDP's simplicity, UDP headers contain fewer bytes and consume less network overhead than TCP. UDP is useful in situations where the reliability mechanisms of TCP are not necessary, such as in cases where a higher-layer protocol might provide error and flow control.

UDP is the transport protocol for several well-known application-layer protocols, including Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and Trivial File Transfer Protocol (TFTP).

DNS (Domain Name System) is a distributed database that maps domain names to IP addresses making it simple to access resources on Internet-based servers only by knowing their names. Internet-connected computers use DNS to resolve URLs (Universal Resource Locators). In this way, a user doesn't need to know the IP address of a Web server – just its name.

Domains are a hierarchical scheme for indicating logical and sometimes geographical venue of a web page from the network. DNS names take the form <domain>.<domain type>, e.g. wapforum.org. While the list of available DNS types is currently being redesigned by ICANN (Internet Corporation for Assigned Names and Numbers), some popular existing types include .edu (educational establishments), .mil (military organisations), .org (non-profit and research organisations), .net (network related), .gov (government agency), and .com (commercial organisations). There are also country-specific domain types, like .ie (Ireland), .jp (Japan) and .de (Germany), etc.

The Internet architecture design is flexible in order to allow interconnection between different types of computers communicating on different types of network architectures. This flexibility also extends the opportunity of developing different types of applications that implement the existing protocols in order to integrate with the existing network structure.

One such application is the Wireless Access Protocol (WAP), which integrates with the Internet architecture by implementing the HTTP protocol as well as the other lower-layer protocols in order to take advantage of the existing Internet structure and the benefits it offers. The next section will detail WAP and related wireless technologies, which take advantage of the existing Internet technology.

2.3.3 Wireless Technologies

Mobile devices, initially able to provide data transfer links only for voice, have evolved over time. Short Messaging System (SMS), a small point-to-point application on top of the existing wireless network protocols, able to transfer a limited number of alphanumeric characters from one device to another in the form of a small message, generated big revenues for the carriers.

The latest application that uses the existing wireless network protocols is the Wireless Access Protocol (WAP) [WAPFORUM]. WAP integrates with the existing wireless network protocols in order to provide data connectivity at an application level across the network. This connectivity is used to establish a connection to a specialised server that acts as a switch between the wireless network and the Internet, and from here on the connection uses the existing HTTP protocol allowing connectivity to web servers. This system provides Internet connectivity to wireless devices using the old GSM networks, typically allowing speeds of up to 9600 bits per second and (lately) up to 14.4 kbps on up to 4 channels (which amounts to 57.6 kbps overall).

In addition to that General Packet Radio Service (GPRS), one of the latest services available on the GSM platform, offers 'always-on' and higher data capacity, as well as Internet-based content and packet-based data services. 'Always-on' means that the device is connected to the network at all times as opposed to the previous way of achieving connectivity through a system similar to the dial-up connection.

Bluetooth, another wireless technology, enables connections between bluetooth-enabled devices like mobile computers (laptops and notebooks), mobile phones, portable handheld devices over short distances. Its implementation includes both link layer and application layer definitions for product developers, which supports data, voice and content-centric applications. It's functioning relies on the use of a spread spectrum, frequency hopping, full-duplex signal at up to 1600 hops/sec. The signal hops among 79 frequencies at 1 MHz intervals to give a high degree of interference immunity. Up to seven simultaneous connections can established and maintained.

2.3.4 WAP and mobile applications

The Wireless Application Protocol (WAP) standard is based on Internet standards (HTML, XML and TCP/IP). It consists of a WML language specification, a WMLScript specification, and a Wireless Telephony Application Interface (WTAI) specification.

WAP is positioned at the convergence of two rapidly evolving network technologies, wireless data and the Internet. Both the wireless data market and the Internet are growing very quickly and are continuously reaching new customers. The explosive growth of the Internet has fuelled the creation of new and exciting information services.

Most of the technology developed for the Internet has been designed for desktop and larger computers and medium to high bandwidth, generally reliable data networks. Mass-

market, hand-held wireless devices present a more constrained computing environment compared to desktop computers.

Because of fundamental limitations of power and form-factor, mass-market handheld devices tend to have [Predescu 2001]:

- Less powerful CPUs,
- Less memory (ROM and RAM),
- Restricted power consumption,
- Smaller displays
- Different input devices (e.g. phone keypads).

Similarly, wireless data networks present a more constrained communication environment compared to wired networks. Because of fundamental limitations of power, available spectrum, and mobility, wireless data networks tend to have:

- Less bandwidth,
- More latency,
- Less connection stability
- Less predictable availability

Mobile networks are growing in complexity and the cost of providing more value-added services is increasing. In order to meet the requirements of mobile network operators, solutions must be:

- Interoperable – terminals from different manufacturers communicate with services in the mobile network;
- Scalable – mobile network operators should be able to scale services to customer needs;
- Efficient – provide quality of service suited to the behaviour and characteristics of the mobile network;
- Reliable – provide a consistent and predictable platform for deploying services; and
- Secure – enable services to be extended over potentially unprotected mobile networks while still preserving the integrity of user data; protects the devices and services from security problems such as denial of service.

Many of the current mobile networks include advanced services that can be offered to end-users. Mobile network operators strive to provide advanced services in a useable and attractive way in order to promote increased usage of the mobile network services and to decrease the turnover rate of subscribers. Standard features, like call control, can be enhanced by using WAP technology to provide customised user interfaces. For example,

services such as call forwarding may provide a user interface that prompts the user to make a choice between accepting a call, forwarding to another person, forwarding it to voice mail, etc.

The WAP specifications address mobile network characteristics and operator needs by adapting existing network technology to the special requirements of mass-market, hand-held wireless data devices and by introducing new technology where appropriate.

The Wireless Application Protocol narrows the gap between the mobile world and the Internet world (TCP/IP networks) by optimising standards for the unique constraints of the wireless environment. It also offers complicated enough security mechanisms and application platform for mobile electronic commerce applications to be developed.

Mobile hand-held devices as facilitated by Mobile IP technologies connect directly within the IP network or by access technologies, like 2G (2nd generation) GSM networks or 3G (3rd generation) networks. One of the main differences between these different network generations is the bandwidth. While the current 2G networks the bandwidth ranges from 9.6 kbps to 14.4 kbps per channel, the so-called 2.5G HSCSD (High Speed Circuit Switched Data) will offer in practice 57.6 kbps and GPRS (General Packet Radio Service) around 112 kbps transmission rates. However, in reality, the GPRS system offers much less data transmission rates than initially advertised, its main benefit being in fact always-on connectivity. The EDGE (Enhanced Data rates for Global Evolution) promises 384 kbps maximum, but in practice the transfer rates are below that. 3G networks should provide 2 Mbps in good circumstances, but in worse circumstances (e.g., weak signal) the bandwidth will only reach a few hundred kbps [Veijalainen 2000].

The integration of the Internet IP protocol technology into the new generation 2.5G GPRS mobile core networks is primarily due to the huge popularity of the IP protocol on the Internet. This will provide for the easy integration into the existing Internet environment and Internet-based applications. Indeed, with the recent advances in optics and routing technology and the impact that these have had on price/performance, IP is gradually becoming a dominating transport technology, especially combined with other key technologies such as IP-based virtual private networks (VPNs). The ease of integration will also reflect on the costs of this implementation, which will provide for lower prices for the carriers that in turn will reflect on better value for users [Hameleers 2002].

The WAP programming model is similar to the WWW programming model. This provides several benefits to the application developer community, including a familiar programming model, a proven architecture, and the ability to leverage existing tools (e.g.

Web servers, XML tools, etc.). Optimisations and extensions have been made in order to match the characteristics of the wireless environment. Wherever possible, existing standards have been adopted or have been used as the starting point for the WAP technology [WAPForum].

The WAP Forum, an industry organization founded in 1997 by Ericsson, Motorola, Nokia, and Unwired Planet and dedicated to developing open standards for wireless communication, has provided a formal specification for WML. The WAP Forum designs and publishes the specifications, including the Document Type Definition (DTD) for WML [WAPForum]. Forum members now represent over 90% of the global handset market, as well as leading infrastructure providers, software developers and other organizations. Big corporations like Alcatel, AT&T, Nokia, Intel, IBM, Compaq, Hewlett Packard and Microsoft are full members of this industry association, contributing to the development of the new wireless standards.

WAP content and applications are specified in a set of well-known content formats based on the familiar WWW content formats. Content is transported using a set of standard communication protocols based on the WWW communication protocols. A *micro browser* in the wireless terminal co-ordinates the user interface and is analogous to a standard web browser.

WAP defines a set of standard components that enable communication between mobile terminals and network servers, including [Hubbard 1999, Alfano 1999]:

- Standard naming model – WWW-standard URLs are used to identify WAP content on origin servers. WWW-standard URIs are used to identify local resources in a device, e.g. call control functions.
- Content typing – All WAP content is given a specific type consistent with WWW typing. This allows WAP user agents to correctly process the content based on its type.
- Standard content formats – WAP content formats are based on WWW technology and include display markup, calendar information, electronic business card objects, images and scripting language.
- Standard communication protocols – WAP communication protocols enable the communication of browser requests from the mobile terminal to the network web server.

The WAP content types and protocols have been optimised for mass market, hand-held wireless devices. WAP utilizes proxy technology to connect between the wireless domain and the WWW. The WAP proxy typically is comprised of the following functionality:

- Protocol Gateway – The protocol gateway translates requests from the WAP protocol stack (WSP, WTP, WTLS, and WDP) to the WWW protocol stack (HTTP and TCP/IP).
- Content Encoders and Decoders – The content encoders translate WAP content into compact encoded formats to reduce the size of data over the network.

This infrastructure ensures that mobile terminal users can browse a wide variety of WAP content and applications, and that the application author is able to build content services and applications that run on a large base of mobile terminals. The WAP proxy allows content and applications to be hosted on standard WWW servers and to be developed using proven WWW technologies such as CGI scripting. While the nominal use of WAP will include a web server, WAP proxy and WAP client, the WAP architecture can quite easily support other configurations.

Although WAP represents a big advancement in the way it is implemented, there are also a number of disadvantages in the way its implementation was designed [Bigelow 2001]. These advantages and drawbacks are shown in table 2.1.

Advantages	Drawbacks
<p>WML is an XML-compliant language. This ensures that a document is well-formed, and portable.</p> <p>It provides a scripting language that has a similar structure to JavaScript, VBScript, and ECMAScript.</p> <p>It has strong adoption among carriers and handset manufacturers, and holds second place in terms of handset units deployed.</p>	<p>It has high carrier and developer costs. These costs are partly due to the need for gateway implementation.</p> <p>Application and Content developers must acquire new or upgrade existing tools in order to author WML pages.</p> <p>There are security concerns about data being passed across two networks that have proved relatively unreliable in the past from a security point of view.</p>

Table 2.1 – Advantages and disadvantages of WAP and WML

Mobile client devices have an interface that provides for user interaction with content and applications resident on a Web ("Origin" Server) via a WAP Gateway. On the mobile

device, user agents handle the interpreting of this content on behalf of the user. The WML browser is one such user agent; it is very similar to a web browser except it handles content formatted in Wireless Markup Language (WML). User agents also typically have a built-in WMLScript Interpreter for running applications. These applications are written in a script language called WMLScript. In addition to the programming language itself, the WMLScript Interpreter also implements a set of libraries that allow the application to access certain services of the user agent. WML and WMLScript are designed for use in wireless, narrowband networks, and they are both binary encoded for optimum transmission efficiency. In most cases, the actual application or other content is located on a Web server, and the content is typically created in WML and WMLScript. In order to create a connection between the wireless networks (to which the mobile device is connected) and the Internet, a special type computer is used. This computer is called a gateway. Special software running on this computer integrates both types of protocols, the one needed to connect to the wireless network and the other one needed for connecting to Internet resources (HTTP).

The figure below presents the WAP architecture and the normal sequence of content requests and responses, which occur between a client mobile device, a WAP Gateway, and a Web (or Content) Server [Poe 2000].

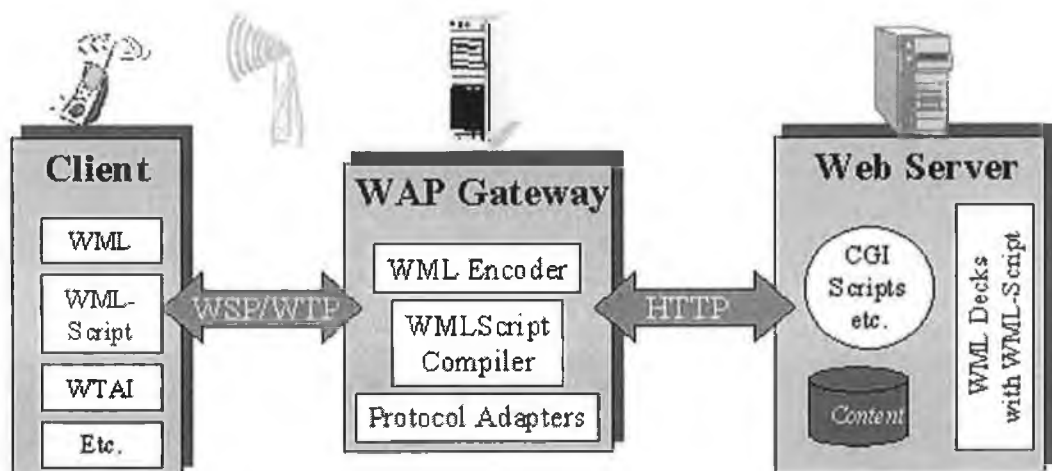


Figure 2.1 – The WAP architecture

The following list presents the succession of events as they occur in a normal transaction between a WAP client and a content server [Hillebrand 2001]:

- The user presses a phone key that has an URL request assigned to it.
- The user agent sends the URL request to a WAP gateway using the WAP protocol.

- The WAP gateway creates a conventional HTTP request for the specified URL and sends it to the web server.
- The HTTP request is processed by the web server. The URL may refer to a static file or to an ASP, CGI or other script application. In the first case, the web server fetches the file. If the URL specifies a script application, the web server runs the application.
- The web server returns the WML deck, either resident (if it is a static file) or the WML output resulted from processing the script application, along with any HTTP headers.
- The WAP gateway verifies the WML content and encodes it, as well as any HTTP headers, to binary form. The gateway then creates a WAP response containing the encoded WML and sends it to the user agent.
- The user agent receives the WAP response. It parses the WML response and (by default) displays the first card of the WML deck to the user.

2.4 Concepts and Technologies used for designing WAP Applications

The following section provides a brief description of the concepts and technologies involved in designing WEB and specifically WAP applications. The first subsection details the concepts of 2-tier and 3-tier applications that can be employed for the development of WEB-based applications. The following subsections detail the structure and the usage of other WEB-related technologies like ASP or the scripting programming languages (VBscript, JavaScript), markup languages (WML) and data access technologies like ADO (ActiveX Data Objects) and SQL (Structured Query Language). The last subsection details the tools used for design and testing of applications, like the Nokia WAP Toolkit.

2.4.1 The concept of 3-tier applications

Most of the applications in use today that use networks to transfer data are designed by employing the client-server concept. This concept states that the transfer of information between the user's computer (the client) and the computer providing the information (the server) is achieved through structured requests from the client towards the server. The client application requests services and data from the server, and the server application responds to client requests. Thus, every computer on the network is viewed as either a client or a server.

Historically, the most used types of servers were file servers (servers providing files to clients) and database servers (providing access to structured databases and sometimes predefined queries for those). With the advent of new technologies, a new type of server

becomes more and more present on the networks: the application server. Essentially, these servers provide some sort of functionality to clients, functionality that involves processing of data. It is not important whether that data comes from a file server, database server, the same server acting as a file/database server or even from the user (the client provides the data to be processed). What is important is that the server provides some processing functionality that the client takes advantage of.

Initially, when they first appeared on the Internet, web servers fell into the file servers category. Offering a very rudimentary set of services, the first web servers introduced with the advent of the Internet would only provide files requested by clients, using the HTTP protocol. Thus, simple tasks like navigating from one server to another, accessing resources and displaying content (like pictures) were achieved by either manually typing in the links or referencing files from within the HTML code contained in other files stored also on web servers. This mechanism relies entirely on the client making directed requests from web browsers.

With the explosive development of the Internet in the early years, it became more and more apparent that complex mechanisms needed to be developed in order to allow development of more interactive applications enabling features like personalisation, content processing and database access.

New mechanisms like Java Server Pages (JSP), Active Server Pages (ASP) and Hypertext Preprocessor (PHP*) were developed, that make use of scripting languages in order to allow the creation of dynamic Web pages for e-commerce and other Web applications. In addition to these scripting based mechanisms, another type of application processing mechanism is the Common Gateway Interface (CGI). The latter uses the same ways of retrieving data from and submitting to clients as the first ones, but instead of employing scripting languages it uses precompiled modules that make the execution faster.

The development of networked business applications first involved the use of 2-tier application concept. This concept describes an application as having all of its processing done on the client's machine, while the server only provides database functionality by providing data to the applications running on the client's side. Thus, the client-side application constitutes the first tier, while the server providing data to be processed constitutes the second tier, as shown in the following diagram (figure 2.2).

* recursive acronym for "PHP: Hypertext Preprocessor"

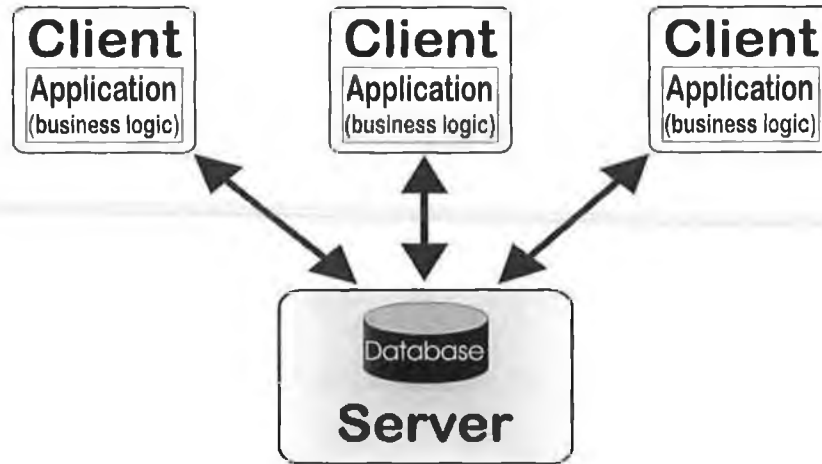


Figure 2.2 - The 2-tier application concept

Early two-tier (client/server) applications were developed to access large databases, and incorporated the rules used to manipulate the data with the user interface into the client application. The server's task was simply to process as many requests for data storage and retrieval as possible.

Two-tier applications perform many of the functions of stand-alone systems: They present a user interface, gather and process user input, perform the requested processing, and report the status of the request. This sequence of commands can be repeated as many times as necessary. Because servers provide only access to the data, the client uses its local resources to perform most of the processing. The client application must contain information about where the data resides and how it is organized in the database. Once the data has been retrieved, the client is responsible for formatting and displaying it to the user.

The client side application is usually a monolithic application that handles the network functionality, processing of data, display of the results to the user and the handling of user-generated events. These types of clients are called “fat clients”.

One major advantage of the client/server model was that by allowing multiple users to simultaneously access the same application data, updates from one computer were instantly made available to all computers that had access to the server.

However the system had many disadvantages, of which some of the most important are:

- Applications are difficult to develop and maintain; since the development is platform dependent, the application needs to be recoded/recompiled for each platform it is running on. It is also difficult to maintain because it has to be reinstalled onto each client's machine individually which can be very expensive, complicated, prone to error and time consuming.

- As the number of clients increases, the server becomes overwhelmed with client requests.
- It leads to increased network traffic. Since the actual processing of the data takes place on the remote client, the data has to be transported over the network, which increases the load.
- Security issues are involved. Unauthorised access into any client's computer compromises the security of the database.

3-tier architectures endeavour to solve these problems. This goal is achieved primarily by moving the application logic from the client back to the server [Gupta 2002]. This type of architecture is described in the following picture (figure 2.3).

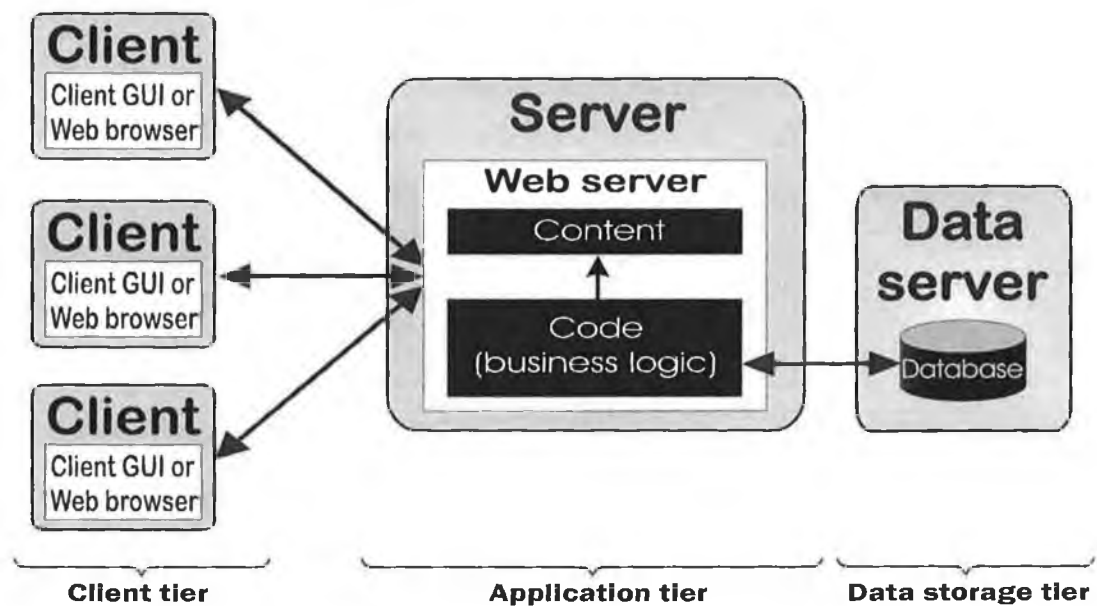


Figure 2.3 – The 3-tier architecture model

The three tiers have the following functionality:

- **Client tier**
Is responsible for the presentation of data, controlling the user interface, receiving user events and submitting them to the application server. The actual business logic has been moved to an application-server.
- **Application tier**
Business-objects that implement the business rules are embedded here within the application code, and are available to the client tier. This tier also protects the data from direct access by the clients.
- **Data server tier**

This tier is responsible for data storage. Besides the widespread relational database systems, existing legacy systems databases are often reused here.

It is important to note that boundaries between tiers are only logical. Although usually each tier is located on a different computer, it is quite possible to run all three tiers on one and the same (physical) machine. The important fact is that the system is structured in an ordered fashion, and that there is a well-planned definition of the software boundaries between the different tiers.

The main advantages of the 3-tier architecture are [Lorriman 2000]:

- Clear separation of user interface control and data presentation from application logic. The three-tier architecture isolates each major piece of functionality, so that the presentation is independent of the processing rules and business logic, which in turn is separate from the data. Through this separation changes made in the application are easier and faster to implement leading to faster development through the reuse of pre-built business-logic components and a shorter test phase. It is, however, required that interfaces remain stable and old client versions are still compatible. In addition such components require a high standard of quality control since low quality components can endanger the functions of a whole set of client applications.
- Re-design of the storage strategy will be transparent to clients. Relational databases (RDBS) offer a certain independence from storage details for the application and (usually) accessing the database from within the application is totally transparent for clients. However, cases like changing database structure make it necessary to adapt the application running on the server and thus changes may appear in the way users perceive the functioning of the application. Usually, even radical changes, like switching between different types of relational databases and Object Oriented Databases (OODBS) will not influence the client. In well-designed systems, the client still accesses data over a stable and well-designed interface, which encapsulates all the storage details.
- Application speed is increased. If the application is designed so that business logic within the application and data storage is brought as close together as possible or that client access and database access take place using different (separated) networks the speed of the application is higher due to increased database access speed and reduced network traffic.
- Further speed increases are easy to obtain through dynamic load balancing. If bottlenecks in terms of performance occur, dynamic load balancing systems can easily

be implemented allowing redirection of clients to several servers running the application and accessing the same database.

- Increased security. Security implementation on the server is easier to implement and more effective than that of thousands of distrusted client machines. Data protection and security are simpler to obtain and therefore it makes sense to run critical business processes that work with security sensitive data on the server.

This model requires much more analysis and design effort up front, but greatly reduces maintenance costs and increases functional flexibility in the long run.

2.4.2 Server-Side Technologies

In order to deliver active content some mechanism is needed that will ensure the processing of the information supplied by the user and the data obtained from different sources like databases or other web servers. Upon processing, the server will deliver the requested content with the proper formatting.

The content needed for wireless devices is WML, which stands for Wireless Markup Language. WML is XML compliant, which means that it conforms to specific rules of syntax that are defined in the eXtended Markup Language specifications. [XML] Since HTTP (WEB) servers have their main designation of delivering HTML content or any other XML compliant markup language, it makes sense that – if properly configured – these servers could also deliver WML.

The following image depicts the structure of the Web server employing ASP pages in order to process WML content, using data retrieved from a database using ActiveX Data Objects (ADO).

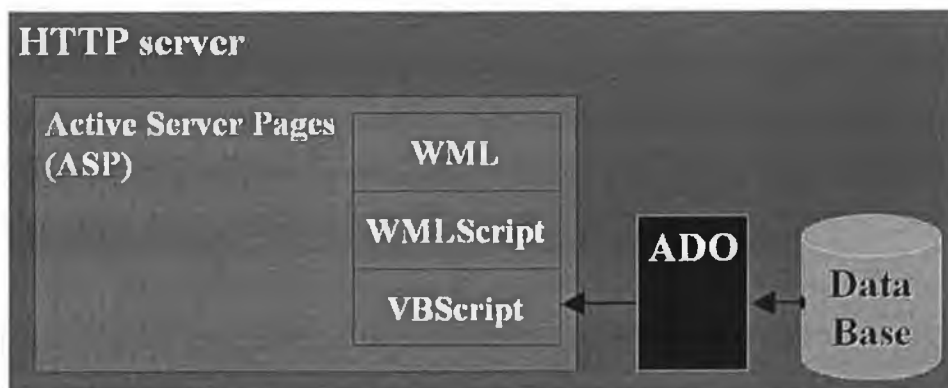


Figure 2.4 – Web server architecture

The WEB server program running the code contained within the ASP pages is able to connect using the ADO (which is not part of the WEB server program) to the database.

The database may or may not be located on the same physical machine as the WEB server program, and for this reason it has been represented as a separate entity.

ActiveX Data Objects (ADO)

ActiveX Data Objects (ADO) is a server-side technology designed by Microsoft and it's intended to provide a common programming model for any OLE DB data source. It is essentially a collection of objects that expose the attributes and methods used to communicate with a data source. ADO uses general OLE DB providers to access unique features of specific data sources; it also uses native OLE DB providers, including a specific OLE DB provider that provides access to Open Database Connectivity (ODBC) drivers. Designed to replace the need for all other high-level data access methods, ADO can access relational, Indexed Sequential Access Method (ISAM), or hierarchical databases, or any type of data source—as long as there is an ODBC-compliant driver.

OLE DB, the foundation of Microsoft's Universal Data Access model, is a set of COM interfaces that provides a standard way for programs to access data. The way applications use ADO functionality will be partially determined by whether or not there is an OLE DB provider for the data. ADO is designed to work with OLE DB, and in most instances ADO components will communicate with databases through OLE DB; ADO can also be used to communicate directly with the ODBC driver, if no OLE DB provider is available.

The following diagram shows the hierarchy in which the ADO components are used.

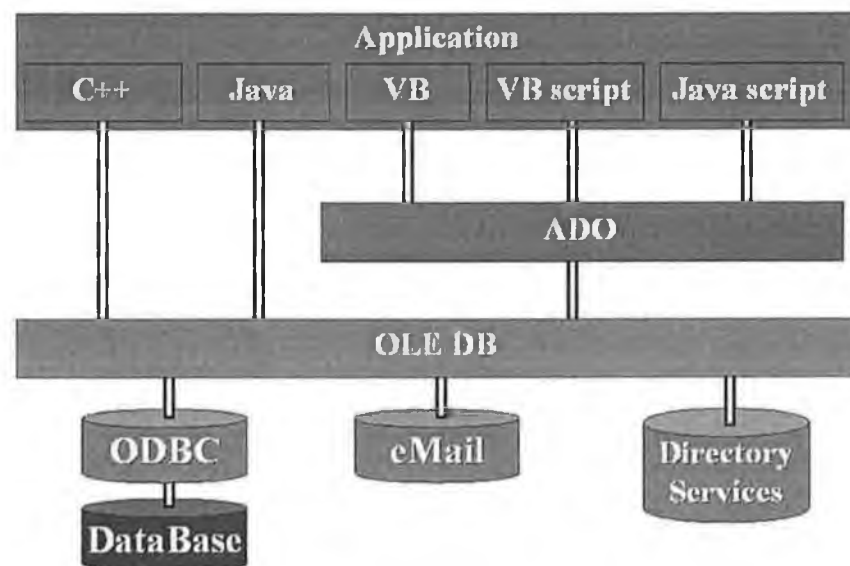


Figure 2.5 – ADO hierarchy

ADO's ease of use, speed, and low memory overhead make it ideal for server-side scripting. In fact, ADO is the recommended technology for data access for ASP

applications. ADO can be called directly from server-side scripts or from business components.

Unlike earlier data access methods, ADO does not require navigation through a hierarchy to create objects; most ADO objects can be created independently, which allows greater flexibility in reusing objects in different contexts and reduces memory consumption. ADO also takes advantage of ODBC 3.0 connection pooling for ODBC data sources, and session pooling for OLE DB providers. This eliminates the need to continuously create new **Connection** objects for each user, which is very resource intensive.

Active Server Pages (ASP)

Active Server Pages (ASP) is a server-side scripting environment that can be used to create dynamic, interactive Web applications. With ASP, HTML pages, script commands, and ActiveX components can be combined to create interactive Web pages or powerful Web-based applications. ASP applications are easy to develop and modify. In short, ASP is an open Web application framework that lets developers combine server scripting with custom components to provide dynamic Web-enabled applications [Blexrud 1999].

Active Server Pages enables server-side scripting for Microsoft Internet Information Server (IIS) with native support for both VBScript and JavaScript.

The fact that ASP is a compile free application environment means that whatever code is contained within the pages (code needed for processing the content) it does not need to be pre-processed (or compiled) into machine code in order to be executed by the computer. The code is rather stored in its original form (in the form it was written) and it will be run directly by the web server's script interpreter, which will perform whatever operations needed in order to process and deliver the requested content.

An ASP file can contain text, HTML or other XML-compliant markup language such as WML, and scripting languages, such as Visual Basic Script (VBScript) or Java Script. Scripts in an ASP file are executed on the server-side, as opposed to the scripts delivered to the client, which are also called client-side scripts. An ASP file has the filename extension .asp

In order to achieve the processing needed for the active content, content that modifies dynamically according to user input and data retrieved from databases, scripting languages are used within the active server pages and on the client side.

The following section provides a brief overview of the scripting languages used for developing applications.

2.4.3 Scripting Languages

Web servers and browsers have the capability of processing scripting code embedded within the content that is delivered/retrieved.

There are three major types of programming and execution models in use today. These refer to how the code is compiled and executed on the machine and by the program that generated it. These models are:

- Compile separate from execution. The programmer designs the initial code (in ASCII format), then the code is compiled using the specific compiler program for which it was written. The resulting code consists of machine code instructions (binary code) and it is specifically designed for a certain machine and platform type (for instance, there are programs specifically designed to run on DEC Alpha machines that are using the Microsoft Windows NT platform). The code is then executed directly onto the target machine, without the need or use of any other additional support from the compiler program. Examples of such programming languages are C++, Pascal or Delphi.
- Run-time compiling. The compilation is performed at execution time. The programming environment compiles the ASCII code when it is instructed to execute the program. After the compilation is performed, the resulting code is launched into execution. Examples of such programming languages are BASIC (and its variants, such as QBASIC), and the scripting languages.
- Pseudo-compilation. When using this mechanism, some compilation is done in the initial stage, and then another compilation process takes place on the target machine just before the program is executed. The mechanism is employed by the Java programming language mainly to achieve platform independence. The result of the initial compilation phase is pseudo-code, a code that is neither in plain ASCII format nor in machine code format. This pseudo-code is then ported to some other machine that is using a certain type of operating platform. Integrated into the operating platform, there is a runtime environment specifically designed for that specific machine architecture and operating system. This runtime environment will then further compile and execute (or better said it will execute the code itself, internally) so that the application can provide its functionality adapted to that specific machine architecture and operating system environment.

The functioning of scripting language applications running on web servers doesn't exactly fit either of these mechanisms. Although it is somehow similar with runtime compiling and execution method, it does not conform to it in the fact that it does not recompile the application code every time the execution is required. For instance, when a scripting application residing on a web server is requested to run for the first time (through the normal HTTP request coming from a client on the Internet), the web server identifies the compiling engine needed to compile the specific scripting code contained within the application. The engine, which is internal to the Web server, then performs the compilation and if no errors are generated, the Web server launches the resulting application code into execution and generates the content requested. On the other hand, when subsequent requests are made for that application, the Web server determines whether the source code contained within the files of that application has changed, and if it did not then the server uses the same compiled version previously generated to obtain and submit the content. The main purpose for employing this method is that it increases server's performance by substantially reducing server's reaction time. This is due to the fact that this method eliminates any additional processing time used for compiling the application code and it also drastically reduces the amount of resources (such as memory) needed to satisfy one request. In the case of servers, the processing of the scripting code starts immediately after a request has been received from the client and ends once the content has been made available for delivery. In the case of the client browsers, like web browsers, the processing can last as long as the content is still loaded into the client's memory (or, for some types of browsers, it is still present on the viewable display area of the browser).

Scripting languages are generally very high-level languages, which confer them the following features [Blexrud 1999]:

- Interpreted execution, so compile-link cycle is required. The scripting languages need to be compiled prior to execution by the application attempting to run the code.
- They have a simple syntax.
- They use untyped variables, which act as strings or numbers depending on what operation is being performed on them.
- Variables are created when referenced, rather than through explicit declarations.
- They do not use pointers or memory allocation.

For reasons that will be revealed in chapter 4, the scripting languages used to build the applications describes in this thesis are VBScript on server-side and JavaScript on client-side (for the Web application).

Microsoft Visual Basic Scripting Edition (VBScript), a subset of the Microsoft Visual Basic programming language, is a fast, portable, lightweight interpreter for use in World Wide Web browsers and other applications that use Microsoft ActiveX Controls, Automation servers, and Java applets. It brings active scripting to a wide variety of environments, including Web client scripting in Microsoft Internet Explorer and Web server scripting in Microsoft Internet Information Server.

In general, scripting languages are easier and faster to code than the more structured, compiled languages such as Java, C and C++, and are ideal for smaller programs of limited capability or that can reuse and tie together existing compiled programs.

VBScript, which is a variant of Visual Basic, is the default server side language for Active Server Pages (ASP). It talks to host applications using ActiveX Scripting. With ActiveX Scripting, browsers and other host applications do not require special integration code for each scripting component. ActiveX Scripting enables a host to compile scripts, obtain and call entry points, and manage the namespace available to the developer.

JavaScript is a light version of the Java language, developed by Sun Microsystems. It owes its popularity mostly to the fact that it was developed by the Netscape Communications Corporation and integrated into their web browser, which became fairly popular. Later, other big corporations like Microsoft adopted and incorporated it into their applications. At this time it is the most popular client-side scripting language used on the web.

Core JavaScript contains a core set of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. It can also be extended for a variety of purposes by supplementing it with additional objects specific to the application that implements it. For instance, client-side JavaScript implements extensions allowing an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation, while server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server.

The JavaScript language itself is standardized by the ECMA-262 Standard. [ECMA 1999]

Structured Query Language (SQL)

SQL (Structured Query Language) is a specialized programming language for querying information stored in relational databases.

The relational database is based on the idea that the database is made up of tables of data, where each cell holds a single piece of data. There can be no real arrays, or struct-like objects in a database of this type. The database contains many tables, many of which have relationships with each other, hence the name relational database.

SQL is an ANSI (American National Standards Institute) standard language for performing database operations. Its syntax is relatively simple, yet it is powerful enough to handle complex data tasks.

A SQL *statement* is a command made up of *clauses* that specify the operation to perform, the data source, and any instructions needed to complete the operation.

Besides retrieving data from a database, SQL statements also can be used to manipulate the records in a database, create and remove data objects, and to run administrative tasks such as setting user access to data sources.

There are many implementations of the SQL database (Oracle, MySQL, Microsoft SQL Server), but each implementation has a core set of SQL commands. Every database contains commands such as select, create, delete, insert etc. but each vendor will add specific functionality in a certain area.

2.4.4 The Wireless Markup Language (WML)

WML is a markup language based on XML (Extensible Markup Language). It is designed for specifying user interface behaviour and displaying content on wireless devices such as phones, pagers, and PDAs (Personal Digital Assistants).

The devices that currently support WML fall into two principal categories:

- Phones - which typically feature text displays of 4 to 10 lines and support user input through numeric and function keys.
- Personal Digital Assistants (PDAs) - which typically feature display resolutions of 100X100 pixels (or better) and support enhanced user input through keypads, pointers, or handwriting recognition.

It is anticipated that as handheld devices with sophisticated capabilities, such as voice recognition, become available, many of them will also support WML.

Because WML supports a variety of devices with different capabilities, a common client device can be described with reference to a "least common denominator device" or "reference device". The reference device has the following characteristics:

- A display area having a width of 12 (fixed-size) characters and a height of 4 lines, including one line reserved for function key labels (described below)
- Support for the ASCII printable character set
- Numeric and alphabetic character entry
- Choice selection (using arrow or numeric keys)
- Two programmable function keys, referred to as `ACCEPT` and `OPTIONS`, the labels for which appear above the key in the phone display area
- A `PREV` key for navigating backward
- Vertical scrolling with arrow keys
- Horizontal scrolling of non-wrapping lines

WML uses the XML document character set - currently the Universal Character Set of ISO/IEC-10646 (Unicode 2.0) - and supports any proper subset of the Unicode character set (for example, US-ASCII, ISO-8859-1, or UTF-8).

WML uses tags - just like HTML - but the syntax is stricter and conforms to the XML 1.0 standard. The WML pages have the extension `*.WML`, just like HTML pages have the extension `*.HTML`.

Unlike HTML, WML is case-sensitive. Elements, attributes, and enumerated attribute values must be specified in all lowercase. The case-sensitivity should also be kept in mind when naming cards or variables.

WML is mostly about text. Tags that would slow down the communication with handheld devices are not a part of the WML standard. The use of tables and images is strongly restricted. WML pages are called *decks*. They are constructed as a set of *cards*, related to each other with links. When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. The phone computer – a small processing unit inside the phone – handles the navigation between the cards without any extra access trips to the server.

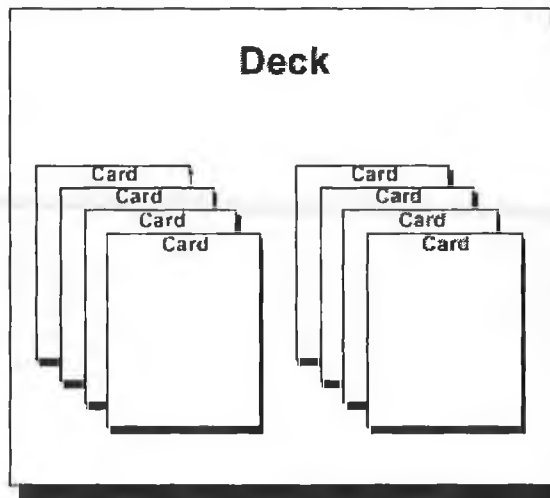


Figure 2.6 – The structure of the WML deck

When a user agent receives a deck, it typically activates the first card in the deck unless directed to a different card as specified by the requesting URL in its reference section. A card element can contain text, markup, links, input-fields, tasks, images and more. Cards can be related to each other with links.

All WML decks must specify the following XML document type declaration at the beginning of each file:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

WMLScript is a scripting language that can be used together with WML to provide client-side procedural logic to WML cards and decks. It can also be used as a stand-alone tool. Based on ECMAScript, WMLScript has been modified to better support low bandwidth devices such as mobile phones.

WMLScript is a light JavaScript language. However, WML scripts are not embedded in the WML pages. WML pages only contain references to script URLs. The scripts need to be compiled into byte code on a server before they can run in a WAP browser.

With WMLScript, the following restrictions of WML can be overcome:

- Check the validity of user input.
- Access facilities of the user agent. For example, on a mobile phone, allow the programmer to make phone calls, send messages, and add phone numbers to the address book or access the SIM card.

- Generate messages and dialogs locally, thus allowing alerts, error messages, confirmations, and so on, to be seen faster by the user.
- Allow extensions to the user agent software and configure a user agent after it has been deployed.

Additional details regarding WML and other markup languages, as well as scripting languages and other related Internet technologies can be obtained from the W3schools Web site (www.w3schools.com)

2.4.5 The WEB Server

In order to be able to deliver content on the Internet, the computer hosting the site needs to be connected to the Internet. In addition to that, in order to make the WAP content available on the Internet, a WEB server must be used. After the Internet connection is in place, the next step is to install the WEB server. One of the most common WEB servers available is the Internet Information Server (IIS) provided free of charge by the Microsoft Corporation. On Windows NT Server platforms, the Internet Information Server 4 (IIS 4) WEB Server is provided as standard and can be installed from the installation CD containing the operating system installation kit. Also IIS ver.5 is already integrated in Windows 2000. On the other hand, when using Windows NT 4.0 Workstation the WEB server is not included in the installation kit, and as such it must be separately downloaded and installed from the Microsoft Corporation's WEB site [Microsoft].

Once downloaded (or otherwise obtained), the Option Pack 4 package can be run and will provide user the ability to install and configure the IIS 4 WEB server. By default the Option Pack 4 installation wizard will create a folder called Inetpub located in the root of the primary partition (usually the C drive). Located in this folder, another folder called WWWROOT will be created. This folder is the actual root of the web site, containing the root index file and all the subfolders hosted on that site. Usually, this default location does not need to be changed. When the installation process finishes, the user is prompted to restart the computer and upon doing so the web server will be automatically started as an Windows NT service and a control icon will be displayed in the tray bar (located on the right-hand side of the task bar).

After installing the WEB server, the WAP application's files will be posted. Since the WEB server does not support WAP content by default, the user must insure proper configuration of the server in order to be able to deliver the appropriate content to WAP

devices. The next step is to configure the WEB server so that it supports the MIME types needed to deliver content required by WAP devices.

MIME (Multipurpose Internet Mail Extensions) is a specification for the format of data that can be sent over the Internet. When the server sends data in response to a request it receives, it sends a MIME type with it. This MIME type can also be explicitly set by the application. Normally, the file extension of the requested file is associated with a MIME type and so the server automatically issues the correct MIME type. Then, when a browser receives information from the server, it checks its MIME type to see what to do with it. If, for instance, it sees that the data has a MIME type of “image/vnd.wap.wbmp” then it knows to display it as a picture. The MIME types accepted by a certain browser can also be discovered dynamically by the WEB server using information included in the request for data. This enables an application to explicitly set the MIME type by determining the supported MIME types of the device.

In order to configure the MIME types for IIS, the Microsoft Management Console for the Internet Information Services must be opened. This is achieved by opening the *Server Extensions Administrator* from the *Start* menu under *Administrative Tools* in the *Programs* menu. The following picture shows the Internet Information Services management console.

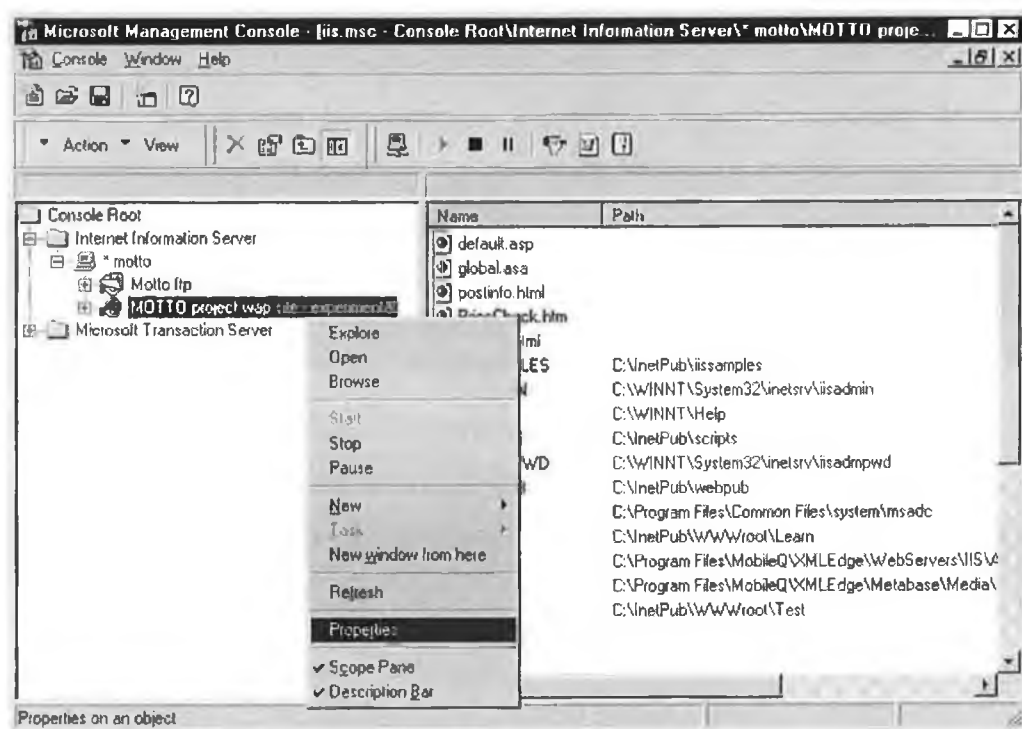


Figure 2.7 – The Internet Information Services management console

Under the heading *Internet Information Server*, by right clicking on the machine name it will bring up a menu from which the *Properties* option should be selected. A properties

window should appear, in which the *HTTP Headers* tab must be selected. Once this tab is selected, in the lower part of the Properties window there is a section called MIME Map. By clicking the button File Types in this section, a window called *File Types* opens, listing the additional file types (besides the default ones like HTML or BMP) supported by the server. By clicking the *New Type* button, it is possible to add the MIME types required.

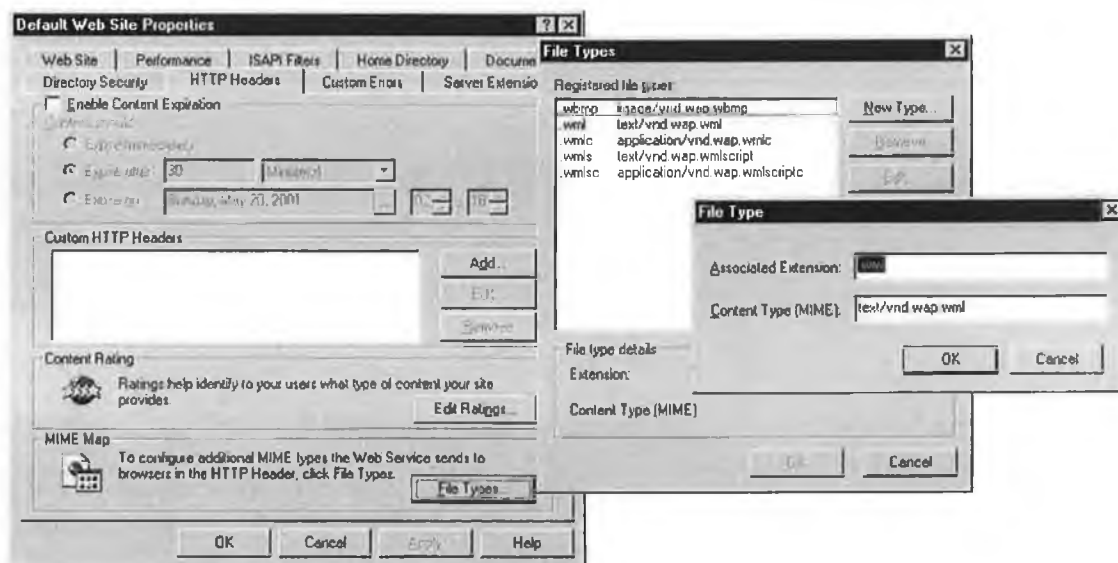


Figure 2.8 – Setting the MIME types

The following MIME types should be added to supplement the ones already added by the installation program:

File extension	MIME type
.wml	text/vnd.wap.wml
.wmlc	application/vnd.wap.wmlc
.wmls	text/vnd.wap.wmlscript
.wmlsc	application/vnd.wap.wmlscriptc
.wbmp	image/vnd.wap.wbmp

Table 2.2 – The WAP MIME types

The first extension type refers to the WML code, which is in text format (uncompiled), while the second extension type deals with WML code in compiled form (binary format).

Similarly, the wmls extension refers to wml script code (text format) while the WMLSC extension deals with the WMLscript code in a compiled form (binary format).

The last extension type deals with the WBMP extension, which is the wireless bitmap format (image format).

Since WAP devices are very limited with regards to the processing power, memory storage capacity and display size, this image format has only one bit color depth (every pixel can be either black or white) and the maximum size is 256x256 pixels (depending on the browser capabilities of the client device used).

2.4.6 The WAP Toolkit

A very useful tool used in the process of creation and testing of mobile applications is the Nokia WAP Toolkit.

The Nokia WAP Toolkit offers developers a PC environment for creating, testing and demonstrating WAP applications, and will also allow developers to test the usability of wireless applications and services with customers. It includes tools for creating WML and WMLScript content, adding WBMP graphics, debugging and simulating WAP applications on WAP-enabled handsets. The latest release, ver. 2.1, provides application developers with their first chance to take advantage of new push functionality.

WAP application developers can create complete applications without a handset or access to carrier infrastructure. Testing and demonstrating WAP applications is straightforward, and developers can navigate and request URLs on any WAP gateway or on any web server on the Internet. In addition, applications can be stored and queried directly from the PC file system.

The Nokia WAP Toolkit is available free of charge on the Internet from the Forum Nokia at <http://www.forum.nokia.com/>

In order to be allowed to download the toolkit software, the user will first have to register with the Nokia developer program. In order to accomplish this, some amount of information must be provided about the company the developer is working for, and also a username and a password must be supplied. The username and the password will be used for subsequent access to the Nokia Developer section. Upon logging on with the username and the password provided, the user will gain access to various developer software programs, amongst which is the Nokia WAP Toolkit.

The key features for the Nokia WAP Toolkit include:

- WML and WMLScript editors for creating and editing static WML and WMLScript content

- Support for WAP 1.2 functionality, including push services, WML and WMLScript
- Device simulators, including an exact Nokia 7110 phone simulator for displaying WAP content
- Server simulator based on Nokia WAP Server if a WAP Gateway is unavailable
- Debugging and testing tools showing detailed information about the application execution
- WBMP Graphics editor allows you to create graphics in WBMP format or convert GIF and JPEG images to WBMP format

Following is a list of the simulators included in the toolkit and the WAP versions they support:

Toolkit Component	Supports WAP Version
Blueprint Phone Simulation	June 2000
Server Simulator	June 2000
Nokia 7110 Phone Simulation	WAP Version 1.1
Nokia 6210 Phone Simulation	WAP Version 1.1

Table 2.3 – Simulators included with the Nokia WAP Toolkit 2.0

The system requirements for the Toolkit are:

- Windows NT 4.0 with ServicePack3 or later, Windows 2000 or Windows 98
- Requires Pentium class 266MHz or faster processor
- 128 MB RAM (256Mb recommended)
- 16-bit colour display with 1024x768 resolution
- 65 MB of hard disk space
- Requires installation of Java Runtime Environment (JRE) version 1.3 or later. During Toolkit installation, this component will be automatically installed if it is not already found installed. It is also freely available from the Sun Microsystems Web site. [SUN]
- The user documentation is provided in Adobe Acrobat PDF format. In order to read it, Adobe Acrobat software is required, which is freely available for download from Adobe Web site.
- To test services available on the Internet, an Internet connection is required.

The Toolkit browser is a fully functional browser and is able to display URLs from a WAP Gateway as well as local files. It provides complete bookmarking and history functions. It also provides an encoder for converting textual WML and WMLScript content to binary format for transmission over a wireless network. Conversely, its decoder transforms received binary content to text for display on the phone simulations.

The device simulations are on-screen images of phones that have fully functional displays and keys for navigating and entering data. The device simulations provided by the Toolkit include the following:

- Blueprint A Java-based phone simulation having no real-world counterpart.
- Nokia 7110 A simulation of the actual Nokia 7110 phone. It requires the WAP Server Simulator (provided) for use.
- Nokia 6210 A simulation of the actual Nokia 6210 phone. It requires the WAP Server Simulator (provided) for use.

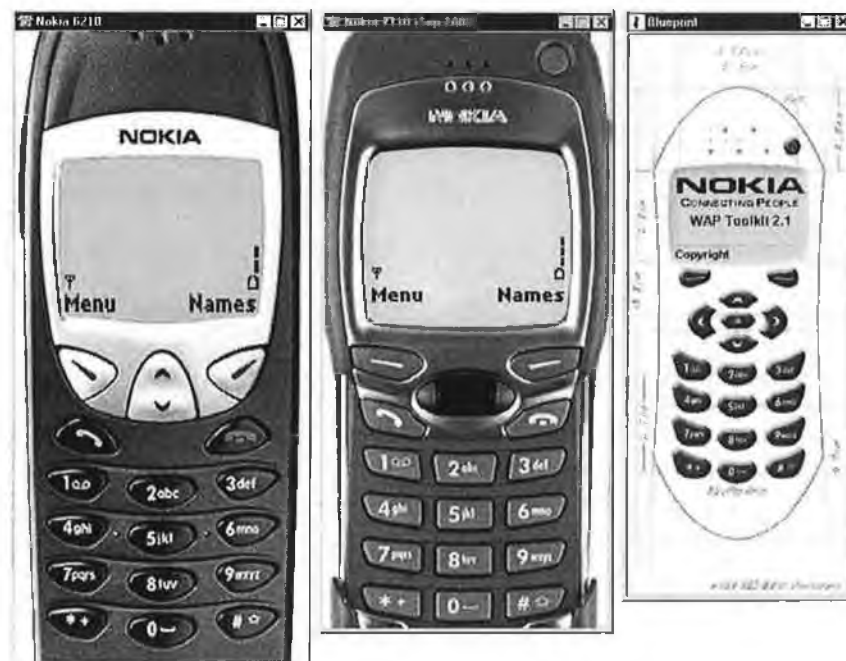


Figure 2.9 - Nokia 6210, Nokia 7110 and the Blueprint phone simulations

The Toolkit provides editors for convenience in developing WAP applications. The following four editors are activated through the *File*→*New* command and produce new tab views of the files being edited: WML Deck, WMLScript, WBMP image, and Multipart Content. In addition, the Push Message Inbox provides three forms-based editors for creating Push messages containing SI, SL, and CO content.

All Toolkit tab views provide information; however, the following tab views are particularly useful for debugging: Toolkit Message Log, WML Browser Context History, WML Browser Context Variables, and WML Browser Session/Cache Information.

There are four network configurations in which the toolkit can be used are presented in figure 2.10. In this diagram, method 1 uses the WAP protocol, while methods 2 and 3 use the HTTP protocol. Method 4 depicts local file access.

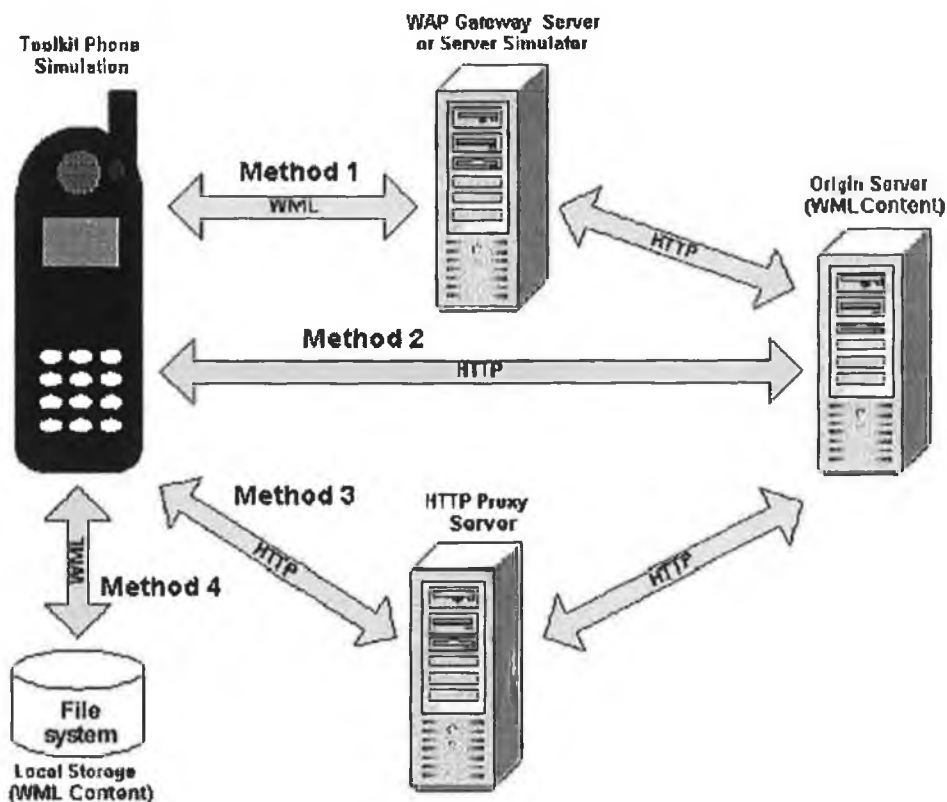


Figure 2.10 – Nokia WAP Toolkit 2.0 network configurations

The description of the diagram is as follows:

- With method 1, the phone simulation is configured to retrieve WAP content through either a WAP Gateway or the Toolkit's WAP Server Simulator. All Toolkit phone simulations may be configured in this manner.
- In method 2, the phone simulation is configured to retrieve WAP content using the HTTP protocol from a Web origin server, providing unencoded, that is, noncompiled, WML. The Toolkit encodes the textual HTTP content and displays the encoded content on the phone display. Only the Blueprint simulation may be configured in this manner.
- Method 3 is very similar to method 2; the only difference resides in the fact that the content passes through an HTTP Proxy Server. The Toolkit encodes the textual HTTP

content and displays the encoded content on the phone display. Only the Blueprint simulation may be configured in this manner.

- In method 4, the Blueprint phone simulation is configured to retrieve and display WAP content (compiled WML and WMLScript) from direct file access. Only the Blueprint phone simulation may be configured in this manner.

The Nokia 6210 and 7110 series simulations are capable of being configured using method 1 only, as depicted in Figure 2.10, which means they require either a WAP Gateway or the Toolkit WAP Server Simulator. The Blueprint phone simulation can be configured to use methods 1, 2, or 3. This means it possesses the additional capability of making HTTP requests to an Origin server without using a WAP Gateway as an intermediary (methods 2 and 3). Whether to choose method 2 or 3 with the Blueprint simulation depends simply upon the existing network infrastructure. If a Proxy Server is used in order to access the Internet, than method 3 must be used. If not, than method 2 should be the obvious choice.

Since the purpose of the Toolkit is to help development of WAP applications that work over a WAP Gateway, configuring the phone to use a Gateway (or the WAP Server Simulator) is basically standard procedure. The capability of configuring using methods 2 and 3, bypassing the WAP Gateway, is provided for the purpose of debugging problems with applications. For instance, if a problem is encountered and it is not clear whether the problem lies in the application code or in the WAP Gateway itself, the Blueprint simulation could be configured to use methods 2 or 3 for testing purposes. If the problem disappears, than it might be assumed that the problem is within the WAP Gateway, since it was bypassed in the new configuration. If not, it is likely that the problem lies in the application.

The WAP standard specifies an image format called WBMP. The Nokia WAP Toolkit, which includes a WBMP editor, can be used to:

- Create new WBMP images of any size.
- Open existing WBMP images in the editor and modify them, as necessary.
- Import GIF and JPEG images and convert them into WBMP.
- Draw pixels, straight lines, boxes and ellipses with outlines in black or white, of variable thickness and with a choice of fill-in patterns.
- Zoom in and out of the image in the editor.
- Cut, copy, and paste selected regions of the image.
- Write the finished WBMP images to a file.

Because of the limited display area on mobile devices, images tend to be small and relatively simple, as for example an up or down arrow to indicate the direction of the stock market, or perhaps an image indicating the weather. Working with such small images means working with pixels, and consequently it is necessary to specify the size of the WBMP image (in pixels) when the editor is opened in order to create a new image. Mobile phone displays vary in size from roughly 80 x 60 pixel areas to 100 x 100, or more. So, it is recommended to specify a starting size of say 32 x 32 or maybe 16 x 16, and then decide by trial and error what exactly works best for the specific mobile device and application being developed.

An important function served by the WBMP editor is its capability to convert existing images from GIF or JPEG format to WBMP format. This is true because it is usually preferable to use some other more robust image-editing program to create the image.

The Toolkit displays Push messages only in the Blueprint phone simulation. There are four content types that the Toolkit is able to receive as Push Messages, and each must be received in encoded form. These types are:

- Service Loading
- Service Indication
- Cache Operations
- WML

The Toolkit displays all Push messages in the Push Inbox view. When a Push message is received, the Toolkit decodes it and places it in its Push Message Inbox without processing it. This allows viewing of the headers and content of the message before processing. In order to initiate Push message processing, the *Activate Msg* button must be enabled. Then, the effects of the push message can be observed on the phone simulation, on the cache contents and on the Push Inbox contents. In order to determine the processing by the Toolkit of the Push messages immediately upon receiving them, the AutoActivate option in the Device Settings must be selected.

The soft keys and number pad on the mobile phone display can be used in order to navigate through menus and cards and for entering text and numbers. If there is only one soft key action specified, the soft key is directly available, and the Options button is not displayed.

The phone simulations use three editors: one for text, one for numbers, and one for passwords. The WML code specifies which editor the phone simulation uses. The following image (figure 2.11) shows the text editor.



Figure 2.11 – The text editor of the Blueprint phone simulator

Letters can be entered into the text editor by clicking the alphabetic keys on the phone simulation's keypad. In the Blueprint phone simulation only, the keyboard may also be used to input data. Note in the above image that both upper and lowercase characters are displayed. Switching between upper and lower characters can be achieved by clicking the pound sign (#); this changes the entry mode. The current entry mode is shown in the top left hand corner, in this case "abc" indicating lowercase. Numbers cannot be entered in the text editor using the phone keys, only alphabetic characters; however, the computer's keyboard can be used for this purpose.

If the WML code specifies number entry, the phone simulation uses the number editor. The number editor is indicated in the phone display by "123" in the upper left corner of the display. The number editor permits only numerals to be entered; alphabetic characters cannot be entered. The password editor is similar to the text editor, except that the characters entered are displayed only briefly (so that the user gets a glimpse of what was entered) and then they change to asterisks (*) for security purposes.

When the Toolkit is launched, its main window displays seven tabs, each of which opens a different view into Toolkit functions and features. These tabs can be removed or added to the main window by using the *Toolkit* → *Show* command on the menu bar. This command also controls whether or not the mobile phone simulation (WAP Device) is displayed. In addition to that, whenever a new file is created or an existing file opened (via *File* → *New* or *File* → *Open*), a new view with an accompanying tab is created.

2.5 Summary

This chapter provided a brief overview of the Internet-related business issues, technical issues and other aspects that are behind the functioning of the Internet and its underlying technologies.

The first section, called Business Overview, analysed business notions such as eBusiness, eCommerce and mCommerce. It also provided a brief overview of concepts

such as Customer Relationship Management, Business-to-Business and Business-to-Consumer.

The second section, titled Internet and Mobile Technology, provided a brief overview of the Internet and its structure by highlighting some of the concepts behind it. It also provided a description of the wireless technologies, WAP and other related notions.

The third section was a review of the existing technologies in the area of web applications. Some of these technologies have been used in the development of the applications described in this thesis. It described notions like the 3-tier applications concept employed for developing web applications, server-side technologies such as ADO and ASP. It also provided a brief overview of the scripting languages commonly used for designing web applications such as VBscript and JavaScript and also a brief overview of Structured Query Language (SQL). The section concluded by presenting some of the tools used in the development of the applications presented in this thesis, such as the Internet Information Server (IIS) from Microsoft and the Nokia WAP Toolkit and the way they were configured in order to achieve the required functionality.

The next chapter will analyse the characteristics of different areas related to electronic business in order to outline the main features and requirements that a mobile application designed for the SME Business to Business CRM area needs to implement.

CHAPTER 3

Requirements for the Development of B2B Mobile CRM Applications

- 3.1 Introduction
- 3.2 Requirements for the area of B2B E-commerce
- 3.3 Requirements for SMEs
- 3.4 Requirements from a CRM perspective
- 3.5 Requirements and Characteristics of WEB Applications Versus WAP Applications
- 3.6 Summary

3.1 Introduction

This chapter discusses the requirements and the main features that need to be integrated into Web-based and WAP-enabled applications that will be specifically designed for the area of Small to Medium-sized Enterprise Business-to-Business Customer Relationship Management. As outlined in chapter one, each of these areas has its own specific characteristics and requirements. An application situated at the convergence of all these three areas must consequently satisfy the requirements for each of them, as described by the following diagram.

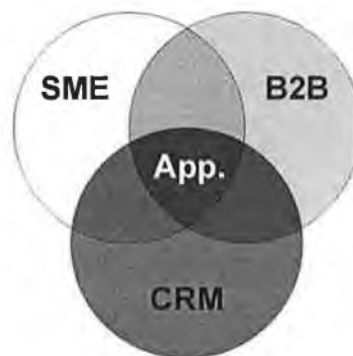


Figure 3.1 – The convergence of SME, B2B and CRM areas

In addition to satisfying the business requirements mentioned before, special characteristics and requirements must also be taken into consideration when designing applications for the areas of PC-based Internet access as opposed to mobile Internet access.

As a result, the area of interest for this application is narrowed down from the wide area of eBusiness and eCommerce to the area of Business-to-Business transactions between SMEs and further more to the specific area of Customer Relationship Management performed using the Web and particularly on wireless devices (WAP). The following diagram depicts this.

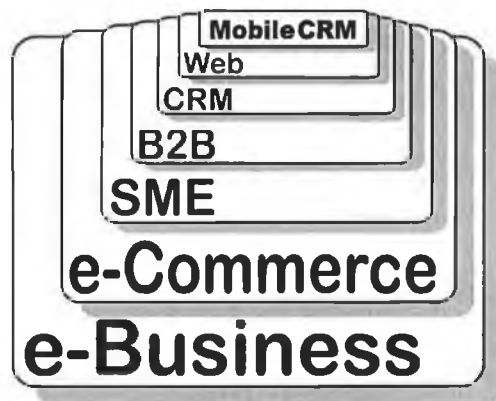


Figure 3.2 – The wireless application hierarchy in the context of e-Business

The first section starts by presenting the general requirements for E-commerce applications in the Business-to-Business (B2B) e-Commerce area. It then continues by presenting the requirements that need to be considered when designing applications that will be used by Small-to-Medium Enterprises (SMEs). The next section exposes the requirements in the area of Customer Relationship Management (CRM).

Finally, the last section of this chapter will detail the specific requirements for WAP applications by highlighting the main differences between pure WEB-based applications and WAP applications and devices.

3.2 Requirements for the area of B2B E-commerce

The application that needs to be developed must be an E-commerce type of application. Further more, it must be specifically tailored for the Business-to-Business (B2B) area, which means that it must serve as an interface between two companies in the supply chain, as opposed to the Business-to-Consumer (B2C) area, which focuses on the interface between a company and its individual consumers. This will be achieved by incorporating features that respond to the following requirements:

- It has to be a platform for mobile E-commerce (M-commerce) in the area of B2B.
- The application must enable customers represented by individual users to purchase goods / submit orders using a mobile device.

- All the transactions will be conducted on behalf of the companies that users represent.
- Customers will have to be registered and validated prior to the transaction.
- The application must provide some sort of security mechanism to ensure privacy.

3.3 Requirements for SMEs

The application must be tailored to accommodate the needs of SMEs, which means they must comply with the following set of requirements:

- **Low implementation costs.** The small price tag for the implementation is required because SMEs have very limited resources for this type of investments.
- **Easy to implement.** The ease of implementation is related to the fact that, on one hand, SMEs lack the technical resources that big companies benefit of (servers and other type of hardware, including mobile devices, and also investments in specialised software), and on the other hand because if the application would be complex it would require employing qualified specialists over a long period of time in order to complete the implementation, which in turn would lead not only to delays in launching the application but also to increased costs put into the actual development of the software.
- **Easy to maintain.** This requires that the application needs a low amount of maintenance (from a technical point of view) and it is also easy to administer. This is needed since employees working in SMEs are usually qualified only for the area of activity that the SME is in (they are not very tech-savvy). Also, the ease of administration is related to the fact that, typically, SMEs lack manpower and cannot afford diverting employees to perform administration tasks for long periods of time.

3.4 Requirements from a CRM perspective

The application must deliver value-added services like order status updates, which will address company's Customer Relationship Management issues.

An important characteristic of such a software system, as stated from a CRM perspective, is the fact that the system should be able to provide as many automated services as possible. This will ensure that those services will be provided with good response times, leading to increased customer satisfaction. The following diagram depicts the simplified structure of a company and its connection to its customers and suppliers. In this context, the role of such an application is described as creating a connection (link) between the company's customers and its production/distribution department.

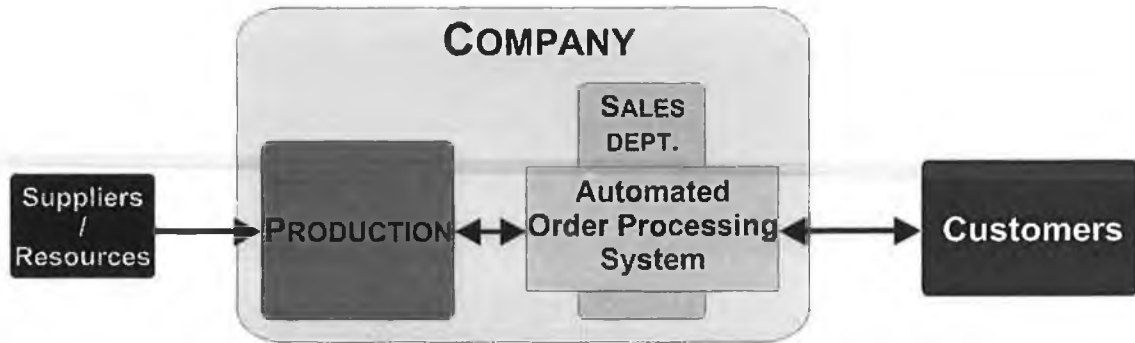


Figure 3.3 – The integration of an Automated Order Processing System

Secondly, due to the fact that these services are provided by an automated system, will reduce the number of persons involved in handling sensitive data (i.e. some customers might want to restrict access from third parties to confidential data like, for instance, what products were ordered, what quantity, type and price).

And last but not least, is the fact that by reducing the number of employees, it will increase the business's profits by cutting the costs.

3.5 Requirements and Characteristics of WEB Versus WAP Applications

In order to allow the implementation of a mobile application into an SME, certain technical conditions must be met. These conditions ensure that all the necessary steps have been taken and that all the conditions from a technical perspective are present that will allow the implementation and normal use of such a mobile application.



Figure 3.4 – Technical requirements for integrating a wireless (WAP) application

The basic condition is that the company has the initial hardware support in the form of Personal Computers (PCs) that will subsequently allow the development of an internal network. This system will then be used to set up an internal Data Management System,

allowing the company to keep track of its customer base, track their orders and maybe keeping track of stocks. At this point, the Data Management System is only used internally, for the purpose of improving company's activity by improving its workflow. The next step in the process is obtaining Internet connectivity and the development of a company Web page. Besides that, another useful service that will become available, allowing the company to interact with its customers, is Email. This will ensure the company's presence on the Internet as the first step in developing Customer Care Solutions. These solutions will initially materialise in the form of a Web application, allowing customers to interact with the Data Management System previously developed. By doing so, the customers will be able to place orders and have access to information from any Internet access point and at any given time. In order to provide even more freedom for customers that are always on the move, away from their office or other normal Internet access points, a mobile application can be implemented on top of the web application. This wireless application will also interface with the Data Management System and will possibly use some other technologies such as email or SMS.

A different set of requirements applies to WAP as opposed to the WEB applications. These requirements are due to the fact that, as already mentioned in chapter two in the literature review, mobile devices have limited capabilities compared to their WEB counterparts (PC-based Web browsers).

These characteristics are presented in the following table:

	WEB	WAP
Bandwidth	High Bandwidth	Low Bandwidth
Input capability	Very good (PC keyboard)	Limited (Mobile phone keyboard)
Memory	High	Low
Processing Power	High	Low
Display capabilities (screen resolution, number of colours)	High (Very high, full colour)	Low (Low resolution, B/W)

Sound	Yes	No
Size	Large	Very small
Portable	No	Yes

Table 3.1 – Comparison between WAP devices and Web devices (browsers)

From these characteristics can be concluded that WAP applications must comply with the following set of requirements:

- Transfer small amounts of data due to the fact that mobile devices have low bandwidth and a small amount of memory.
- Require little input from the user due to the limited input capabilities of the mobile devices.
- Must not require big amount of processing being performed on client-side due to the low processing power of the mobile devices.
- Must not rely on providing information using graphics, due to the poor display capabilities of mobile devices.
- Must not rely on providing information using sound since WAP specifications do not include sound capabilities.
- The application can rely on ubiquitous access, whenever needed, since mobile devices are small in size, portable and are able to access the Internet from any location (as long as they are in the coverage area).

3.6 Summary

This chapter highlighted the features and requirements for a generic application that would be specifically designed for the targeted area that this thesis addresses: B2B mobile Ecommerce and CRM for SMEs. It was intended as a guideline that underlines the conclusions drawn from the business and technology review performed in chapter two. There are three areas with the requirements of which such an application must comply: the area of B2B eCommerce, the area of applications designed to accommodate the needs of SMEs, and finally the area of Customer Relationship Management and value added services. In addition to that, the application that needs to be developed and implemented must also deal with the specifics of the WAP and wireless commerce applications, which

means it has to adapt its design to accommodate the limited capability of offered by WML (Wireless Markup Language, a language similar to HTML but specifically designed for mobile devices such as mobile phones) and other specifics of ubiquitous commerce (mCommerce).

As a result, the chapter was structured into four sections, each of which underlined the specific requirements for an area. These areas are:

- B2B eCommerce
- Area of SME
- Customer Relationship Management
- WAP versus WEB

The latter outlines the differences between mobile devices as opposed to WEB or WEB-based devices and browsers, that needs to be taken into consideration when designing mobile applications.

The following chapter will illustrate the integration of all the outlined requirements with the wireless technology, which resulted into a proof-of-concept prototype application. A detailed description of this application and its features will be given, which in turn will provide an overall view of the possibilities and limitations of the mobile technology used.

CHAPTER 4

A Prototype Solution for Mobile CRM

- 4.1 Introduction**
- 4.2 The Mobile Order Entry System**
- 4.3 Application design and implementation**
- 4.4 Coding details**
- 4.5 Summary**

4.1 Introduction

The prototype developed is a WAP Mobile Order Entry System aimed towards the Business-to-Business (B2B) environment. It actually consists of a WAP application hosted on an IIS* 4.0 WEB server, running on a Windows NT 4.0 or Windows 2000 platform. The code of the application can be obtained from Appendix 1. The WAP application enables a mobile device user (customer) to browse through a list of available products and to place an order for a specific product listed.

Important features of this prototype application are:

- Personalisation
- Security features, including:
 - Authentication
 - Registration
 - Limiting access to sensitive information for non-validated users
- Help system
- Info and News sections
- Changing the user details
- Products database listing and search
- Order information storage into the database
- Database updating
- Error handling

* Internet Information Server

4.2 The Context Data Flow Diagram

The first level of the Data Flow Diagram (fig.4.1) defines the scope of the system and includes all external entities/agents and their major interactions with the system. Normally in the context diagram, there is only one process in the system.

The context diagram of the prototype system is illustrated in the figure below:

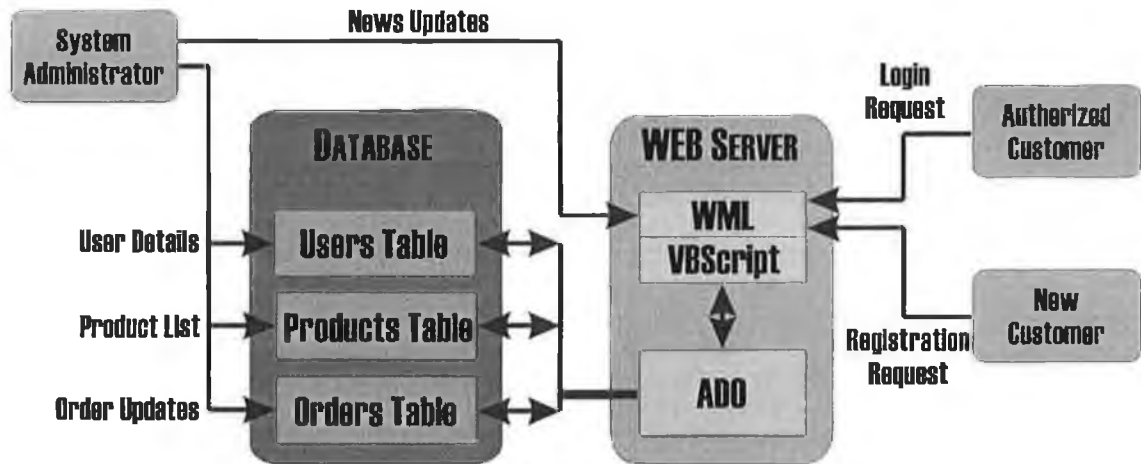


Figure 4.1 – The context data flow diagram

In the context diagram, there are three external entities: System Administrator, Prospective New Customer (required to register in order to access the system) and authorized B2B Customer (this user has a validated account set up, consisting of user name and password, which is held in the system's centralized database, controlled by the administrator). They interact with each other within the application. Their roles in the system can be described as follows:

System Administrator:

- Examines the requests from customers by examining the users table in the database.
- Administrate the MOTTO system by:
- Validating the users that need to be validated and meet the criteria established by the company's internal procedures.
- Updating the products table to include the latest product details and remove the products that are no longer in production.
- Retrieving the newly submitted orders, examining them to the departments in charge of production.
- Modifying order details to reflect the actual order status.

- Send notification (using email addresses stored in the database) to customers by submitting product and order details for users that requested or are willing to receive such notifications.
- Correlate info on new registrations. By using either an internal data management system or hard copies, the newly registered customers/companies will be recorded into the database / archive.
- Update News Bulletins that will be disseminated to system users.

New Customer:

- Will be required to register and submit email address; this will enable an employee from marketing to contact him. The registration can be regarded as a type of CRM data-mining application.
- Submit request to the system. Upon request approval, the user will be promoted to authorized customer, and will then be capable of viewing detailed information and placing orders.
- Once registered, this user will have full access to the system, up to the point of purchase. The user will then be able to access detailed information on the products listed.

Authorized Customer:

- Send product requests and receive notifications of order placement from system.
- The authorized customer, like the unauthorized one, will be able to read up-to-date information on the company in question, from the section titled 'News'.

However this user, unlike the non-validated customer, will be able to see detailed product information, including quantity in stock (availability) and price.

4.3 Application design and implementation

The prototype application consists of 23 files, of which 20 files contain the actual code in ASP format, one file is the ADO constants definition (adovbs.inc), one file contains the WML scripts required for providing client-side procedural logic (called scripts.wmls) and another file is the Microsoft Access database called prototyp.mdb. Each file constitutes a deck that will be submitted by the server to the client device upon processing of the code contained.

4.3.1 Functional Data Flow Diagram

This section will describe the functioning of the prototype application by presenting data flow diagrams illustrating the functionality and discussing them.

a) Login and Authentication

When a user attempts to access the application a complex process takes place by which the application attempts to identify the client device. Upon identification, the user is guided through the authentication process or (if he doesn't have an account with the system) through the registration process. Upon authentication or registration, the user is directed to the main page, where he will be able to take advantage of all the functionality offered by the application. The authentication process is described in a clear fashion by the following data flow diagram (figure 4.2).

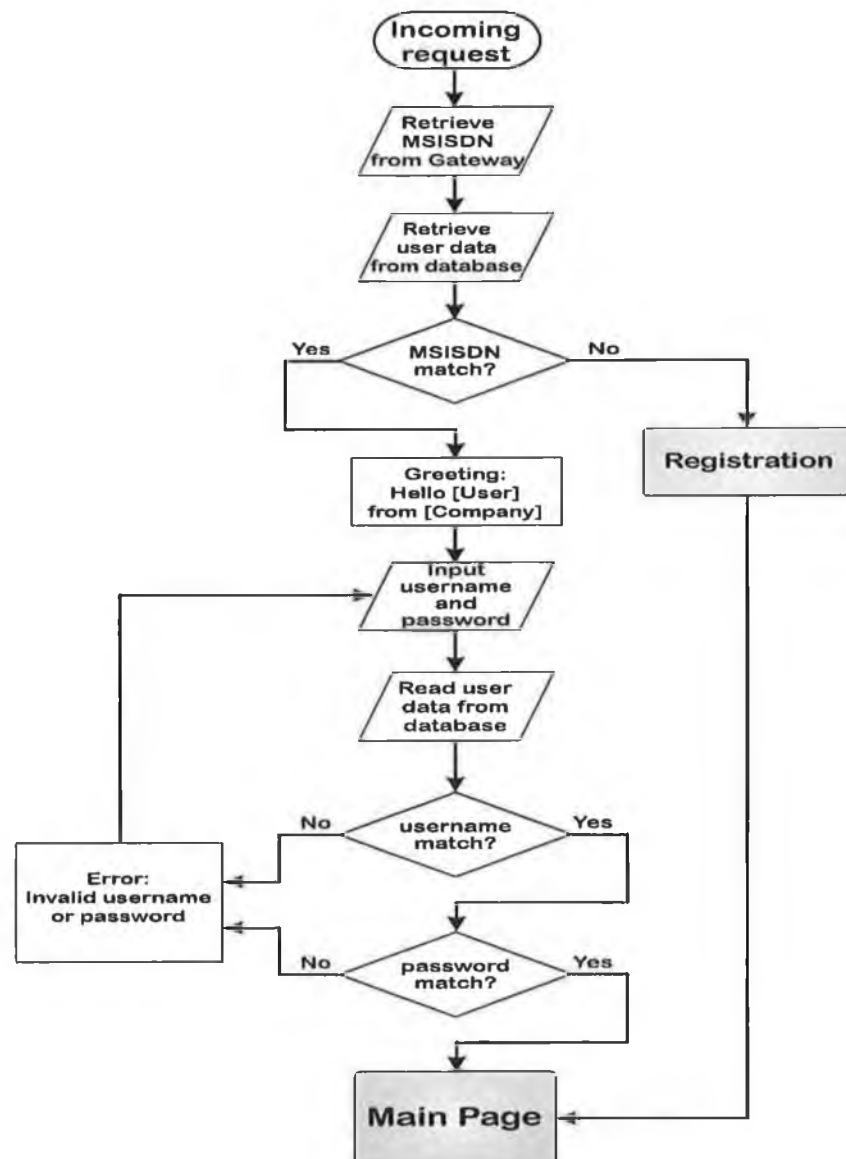


Figure 4.2 – The login and authentication data flow diagram

One of the many important features included in the prototype application is personalization. In order to achieve personalization it is necessary to identify the customer or the customer's client device. This can be achieved in two different ways.

One way is by authentication, which is by requesting the user to authenticate itself by typing in a username and optionally a password. The second approach is by employing a method by which the server automatically identifies the user's client device (the device that the customer is using to access the site i.e. mobile phone).

The prototype application incorporates both of these methods.

In order to provide means to implement the second method, the WAP architecture specifications include a gateway cookie request that provides the server with the ability to retrieve the client device's ID number (identifier). This number, called Mobile Station International Subscriber Directory Number (MSISDN) is actually composed of the country code (first 3 digits), the mobile operator identifier (2 digits) and the phone number (the number that uniquely identifies the mobile phone within that specific mobile operator network).

This request is addressed towards the gateway, which retrieves the requested information from the wireless network servers and passes it on to the WAP server.

The syntax of this request, placed within the VBscript code, is as follows:

```
Request.Cookies("User-Identity-Forward-msisdn")
```

The first part instructs the WAP (WEB) server to send a cookie request to the client. The second part is the part that the gateway will intercept and is the actual request for the client's ID number.

Within this prototype application, each user account in the user's table has an ID number used for indexing the records in that table. The main characteristic of this index numbers is the fact that the number has a unique value for each record in the table. Since the MSISDN number has this characteristic of being unique for each mobile device, it has been decided that this number will be used to index the table and to constitute the ID number for that specific customer.

Although the MSISDN request method is part of the WAP 1.0 Specifications, for privacy reasons mobile operators do not endorse it, and as a result it is not implemented or has been disabled in the gateway software. Since the prototype software relies on the use of this client ID number as the means to identify clients (client devices), a workaround has

been designed that will allow the retrieval of this ID number for testing purposes by requesting it from the user.

In the normal operation of the application, when a user enters the application (by accessing the `initiate.asp` file, which is the entry point), the application attempts to retrieve the MSISDN number from the gateway, as shown in the diagram. Once this number has been obtained, the application attempts to locate the account record within the database based on this number. If, based on the MSISDN number, the user account cannot be identified (there is no corresponding account record in the database to match the MSISDN number as ID), then the user is given the chance to register (create an account). If the number has a corresponding account, then the account data is retrieved and used for the next screen containing a personalised greeting, as seen in the diagram.

After the greeting screen, the user is directed to the login page where he is asked to input his credentials (username and password). After retrieving the credentials from the client, the application compares them against the ones stored in the database. If the result of the comparison is successful, the client is directed towards the main page of the application, otherwise the user is presented with an error message stating that the authentication has failed and allowed to retype the data. The card informing the user about the authentication failure does not provide details as to the specific cause of the failure (such as which value in the pair of values, the username or the password, was erroneous), but only states that one of them or maybe both are not valid. If the username and password submitted by the client match the ones retrieved by the application from the database, the user is directed to the main page.

An important feature of the application is that whenever an user that has already been authenticated by the application submits a request for a page (deck) within the application, he also submits along with the request his User ID number. This number has been established when the user has logged on and it will be passed on from one page (deck) to another using the client's requests as the means to identify the user. This method of identification has been used since WAP clients lack the ability to store session ID cookies, and as a consequence the connection is stateless, which means that the WEB server is not able to create and manage sessions and store session data (as it will be described in the following chapter, chapter 5).

Another feature of the application is that every user account has two distinct clearance levels. The basic level allows the user to search products and browse through the list of products in the database, as well as all the other features present in the application (like help, info, and news sections, and modify his settings in the settings page), but he is

forbidden to see detailed product information like availability or stock, and he is also not allowed to submit orders. An authorised user (a user with the second level of clearance) is allowed full functionality from the application.

The authorisation is manually assigned by the application administrator for each individual user, and every new user that has registered from within the application (by using the registration facility present in the application) is automatically assigned first level of clearance (he is not an authorised user). Once the application administrator has examined the newly created account and decided that the user is legitimate, he will grant level two clearance and promote the user to authorised user status, enabling him to see sensitive information and interact with the application by submitting orders.

If, in a user's authentication process, he unsuccessfully attempts to log on to the application (by inputting his credentials) for more than three times, this is interpreted as an attempt to hack (break into) the application, and the user will automatically be downgraded by the application to the first level of clearance (unauthorised user), so even if (after a certain number of attempts) the attacker manages to break into the application, he will not be able to gain access to any sensitive information or do any damage by submitting false orders.

b) Using the application

After it has been properly authenticated, the user is directed to the main page of the application. The following functional flow diagram (figure 4.3) illustrates the functionality provided by the system from the main page of the application. In this diagram, the following conventions have been made:

- Boxes having thick border represent listings or some other information presented to the user. When a list of items is presented, the user can select one item from the list by editing the list field. With WML, when a list located in a card is presented to the user, the list appears in the card as a list field (a sort of link to that list). The list's name displayed in the card is that of the item that has been selected in the list. For instance, if a list containing items *One*, *Two* and *Three* is present in a card, upon editing the list field (selecting the list field, pressing the *Options* button on the mobile device and selecting the *Edit* option) and selecting item *Two*, the card containing the list will display the list field that reads *Two*.
- Boxes with thin border assume some sort of direct user input. For instance, the *Edit Settings* page assumes input from the user in the sense that the user should input data in order to modify the account settings.

- Small rectangles overlapping a card's border at the point where an arrow leaves the card, suggest the name of the link that needs to be selected in order to obtain the functionality. The links can be selected either from the *Options* menu of the mobile device or directly from the card.

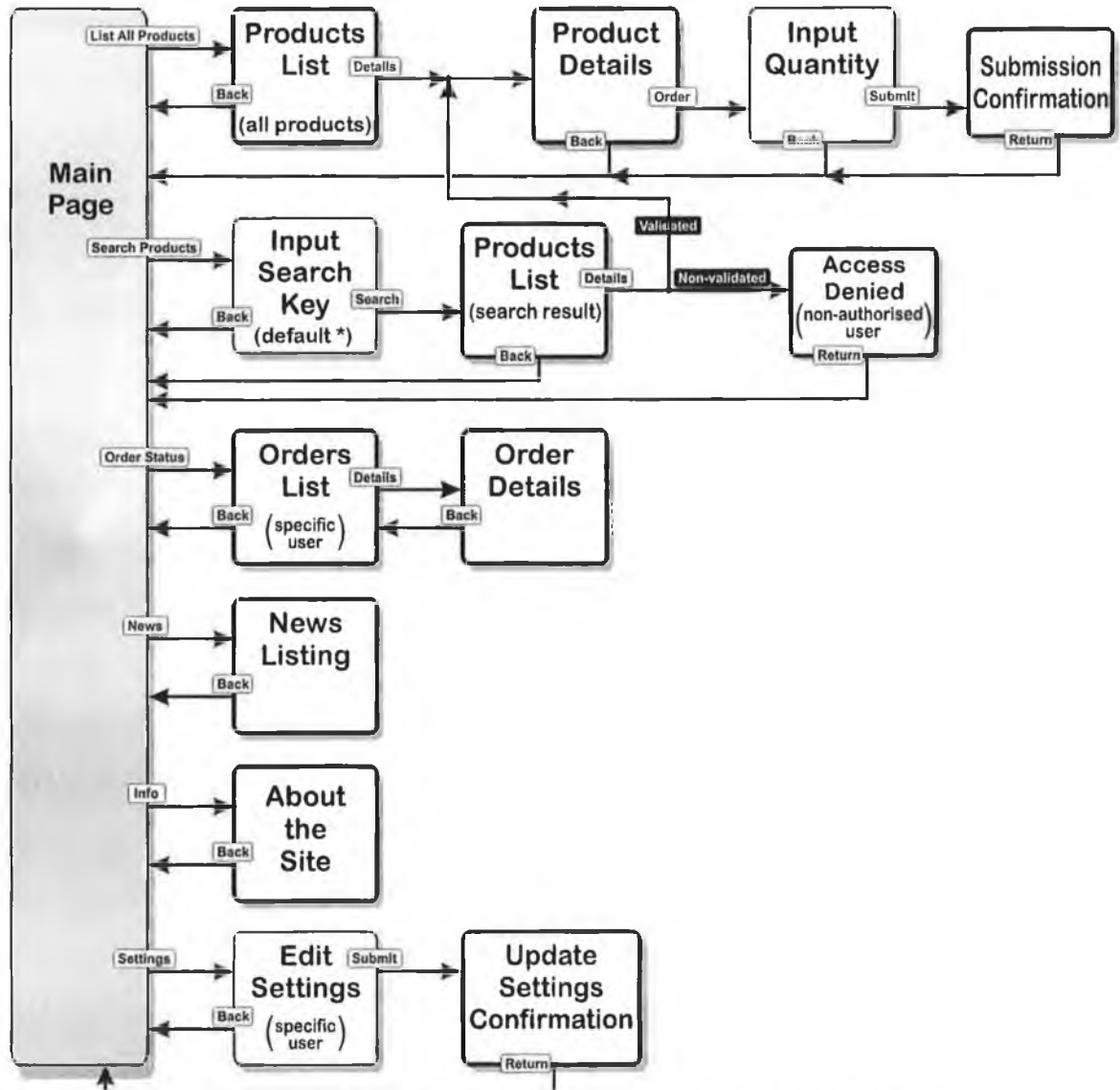


Figure 4.3 – Prototype application functional flow diagram

From the *Main Page* the user is presented with the following list of choices in the form of links:

- **List All Products** – displays a list of all the products in the database. Upon selecting this link, the user is presented with a list of products from which he can make a selection. After making the selection, activating the *Details* link will present the user with a page containing the details for the previously selected product. The user will then be allowed to the *Details* page only if he possesses the second level of clearance (authorised user), otherwise he will be directed to a page informing him that access to

details was denied. If access is granted to a validated user, upon examining the product details, the user has the opportunity of selecting the *Order* link, which enables him to input the quantity ordered in the next page. After inputting the appropriate value for the quantity and selecting the *Submit* link, the user will be presented with a submission confirmation page. In this page, the user is able to retrieve the order ID number and other details pertaining to the newly submitted order. A link is also present in this page directing the user back to the *Main Page* of the application.

- **Search Products** – enables the user to perform a search through the products contained in the database using matching criteria. The matching criteria for the search can be type using a combination of characters (letters or numbers). The default for the search criteria is the * symbol, which stands for “all products”. If a number of characters are entered, all the products containing in their name the respective combination will be displayed in a list presented in the next page. Upon selecting a product from the list and activating the *Details* link, the whole process continues in an identical fashion as presented in the previous part (described in *List all Products*).
- **Order Status** – enables the user to view a list of previously submitted orders. A user can only have access to orders that were submitted using his order ID number (were submitted using the same account). The list of orders is presented as a list of links, each link specifying an order, so that when the user selects the link corresponding to the order he is interested in that link will direct him to the next page where he will be able to read specific details about the selected order. From here, the user can either return to the previous page, containing the list of orders, or he can return directly to the main page of the application.
- **News** – selecting this link will direct the user to a page containing a listing of corporate news or other useful information posted for system users. The page contains a image (in the *wireless bitmap* format) as heading. The page also contains a link that will return the user to the main page.
- **Info** – this links to an info page presenting the user with information about the site’s owner (similar to the “About” page in other applications, brought up by the “About” item commonly found in the help menu of many software applications or even web sites).
- **Settings** – this will enable the user to change his account-related details, like username, e-mail address, company name or password. The actual values for the specified parameters are already present in the edit fields when the user enters the page, so that he can directly edit some of them or leave others unchanged. Upon submitting the

changes (using the *Submit* link), the user is notified of the successful outcome of the update process and directed back to the *Main Page* of the application.

The following section will present a detailed description of the application by detailing the screen flow of all the major functions present in the application.

4.3.2 The Screen Flow

The following section will describe the application in detail by presenting the screen flow for the main function included. The images used in this screen flow were generated using the Nokia 7110 mobile phone emulator included in the Nokia WAP toolkit. The functioning of the application is identical when using a real Nokia 7110 mobile phone. However, when using a different type of mobile phone not manufactured by the Nokia Corporation (for instance the Siemens M35i with a microbrowser produced by OpenWave), the screen layout might look different, but the functionality will remain unchanged (unless otherwise noted).

a) Login and Authentication

The file that constitutes the entry point for the application is called *initiate.asp*, and is automatically delivered by the WEB server upon receiving an unspecified file request for the folder hosting the WAP application, for instance www.domain.com/motto/ (where *motto* is the name of the application folder).

As already mentioned, each user that has an account in the database has also a unique ID number associated with his account, number that will identify him when placing orders. Upon entering the site the user will be required to enter the ID number.



Figure 4.4 - Inputting the user ID number

If the user has not registered with the system yet, the software will also provide the first available user ID number (the first consecutive number that hasn't been assigned to any

other user). This ID number is provided as a link and by selecting this link the registration process will be initiated.

The user has the choice to either type in the desired ID number by using the *Options*→*Edit* command (if he is already registered) or to select the suggested ID link (if he's a new user). In the first case, after going through the introductory screens, he will be greeted using the username and the company name stored in the database, and afterwards he will be taken to the *Login* page.



Figure 4.5 – The welcome screens

The *Login* page will prompt the user to input the username and the password for authentication purposes. The entered values will be matched against the ones stored in the database. If the match is successful the user will be linked to the *Main Page*, otherwise an error message will be displayed.

Upon receiving the error message the user will be presented with a choice to go back to the previous page (the *Login* page) and re-enter the password (or username) in case a typing mistake has been made.

Another useful feature is the masking of the password. This means that when the user is typing the password, upon typing each letter of the password it will be displayed as an asterisk (*) after a few seconds. This will ensure that the password will not be disclosed to a person standing next to the user.



Figure 4.6 – Authenticating with the system

In the *Login* page, the user will also be presented with a message stating that after three unsuccessful attempts the account will be invalidated. This feature adds an extra security aspect to the application.



Figure 4.7 – Successful authentication

In the second case, when the user is not registered, he will be prompted to type the suggested user ID number (or a ID number that is not already stored in the database). The suggested user ID number is a feature included for the purpose of providing the user with the first consecutive ID number that has not been used yet, so that the user will not have to guess an unused ID number. The suggested user ID number is obtained by running a query

against the database and processing the data, so that the lowest unused ID number found in the database is used by the system.



Figure 4.8 – Starting the registration process

After the introductory screens, the user will be prompted to go to the registration page in order to provide details that are going to be stored in the database as the user's profile.



Figure 4.9 - The registration process

The details provided by the user, which are stored into the database, are:

- User Name
- Email address
- Company on behalf the user is acting
- User's password

The user is also asked to re-enter the password in order to make sure it has been properly typed in.



Figure 4.10 – Editing user details

The data entered must be in the following format:

- The user name may have a maximum of 50 characters.
- The email address may be in the normal format (mailbox@provider.com) and can have up to 50 characters.
- The company name may have up to 50 characters.
- The password may have up to 20 characters and must match the re-entered password.

The user may enter the data by editing the respective fields (highlighting them and selecting *Edit* from the *Options* menu).



Figure 4.11 - Using the edit function

Special symbols like (@) or the dot (.) can be typed in on the Nokia phones by pressing the star (*) sign, located on the lower-left side of the keypad, and scrolling through the symbols using the cursor keys (or the roller key on Nokia 7110 mobile phone).



Figure 4.12 - Inputting special characters on the mobile device

The uppercase/lowercase change can be triggered by pressing the pound (#) key located on the lower right-hand side of the phone's keypad (Nokia 7110).

Another important feature becomes obvious when typing passwords. When a password is entered the last typed-in letter remains visible only a few seconds for the user to receive feedback of the input, after which it turns into a star (*) sign so that the password would not be disclosed easily.



Figure 4.13 – Editing password fields on the mobile device

After entering the password twice, the user can select the *Submit* link and by doing so all the entered data will be submitted to the WAP server. The server will conduct an analysis on the data, testing to see if the two entered passwords match. If they do not match, the user will be presented with an error message and prompted to revise the submitted data.



Figure 4.14 – Authentication failure

After a successful registration the user is presented with an acknowledgement screen.



Figure 4.15 – Successful authentication

b) Using the application

After a successful login or registration process, the user will be directed to the *Main Page* of the application. From here, the user has full access to all the functionality provided by the application.

The Main Menu

The main page of the application contains a picture symbolising the logo of the Motto project, and the main menu of the application. The logo actually consists of a black and white picture in the wireless bitmap (WBMP) format.



Figure 4.16 – The main application page

The *Home Page* (main page) of the application will present the user with the following headings:

- *List all Products* –lists all the products contained in the database
- *Search Products* –allows the user to conduct a search through the database and select a product
- *Order Status* – allows the user to examine the order status details for a specific order submitted
- *News* – allows the user to read the latest news about the company and product updates.
- *Info* – allows the user to get information about the WAP site and related development issues.
- *Settings* – allows the user to modify the stored profile information, like username, password, company name and email address
- *Help* – provides help about site’s functionality and features

The following sections will describe each of these items present in the main menu (*Home Page*) of the application and the functionality they provide.

The Settings page

By selecting the *Settings* link in the *Home Page*, the *Settings* page can be accessed. This page allows the user to change the data defining its user profile.

Figure 4.17 – The *Settings* form

This page has also a password check feature ensuring that no typing errors were made.



Figure 4.18 – Editing user settings

The News, Info and Help sections

The *News*, *Info* and *Help* sections can be accessed via the links in the main page (*Home Page*).

Figure 4.19 – The *News* card

In the *News* section the user will be presented with the latest company news. The user may scroll through all the screens containing the news and may return to the main page at any moment.

The amount of information stored in this section is limited by the average mobile phone's memory amount to only a few paragraphs but, in the future, as mobile device technology is evolving and more powerful mobile devices are produced, the amount of information displayed is likely to increase.

The *Info* section provides information related to the Motto WAP site. It also includes logos of the institutions involved in the project in the Wireless Bitmap (WBMP) format.



Figure 4.20 – The *Info* card

The *Help* section provides information about menu functions and other features.

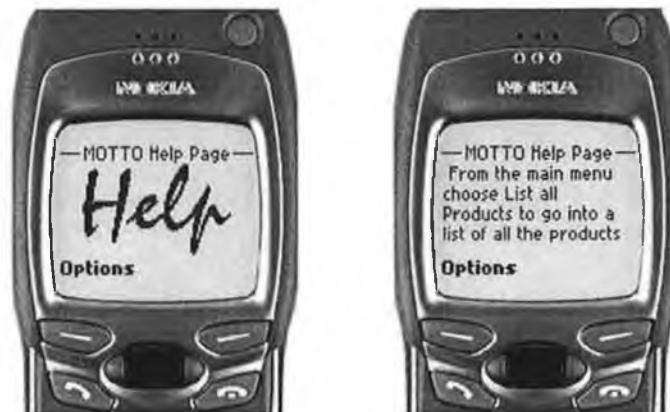


Figure 4.21 – The *Help* card

List all Products and Search Products sections

By selecting *List all Products* from the main page, the user will be presented with a listing of all the products contained in the database.



Figure 4.22 – The *List all products* page

By selecting the *Choose a product* link in the *Available products* page, the user will be presented with a product list. Moving the cursor over the desired product and pressing the *Select* soft key can make a selection.

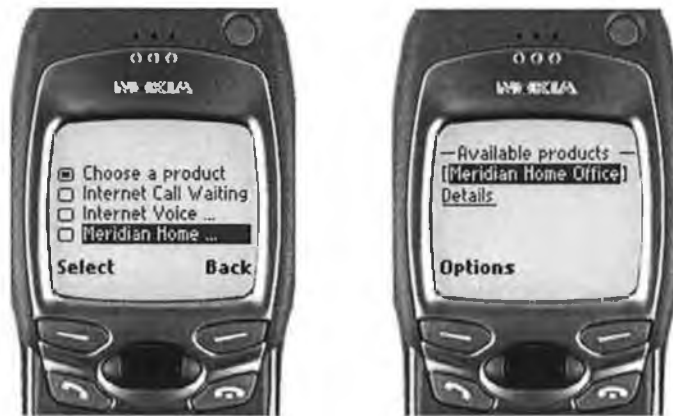


Figure 4.23 – Selecting a product from the products list

Upon selecting a product, the name of the link will change to reflect the product's name. The same functions are present in the *Products Search* page, but the user will first be presented with the opportunity of typing-in the search key for the product name.



Figure 4.24 – Using the search function

By default the search key will contain a star (*) sign. The star (*) sign is a wildcard and stands for “everything”, so by running the search using this search key, all the products in the database will match the condition and will consequently be selected and presented in the search result list. This is an extra precaution ensuring that errors will not occur even in case the user runs the search query without actually entering a search key.

In order to enter a specific search key, the user will first have to delete the star (*) symbol and then type in the letter or letters contained in the product name that needs to be looked for. The search key is not case sensitive. Upon entering the letters, the search key may be submitted and the results will display in the same form as the ones displayed for listing of all the products.



Figure 4.25 – Editing the search field



Figure 4.26 – Selecting a product from the search results list

By selecting *Details* link the following details will be displayed:

- Product Name
- Unit price
- Stock quantity for that specific product

The user will be presented with the choice to either go on to the order page or return back to the Main Page.



Figure 4.27 – The product details card

In the *Order* page, the user, having the details of the product displayed on-screen, will be prompted to enter the required quantity.



Figure 4.28 – The order form

The default value for the input quantity is 1. This will ensure that a reasonable value will be submitted if the user accidentally cycles through the page. If the quantity entered is less than or equal to zero or exceeds the stock, an error message will be displayed and the user will have the chance of returning to the order form and adjusting the quantity. Upon a successful submission of the required quantity, an acknowledgement page will be displayed, informing the user that the transaction has successfully been completed.

In the *Submission Confirmation* page, order details such as order number and submission date and time will also be supplied.



Figure 4.29 – The *Submission Confirmation* card

The Order Status section

The order status section is available by using the link located in the *Home Page*. It incorporates a personalization feature, which is the fact that it will only display the details of transactions submitted by the respective user requesting the *Order Status* details at that moment.

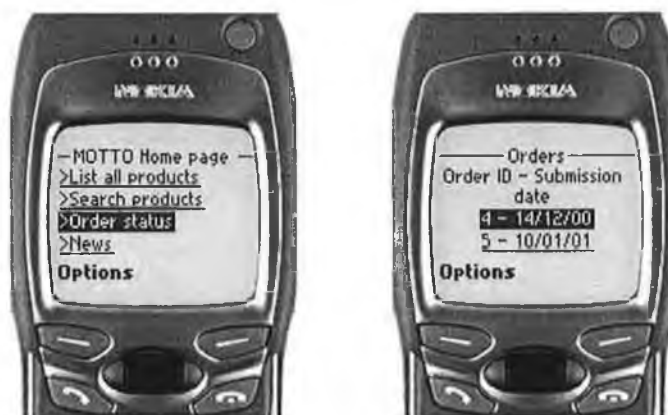


Figure 4.30 – The *Order Status* section

After a selection is made, the details of the selected order are displayed. These details are:

- The ID of the user who submitted the order

- The company the user acts on behalf of
- The ID number of the product ordered
- The product name
- The unit price
- The quantity ordered
- The submission date
- The submission time
- The status of the order
- The date when the product is due to be delivered (if available)



Figure 4.31 - The order details card

After reading the details, the user is presented with the option of returning to the main page (*Home Page*) of the application.

4.4 Coding details

This section underlines the main issues involved in designing and writing code for this prototype software, illustrating the chosen coding solutions with examples from the prototype software code.

The first part analyses some of the WML statements and syntax. The second part of the section offers details about working with databases using ADODB connections and Microsoft's database Jet engine, while the third section deals with VBScript programming details and parameters transfer using VBScript within ASP pages.

The last part describes the issues related to writing WML scripting code used in this application in order to provide functionality on the client-side.

4.4.1 The WML code

Valid WML content must always begin with the standard Document Type Definition (DTD) header, which is sometimes called the standard prologue.

The Document Type Definition supplies information to the gateway, information needed to encode (compile) the content from the original WML code (which is plain text code) into WMLC (WML compiled code, binary code), code that the client device will accept and interpret.

A standard DTD has the following format:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

The first section, delimited by the first pair of angled brackets, informs the gateway about the type of XML language that the content consists. As it was already mentioned in chapter 2, WML is an XML-compliant language. XML is a general definition for standard markup languages, which defines the rules and syntax of certain programming languages, mainly used for transferring content over internet and WANs (Wide Area Networks). In the case of WML, the structure of the language insures compatibility with XML version 1.0

The second part of the DTD supplies further details about the XML-compliant language included, by stating that this language is in fact WML. It also indicates that the complete type definition for the language is provided by the WAP forum, that the version number for the WML content is WML 1.1 and the character encoding is defined by the English ASCII character set. It also supplies the gateway with the location of the Document Type Definition on the Internet, which is the location of the file containing the complete definition for WML version 1.1 on the wapforum.org server.

After the standard prologue, the WML content must begin with the <WML> tag, indicating the beginning of the deck to be displayed by the WML browser.

The beginning of a card must be delimited by the following statement:

```
<card id="card_id" title="card_title">
```

The first parameter is the card's ID, which is an identifier used for referring the card from within the code. The second one is the title of the page, which will be displayed at the top of the client device's screen. The reason for using two distinct values is due to the fact

that, since the card ID is a variable, it must comply with a set of rules for defining the names of variables (like, for instance, they cannot contain spaces or special characters nor start with a number), which might not be convenient for designing card names.

For instance, if the title of the card is “Home Page”, although it will display properly as a card title, it will not be a good identifier for this card if used within the code, because the space character between the two words might lead to ambiguities (the interpreter might treat the two words as separate variables when in fact they form only one).

The next thing that is typically present in a card is the button assignments. A mobile device typically has two soft keys, one is the options key and the other one is the back key. While it is obvious what the back key does, the options key can be assigned with different functions. One of the functions that are typically assigned to the options key is navigation to the next card. By pressing this key when a link is selected, the user is offered the choice to follow that link. The selection can also be made by using the scroll keys that are sometimes present on mobile devices, for instance on the Nokia 6210 mobile phone.

If the key is pressed when an editable field is selected, the user is presented with the opportunity to edit that field. On some mobile devices, the same functionality is also achieved by pressing the roller key, for instance on Nokia 7110. The roller key on this mobile device has a double functionality, allowing the user to scroll through a list of options or links by rotating the roller, and also enables him to select an item from that list by pressing the roller key.

Functions can be assigned to the options button by using the following instruction:

```
<do type="options" label="Next >">  
  <go href="#Welcome"/>  
</do>
```

Upon pressing the options key, the browser will be instructed to present the option of navigating to the next card (which, in our case, is the card having the ID “Welcome”, located in the same deck). The label parameter contains the text label to be displayed on the browser’s screen just above the options key on some phone models, or to be presented as an item in the list of options that is brought up when the options key is pressed on others.

The second line contains the reference for the card that is going to be linked (in this case it is actually another card located in the same deck), while the third line is the ending tag for the do statement.

Another example is the assignment of the back button on the mobile device:

```
<do type="prev" label="Back">
  <go href="initiate.asp#card1"/>
</do>
```

The type parameter in the first line indicates that the key to which the function is to be assigned to is the back key, and the label parameter indicates the text to be displayed next to this key.

WML uses paragraphs in a way similar to HTML. The following example illustrates the use of a paragraph in this prototype application:

```
<p align="center">
  <br/>
  Sponsored by
  <br/>
  <big>
  Nortel Networks
  </big>
</p>
```

This will produce center-aligned text on the display of the mobile device. The paragraph tag, like any other tag, must be properly concluded by the use of the end tag (</p>). In addition, every card tag must be terminated with the </card> tag, at the end of the card. The end of the deck is marked by the use of the corresponding end-WML (</wml>) tag, which also marks the end of the WML content.

4.4.2 Working with databases using ADODB

In order to transfer data to and from the database, a connection must be established. In this prototype application the connection is achieved by using Microsoft's Jet engine.

The following lines are the database connection string used from within the VBscript code within the application:

```
Dim conn
Set conn = Server.CreateObject("ADODB.Connection")
conn.open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("prototyp.mdb")
```

The first line dimensions (creates) a variable called `conn`, variable that in the second line of code will inherit an object structure (connection object of type ADODB). The third

line links this connection object previously created to the Microsoft Jet database engine. Also in this third line, the data source is indicated by pointing towards the location of the database file, which in this case is called `prototyp.mdb` and is located in the same folder as the ASP files. Once a connection is opened, data can be submitted to or retrieved from the database by using SQL queries. Each time a request is made towards the Web server, the server processes the ASP page, creates a connection and manipulates the data in the database. If the number of requests is very large (a big number of customers access the application), the server will create a big number of connection objects leading to huge amounts of memory being used for creating these connections and a considerable decrease in speed of delivery of the content by the web server. If the amount of requests reaches a certain point and the number of incoming requests exceeds the number of requests processed (dismissed), the server will no longer be able to handle the requests and will eventually crash, failing to deliver the content and generating fatal system errors.

A good programming practice is to close the connections once they are no longer needed. But for the cases where the application fails to complete the execution of a page and close all its connections, in order to prevent clogging that lead to crashes, servers are designed to automatically drop (discard) old connections after a period of time ranging from a few seconds up to 10-20 minutes. This interval can be specified by using a VBscript statement. Nevertheless, it is recommended that in designing scripting code, the programmer should pay close attention to managing database connections and specifically close all connections once they are no longer needed.

In order to achieve that, the connection with the database should be closed and the connection object destroyed:

```
conn.close  
Set conn = Nothing
```

Each time an ASP file deals with data from a database, a connection should be opened at the beginning of the file and closed at the end of it. Since a site is usually composed of a large number of ASP files, if the database type, configuration or access method is changed, modifications must be made to all ASP files to reflect the changes. This may lead to a big amount of effort and opens the possibility of making mistakes in modifying all the connection strings in all the ASP files.

In order to prevent that, the connection string may be isolated into a separate file which will be subsequently included at the beginning of all the ASP files using the include directive:

```
<!--#include file="dbcon.asp" -->
```

This way, if the connection details change, the modification must only be performed on the connection string within the dbcon.asp file, while the connection object being used within the ASP pages (conn) and the name of the included file (dbcon.asp) remains the same.

A similar include directive must be placed at the end of each file that requires a database connection, pointing towards an ASP file containing the string needed to close the connection.

```
<!--#include file="discon.asp" -->
```

Once the connection has been opened using the connection string, data can be retrieved from the database using SQL queries. An example of an SQL query that retrieves user names and company names from the *Users* table based on the user ID number is presented in the following lines:

```
SQLquery = "SELECT username, company FROM users WHERE user_id = " &  
userid  
Set ReqUser = conn.Execute(SQLquery)
```

In this example, the query to be submitted to the search engine is first stored in the variable called `SQLquery` after which, on the second line, the connection object is called with this variable as a parameter. The result of the query will be stored in the variable called `ReqUser`, which will automatically inherit the recordset structure (an object with the structure of recordset type). Since this variable becomes an object, it must eventually be destroyed when it is no longer needed.

Every field of every record stored in this variable can be accessed in the following way:

```
ReqUser("username") - is the field called username from the first  
record  
ReqUser("company") - is the field called company  
ReqUser.MoveNext - incrementing the index to the next record
```

The following is a piece of code that uses this way of incrementing the record index to generate a list of products stored in the database:

```

<p align="left">
<select name="prodid">
  <option value='0'>Choose a product</option>
<% Do While Not SearchRes.EOF
  Response.Write "<option value='" & SearchRes("prod_id") & "'>" &
SearchRes("prod_name") & "</option>" & vbcrLf
  SearchRes.MoveNext
Loop %>
</select>

```

This portion of code is in fact composed of VBscript routines embedded in the WML code. The VBscript routines (delimited by the <% and %> symbols) generate the option fields required by the select instruction. While the *Do-While* loop runs, the fields in the SearchRes variable will be inserted into the option parameters, and with every loop, the record index is incremented by one exposing the next record in the variable. Each time the *DoWhile* loops, the SearchRes variable is tested to determine whether it has reached the end. This is achieved by evaluating the “Not SearchRes.EOF” expression, which is a VBscript function specifically designed to test for the end of a data stream contained within structured variables. This portion of code is used to generate the products list in the prototype application.

The following section lists the code used to store the submitted orders into the database:

```

SQLquery = "INSERT INTO orders VALUES (" & corder & ", " & userid & ",
'" & company & "', " & prodid & ", " & qty & ", '" & date & "', '" &
hour(time) & ":" & minute(time) & "', 'Submitted', 'undetermined');"
conn.execute(SQLquery)

```

This piece of code works in a similar fashion as the previous query that reads data from the database, except for the fact that in this case data is written into the database. The SQLquery variable is constructed by appending other variable’s data into it.

The following section lists the code used to retrieve the orders from the database for a specific user indicated by the user ID number:

```

Set OrdRes = Server.CreateObject("ADODB.Recordset")
OrdRes.CursorType = adOpenStatic

SQLquery = "SELECT * FROM orders WHERE orders.user_id = " & userid &
";"

```



```
Set OrdRes = conn.Execute(SQLquery)
```

By default, when data retrieved from a database is placed into a variable, the server automatically assigns an ADO DB recordset structure to that variable. This default recordset structure has a predefined set of properties. If, for some reason, the programmer wishes to alter those properties, the variable must be defined beforehand and the properties needed must be assigned at that time, so that when the variable is used for storing the data, it's custom structure is already set in place. In this example, this is achieved by the first two lines. The third line constructs the query and the fourth runs it using the existing connection (`conn`) and stores the result in the `OrdRes` variable previously defined.

After using the variable by retrieving the data from it, this variable must be destroyed, which is achieved by the following two lines:

```
<%  
OrdRes.Close  
Set OrdRes = Nothing  
%>
```

4.4.3 The VBscript code

Within the prototype software, VBscript is used to achieve all sorts of programming tasks required to manipulate the WML code in order to create active content. One of these tasks is to transmit different parameters from the server to the client device and vice versa and at the same time to pass on data from one ASP file to another.

The method used to transfer these parameters is the get method. As opposed to the post method, which instructs the server to “pull” (request) the data from the device, the get method rather sends these parameters to the server along with the request. Both methods are commonly used on the Internet, and are defined in the WWW protocol specifications. The problem is that the existing gateways, because of the way the WAP protocol is designed, do not support the post method.

As was already stated, the transfer takes place from the client device to the web server and, consequently, the posting of the parameters must be achieved from the WML code, since these clients only support WML as recognized language.

The following piece of WML code will submit a certain number of parameters to the server and across two ASP files (`index.asp` and `register.asp`) using the get method:

Index.asp:

```
<do type="accept" label="Submit">
  <go href="register.asp" method="get">
    <postfield name="userid" value="$(userid)"/>
    <postfield name="username" value="$(username)"/>
    <postfield name="email" value="$(email)"/>
    <postfield name="company" value="$(company)"/>
    <postfield name="password" value="$(password)"/>
    <postfield name="rpassword" value="$(rpassword)"/>
  </go>
</do>
```

register.asp:

```
Dim userid, username, email, company, password, rpassword

userid = Request.QueryString("userid")
username = Request.QueryString("username")
email = Request.QueryString("email")
company = Request.QueryString("company")
password = Request.QueryString("password")
rpassword = Request.QueryString("rpassword")
```

In the first section (index.asp), WML is used to submit the data, by using the postfield instruction. The second part is included in the VBscript code in the ASP file and uses the Request.QueryString instruction (method) for retrieving the variable values.

The drawback of this method is that, since the transmitted parameters are actually appended in plain text format at the end of the address of the requested page, when a PC-based WAP emulator is used, the user actually sees these parameters that are transmitted to the server and their format in the address bar, along with the address of the requested page. The following request line is an example:

```
http://pilot.nuigalway.ie/motto/login.asp?userid=3&username=Eamon&password=ew
```

Besides exposing sensitive information like passwords and other sensitive data, this method of transfer also reveals sensitive information about the structure of the site hosted on the WAP server, which might lead to unwanted security breaches. In order to prevent this from happening, a secure method of encryption for connecting to the server will have to be developed in the future.

4.4.4 The WMLscript code

WMLscript code has been used in the prototype application to achieve client-side functionality.

WMLscript code is fairly similar to JavaScript, although there are some differences in syntax and programming object model. The major difference is that while the JavaScript code is interpreted on the client side by the browser, WMLscript code is encoded by the gateway and is executed by the client device in binary (compiled) form. The encoding process takes place in the gateway's encoder, and produces, as a result, binary (machine instructions) code, which is the code processed directly by the mobile device's microprocessor.

The WMLscript code that the web server delivers when a request is made is in plain text, but with a specific MIME type attached to it so that the gateway will recognize it as scripting language and process it accordingly.

The WML functions used in the prototype application accomplish different tasks like verifying user input for illegal values, conditional navigation between cards and decks and password verification. By accomplishing these simple tasks, the application increases in speed. This is achieved by eliminating the time that the user would have to wait while a new request is made to the server (through the slow connection of the mobile device), the time it would take to process the request by the server and the trip back to the device. Instead of dealing with all those latencies, by the use of WMLscript routines the task is accomplished almost instantaneously by the mobile device. Another advantage resulted from using WMLscript is that eliminating the additional trips to the server, the traffic through the network will be reduced, thus improving the response time in which content will be delivered when other devices request useful content from servers.

The WMLscript code is structured into functions, called from within the WML code. According to WAP specifications, the parameters for these functions can either be passed directly to the script function when the call is made, or they can be retrieved by the script program from the WML browser application when the scripting code is executed.

In practice, the situation is different. On some mobile phones (like, for instance, Nokia 6210), if the WML code attempts to submit to the script function more than one parameter, the execution of the program will fail and the user will be presented with a message stating that the script could not be accessed. The workaround (if more than one parameter is to be passed to the script function) is to actually request the parameter from within the script file. This procedure works flawlessly every time.

The following piece of code is an example of how the parameters can be passed to the script function by using this method.

The WML code calling the function:

```
<anchor title="Submit"> Submit form
  <go href="scripts.wmls#qty()" />
</anchor>
```

The WMLscript function:

```
extern function qty()
{
var sq = WMLBrowser.getVar("qty");
var stock = WMLBrowser.getVar("stock");
var q;
var s;
q = Lang.parseInt(sq);
s = Lang.parseInt(stock);
if ((q < s) && (q >= 0))
{WMLBrowser.go("#Submit");}
else
{WMLBrowser.go("#Error");}
}
```

The first part of this function requests the two variables from the WML browser, after which it converts the data stored into those variables from text to integer numbers. It then performs a comparison based on the two numbers and, according to the result, it either jumps to the Submit card, which means that the quantity ordered is in the proper range (smaller than the stock and at the same time greater than zero), or to the Error card, which will display an error message and will give the user a chance to re-enter the value.

Another script that uses more than one variable is the one that compares two passwords entered by the user to verify that they were entered properly:

```
extern function compare()
{
var pass = WMLBrowser.getVar("password");
var rpass = WMLBrowser.getVar("rpassword");
if (pass == rpass)
{WMLBrowser.go("#Submit");}
else
```

```
{WMLBrowser.go("#Settings");}
}
```

The two passwords are retrieved from the WML browser by using the `WMLBrowser.getVar()` instruction, after which they are compared to determine if they match. If the result is true, the execution will jump to the Submit page in the WML code, otherwise the execution will be resumed in the previous (Settings) page, where the user can re-enter appropriate values for the two passwords.

The following script function verifies that the user is validated before supplying details about products in the database.

The WML code:

```
<card id="chkid" title=" Please wait... ">
  <onevent type="ontimer">
    <go href="scripts.wmls#id('<%=User("validation")%>')"/>
  </onevent>
  <timer value="1"/>
  <p align="center">
    <br/>
    <br/>
    Loading...
  <br/>
</p>
```

The WMLscript code:

```
extern function id(sv)
{
  if (sv == "True")
  {WMLBrowser.go("#Order");}
  else
  {WMLBrowser.go("#NoOrder");}
}
```

When making the request for the `id` function, the parameter (`sv`) is sent along with the request. If `sv` (which actually stores the validation data retrieved from the database) has the value of true, the user will be allowed to see detailed info about the product, otherwise he will be presented with an error message.

4.5 Summary

This chapter provided a detailed description of the prototype wireless application developed during the course of this project. The first part of this chapter provided a description of the application structure by presenting the Context Data Flow Diagram. The Functional Data Flow Diagram was presented in the second part to describe both the authentication process and the functionality provided by the application itself once the authentication took place and the user has logged-on to the application. The subsequent part provided a suggestive description by presenting the screen flow for the main functions of the application as seen from a user's perspective.

The last section of this chapter outlined the coding details for the application by providing a brief overview of some relevant portions of code that were either typically used in the application for performing common tasks or they perform the most complicated functions within the application. It covered details about the WML, VBscript and the WMLscript code used, and also the use of ADODB in the process of achieving database connectivity.

The next chapter will describe the application developed for AMT Ireland, which actually consist not only of a WAP application but also a Web application.

CHAPTER 5

Implementation of a WEB and Wireless Application for an SME

- 5.1 Introduction**
- 5.2 Company overview**
- 5.3 User specifications**
- 5.4 Functional requirements**
- 5.5 The content type-based redirection**
- 5.6 Web application design and implementation**
- 5.7 Mobile implementation**
- 5.8 Summary**

5.1 Introduction

The successful development of the prototype application presented in the previous chapter (chapter four) attempted to explore all the possibilities offered by the use of cutting edge mobile technology. This was done in order to analyse what functionality was useful and could be employed to create a competitive application adapted to the needs of companies in the industry sector and to achieve a better understanding of the mobile technology and the advantages it offers.

As such, it was decided to develop an application that will address the needs of an SME. In order to achieve this goal, a real-life implementation for a customer care application was developed. The purpose of this implementation was to provide an SME with the means to address its customer relationship management needs by providing customers with value-added services. This was achieved by designing an application that employs cutting-edge mobile technologies to enable customers to access information using wireless devices.

This chapter first describes and analyses the issues related to the implementation environment in which the application had to be developed, and the user requirements that will provide functionality specifically trimmed to meet the demands of this specific area. It then moves on to the analysis and design of the application. This is the decision-making

process that actually shaped the application to meet the technical demands and user requirements.

The next section provides a description of the application, the functionality it offers and all the features included within the application. This section is divided into three parts, the first one describing the Data Management System incorporated into the application for the purpose of managing the database (the administrator view), the second part describes the WEB side of the application, designed to provide a better customer interaction, and the third part describes the wireless implementation, which is the part that allows customers to access the application using wireless (mobile) devices.

5.2 Company overview

The application was developed for a branch of AMT (Advanced Manufacturing Technologies) Ireland. This company provides its customers with expertise in the areas of business process analysis, product design and development, and manufacturing facilities. In addition, AMT also provides its customers with training in the areas Business and Process Improvement and Electronics Manufacturing. It also collaborates closely with organisations like Enterprise Ireland and a number of Irish universities such as UCD (University College, Dublin), UCC (University College, Cork), NUIG (National University of Ireland Galway) and UL (University of Limerick).

Although the company is rather large and it might not fall under the SME category, it actually comprises of more small-sized departments that act in an independent fashion up to a point, making their own activity-related decisions and even some investments decisions. The small branch of AMT, called the Materials Research Lab, for which the application was developed functions independently up to a certain point and can be regarded as a small SME conducting its activity in the commercial environment as any other company of its size. It falls under the category of small enterprise by having five employees and working with about one hundred customers (companies), averaging in size from small enterprises to large organisations such as Motorola, Dell, Analog Devices, Matrox, APC, 3Com Technologies and Loctite.

This particular department for which the application was developed specializes in the area of electronic components assembly by providing services like research and development for electronic products assembly, PCB prototyping, low volume PCB assembly, PCB rework and repair, and assembly reliability analysis.

The requirements for a Customer Relationship Management type of application that they needed implemented include providing customers with an order entry system,

enabling them to submit Requests for Quotations, purchase orders and obtain information about the status of their orders. It achieves this by using the available technologies such as the Internet and WAP-enabled wireless devices.

Further details about AMT and the analysis of the impact of this application on its CRM business process can be found in other MOTTO project work [Butler 2002].

In order to better understand the functional specifications for this application, a detailed analysis of the user requirements must be performed. This analysis is detailed in the following section.

5.3 User specifications

From a user's perspective, the application needs to include the following functions, enabling users to:

1. submit Requests for Quotations (RFQs)
2. submit orders.
3. obtain information regarding the status of the submitted orders (Order Status)

In addition to that, the application must also include the following features:

1. it must be a secure application
2. the application must be fast both in terms of transfer speeds and processing power
3. the application must be reliable and the information obtained must be up-to-date at all times
4. the information must be well organised and the application easy to use

While the first three requirements are specific to this particular implementation, the last 4 features are generally required from an E-commerce application, whether the application involves the use of the Internet and the Web by using a web browser, or more advanced technologies such as mobile (wireless) devices.

1. The first function enables the user to submit requests in order to obtain quotations in a timely manner. The application operator will receive these requests, process them and submit the quotation back to the user by either using electronic means like email or SMS or by using more conventional means such as faxing, phone or even hard copies if necessary. The application cannot provide useful means for transferring this kind of information because:

- The time lapsed from the moment when the initial request was received and the moment when the quotation is ready to be submitted cannot be quantified. Indeed, due to the specifics of this activity it is almost impossible to quantify the services provided so that a standard procedure is set in place for evaluating costs and delivery dates. Provided that most of the orders consist of services implying a succession of operations that first need to be thoroughly analysed and a manufacturing procedure set in place, it is impossible to predict the time needed for this analysis.
- Furthermore, the data resulted from this analysis and the quotation itself is hard (if not impossible) to be somehow integrated into a standard data structure suitable for storage into a simple database structure so that it will be subsequently presented to the user especially if using a mobile device.

For these reasons it is more efficient to provide the user with the data by using the method of its choice, and the details needed for submitting the quotations (like email address, fax/phone/mobile number or street address) can easily be stored into application's database.

2. The second function gives the user the opportunity to submit orders based on the information received within the quotation. The system must store the order submission information and present it to the application operator so that the latter will be able to assign a project number for that specific project.

3. The third function will provide the user with the ability to receive status updates for the ongoing projects including but not limited to the status of the order (including the status of the current operation being performed), the due date, the assigned project number, the description for the project and the delivery address.

As for application's features:

1. The first feature, which is a generic requirement for most E-commerce applications, requires that the application be secure. This means that the application must employ some sort of authentication mechanism in order to provide confidentiality for the data being transferred and only the users that are entitled to see certain details can do so. Since the most common type of authentication encountered on the web (which also proves to be one of the most reliable) is the username/password type of authentication, this method can be implemented into the application.

2. The second feature refers equally to the transfer speed of the content (in terms of both quantity of bandwidth and latency) and the processing time both on the server-side and client side. It has been said that as much as 50 percent of E-commerce transactions

might be lost because of long delays in accessing web sites and the lack of ability from the content providers to display the content in a timely fashion. [Dennis 2001]

As it was already mentioned, these delays are due to mainly four reasons, listed here in the order of their importance:

- Server request processing delays; these delays usually manifest themselves due to both heavy traffic on the servers and bad script design for applications which leads to long script processing times. In most of the cases, web servers exhibit a mechanism that allows dropping a request if the processing time for that request exceeds a specified amount of time. Although this mechanism causes the server to discard a number of requests, it is a necessary drawback since otherwise the server would accept and process all the incoming requests and eventually, due to the inability to finalize them, will grind to a halt. Sometimes (in rare cases) the delays are due to the fact that the information requested needs to be retrieved from third-party servers or in some cases dedicated database servers, thus introducing additional latencies in the process of content retrieval (this is especially true for some search engines).
- Delays due to low bandwidth; this is especially critical for mobile applications, where the rate of data transmission is fairly low and the transmission itself is unreliable. Besides low transmission rates, additional delays are introduced due to the fact that data packets bounce through the WAP gateways so that content is compiled and encrypted according to the WAP/WML specifications.
- Large amounts of data to be transferred at once; this is due to bad application design. The content supplied by the application should be organised into reasonably sized web pages (WML cards) so that even under heavy server loads conditions the content should reach its destination within a reasonable amount of time.
- Long processing times on the client-side; this is mainly due to poorly designed scripts on the client-side which either cause long delays in displaying the content or they simply crash before finalizing the processing (for instance web page applets that are unable to run due to missing java classes and the inability to retrieve those classes from the server, or JavaScript processing errors that make the script crash, preventing the page from displaying properly and sometimes report errors).

3. The third feature is the reliability of the application and the freshness and accuracy of the data. The reliability of the application relates mainly to the fact that the processing of scripts should take into consideration all the combinations of user data input so that incorrect or bad-formatted data retrieved from the client would not crash the application or leave it in an undefined state. One important aspect is also error control, which ensures that

the user has supplied the data in a consistent manner (has filled all the required fields with appropriate values).

Regarding the freshness and accuracy of the information transferred, this requires that the data retrieved from the user should be committed to the database as soon as it was checked for consistency and validated, so that in the event of an application crash or the user suddenly deciding to leave the application for some reason, the data would not get lost. It also implies that the data submitted to the user is retrieved directly from the database bypassing caches or other means of local storage that will cause the data to become stale.

4. The last issue relates to the usability of the application. It outlines the fact that the application should be organised in a rational manner so that the user can locate the information it needs with the minimum amount of effort (number of requests by following links or submitting queries) possible. It also emphasizes the fact that page design should be clear and straightforward in order to present the information requested by the user [Lyons 2001].

In addition to these requirements, additional features for the application are more or less implied. One of these features is the fact that all the data pertaining to the orders, users list and other types of information related to the normal operation of the application should be stored into a relational database. This will provide an optimal way of organizing the data and will also insure easy and effective retrieval, manipulation and storage of the data by the application.

Another implied feature is the need for some mechanism allowing application management. This means that an operator must be able to access any piece of information from the database, pertaining to any user or order, and eventually be able to make modifications for the purpose of managing users accounts, updating order status, assigning project numbers etc. The most obvious way to achieve this is by simply editing the database file by using the program used to generate the database. For reasons that will be subsequently presented, this is not possible because:

- first of all physical access to the database file cannot be allowed since access to the database from the application's point of view is achieved using the Microsoft Jet engine. When the Microsoft Jet Engine is accessing the database, it automatically locks it, so that other type of access using, for instance, MS Access, is denied. It is also not always possible to physically access the database file since the operator is not usually on the same intranet with the web server running the application.

- secondly because employees in an SME are not usually IT experts specialised in running complex applications like database management systems. And even if they were, it will still be tedious and counter-productive to manually manage the database and ensure that every bit of data that is edited into the it is in accordance with the database structure, is in the valid format and doesn't corrupt its structure. For these reasons there must be some fail-safe mechanism that will guide operator's actions so that he is not allowed to perform operations that will generate erroneous data or even damage the database.

Keeping this in mind we will find more arguments that will eventually lead to a solution.

It is a well-known fact that mobile devices lack one very important usability feature: they are not suitable for inputting large amounts of data [Groves 2000]. Indeed, due to the fact that mobile phone keyboards are small, they do not allow direct typing for letters, but instead the user has to cycle through the letters available on that key. For instance, if a user needs to type letter s using a mobile phone keyboard, he will have to press four times the "7" key in order to cycle through letters P -> Q -> R -> S. This, combined with the fact that the keyboard is so small and usually has to be operated with a single hand, makes text input using a mobile phone next to impossible. Although applications like T9 predictive text input have been developed that run on mobiles and help the input process, it is still a very difficult task to input large quantities of text.

Having said that it comes as a logical conclusion that the customers using this application will be reluctant to submit orders necessitating a tremendous amount of input for addresses, order descriptions, due dates or purchase order numbers. For this reason it has been decided to use the mobile (wireless) application only for delivering information.

Since the orders still have to be placed, it has also been decided to develop a web application interfacing to the same database. Indeed, since hard-line internet access is much more common at this time than wireless internet access, it is reasonable to assume that a web application is more likely to be used by customers interacting with such an application, in a real life implementation. This application will allow users to perform all the necessary operations like submitting or activating orders, managing their accounts and also obtaining status information pertaining to their orders.

In this case, the implementation conditions dictate that the mobile application should only come as an extension to a web customer care application, an extension enabling customers to obtain up-to-date information while they are away from their office, and at any time.

The next section will analyse the technical design issues involved in the development of the two combined applications.

5.4 Functional requirements

As discussed in the previous section, the data handled by the application will be directed or will originate from a database. The design and functioning of the application is described by the 3-tier application concept detailed in chapter three - Application Specifications, section 3.4. The common layer for the two applications (as described in the 3-tier application concept) is the database layer. The other two layers are designed specifically for each application, with the third layer (the presentation layer) designed with the use of HTML for the WEB application and with the use of WML for the mobile application.

As already discussed in chapter three, the application needs to comply to the requirements of an SME, that is low implementation costs, easy to implement and easy to maintain. The necessity for low implementation costs comes from the fact that most of the SMEs are still reluctant to make big investments in technologies enabling them to infiltrate into the E-business are, and very few are willing to support the cost usually associated with this. Due to this reason the application was designed to run on a web server installed on a Windows NT 4.0 Workstation platform. The application runs on the Microsoft Personal Web Server (PWS) 4.0 included with Microsoft Windows NT Option Pack 4, a software package that can be freely downloaded from Microsoft's web site (www.microsoft.com). In legal terms, the license for the Option Pack 4 is free of charge provided that the Windows NT 4.0 platform on which the software applications contained within the package will run. The license for the use of the PWS included in the Option Pack 4 package grants the right to have 10 concurrent connections to the web server, which, for personal use is usually enough. However, for a corporation, even if it is an SME, it might prove to be too restrictive and inhibit the normal functioning of a web or wireless application especially if that application is designed so that the Web or WAP pages make more than one request for each page/card in order to get the necessary content (i.e. a Web page that is retrieving more pictures at a time, pictures needed to be displayed on that page). One (temporary) solution would be to set the maximum number of concurrent connections to the maximum allowed by the web server configuration program, which is 40 (hard-wired into the program). However, this constitutes a violation of the license agreement, and as such the program states it when the setting is attempted. The problem is similar even when using other types of windows platforms, such as windows 95/98 or

windows 2000 professional (which comes with the personal web server already included in the installation kit). The only way to overcome this limitation is by using either a Windows NT Server operating system or a Windows 2000 Server platform which are both capable of running Microsoft Internet Information Server (IIS) 5.0 but also cost about four or five times more than the workstation version. Either way, the design and implementation of the application is almost identical in any of the presented cases.

As already mentioned, the design of the application includes two forms of presentation for data: a WEB presentation and a WAP. For this reason two applications were developed that connect to the same database using the ActiveX Data Objects (ADO) mechanism already described in chapter 3, which in turn relies on the Microsoft Jet Engine to perform the database access. This mechanism enables the application to open database connections from within ASP files running scripting code, and submit SQL queries towards the database for the purpose of retrieving or submitting information from/into the database.

Also, one of the reasons that determined the choice for the database is the same reason stated previously, that is low implementation costs. The database is, as in the prototype software discussed in chapter 4, a Microsoft Access database. The design of the database is achieved by using Microsoft Access, the database management application included with the Microsoft Office suite. Although the design and development of the application requires the use of Microsoft Access program, its functioning does not, since database access needed for the normal operation of the web or wireless applications is achieved using ADO and the Microsoft Jet Engine.

Both applications use a similar authentication method, based on username/password pair of values that are stored into the database. The username/password pairs are specific to each user and are stored into the database together with other relevant user information such as the full name of the customer, the company on behalf he acts, both invoice and delivery addresses, contact details specific for that user (email, fax number, phone number and mobile phone number).

The MS Access relational database structure actually consists of two database files. The first file is split into 2 tables, the first one containing details about users and the second one details about orders. The second database file consists only of one table, containing details about user connections (logged connection events).

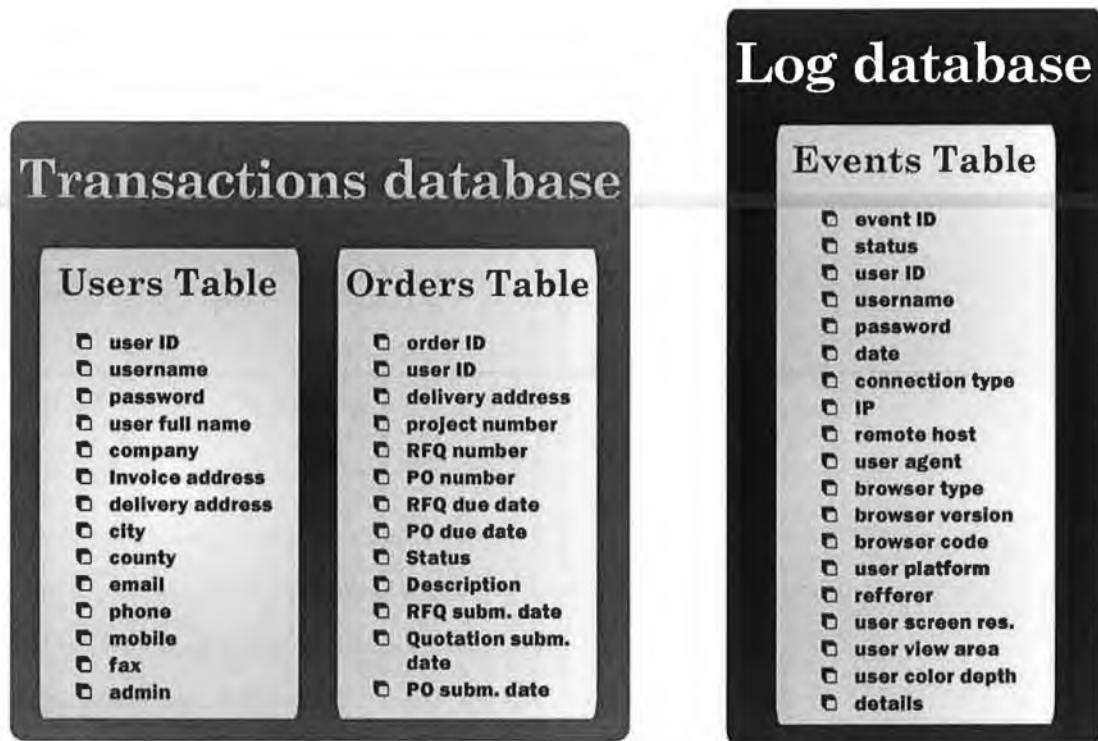


Figure 5.1 – The AMT application database structure

An important note is that each table in the database contains a field indexing the records in that table (for instance User_ID in the Users table, or Order_ID in the Orders table). This number is unique for each record in that table, and every time a record is added to the table a specially designed routine within the application analyses the table and calculates the first available ID number for the new record to be added.

Usually these ID numbers are in consecutive order, so that the next available one can be obtained by determining the highest number in the list of ID numbers obtained by retrieving them from the database and incrementing this number to obtain the next available value. This method makes sense for databases that are only used to store data incrementally, so that most of the data transactions add new data into the database. Since, in our case, a certain number of transactions also remove data from the database (by removing requests for quotation that did not materialise into orders, or removing old orders that have been completed or removing users that are no longer needed from the users list) this simple procedure is not good enough. If this method of generating new ID numbers is used in this case, after a certain period of time the new ID numbers generated in the normal operation would reach astronomical values until it will eventually overflow the allocated field size for the ID number in the database.

One solution to this problem would be to reserve a big amount of space for that field size so that even if the numbers would get very high they would still fit into those fields. Although this solution might seem very simple and straightforward, it still has important

disadvantages. One of these disadvantages is that reserving extra space for the ID number field in every table of the database would lead to important increase in database size, which in turn would result in more information being moved from and to the database and longer processing times which would make the application slower. Another disadvantage to this method is the fact that big order ID numbers (eight or ten digits or even more) are hard to handle or remember by customers which might want to reference orders this way. Since the order ID number is supplied to the user every time he submits an order or request for quotation, it is normal that customers might want to reference orders that they have submitted by using this unique number when they make enquiries about their orders (either by querying the application, by phone or by using email). If long ID numbers are used, this might lead to difficulty in handling these numbers and will result in confusions being made.

An elegant solution to this problem is to retrieve and analyse all the ID numbers in that table, in order to obtain the smallest ID number that is not used by any record in that table. By doing this, the application will fill the gaps in the sequence of numbers used and will effectively reassign numbers that were used before but are no longer needed. This simple routine listed below does just that, by obtaining a recordset variable containing all the ID numbers within that table, and cycling through them to see which number is missing from that sequence and returning that number as the result. If, on the other hand, the sequence is complete and no gaps are found, the routine will return as result the highest ID number incremented by 1 (the next number in the sequence).

The following VBscript routine is used in the administrative view within the application to determine the first available order ID number for adding a new order.

```
Set ROID = Server.CreateObject("ADODB.Recordset")
    ROID.CursorType = adOpenStatic

SQLquery = "SELECT Orders.Order_ID FROM Orders"
Set ROID = conn.Execute(SQLquery)

unique = 0
faOID = 0

If Not ROID.EOF Then

    Do While unique = 0
        faOID = faOID + 1
```

```

        unique = 1
        Do While (Not ROID.EOF And unique = 1)
            If faOID <> ROID("Order_ID") Then
                ROID.MoveNext
            Else
                unique = 0
            End If
        loop
        ROID.MoveFirst
    loop

End If

```

The first two lines create a static recordset type variable that is used in the following two lines to store all the ID numbers used in that table (orders table) by using a database query. In the next two lines two more additional variables are created, *unique* and *faOID* (first available Order_ID). The first one is used for the purpose of designating the uniqueness of an ID number within the table and contains a Boolean value (0 or 1), and the second one for the purpose of storing for later usage of the result of the computation. The following line contains an *if* statement, used to determine whether at least one ID number is present in the recordset variable (the table is not empty). The following lines consist of two loops, one of them embedded into the other. While the first one generates a series of numbers (starting with zero) in ascending order, the second one tests whether that number is already used as an ID number (it is present in the *faOID* recordset variable). It does this by cycling through all the values stored in the *ROID* (Retrieved Order_ID) recordset variable and comparing the value stored in the current position with the number generated by the previous loop. The result of the testing is stored in the *unique* variable, which has the value of 1 if the number hasn't been used before and 0 if the number is already used. If at the end of the second loop the *unique* variable contains the value 1, then it means that a unique number has been discovered, the execution is stopped and the number is returned as a result and it will subsequently be used as the assigned ID number for that newly submitted order.

A slightly modified version of this routine has been used throughout the application, whenever it was needed to generate a missing number in the sequence of ID numbers used in a table. The same slightly modified routine was also used in the WAP application presented in chapter four.

Besides offering users the ability to submit orders or receive information like status updates, the two combined applications (Web and wireless) need a simple database management mechanism that will enable a system administrator to manage the information in the database. This includes managing the users list (adding or removing users, or changing their account details), orders table (adding or removing orders, changing order status or details, etc) and examining the application log. In order to achieve this, the web application is split into two parts, providing a user view and an administrator view. The user view will allow a normal user (customer) to view its orders, place new orders, activate RFQs (Requests for Quotation) or change his account details. The administrator view enables the system administrator to gain access to the whole database, including users list and orders. In addition to this functionality, the administrator also has the ability to view the application's logs. The information is contained into a Microsoft Access database. The administrator will be able to make changes into orders, into the users list (by editing individual user details) and submit emails to user email addresses stored within the application by using an external email program (like Microsoft Outlook, Outlook Express or Pegasus Mail). This administrative feature is only available from within the web application and users with administrative privileges are distinguished from ordinary users at login time, by using a special field stored within the user's profile in the database.

The functioning of the two connected applications will be explained in detail in the following sections.

5.5 The content type-based redirection

In order to make a distinction between customers attempting to access the application using either web browsers or mobile devices, a content-type redirection is made when the client first connect. This redirection is not only necessary because of the two different content types required by the client device accessing the application (HTML for web and WML for wireless devices), but it is also useful to design the application using a single entry point. This will make the application much easier to use, since the customer will only have to remember one single address in order to access the application (in the case of this implementation, the address is <http://amt.ul.ie/motto/>). This is the single entry point from which customers will be automatically redirected to the specific part of the application dependent on the fact that they are using a mobile device or a web browser.

When the user tries to access the application folder, the server is configured so that it automatically supplies the index.asp file (this is done by specifically configuring the web server to indicate that index.asp is the default file for this folder if no specific request is

made). By automatically directing the client to the index.asp file, the code contained within the file is launched into execution.

When a browser submits a request towards a web server, along with the request it also submits information about itself and the device that it's built into. It does this by including lines of text (called headers) at the beginning of the request message. The header includes information about the type and version of the browser, the mime types accepted by the browser, the address (IP address) of the host on which the browser resides, the operating system of the host, details about the user's display, etc. This information conforms to the CGI/1.0 specification and is stored by the web server to be either used by the applications running on the server or to be possibly included in the server's logs (if the server is specifically configured for storing this data). Within the ASP, in the Visual Basic Object Model employed with the use of VBScript there is a specific method within the *request* object, method that retrieves the value of these headers for use in applications running on the server. This method is *request.ServerVariables()*.

Within the headers obtained by the server, there is one containing information about the content types accepted by the browser. This is called HTTP_ACCEPT and it is a string comprising of comma-separated values describing the MIME type in the following format: type/format. For instance, the HTTP_ACCEPT header generated by a common web browser (IE6) is:

```
image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/msword,  
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/x-  
gsarcade-launch, */*
```

The application analyses this mime-type list to determine the capabilities of the browser and redirects it to the specific part of the application as follows:

- If the browser supports WML content but not HTML content, it will be automatically redirected to the WAP application
- If the browser does not support WML content, it is automatically assumed to support HTML content and is redirected accordingly, to the web application.
- If the browser supports both WML and HTML content, it is automatically redirected to the web (HTML) application.
- In all the other cases, including the inability to determine the supported content type for the browser (this might happen with some "exotic" browsers like Opera, which allow

the user to specify a number of parameters for the browser, including mime-type declaration), the request is redirected to the web (HTML) application.

This decision process was designed due to the following considerations:

- Whether or not the client device (browser) supports WML, it is preferable to direct the client to access the web (HTML) part of the application (if HTML is within the accepted mime types in the HTTP_ACCEPT header), since the web version allows a better design, enhanced graphics capabilities and has more functionality than the WAP version. This is due to the fact that generally speaking mobile devices and WML have modest rendering capabilities and reduced functionality due to their small screen, rudimentary input capabilities (small keypads) and low processing power (as discussed in chapter 2.2 and 3).
- If the browser does not accept WML, it is probably not a mobile device and therefore it is assumed to support HTML (which is much more common than WML anyway, since the request is coming from the Internet)
- In the near future, it is likely that common web browsers will incorporate WML functionality to enable internet users to view and interact with WAP sites even when they are browsing them using desktop computers.

Once the client device (browser) is redirected to the appropriate application, it requests and loads the initial page (or deck) of that specific part of the application. The two parts of the application, the Web version and the WAP version, will be described in detail in the following sections.

5.6 Web application design and implementation

After the initial redirection of the client device (Web browser), the Web application will direct the user to perform the authentication. The authentication process and the technique employed to keep the client device authenticated at all times during the usage of the application will be detailed in the following section.

5.6.1 Authentication and security issues

The Web application starts by supplying the browser with the login page (login.asp). This page allows the user to input the username/password pair of values for the purpose of authentication. Authentication is employed not only for the purpose of providing the user with personalised content, but also to enforce security for the application.

Once a user has logged on, subsequent accesses to the application made by requesting different ASP pages identify the user's browser by the means of session cookies, so that a user utilizing that browser remains authenticated while requesting different pages within the application.

One of the challenges to developing Web applications is maintaining user information over the course of a visit, or session, as the user travels from page to page in an application. Since HTTP is a stateless protocol (meaning that Web servers treat each HTTP request for a page as an independent request) the server retains no knowledge of previous requests, even if they occurred only a short time prior to the current request. This inability to remember previous requests means that it is difficult to write applications able to keep track of items a user has selected while jumping between various pages.

Cookies are a general mechanism which server side applications can use to both store and retrieve information on the client side. When a cookie is stored on the client browser, along with the data stored within the cookie an expiration date is also stored. This expiration date specifies the date (and time) after which the cookie is considered not to be valid anymore and is removed. If no date and time are specified, the cookie lives until the web browser program is stopped (closed).

ASP uses cookies to manage session information. Using the ASP Session object and a special user ID generated by the web server and stored within a cookie on the client-side, the web server is able to store a number of variables associated with a specific user (session data) associated with and identified by that user ID.

The following diagram visually describes the session mechanism.

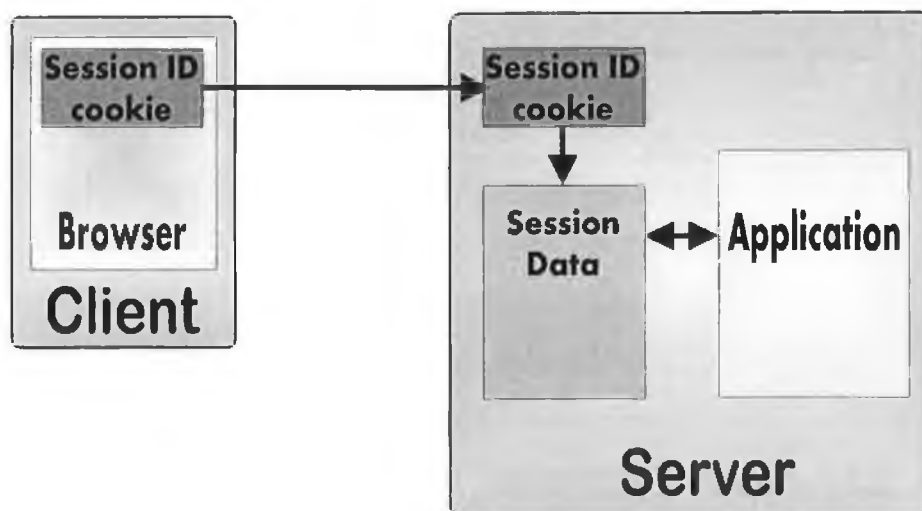


Figure 5.2 – Using the cookie mechanism to maintain session data

ASP sessions have the following features:

- The session data is stored on the server.
- In order to keep track of the data for a specific user, the server uses the session ID cookie received to identify that user.
- The first time a browser requests an .asp file from a given application or whenever a valid session ID cookie is not found on the browser when a request is made, ASP generates a Session ID number and stores it as a cookie on the client's browser.
- Session ID cookies are randomly generated numbers produced by a complex algorithm.
- The time after which a session expires (if no request is received) can either be defined in the web server's settings or can be modified programmatically (from within the application code).
- Once a session has expired, all the data pertaining to that session is discarded (lost) by the web server.

Sessions are used in the application to store validation information for users that were properly authenticated. This way, once authenticated, users can access ASP pages in the application by presenting a valid Session ID cookie that has its valid corresponding session data. However, since this mechanism was not specifically designed as a system for providing powerful security, the application cannot be considered bulletproof. [Endler 2001] The system can only be regarded as a simple mean to enforce basic security and identify users for personalisation purposes.

The first time a user accesses the application through the normal entry point (index.asp), its browser is instructed to store the session ID cookie. This automatically assumes that the client's web browser is able to store cookies and also the user allows it to accept them through the browser's settings. If not, the user will not be able to successfully authenticate.

In the login page (login.asp) the username/password pair of values are inputted into two fields, which are part of a form that uses the post method to submit the data.

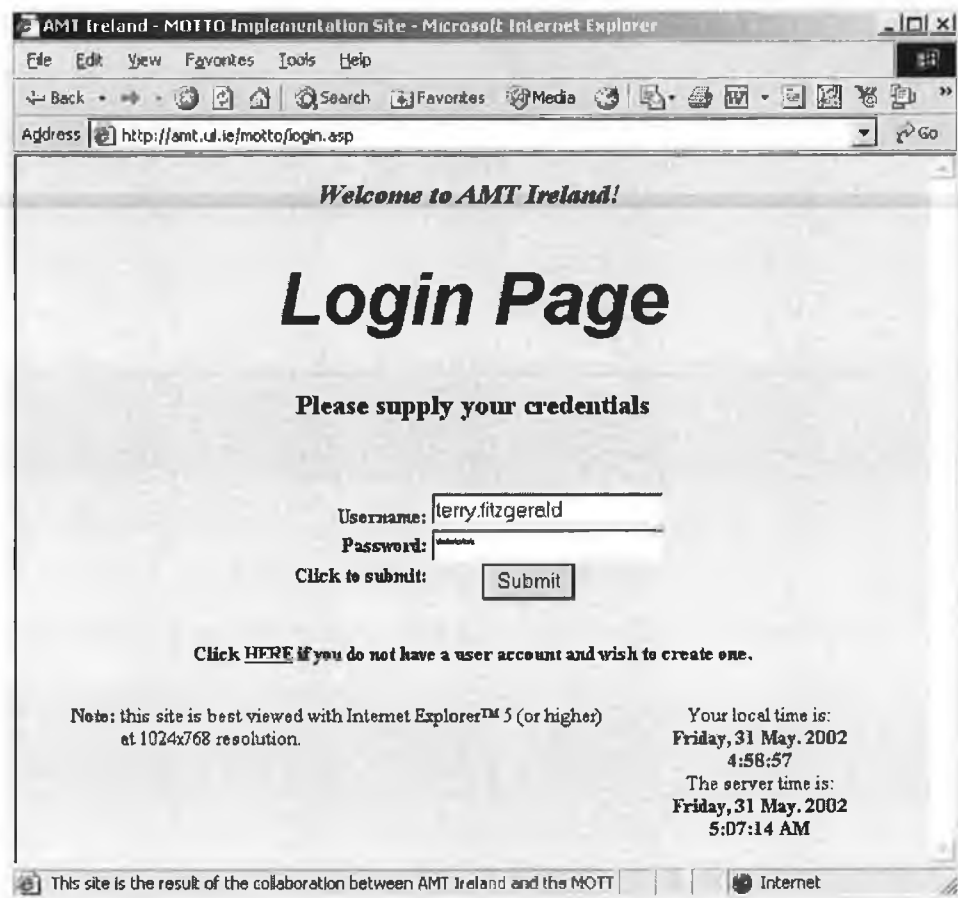


Figure 5.3 – The login page

After filling in the username/password pair of values in the login page, the user presses the submit button which will access the verify.asp page and submit the two values.

As seen in section 5.4 the transactions database contains a users table. This table stores specific data about registered users accessing the system.

When the verify.asp page is launched into execution, it retrieves the two values submitted by the user and opens a database connection (towards the transactions database). After establishing the connection, it uses an SQL query to retrieve the password value from the users table for the user having that specific username, and compares it to the password value submitted by the user. If the comparison is not successful the client is informed that the authentication has failed and directs it back to the login page after a small delay.



Figure 5.4 – The authentication failure page

It is important to note that the system deliberately does not supply users with information indicating which of the two values was not correct (username or password) even if this would be easy to achieve in the program, since this would facilitate easy discovery of usernames stored into database and this, in turn, would make it easier to break into the system. Instead, the application only states that the pair of values is invalid.

It is also important to note that in order to be consistent with the current practices regarding the usage of usernames and passwords in modern authentication systems, the application provides case sensitivity only for passwords but not for usernames. If the comparison is successful the user is directed towards the main page of the application.

Besides the username and password fields, the users table in the database also contains a field called Admin. This field has a Boolean value (1 or 0, true or false), which identifies whether the user is an application administrator. In order to provide personalised content the application uses a different set of pages to display different functionality to application administrators as opposed to normal users. After a successful authentication, the value in the Admin field is analysed and if the user is found to be an application administrator (the Admin field has the value of 1 or true) it will be directed to the administrative view of the application. The application also stores a variable into the session contents collection (the

session data) for the user, variable that will subsequently identify that user as an administrator every time he will attempt to access pages with administrative functionality. If the Admin field contains the value 0 or false, the user is identified as a normal user and directed accordingly to the user view of the application.

The following diagram illustrates the process that takes place when users authenticate with the system.

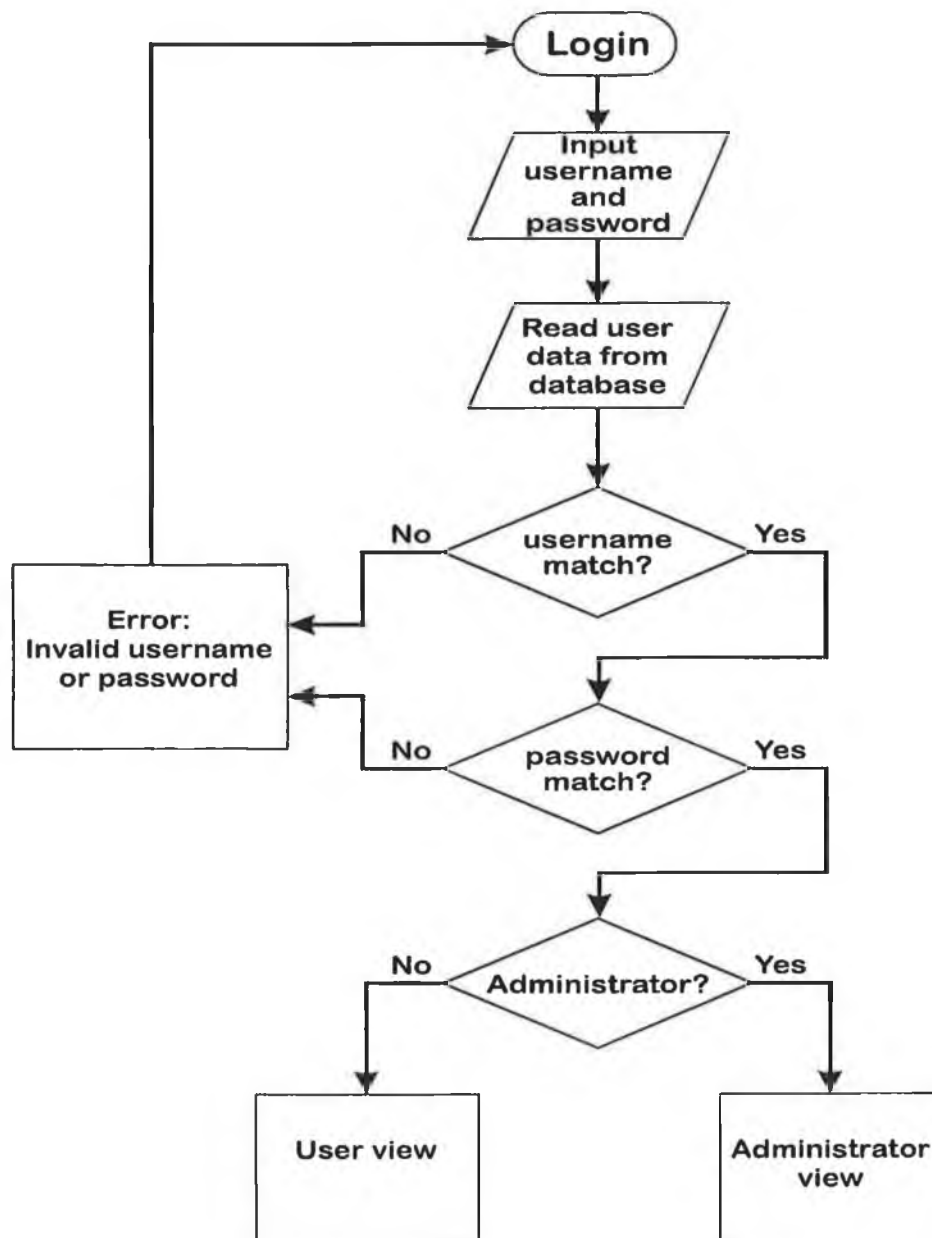


Figure 5.5 – The authentication data flow diagram

The following sections will describe these two parts of the application, the normal user view and the administrative view.

5.6.2 The user view

After the user has been successfully authenticated he is presented with a message to that end and redirected to the main page of the user view.

The main page offers the following functionality:

- Submit requests for quotations
- Accessing the order status for current submitted orders
- Activation of orders (for orders that are in the RFQ stage)
- Change the user's account settings



Figure 5.6 – The main page of the user view

In addition to this functionality, the user can also log off at any time by clicking the *Log Out* button located at top-right-hand side of the window.

By following the links provided in the main page, the user accesses a number of pages and forms providing different functionality. Every time it does that, a routine stored in a different file and included in each page (by the means of the `#include` directive) gets executed. The routine, named *UID* (short for UserID), checks whether the *UID* variable exists in the session contents collection and if it contains a valid user ID number. If the variable exists it means that the user is properly authenticated and a valid session ID cookie was submitted, and therefore it grants access to the functionality provided within

that page. Otherwise, the user is presented with a page informing him that there was an authentication failure, and is subsequently redirected to the login page as shown in the following image.



Figure 5.7 – Authentication failure due to unauthorised request

The first link provides the user with the ability to submit requests for quotation (RFQs). The user is presented with a form enabling him to enter details about the order, like delivery address, request for quotation number, purchase order number, due date and a description of the order.

AMT Ireland - MOTTO Implementation Test Site - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media Print Mail

Address <http://amt.ul.ie/motto/urfq.asp> Go

User: **terry.fitzgerald**

AMT Ireland

Submit Request For Quotation

Submit Request for Quotation with the Order ID 13

OrderID:

UserID:

Delivery address:

Account Delivery Address: 52 Airways Industrial Estate Dublin 17

Account Invoice Address: 52 Airways Industrial Estate Dublin 17

Other:

Project Number:

RFQ Number:

PO Number:

DueDate:

Description:

Note: All fields marked with * are mandatory

This site is the result of the collaboration between AMT Ireland and the MOTTO Project team. For more de

Figure 5.8 – The *Submit Request For Quotation* form

Before submission, scripting functionality running on client-side verifies whether the fields in the form contain less than the maximum amount of data accepted (maximum field size from the database) and if the mandatory fields contain the required information.

Also, as mentioned in section 5.4 of this chapter, an automated routine running on the server-side (within the asp page) analyses the data in the Orders table and identifies the first available order ID number, which can be used for the current order.

After filling in the form and submission, the application retrieves the data, makes additional verifications to determine whether the data is consistent, opens a connection to the database and adds a new record into the Orders table using an SQL statement. Upon adding the information into the database, the user is presented with a report stating that the data transaction was successful.

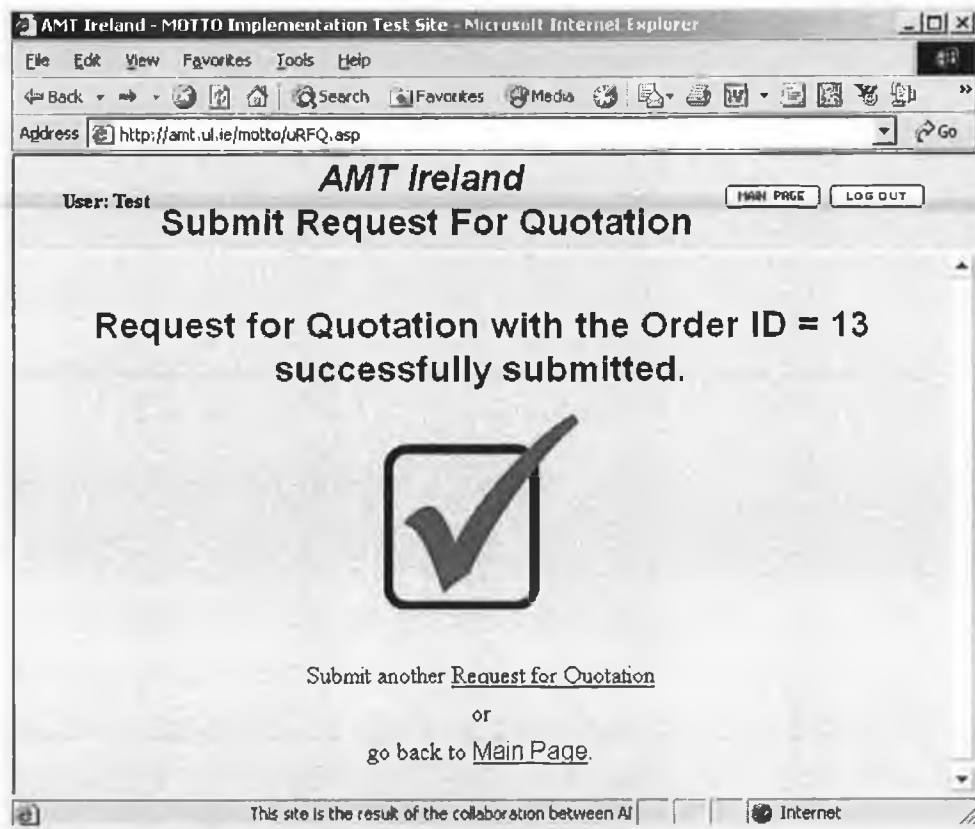


Figure 5.9 – The Request For Quotation submission confirmation page

The second link on the main page (*Order Status*) provides the user with the ability to obtain a list of orders, specific details about these orders and assess their status. The user is presented with a list in the form of a table containing all the orders in the database registered on his name and some details about these orders.

The screenshot shows a Microsoft Internet Explorer browser window displaying the AMT Ireland website. The address bar shows the URL <http://amt.ul.ie/motto/OS.asp>. The page content includes the following elements:

- Header: "AMT Ireland" with a "MAIN PAGE" button and a "LOG OUT" button.
- User information: "User: sean.sidley.".
- Main heading: "Order status".
- Table of orders:

Details	OrderID	Project Number	RFQ Number	PO Number	DueDate	Status	Description	RFQ SDate	Q SDate
DETAILS	6	02L5068 - batch 2	-	013541	23/04/02	project complete	samples 4 - 8 (batch 2) for bare board validation	-	4/23/2002 9:50:34 AM
DETAILS	3	02L5068	-	-013541	-	Project Complete	Bare board evaluations on 3 part numbers	4/4/2002 4:33:26 PM	-
DETAILS	11	unassigned	-	013541	-	project completed	Batch 3 of bare board evaluation	5/20/2002 3:14:13 PM	-
DETAILS	12	unassigned	-	013541	-	working on samples	batch 4 of bare board evaluation	5/20/2002 3:14:51 PM	-

There is a total of 4 orders.

Sort orders by: [OrderID](#) [Project Number](#) [RFQ Number](#) [PO Number](#) [RFQ Submission Date](#) | [RFQ DueDate](#) |

Your local time is: Friday, 31 May, 2002 8:44:38
The server time is: Friday, 31 May, 2002 8:52:56 AM

Footer: "This site is the result of the collaboration" and "Internet".

Figure 5.10 – The Order Status page

Additional details about these orders can be obtained by clicking on the corresponding *Details* button located in the first column of the table. This button will open an additional window containing all the details available in the database for that specific order so that the user can analyse them or print them.

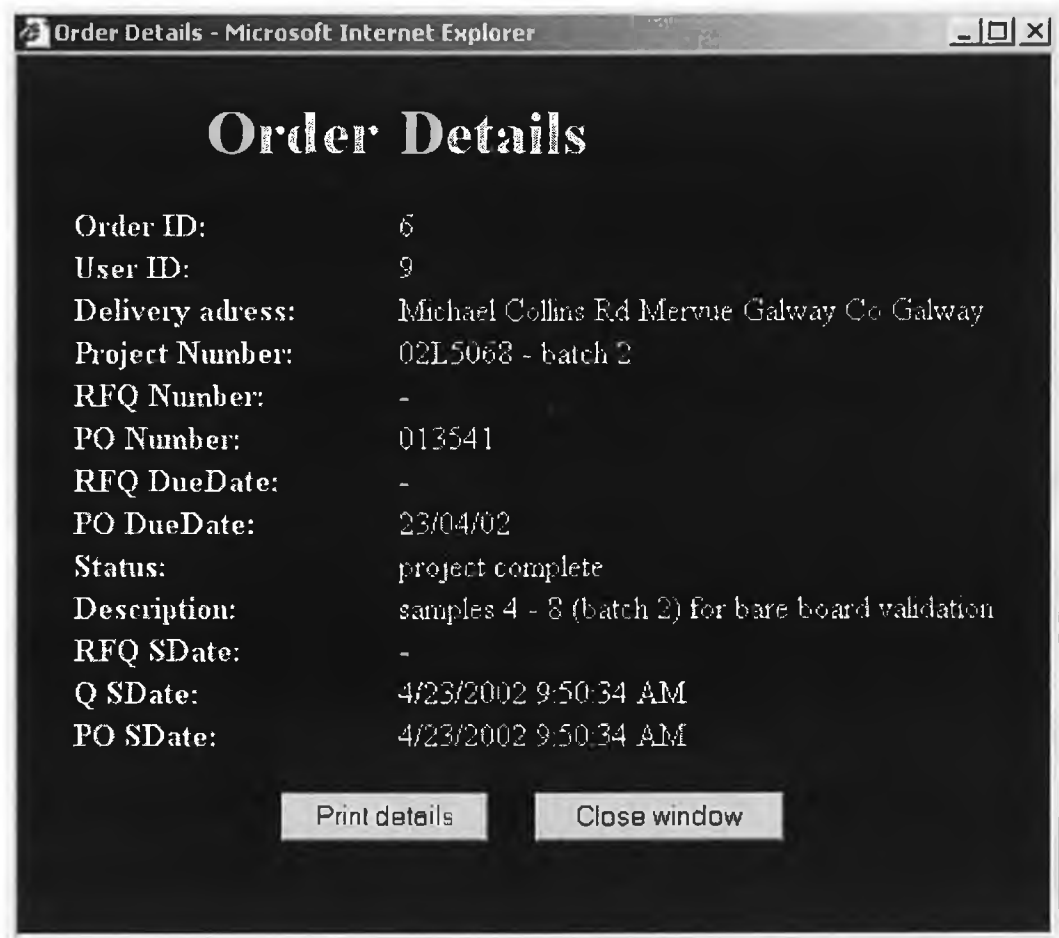


Figure 5.11 – The Order Details page

An important note is that the order listing, as well as all the other functions accessible from the main page, are personalised. This means that the user will only see orders or RFQs submitted by him, and will only be able to submit new RFQs/orders that originate from him.

A second observation is that the user cannot remove orders or RFQs once they are submitted. This task can only be performed by the application administrator, after a request to that end has been made by the user (or user's company).

The third link in the main page enables the user to activate an order. From the application's perspective an order is considered to be active if either it has an assigned project number in the Project number field, if the Purchase Order number field has either a valid Purchase Order number or if this field contains the word "Active". This link enables users to see a list of inactive orders (RFQs) and activate one of them by clicking on the

Activate button and either inputting a PO number in the form that opens or just by simply selecting the *Activate Order* button in that form (which will just add in the PO number field the word “Active”). Once an order has been activated, if it does not have a Project number associated with it, it will flash in the orders listing in the administrator’s view signalling that it needs to be attended. Also, when activating an order, the user gets the chance to change (edit) details about that order, like delivery address, due date or description.

The fourth option present in the main page allows the user to change his account settings, like the username registered for that account, the full name, both invoice and delivery addresses, city, county and personal details like phone numbers, email address and the password.

The following diagram illustrates the screen flow for the whole user view part of the application (user screen flow diagram).

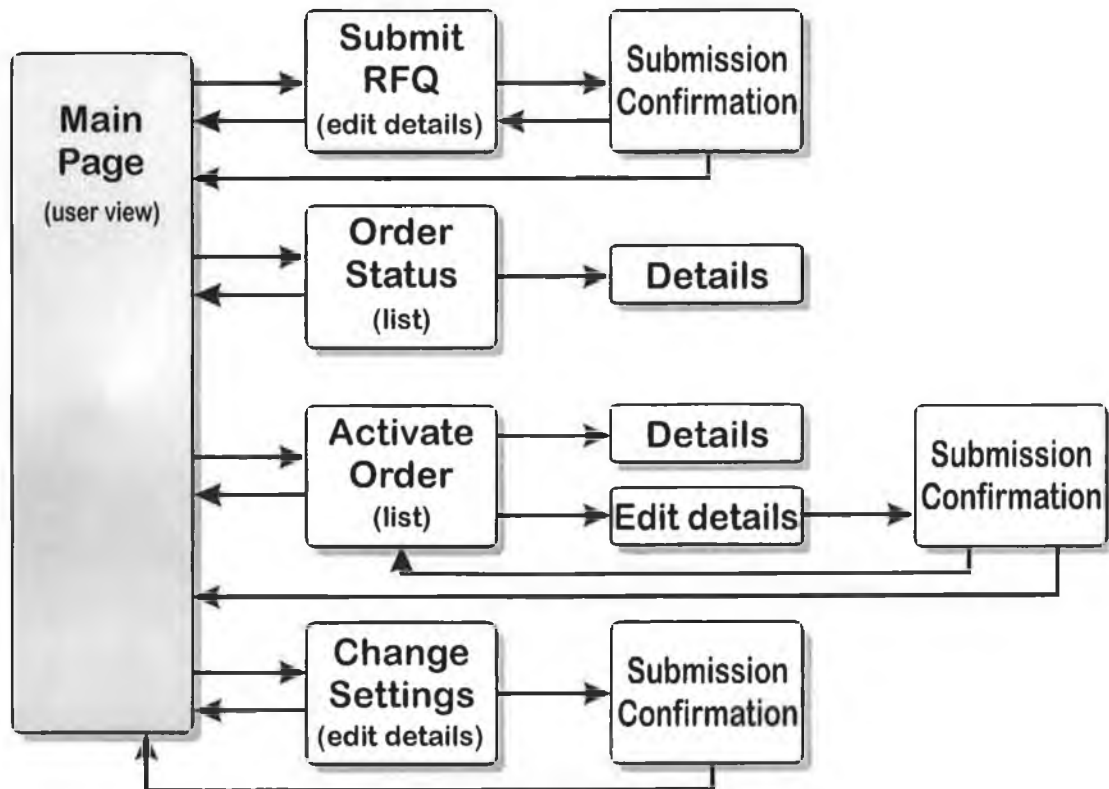


Figure 5.12 – The user view screen flow diagram

The application has also a hit counter, displaying the number of users that were logged on to the application. This counter, displayed in the header of each page (including the main page), is not incremented each time the user accesses the page, but rather when the user first tries to access the application by requesting a page from within the application (typically the index page).

When the request is made, the Web server examines the cookies that were sent along with the request. If the session ID cookie is not present amongst these, the web server generates a session ID number (using an internal random number generator) and passes this number to the browser as a cookie, thus creating a new, valid session ID cookie as described earlier in this chapter. At the same time, the server also allocates memory space for a data structure that will be used to store session variables from within the application, and labels this structure with the session ID number, so that subsequent accesses to the server from the same browser will be identified and immediately associated with the set of variables stored in that data structure.

As part of the same process, after completing assigning the session state data, the server will also launch into execution a set of routines stored in the *global.asa* file. This file stores a number of procedures that are to be executed when certain events occur. These events are:

- *Application_OnStart*; this subroutine is launched every time the application is started, which usually happens when a page within the application is first accessed after the Web server has been restarted (or started after the server has been switched on). It is typically used for initialising applications and setting up application state variables.
- *Application_OnEnd*; this subroutine is launched every time the application is stopped (terminated), which usually happens when the web server is shut down or when a special command to that effect is issued from within the web server. It is typically used for cleaning up the application variables and performing other tasks that are necessary for an application, like, for instance, saving different state variables onto the disks.
- *Session_OnStart*; this subroutine is launched every time a session is created. It is typically used to create session variables as well as performing different other tasks pertaining to the session.
- *Session_OnEnd*; this subroutine is launched every time a session is destroyed.

While the first two subroutines are used for performing different internal tasks within the application (like setting up application variables and application path data), the last two are used for incrementing the variable that stores the number of hits for the application (`Application("Hits")`) and storing that variable on the disk for future reference (`Session_OnStart`), and for committing log information into the log database (`Session_OnEnd`). Thus, the number of hits on the application is not incremented every time a user requests a certain page within the application (which can happen many times during the normal use of the application) but rather each time the user has a working session with the application.

5.6.3 The administrator view

As mentioned earlier in this chapter (in section 5.4 - Functional requirements), besides the normal user view enabling users to interact with the system, the web application also exhibits an administrative view. This view enables the application administrator to keep track of user-submitted orders and RFQs, manage the orders table and users list, and examine the system log. Also, as already mentioned, the distinction between ordinary users and administrators is made by the application at login time, by examining the Admin field within the user's profile stored in the database. The login process takes place in the same fashion as for every user, except for the fact that once the user's profile is retrieved from the database and the Admin field is found to contain the value of 1 (or true), the respective user is granted administrative privileges by storing a variable called Admin in the session contents collection and assigning it the value 1.

```
<%If URecord("Admin") = True Then%>  
...  
<%Session ("Admin") = 1  
...  
End If%>
```

This variable will be used during the whole session to identify the user as an administrator and grant him access to the administrative view pages.



Figure 5.13 – The successful administrative logon page

Once the user has been identified to be an administrator, he is automatically directed to the main page of the administrative view.

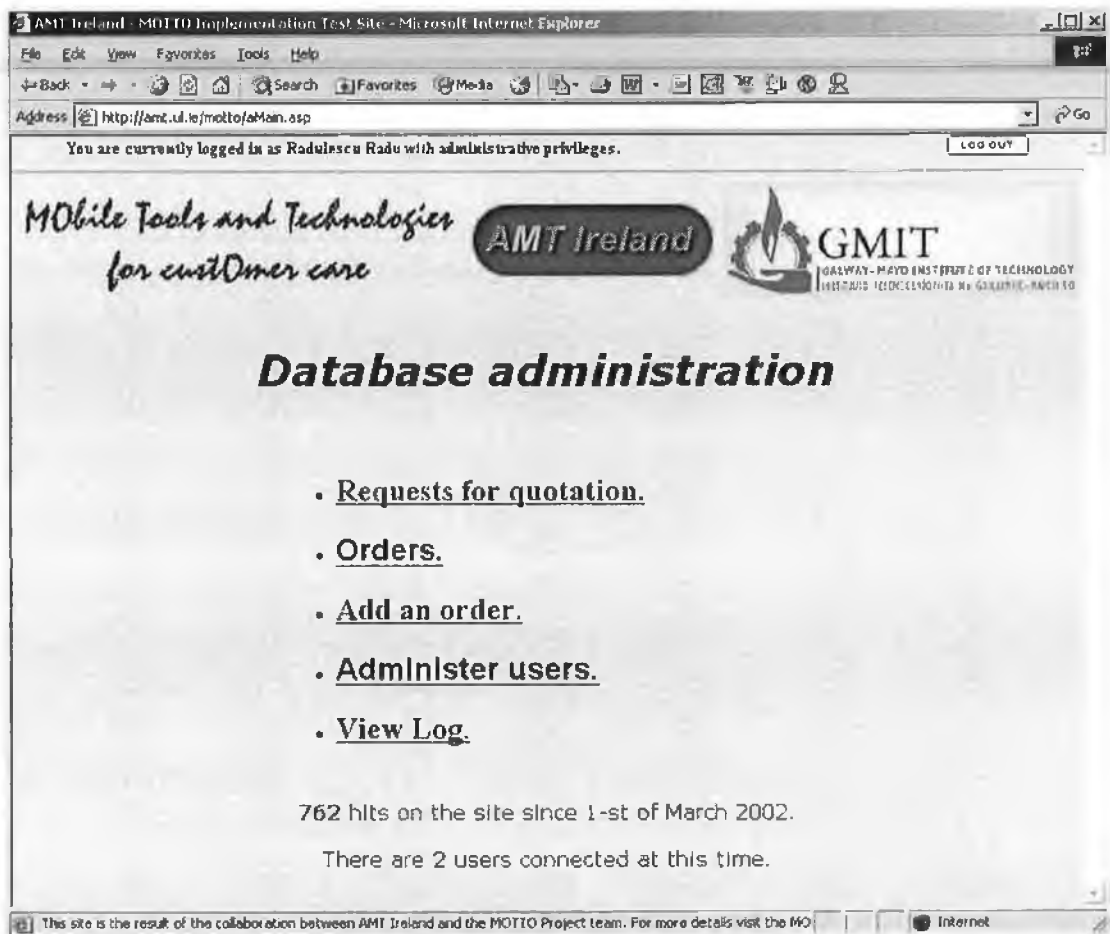


Figure 5.14 – The administrator main page

From here, the user (application administrator) can perform one of the actions listed in the menu as shown in the following diagram, the administrator view screen flow diagram (figure 5.15).

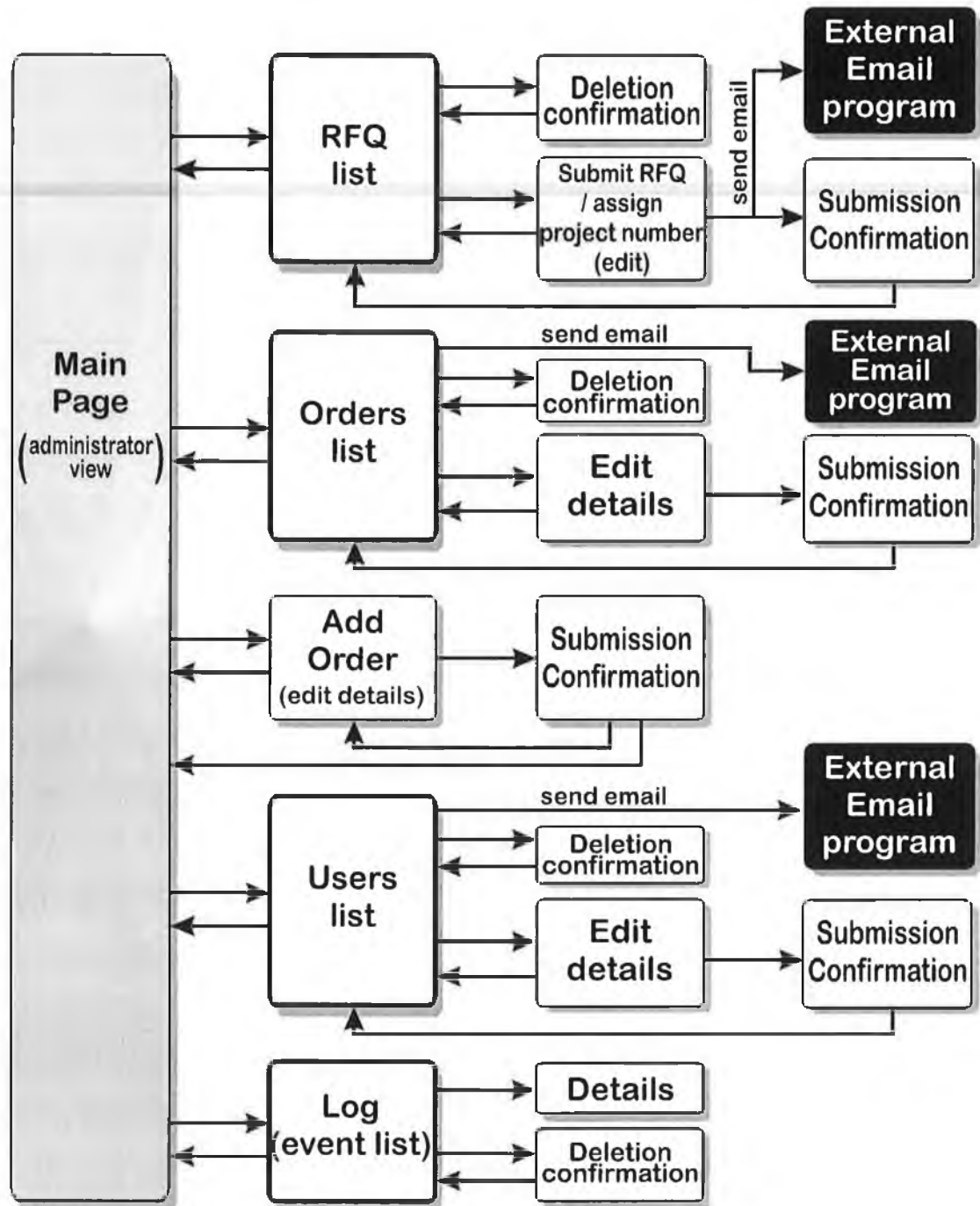


Figure 5.15 – The administrator view screen flow diagram

The first link provides the administrator with the ability to list all the requests for quotation stored in the database. Orders can be listed using the second link. Essentially, both orders (active or inactive) and requests for quotation (RFQ) are stored in the database in the same table and in identical type of records. The application distinguishes an order from an RFQ by analysing information stored in its fields.

Every order has the following characteristics:

- An order ID number (already described earlier in this chapter, in section 5.4)

- The ID number of the user who submitted this order (used for linking the order with user's details, like username, full name, company name, phone and fax numbers and email address)
- Delivery address for that order. Since a company, besides headquarters, might also have more than one subsidiary, the order's delivery address cannot be linked to (retrieved from) the user's delivery address. For this reason, since a user residing at headquarters might order a product that needs to be delivered at different branch of that company, the order must contain its own delivery address.
- Project number. This is an internal number assigned to each order. It must be assigned to orders before they are launched into production.
- Request for quotation (RFQ) number. This is an internal number assigned to each quotation submitted to a customer.
- Purchase order (PO) number. This is an internal reference number assigned by the client company for every purchase order issued.
- RFQ due date and PO due date. The first one is the due date requested by the client at the time of the RFQ submission. It is essentially the due date that the customer says he needs when he requests the quotation. The latter is the due date agreed between the producer and the client after the project has been analysed, discussed and agreed upon. Sometimes these dates do not coincide due to the fact that, sometimes the order cannot be completed in the required time, and some other times because the order actually requires less time to be completed than it was initially estimated.
- Status. Describes the current status of the order.
- Description. Provides a brief description for the order (no more than 100 characters are stored in the database)
- RFQ SDate, Q SDate and PO SDate. The first one is the date when the request for quotation (RFQ) was submitted. If the order was submitted using the application, this date is automatically recorded by the system in the database. The second one represents the date when the quotation was submitted. It is also recorded by the system if the application was used for submitting the quotation (by bringing up the external email program for submitting the quotation). The third one, PO SDate, is the date when the project was activated and is treated by the system as a fully qualified order.

An order is considered to be an RFQ (request for quotation) if it meets both of the following conditions:

- It does not have an assigned Project Number (the database contains either a dash (-) or the word "unassigned" in the Project Number field, or the field is empty).

- It does not have an assigned PO number (the database contains a dash (-) in the PO number field or the field is empty)

An order is considered to be an active order if it meets either of the following conditions:

- It contains a project number in the Project Number field in the database
- It either contains a purchase order (PO) number in the PO number field in the database or instead it contains the word “active” in this field (the field is of type text, so it can contain either numbers or text)

In addition to that, if the order does not have an Project Number assigned but it does have a PO number assigned by the client (or has just been activated by the client, which means the word “Active” appears in the PO number field in the database) then the order is considered to be active but has not yet been taken into consideration by the manufacturer. This determines the fact that orders (which only appear in the orders listing, since are no longer considered to be RFQs, but active orders) will have the word “Active” flashing in red in the first field in the orders listing. This is depicted in the following screenshot:

AMT Ireland Orders						
User: Ratu (admin)						
REMOVE	10	Frank Glackin	Blair	-	-	
Active REMOVE	12	Sean Sidley	Tyco Healthcare Ltd	091753771	-	Sean.S
REMOVE	5	Hezcon Computer	University of Cambridge			

Figure 5.16 – Active order detail

Returning to the main menu items, the first link provides the application administrator with the ability to list RFQs by using the first link (titled “Requests for quotations”). This will bring up the list of RFQs, which are essentially – as described earlier – orders that meet certain criteria. The second link in the main menu lists all the orders contained in the database, either active or inactive (in production or completed), and it includes the RFQs. This was provided to insure full control of all the orders by the administrator in case something goes wrong in the process of promoting orders from the status of RFQ to active and normal orders (being processed or completed).

In the first listing (RFQs listing), the administrator can make an order active by assigning a project number by double-clicking any RFQ in the list. This will bring up a

different page containing input fields that already contain the details for that specific RFQ. The administrator can then change those details or fill in the empty fields and, upon submission, he will be given by the application the opportunity to submit an email message to the user related to that order.

In the second listing orders can be edited in the same manner previously described, with the only exception that the administrator will not be asked by the system whether he wants to submit an email to the user for notification purposes, but instead the administrator will have to do so by manually clicking on the user's email address to bring up the email program with the user's address already present in the "TO" field.

The third link in the main menu enables the application administrator to add an order. This option was included for the eventuality that an order (or a request for quotation) has been submitted by a client without the use of the application, and a record of the transaction needs to be inserted into the database for processing within the system.

The page consists of two frames. The administrator will be given the opportunity to select the company on behalf the order is issued from a drop-down list located in the first frame of the page. After selecting the company, a second drop-down list located in the same first frame will be automatically and instantly updated to contain the list of users working with that company, so that the administrator may select the specific user on behalf which the order is considered to be submitted.

The screenshot shows a web browser window titled "AMT Ireland - MOTO Implementation Test Site - Microsoft Internet Explorer". The address bar shows "http://ant.u.ie/motto/AddOrder.asp". The page content includes a header with "User: Radu (admin)", "AMT Ireland Add Order", and buttons for "HIGH PRICE" and "LOG OUT". The main section is titled "Submit Order with the Order ID 13". It contains a form with the following fields and options:

- Company:** A dropdown menu with "APC (Castlebar)" selected.
- User:** A dropdown menu with a list of users: "AMT Ireland", "Analog Devices", "APC (Castlebar)", "Aragon Services", "Beta Layout", "BMR", "C&D Technologies", "Cemtech", "Dens", "Dell", and "Dell Products".
- Delivery address:** A text input field.
- Account Delivery Address:** A text input field.
- Account Invoice Address:** A text input field.
- Other:** A text input field.
- Project Number:** A text input field.
- RFQ Number:** A text input field.
- PO Number:** A text input field with a radio button for "Active".
- Due Date:** A text input field.
- Description:** A text input field with an asterisk indicating it is mandatory.

At the bottom of the form are buttons for "Cancel", "Reset", and "Submit Order". A note at the bottom states: "Note: All fields marked with * are mandatory".

Figure 5.17 – The Add Order page

Upon selecting the user from the second drop-down list, the second frame within the page will be updated (automatically refreshed) so that it contains input fields customized for the selected user. The administrator can then fill in other details related to the order and submit it to the server to be stored into the database. This is depicted by the following screen shot.

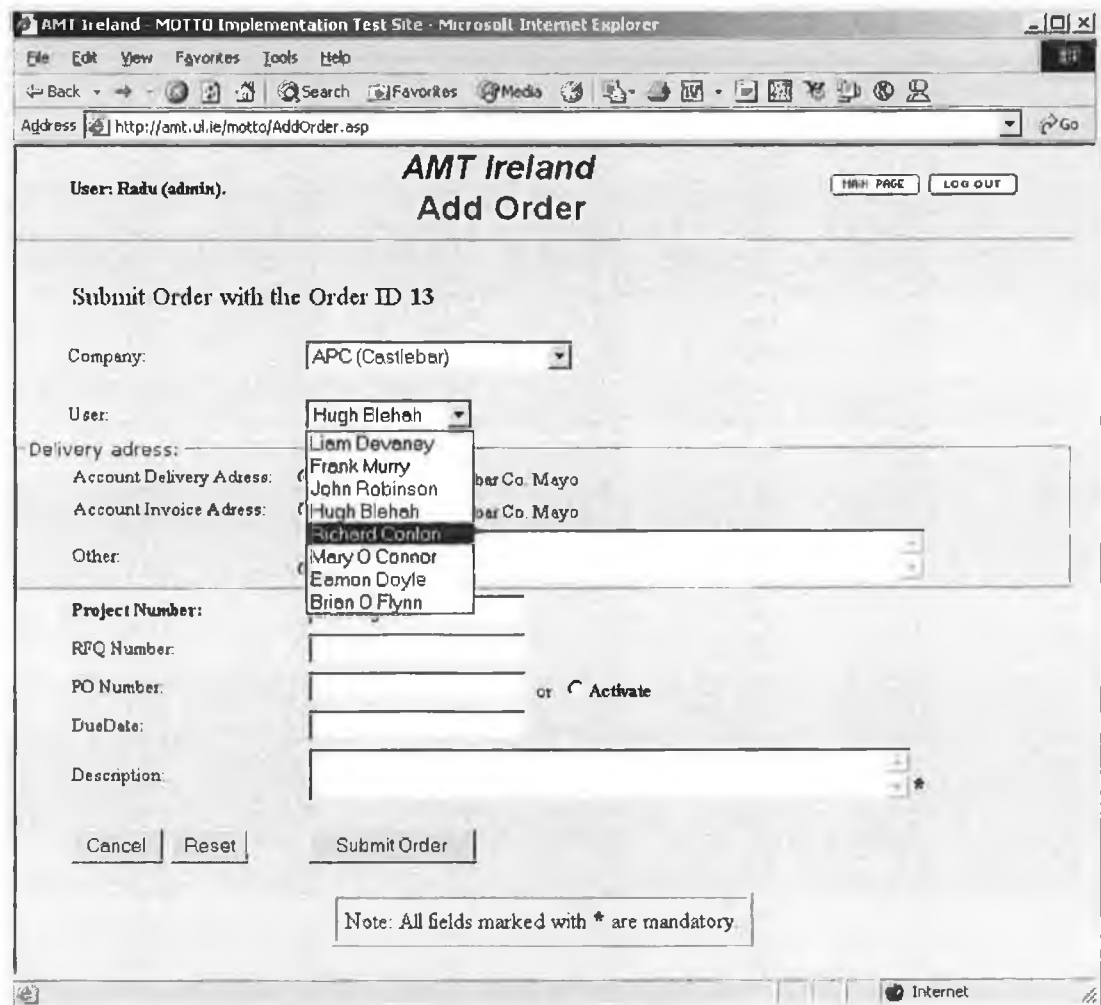


Figure 5.18 – Selecting a user from the customised list

The fourth link in the main menu offers administrators the possibility of managing the list of users recognized by the system. Within this page, any action can be performed on a user account, including changing user's details, promoting the user as an administrator (granting administrative rights) or even completely removing the account.

The administrator is presented with a list of users and their details (UserID, username, user's full name, company, phone and fax numbers, email address, password etc). As already described in the previous section regarding the RFQ and orders list, by double-clicking a record (row) in the list containing details for an account, a different page is presented containing input fields. The input fields contain user's details, and can be edited

in order to make changes to the account. Upon submission, the new details will be stored into the database.

The last piece of functionality provided by the application offers administrators the opportunity to view application's log, containing details related to users connecting to the application. These details are:

- Event ID number (a number unique for each event that took place, used for indexing this table)
- The date and time when the connection attempt took place
- The username and password used for the attempt (if any)
- Whether the attempt was successful or not (status of the attempt)
- The type of connection (web or wireless)
- The IP address of the machine used for the connection attempt
- The same IP address for the connection but determined using an alternate method (the DNS lookup method, RemoteHost)
- The type of browser the remote client was using (also determined using two distinct methods, the one provided by the ASP programming object model and a safer method using client-side JavaScript)
- Other details regarding user's browser, like, for instance, the version of the browser used for the connection (e.g. IE 6.0), or browser's compatibility string
- The operating system platform of the remote machine (e.g. Windows NT 4.0)
- The referrer for the application page attempted (the address of the web server - if any - containing the link to the application)
- Remote client's screen resolution, viewable area and its display colour depth
- Details about the connection, like whether the user has connected as an administrator (if the event was a connection), whether the user disconnected itself or has been automatically disconnected after a certain period of inactivity (if the event was a disconnection), whether the user was not actually connected to the application due to authentication failure, etc

All these event details are important on one hand for keeping track of application usage and generating statistics, and on the other hand from a security perspective, by enabling the system administrator to determine break-in attempts in early stages and take appropriate action like, for instance, preventing further accesses from those remote machines.

After examining the list of events, the administrator has the ability to delete these events either one at a time or all of them at once.

An important thing worth mentioning is that on every page containing a list, whether in the user view or administrator view, a number of links allow the user to rearrange the list using different sorting criteria. For instance, in the orders list within the administrator view, orders can be sorted (arranged) in the ascending order of the Order ID, User ID, Project Number, RFQ Number (request for quotation number), PO Number (purchase order number), RFQ Due Date (request for quotation due date), PO Due Date (purchase order due date), RFQ Submission Date (request for quotation submission date), Quotation Submission Date or PO Submission Date (purchase order submission date). This offers better functionality by enabling the user to easily find a certain record (row) within the list by re-sorting the records using the field (column) he is looking for, and scrolling down until he finds the fields with the corresponding letter(s). In addition to that, the user can also employ the search facility integrated within the browser (for instance, using the edit menu in Internet Explorer: *Edit -> Find (on this page)*... or keyboard shortcut Ctrl-F) to quickly locate a certain item on the page.

In addition to the sorting capability, a number of other useful functions are present in most of the pages containing listings. These functions include print buttons that generate output of lists in a format suitable for printers, buttons that allow the user to return to the main menu, buttons for logoff (disconnecting from the application) and fields presenting the user with the server time (local time set on the server hosting the application) and his own local time (for users travelling abroad).

Also, within tables (listings) containing email addresses, these addresses are presented as links which, when clicked, activate the default mail program on the user's computer (administrator) enabling him to bring up a new message window within its favourite mail program at the click of a button, and automatically have the address present in the "TO" field of the message. For instance if the administrator of the application wants to send an email message to a certain client (informing him about changes in the status of his orders, or requesting information about an order, etc) all he has to do is bring up the list of users (by clicking on the *Administer users* link in the main menu), locate the record for that user either by rearranging the records in the list using, for instance, the user's full name as a sort criteria and scroll down to the corresponding letter, either by simply using the web browser's search functionality (*Edit -> Find (on this page)*... in IE6), and after locating the user record by simply clicking on his email address listed in the record. This will bring up the email program and automatically fill in the user's email address, and thus the

application administrator can manage sending status updates or request information in an efficient manner.

Another useful feature of the application is the fact that, when an administrator displays the main page (on the administrator view), he is presented not only with the number of hits on the application (the number of visits that took place since the last initialisation of the counter), but he is also informed about the number of users connected to the application at that moment. One of the advantages offered by this feature is, for instance, the fact that the administrator is able to determine whether there are users connected to the application at that moment, before he might attempt to shut down the application for maintenance.

While the Web version of the application offers a big amount of functionality both for normal users and for application administrators managing the database, as it was already outlined in section 5.3 (detailing the user's specifications analysis), due to bandwidth and functionality restrictions for mobile devices some of this functionality is not possible or practical to achieve for the wireless application (or the wireless view of the application).

The next section will describe in detail the implementation of the wireless version of this application.

5.7 Mobile implementation

The implementation of the wireless version of the application was designed using the same programming language tools as for the prototype development presented in chapter 4.

Programming languages used for this WAP application are:

- VBscript code embedded into ASP pages on server-side. These pages use the Internet Information Server (IIS) as the web server platform running the application.
- WML (Wireless Markup Language) as the XML-compliant language required by mobile devices (such as mobile phones).
- WMLscript for providing client-side functionality for the application.

The WAP application is simpler than the one developed with the first prototype WAP application (described in chapter 4 of this thesis), due to the decision made (and explained earlier in this chapter) to only provide order status information and minimal capabilities for managing the account. Both of these functions are achieved by employing personalisation achieved by using simple means of authentication.

As already described earlier in this chapter (section 5.5 - The content type-based redirection) the application starts by analysing the accepted mime types header

(HTTP_ACCEPT header) from which it determines to which part of the application to direct the client. Once it has determined that the client can only accept WML content, the application issues a redirect command that will determine the client to request the initial deck in the WAP application.

All the files pertaining to the WAP application are stored into a directory called WAP, located in the main application directory.

The initial deck loaded by the client is actually a greeting page, flipping through a pair of cards at a 5 seconds interval. This generates the appearance of more dynamic content, which is usually difficult to achieve on WAP (mobile) devices.



Figure 5.19 – The AMT WAP application greeting card

The following deck constitutes the login screen, allowing the user to input the credentials for his account (username and password). If either username or password is not correct, a message to that effect is displayed but, as with the web application, it will not specify which one is wrong.

After a successful authentication, the user is directed towards the main page of the application.



Figure 5.20 - The authentication and main page

The links in the main menu are as follows:

- Order status; enables the user to display a personalised list of orders and obtain details about them.
- Info page; a page providing information about the site.
- Settings; a page offering users the ability to change a limited number of account details like username, email address, phone/mobile/fax number and the password.
- Help; a page providing help with the application.

The order status page provides personalised content by selecting only orders related to the user.



Figure 5.21 – The *Order Status* section

Details that can be obtained after selecting an order from the list include:

- Order ID number
- Status
- Description of the order
- Submission date
- Due date
- Purchase order number
- RFQ number
- Project number
- Delivery address
- User ID number
- Username
- Company
- City

After examining order details, the user is able to return to the main page.

The *Info* page, besides providing a brief description about the site, also provides images, delivered to the device in the WBMP (wireless bitmap) format.



Figure 5.22 – The *Info* card

Pictures in the same format are also present in the *Help* page and *Main Page*.

5.8 Summary

This chapter presented the application implemented in an SME, as well as the requirements and other specifications that help shape the application.

The first sections described the AMT Company for which the application was developed, and also provided a brief overview of the implementation conditions present on-site. The second section provided an overview of the user requirements based on which the main features of the application were outlined. The next section provided an overview of the actual features integrated into the application in order to accommodate the requirements previously described. Based on these features a set of technical requirements and specifications were outlined and described in detail. One of the most important features is the fact that the application has a Data Management System incorporated and it also has a dual way of interfacing with clients by using the Internet. Besides the wireless view – which constitutes the initial purpose of the application, it also incorporates a web view, enabling users not only to interface with the application by using wireless devices and WML, but also to be able to interface with users by employing the normal web interface and HTML.

The following section of the chapter described in detail the automatic redirection routine incorporated into the program, enabling the application to automatically direct incoming requests, according to the type of browser used. When accessed, the application will direct browsers according to their list of accepted Mime-types.

The next section (section 5.6) detailed the structure of the web view of the application, containing both the user view (customer interface) and the application administrator view. This section provided an in-depth analysis of the system by presenting and discussing the application data flow diagram for both the user view and the administrator view.

Finally, a similar detailed description was provided for the wireless view of the application, in section 5.7. A similar data flow diagram was also presented and detailed in this section, as well as the screen flow of the wireless application.

The following chapter will provide an overview of testing methodologies generally used for assessing software applications for the purpose of isolating the useful testing procedures employed in order to evaluate both the prototype and the implemented applications.

CHAPTER 6

Testing and Validation

- 6.1 Introduction**
- 6.2 Development and Testing Methodology**
- 6.3 Prototype Testing**
- 6.4 Testing and Validation for the AMT Application**
- 6.5 Outcome**
- 6.6 Summary**

6.1 Introduction

This chapter discusses the testing and validation phase of the two applications described in chapters four and five, applications that were developed during the course of this project.

It starts by presenting the general testing methodology commonly used for testing this type of applications. Even though it is not intended as a comprehensive overview of the testing procedures and methodologies employed for an in-depth analysis of software applications, it provides a set of general guidelines some of which are later employed for testing the applications developed during the course of this project.

The second part of this chapter describes the stages and methodologies used to analyse and validate the initial prototype application developed in chapter four. The third section will describe the testing methodologies used for the application implementation described in the previous chapter.

Finally, the last section of this chapter will discuss the outcome of the application implemented in AMT as detailed in chapter five.

6.2 Development and Testing Methodology

The goal of every software development project is to develop an application that provides the best functionality possible. It also has to be well designed, error-free and reliable and - if possible - fast.

In order to achieve these objectives, every software application must undergo a number of stages in its development process.

6.2.1 Development methodology

The most common principle in software design and testing is known as the waterfall model. This model is generally illustrated by the following diagram, which also includes references to versioning:

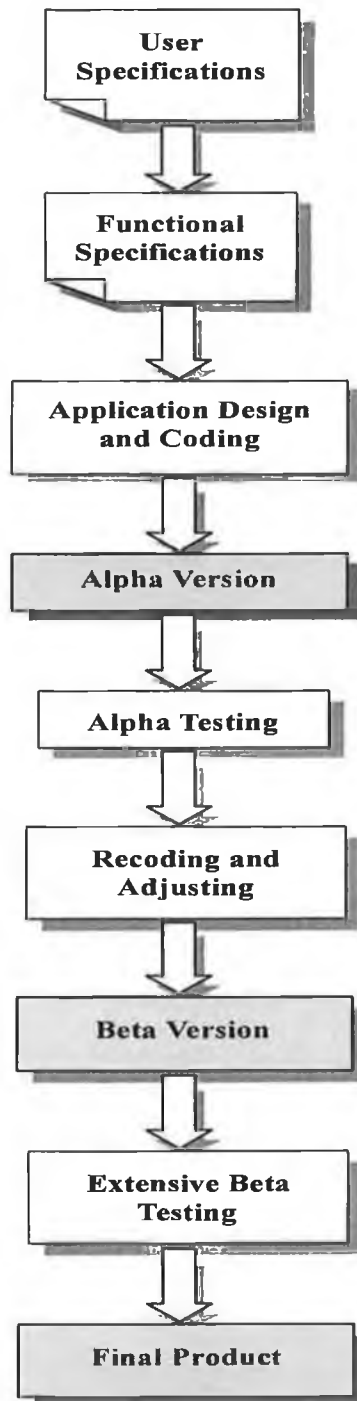


Figure 6.1 – The Waterfall model

User Specifications

This phase is needed in order to define the goal(s) of the application and identify user's requirements. During this phase the application's functionality is outlined and its main functions are sketched.

Functional Specifications

During this phase the specifics of the application from a technical perspective are designed. It provides a rough description of the application as well as describing the technical means used to achieve the design. Some authors separate this phase into two different phases, namely Architectural Design Specification and Detailed Design Specification [IPL 1996].

Application Design and Coding

During this phase the application is coded. Also known as the prototyping stage, it is the stage when most of the design work takes place. The result of this phase is the *Alpha Version* of the application, which is basically a prototype.

Alpha Testing

This is the stage where the initial version is tested. As a result, most of the bugs (program malfunctions) are identified and new functionality is recommended for implementation. This phase is usually accomplished by using a small number of testers, people specialised in the area for which the program is designed.

Recoding and Adjusting

In this phase the bugs identified in the previous phase are eliminated and the new recommended functionality is added to the application. The result of this phase is a new version of the application called *Beta Version* that has added functionality and improved reliability and speed of execution.

Extensive Beta Testing

This is the phase where the application is released to a fairly large number of users called Beta testers in order to assess its potential. Sometimes, the application is simply released to the public and the feedback used to make the final touches and adjustments. It is also at this stage that the debugging process is almost finalised. The result of this stage is the *Final Product*.

After the final product has been released, maintenance and further development will still be performed on the product, mainly by releasing patches (service packs) that will fix bugs (problems) for the existing version or by releasing new versions.

Usually, the *Alpha* and *Beta* testing phases are the longest phases of the project. They are conducted using procedures designed according to applications objectives as defined in the first two phases (User Specifications and Functional Specifications).

6.2.2 Testing methodologies

Software testing is defined as the process of executing software in a controlled manner, in order to determine whether the software application behave as specified [IPL 1996].

The task of testing software applications is a difficult and time-consuming process, requiring technical sophistication and proper planning. This process usually involves modelling the software's environment, selecting the testing scenarios, running and evaluating them and measuring testing progress [Whittaker 2000].

An important approach in software testing is by analysing the application from two perspectives [IPL 1996]:

- The *Static Analysis* investigates the source code of the software, analysing its structure and looking for problems without actually executing the code.
- The *Dynamic Analysis* looks at the behaviour of the software while it is executing in order to provide information such as execution traces, timing profiles and test coverage information.

The testing of software applications by performing a *static analysis* can be conducted by either manually going through the code in an attempt to identify errors, or by using an automated process and specially developed software tools such as AdaTEST or Cantata++ developed by IPL*. These tools are used mainly for parsing through the application code (written in C++) and discovering problems in the design stage.

The *dynamic analysis* is mainly performed in a manual fashion, by running the application and evaluating its functionality according to a set of specifications. It also determines abnormalities (bugs) in the application's behaviour. For web-based, networking or scripting type of applications there are also tools and procedures set in place in order to automate the testing process [Zambelich 2000a, Zambelich 2000b].

Although a big number of general testing methodologies have been suggested and discussed during the time, it is hard to determine the best testing procedures for a general type of application. Considering the rapid development in the software area and the

* A UK-based software and systems company located in the city of Bath.

diversity of platforms, operating systems and programming languages used for applications design, it has been suggested that it is very difficult if not impossible to create tools and general procedures for testing [Biggs 1999]. This is especially true for web applications that employ concepts like 3-tier design. By definition web applications must be able to display content and extend their full functionality on a number of browsers that are most used on the Internet (typically Internet Explorer and Netscape) running on different platforms. Their design must also integrate different types of applications and standards defined by current web server technology (like ASP[†], JSP[‡] or PHP[§]). In addition to that 3-tier applications have to integrate additional standards in order to interface with different types of databases such as Oracle, Sybase, MS-SQL, ODBC etc.

The testing methodology for a generic application can be approached in two ways. The first approach is from the user's perspective, while the second is from an application design perspective.

User Perspective

This aspect of the testing methodology relies to the User Requirements as the basis for defining the testing procedure. From this perspective, the application must comply with the following set of general requirements:

- **Functionality.** This tests application's ability to provide the required functionality as defined in the application's objectives. It also evaluates application's ability to provide less important functions like, for instance, ascending or descending ordering of items in a list using different criteria and other such small tasks that are not essential to the functioning of the application but are appreciated by users, and usually greatly increase users' satisfaction.
- **Usability.** This relates to the ease of use of the application. It essentially evaluates how easy it is to use the functionality provided, how intuitive is the application and how much help is provided in order to make the application more accessible.
- **Reliability.** This tests application's robustness and ensures that it performs well under different testing conditions. For a web application this ensures that the application runs properly using different testing environments and different browsers as well as different types of data input.
- **Speed.** This evaluates application's ability to perform its functions at a reasonable rate. For a web application, this relates to application's ability to deliver the content in a

[†] Active Server Pages, a technology developed by Microsoft

[‡] Java Server Pages, a technology developed by Sun Microsystems

[§] Recursive acronym for "PHP: Hypertext Preprocessor"

timely manner. For this type of application delays in delivering the content may appear as a result of cumulated delays introduced by different parts of the application. For instance, in a 3-tier architecture, the application cumulates the delays introduced by database access. Indeed, since databases may be accessed using dedicated database server programs running on different machines, they are communicating with the application server through the internal network, or they might even reside on the Internet. By running a large number of concurrent queries to the database, the application server might encounter delays in retrieving the requested data. Other delays are also introduced by low speeds for application code execution (either due to heavy server loads or poor code design) and, also, another important factor is the low traffic bandwidth on the Internet when delivering content to users.

Application design perspective

This evaluates application design as the means for testing application's ability to perform its functions. It generally involves complex analysing procedures and it is mainly applicable to software applications designed using advanced programming languages such as C++, Java or Delphi. One such type of analysis is the testing methodology using the Cyclomatic Complexity Metric. This testing method consists of drawing the control flow graph for that application and based on that to determine the Cyclomatic Complexity. Upon determining the Cyclomatic Complexity, the next step is to redesign the application structure so that the number of paths needed to cover all the control branches in the module is reduced to a minimum. This lowest number of paths required is equal to the Cyclomatic Complexity number for that given design/graph [Watson & McCabe 1996].

One useful area for Web application design and testing using this method is by applying it in the design of executable (binary) modules running inside web applications using CGI, applications that were generated using complex programming languages like C++ (or other such programming tools that generate compiled code).

While this is a complex and efficient method of evaluating and improving application design, it is hardly applicable for pure Web application design. This is due to the fact that the design of such programs is much more complex and typically involves more than one programming language (scripting language). For instance a Web application usually implies the use of at least one markup language (typically HTML) and might also have its server-side functionality achieved by the use of PerlScript code and its client-side functionality achieved by using JavaScript. In addition to that, some Web applications also interface with a database by connecting to a database server such as Oracle, Microsoft SQL

Server, MySQL etc. and the data transactions are performed by the use of SQL (Structured Query Language). For these reasons, the Cyclomatic Complexity Metric cannot be applied in this case, because even if it could provide some useful estimate for a scripting Web application, that estimate would only be for one small part not for the whole application,

The next section details the development and testing phase for the initial prototype developed during the course of this project, prototype described in chapter four.

6.3 Prototype Testing

The prototype wireless application described in chapter four was tested by analysing the application from the following perspectives:

- The amount of functionality provided by the application to the users
- The functional analysis of the application, performed in order to determine application speed, reliability and number of supported devices.

In order to design an effective WAP software application, specifications that met the functionality requirements for that specific type of application were designed. The design of the application was shaped by the use of a flow diagram, outlining the screen flow and the main functions of the user interface.

The next phase was the development of the code. General programming rules applied in this phase, which ensured that the application code was designed in a proper manner. For instance, special precautions were taken to ensure that the code was comprehensible, so that subsequent modifications and improvements could be easily made.

After the coding phase was completed, extensive testing was conducted to ensure that the software complied with the specifications and that the required functionality was present and available in different conditions and using different client browsing capabilities.

The first release (alpha version) of the prototype application was developed in approximately 7 weeks. This interval was divided as follows:

- The first phase took 2 weeks and three people were involved in designing the specifications and outlining a generic flow diagram.
- The code-designing phase involved only one programmer and took approximately 5 weeks. During this period both server-side scripting, the markup code and client-side scripting code was designed.

The design of the application made use of the following languages and technologies:

- The application was designed to be hosted on a IIS 4.0 Web Server running on a Windows NT 4.0 Workstation platform.
- Active Server Pages technology was used as the format in which the application was designed.
- The Markup language used was WML (Wireless Markup Language) as defined by the WAP specifications.
- Server-side functionality was achieved by the use of VBScript (Visual Basic Scripting Edition language from Microsoft).
- Client-side functionality was achieved by the use of WMLscript, as defined by the WAP specifications.
- Database connectivity was achieved by employing ADO (ActiveX Data Objects) for interfacing with the Microsoft Jet Engine. The MS Jet Engine is the actual database server, running on the same machine as the WEB server. It connects to a MS Access database also mapped on a local path (located on the same machine).
- The devices used for testing and debugging the code during the design phase were the Blueprint Phone and the Nokia 7110 emulators included in the Nokia WAP Toolkit 2.0 pack.

The testing of the prototype application was performed in two stages:

- Design-time testing. As already mentioned, the design process was based on a trial-and-error method. This method, employed by a large number of programmers at present, consists of designing small portions of code and attempt to run the code and display the results on the mobile device. By using this method, a large number of bugs and malfunctions were eliminated in the design stage.
- Trial testing. This testing stage took place after the initial development of the software and helped eliminate most of the remaining application bugs through subsequent patches. The testing covered the following aspects:
 - ▶ Test of all of the functionality provided by the application (all the major and minor functions provided with the application).
 - ▶ Test of all the input forms by using different combinations for data input, including erroneous values.
 - ▶ It also involved repetition testing, which means that the main functions were repeatedly tested for a large number of times, as much as it was possible considering that this was a manual testing process (i.e. not automated). The tests

were performed manually since the designing of automated testing software is beyond the scope of this project.

After extensive testing, four major types of errors generated by the system were identified:

- WML errors, which were reported by the Nokia Toolkit phone emulator when syntax errors were encountered while parsing the WML code in the attempt to display it on the phone emulator.
- WMLscript errors which resulted in one of the following:
 - ▶ Either the request was suddenly dropped without a reasonable explanation or
 - ▶ Sometimes cryptic error messages were displayed by the handset emulator or
 - ▶ Most often the card would be processed and loaded with no error message at all, but the required functionality was not present.
- VBscript errors, displayed by the VBscript engine running the code on the WEB server. These were mainly compilation errors due to either syntax errors or incorrect variables or object usage. Sometimes different errors occurred, resulted from the processing of the VBscript code. It is unclear whether these errors appeared in the compilation of the code or in the execution stage, since the web server did not provide any relevant details about these errors. These errors later proved to be bugs in the Web server program itself.
- The third type were data access errors displayed by the ADO engine running on the machine in conjunction with the web server. These were either SQL syntax errors or database access errors (including lack of sufficient rights for performing various transactions or inconsistent requests to the database)

During the three months intensive testing phase, this first version proved to deliver a reasonable amount of functionality. After intensive testing using PC-based emulators and a small number of WAP-enabled mobile phones, a number of bugs were identified and removed. It was also decided that new functionality should be added in order to improve the application. Subsequently, a number of features were added, improving the functionality of the software and enhancing the capability of delivering useful content. One of these features was the *Modify Settings* section, allowing the user to change its account information needed for connecting to the site.

The second version of the software was released after a period of 3 months of testing and the new functionality included:

- Order Status
- Improved security features

- Enhanced error-handling routines embedded within the application, providing users with more detailed error information or preventing the user from submitting malformed or erroneous data.

The tools and devices used for testing the application fall into two categories:

- PC-based WAP mobile phone software emulators and Web-based phone emulator applications.
- Real WAP-enabled mobile phone devices accessing the Internet by using a WAP connection.

Mobile phone emulators (whether PC-based or web-based) offer the possibility to test the application on numerous types of devices using a large number of gateways or gateway emulators. Although they are a cheap alternative, they do not usually provide the best testing results. Mobile devices (mobile phones) on the other hand, although they offer the most accurate and reliable test results, are expensive to buy and use because of the purchasing price and subscription fees, and only offer the possibility of testing the application with the use of a very small number of gateways (one gateway for each mobile operator that has been subscribed to). Having said that, it comes as a logical conclusion that the best approach to test an application is to use both types of devices and use the results obtained in a rational fashion.

Another important issue that needs to be discussed here is the fact that there are two major types of WAP microbrowsers used in mobile phones.

- The Nokia browser – a browser developed by Nokia Corporation for use in its own mobile phones.
- The UPbrowser – developed by Phone.Com, an independent software corporation more recently known as Openwave, designing customised WAP microbrowsers for the rest of the major mobile phone manufacturing corporations (such as Motorola, Siemens, Ericsson or Alcatel).

There are some differences in the way these two microbrowsers function. The most notable difference is in the way the two browsers display content. While Nokia browsers have a better graphical appearance and display the content in an organised fashion similar to the PC-based web browsers, the Phone.Com's UPbrowser does not always display the content as it is expected according to the WML card design. Due to the fact that the UPbrowser needs to be implemented on a large number of mobile devices that have different specifications and capabilities (as opposed to the Nokia browser which only need to accommodate a small amount of mobile phone models, the ones manufactures by the

Nokia Corporation), it does not take full advantage of devices' enhanced graphics capabilities (if they are available) nor uses too much processing power or memory. As a result, the UPbrowser tries to cope with little resources by displaying the content in a different way. For instance, if a card contains some text and a selection item (similar to a drop-down box used to make a selection from a list of items), the card containing the text and the list will be rendered on the screen separately. This creates a noticeable difference in the way content is displayed and functionality is organised.

The following device types were used for compatibility testing of the developed application.

PC-based and web-based software emulators used for development and testing:

- Nokia Blueprint phone emulator (Nokia WAP Toolkit 2.0).
- Nokia 7110 WAP phone emulator (Nokia WAP Toolkit 2.0).
- Nokia 6210 WAP phone emulator (Nokia WAP Toolkit 2.0).
- UP.Simulator 4.1 WAP mobile phone emulator (included with UP.SDK Release 4.1 Toolkit from Openwave Systems Inc.)
- Ericsson R520m WAP mobile phone emulator (WapIDE 3.0 from Ericsson Radio Systems AB)
- Nokia 7110, 6210, Wapsilon PDA and the web page version WAP emulators available as a Web application developed by Convolution, a Dutch web design agency (<http://wapsilon.com>).

In addition to these WAP phone emulators, the application code was also tested for XML conformance to standards according to W3C Recommendations. The test was performed using the W3C Validation Service [Oskoboiny 2001].

Mobile phone devices used for testing:

- Motorola Talkabout Wap-enabled mobile phone (employing Phone.Com's UPbrowser)
- Nokia 7110 mobile phone (Nokia browser)
- Nokia 6210 mobile phone (Nokia browser)

Tests using these mobile devices and different configurations were conducted by the MOTTO project team members and Nortel Networks researchers from Ireland, USA and UK. The results of the testing will be discussed in the last section of this chapter and, in the form of conclusions, in the next chapter.

6.4 User Testing and Validation for the AMT Application

As already described in chapter five, The AMT application consists of independent applications interacting with the same database. From a functional perspective, the application can be divided into three parts:

- Administrator Web view.
- Customer Web view
- Customer WAP view

As such, the application must be tested in three independent stages and user feedback must be specifically targeted towards the part of the application that they have experienced.

6.4.1 Application development details

The Web application (consisting of both the administrator and customer Web views) is composed of the following files:

- 64 ASP files
- one file containing global application functions (global.asa)
- one file containing the style sheet for the application (AMT.css)
- one file containing the ADO constants definition (adovbs.inc)
- one file containing the counter information needed during the normal functioning of the application (counter.txt)
- 17 GIF image files and 9 JPEG image files providing the graphics for the Web application
- two MS Access database files. The first one, called AMT.mdb, is used by the application for storing the users table and the orders table. The other database file, called Log.mdb, is used for storing log information in a table called Events (for the purpose of tracking application usage information). Both files are stored in a separate folder called DB, which is configured through the Web server's settings with the anonymous access disabled (the server does not grant direct access to that folder to users that are not properly authenticated by the server).
- An application error log file (in text format) called ErrLog.txt
- 5 JavaScript files (with the extension .js) for performing common client-side tasks and 5 include files (files that are used to load the scripts into the client machine's browser)

Overall, the Web application consists of 107 different files of which 26 image files (in the GIF or JPEG format), two MS Access database files and two text files. The remaining 77 code files contain approximately 6500 lines of code.

The WAP application includes the following files:

- 17 ASP files
- one file containing the ADO constants definition (adovbs.inc)
- one file containing the WML scripts required for providing client-side procedural logic (called scripts.wmls)
- 7 Wireless Bitmap images (having the extension .wmls)

Overall, the WAP application consists of 26 different files of which 7 image files in the Wireless Bitmap format. The remaining 19 code files contain approximately 1250 lines of code.

The application was developed using EditPlus text editor program ver.2.10c as the tool for writing the code. The advantages offered by this application include customized plug-ins for editing ASP pages and VBscript code, support for designing XML-compliant and specifically WML code, and for the web application JavaScript code. Since WMLscript code is in many respects similar to the JavaScript code, the existing plug-in was successfully used.

The development process that resulted in the release of the alpha version of the application took approximately three months. The user specification design took approximately 3 weeks and involved 4 people, three MOTTO project researchers and one AMT employee. The rest of the time was used by the design of the functional specifications (1 week) and by the application design and coding itself.

In order for the application to function properly and to provide the ability to manage the data in the database the administrator view was the first one designed. The second part of the application in the order of development was the customer Web view and the last part added to the application was the WAP view.

The initial release of the alpha version of the application included the following functionality:

Administrator Web view:

- The ability to list all the Requests for Quotation and manage them (modify details, promote as full orders, etc.).
- The ability to list all the orders recorded in the system's database and manage them (modify details, remove, etc.).

- The ability to list all the user accounts recognised by the system and stored into the system's database, and manage these accounts (including changing the details such as usernames, passwords and any other details related to the account).

Customer Web view:

- The ability to submit Requests for Quotation (RFQs) and providing details about them.
- The ability to obtain order status, which means obtaining a list of submitted orders and RFQs submitted by that specific user and view all the relevant details pertaining to that order.
- The ability to modify user's own account settings (which implied editing the values for username, password and all other relevant details except for user's company).

Customer WAP view:

- The ability to view a list of submitted orders and Requests for Quotation (RFQs) and obtaining details about them.
- The ability to view information about the application (application's "about" page)
- The ability to change account details (such as username, password, email address, phone, fax and mobile number).
- The ability to view a help page about application's functionality.

After the initial release of the alpha version and its deployment on the AMT server, feedback regarding application's functionality was received from the system's users. In response to the feedback received, new functionality was incorporated into the application as follows:

Administrator Web view:

- The ability to add orders and edit their details (by selecting the user and company on behalf which the order should appear as being submitted).
- Logging facility for keeping track of application usage (including numerous details about the client's machine, the username and passwords used for connection, internet address and the date and time of the connection)
- Hit counter for the application and an indicator providing the number of users connected to the application at a given time.

Customer Web view:

- The ability to activate Requests for Quotation (promote RFQs to the status of active orders) and edit order details.

The new functionality was developed over a period of about three months and involved visits to the AMT Ireland facility in Limerick and interviews with AMT employees.

In order to achieve a better control over the application development and deployment, a direct encrypted connection to AMT Ireland's Web server was set in place, connecting the developer's computer located in Galway to the AMT Web server located in Limerick. The application used to achieve direct desktop connectivity was the VNC (Virtual Network Computing) software package. VNC is a remote display system consisting of a server application running on the target machine (the machine that is going to be accessed and controlled) and a client application running on the source machine (the machine that is going to initiate the connection and access the target). The application allows the user working on the source machine to initiate a connection and to view the 'desktop' environment on the target machine where the server application is running. The connection can be established over the Internet, on a wide variety of machine architectures and can be encrypted using a number of powerful encryption algorithms. The software package was developed by the Department of Engineering at the University of Cambridge in UK, in collaboration with AT&T. The VNC package used was the Windows-based package, ver. 3.3.3r9, released on 19 March 2001.

Using the remote connection established using the VNC software, a number of patches and updates could be performed remotely on the implemented application hosted on AMT Ireland's Web server in Limerick. The number of updates and the dates at which they were performed are as follows:

- 16 Feb. 2002; this was the first patch after the initial release of the application at the beginning of February 2002-08-03
- 20 Feb. 2002; subsequent fix.
- 22 Feb. 2002; implementation of the beta version.
- 1-st March 2002; this concludes the implementation of the beta version. This was a major update to the application, adding new functionality including the hit counter and the application log.
- 10 March 2002; critical application patch.
- 18 May 2002; application patch.
- 9 July 2002; non-critical application patch.

The patches and fixes to the application were considered critical if the fault that they prevented were essential in the application functioning (fatal application errors such as the inability to display the orders list in the administrator view), and were considered non-critical if the application's functioning would remain unaltered except for error or warning messages displayed when accessing application's functionality. In between, a whole range

of other faults affecting the functionality (faults that were more or less important to the functioning of the application), were remedied through ordinary patches.

6.4.2 Application testing

By far, the most complex part of the application is the administrator Web view. As such, this part was tested extensively in order to ensure that it functions properly and that at the very least does not corrupt the data in the database.

The testing of the AMT application was conducted in a similar fashion as the prototype application, with the notable difference that in this case, not only the WAP application needed to be tested, but also the Web functionality of the application. The testing procedures performed involved the use of the following devices and browsers:

- For the Web part of the application:
 - ▶ Internet Explorer versions 4.0, 5.0, 5.5 and 6.0 (from Microsoft).
 - ▶ Netscape Communicator ver. 6.2 (from Netscape).
 - ▶ Opera Web browser ver. 6.02 (from Opera Software).
- For the WAP part of the application:
 - Software emulators:
 - Nokia Blueprint phone emulator (Nokia WAP Toolkit 2.0).
 - Nokia 7110 WAP phone emulator (Nokia WAP Toolkit 2.0).
 - Nokia 6210 WAP phone emulator (Nokia WAP Toolkit 2.0).
 - Nokia Mobile Browser ver. 3.0.1 mobile device emulator (Nokia Mobile Internet Toolkit ver. 3.1).
 - Nokia 6210 mobile phone simulator (Nokia Mobile Internet Toolkit ver. 3.1).
 - Nokia 6590 mobile phone simulator (Nokia Mobile Internet Toolkit ver. 3.1).
 - Nokia 8310 mobile phone simulator (Nokia Mobile Internet Toolkit ver. 3.1).
 - Nokia 3330 mobile phone simulator (Nokia Mobile Internet Toolkit ver. 3.1).
 - Nokia 7110 mobile phone simulator (Nokia Mobile Internet Toolkit ver. 3.1).
 - UP.Simulator 4.1 WAP mobile phone emulator (included with UP.SDK Release 4.1 Toolkit from Openwave Systems Inc.).
 - Ericsson R520m WAP mobile phone emulator (WapIDE 3.0 from Ericsson Radio Systems AB).
 - Nokia 7110, 6210, Wapsilon PDA and the web page version WAP emulators available as a Web application developed by Convolution, a Dutch web design agency (<http://wapsilon.com>).

Real mobile phones:

- Nokia 7110 mobile phone (Nokia browser).
- Nokia 6210 mobile phone (Nokia browser).
- Motorola Talkabout Wap-enabled mobile phone (employing Phone.Com's UPbrowser).
- Siemens M35i mobile phone (UPbrowser).
- Compaq iPaq PDA device with WAP browser installed.

As stated in the application's login page, the Web part of the application was designed to be best accessed with the Internet Explorer Web browser ver 5.5 or higher, since it is the most commonly used and popular browser program on the Internet. Testing, however, was also performed using the Netscape Navigator Web browser (second most popular web browser on the Internet) and the Opera browser.

Since the development process was similar to the initial prototype application in many respects, it employed the same trial-and-error approach. The devices and browsers most commonly used during the development process were:

For the Web part:

- Internet Explorer ver. 6.0 (Microsoft)

For the WAP part:

- The Nokia 7110 mobile phone simulator (Nokia Mobile Internet Toolkit ver. 3.1).
- Nokia Mobile Browser ver. 3.0.1 mobile device emulator (Nokia Mobile Internet Toolkit ver. 3.1).

The trial testing of the application, after it was implemented, consisted of the following issues being tested:

- **Functionality** – testing performed in order to determine whether the application offers sufficient functions.
- **Usability** – testing performed in order to determine whether the functionality provided behaves as expected and whether the application is easy to use and intuitive.
- **Reliability** - testing performed in order to determine whether the application could withstand abuse (mainly in the form of erroneous data entry) and whether it performs similar after a large number of similar requests.
- **Speed** – testing performed in order to determine the application's speed of execution and whether the application performs well under test conditions involving low bandwidth.

Another source of information used to evaluate the application is the data that was automatically gathered and obtained with the use of the application's log facility. This raw data together with the data obtained from the hit counter will be analysed in the following section to provide indications about application usage.

The outcome of the design and testing of the application will be presented in the following section, while the conclusions related to the global process of application development and the overall conclusions obtained from user feedback will be presented in the following chapter.

6.5 Outcome

The two applications developed during the course of this project were designed with different goals in mind. While the initial prototype application was developed with the primary goal of assessing the state of the technology at that time and to determine whether it could constitute the means for developing decent business applications, the second application was designed to be a real life implementation specifically designed and developed to accommodate the needs and requirements of a specific SME company and its customers.

As such, the testing processes for the two applications, although fairly similar, are aimed at obtaining slightly different information. While the initial prototype attempted to find answers regarding the technology and how it can be used for this type of applications, the second application tried to assess the impact of this technology on people and more specifically onto the SME environment.

6.5.1 The prototype application

During the development phase, the lack of technology support for directly integrating support for services such as SMS or email became obvious.

One of the major drawbacks was also the fact that the development was not conducted by using an effective, powerful IDE (Integrated Development Environment) since those environments were not available on the market or they were very expensive. Since at the time of the development WAP technology was new and relatively untested, the development process was conducted using a trial-and-error method. This meant that the ability to identify the problem and eliminate the malfunction resided in pinpointing the exact source of the error. Since the design of the system was based on a number of different technologies and programming languages, this proved to be a complicated task.

Often, error messages were confusing, not supplying enough information as to the cause of the error. This led to difficulties and delays in the development process.

During testing phase the following problems were identified:

- It was discovered that the gateways lack the ability to supply the MSISDN (Mobile Station International Subscriber Directory Number) identifier (the client ID number, which is actually the mobile device's phone number). It was later discovered that this functionality has been disabled by the service providers for privacy reasons.
- The POST method used for submitting information from browsers towards Web servers does not work with WAP. The GET method was used instead. Besides the privacy and security issues derived from this, it also implies that the quantity of data that can be submitted from the client device to the server is greatly limited since the GET method uses the reference (address) of the target web page for embedding and thus submitting the information. Since the addresses requested cannot be very large in size, the data submitted using this method is limited.
- Although according to the specifications and syntactic rules for WML accesses to WMLscript functions from the WAP browser cannot directly pass more than one parameter to the function. If the attempt is made to submit more than one parameter when calling the function, only the first one will be retrieved, the rest will simply be ignored. If more than one parameter needs to be made available to a function, they need to be retrieved from within the WMLscript directly.

Another problem related to the development of the software was the initial inability to test the software using a large number of mobile devices. This problem was partially solved by gaining access to a small number of devices and software emulators.

6.5.2 The AMT application

A number of application users were questioned in order to determine different aspects of the experience they had with the system. The main points of the questionnaire that was made available to the users focused on the following four issues:

- Functionality
- Usability
- Reliability
- Speed

From the user feedback received, the following estimates were obtained:

For the Administrative section:

- The *functionality* provided was rated by users between 75% and 80%
- The *usability* of the application (the ability to interact with the application in an easy and intuitive fashion) was rated between 75% and 85%
- The *speed* of the application was rated between 90% and 95%
- The *reliability* was rated between 65% and 85%

When users were questioned as to what they think are the biggest issues with this part of the application, the answers were centred on these issues in the order of their importance:

- The lack of help for this part of the application
- The absence of a user manual that would provide a detailed description of different functions and features of the application
- The lack of user experience with this type of applications
- The absence of field examples (on-line examples showing how different forms should be completed)

The customer Web view:

- *Functionality* was rated at around 80%
- The *usability* of the application was also rated at about 80%
- The *speed* of the application was rated on average at approximately 95%
- The *reliability* of the application was rated on average at around 90%

The greatest issues related to the usage of this part of the application were identified as

- The lack of help
- The need for a better graphic design

The customer WAP view:

- The *functionality* of the WAP application was rated on average at around 70%
- The *usability* was rated on average at around 40%
- The *speed* was rated at about 95%
- The *reliability* of the WAP application was rated between 45% and 60%

The point that was made by the users is that the WAP application functions very well in what it does. The low scores do not actually reflect the quality of the design of the application, but rather the limits of WAP technology such as low functionality and low transfer speeds.

From the data presented and the feedback received it becomes apparent that the application was generally well received. Although the data obtained suggests that the administrator Web view needs improvement from a reliability perspective and from the on-line and off-line help and support, on average the application was well rated and the users interacting with it were overall satisfied.

The customer Web view was the part of the application that had the best overall rating. Although it still needs some minor adjustments in terms of help and graphic design, this proved to be the fastest and most reliable part of the application.

The customer WAP view is apparently the least appreciated part of the application. Although users had no special complaints about the design of the application and it has been said that it functions very well in what it does, the application was poorly rated altogether. As it has been subsequently learned, this is due to the fact that the usability offered by current mobile technology is far from being perfect and that the reliability of the application was also low. The low reliability factor rises from the difficulties encountered when using this technology, such as the difficulty of establishing WAP connections and maintaining them over longer periods of time. The low usability score is mainly due to the limitations of mobile devices, such as limited input and display capabilities and low processing power and memory for devices. Another factor that was briefly mentioned was the price of WAP usage. While current rates for the WAP service are a bit lower than the ones employed for voice calls, the specifics of using WAP services (such as low transfer speeds and time consuming tasks like inputting text or browsing through multiple data screens) suggest that these rates are not very encouraging at the moment.

A different source of information about application usage constitutes the application log and the hit counter. The hit counter registered over one thousand one hundred hits (1107 hits to be exact) in the period of time since the hit counter was initially implemented into the application (1-st of March 2002) and the time this analysis was performed (end of July 2002). This suggests that the application was not intensively used during that period.

As already described in chapter five, the application log contains detailed information about the users that connected to the application (or attempted to connect), their IP addresses and other technical details about their computer. By analysing the information contained in the log database, the following conclusions could be drawn:

- More than eighty percent of the connections to the application were made using Web browsers.
- Of the remaining approx. twenty percent of wireless connection recorded, about half were performed for testing purposes.
- It was estimated (based on the IP address analysis) that between twenty-five and thirty percent of the Web connections were generated by search engines (Internet search robots) or random hits.
- Of the remaining fifty or fifty-five percent of legitimate Web connections, about half were performed for testing purposes.

The data is better presented in a more suggestive way in the following diagram:

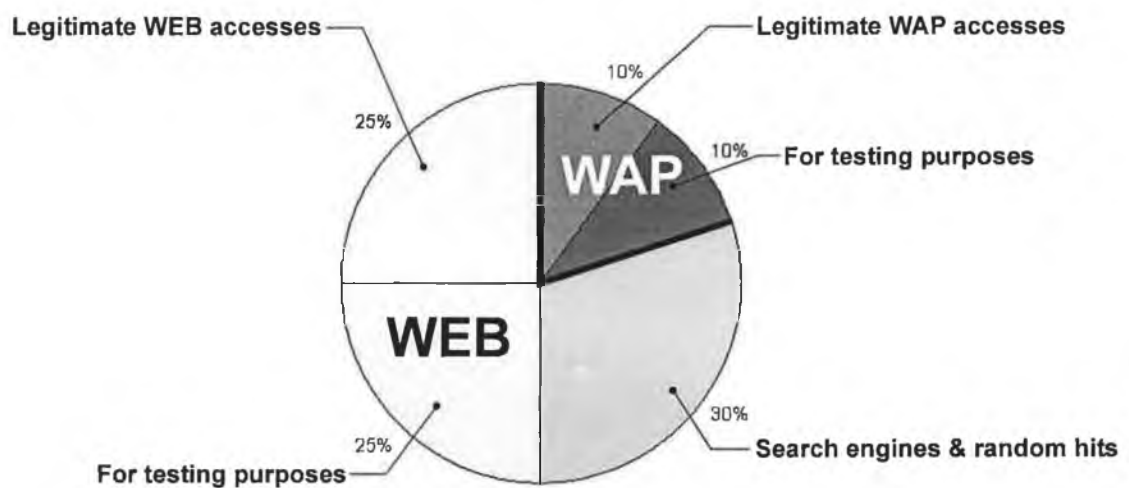


Figure 6.2 – Application usage

From the diagram it becomes apparent that only approximately thirty-five to maybe forty percent of the hits were actual legitimate application accesses, while the rest were either random hits or accesses performed for testing and development purposes. This confirms the previous result obtained by analysing the application hit counter, which is the fact that the application was not intensively used. It was later learned that the SME for which the application was developed (AMT Ireland's Materials Research Lab) was reluctant to promote the application to all of its customers, fearing that the results might prove negative. Instead, they chose to extensively test the application themselves, in order to familiarise themselves with the application and its technology.

The results of the application testing using different devices proved that there are some issues related to device compatibility. While some of the results suggested some incompatibilities between Web browsers (for the Web view), others suggest that there are

some major differences in how mobile devices deal with the WAP application. Tests performed on the Web view of the application proved that some pages in the administrative view do not maintain their appearance, although they maintain their functionality. One single page of the application, namely the *Add Order* page in the administrator view, proved to lose its functionality when accessed using the Netscape browser or the Opera browser. This is not considered a major issue, since the fault is present in the administrative view of the application and since this part of the application is only accessed by a very limited number of users (normally a single user, in charge of administering the application). The other issue relates to the fact that some WAP pages (decks) within the application, after they have been compiled, prove to have a too greater size to be accommodated by certain devices. This is due to the fact that these devices, usually old, do not have enough resources (specifically memory) to accommodate the deck. This results in the deck being rejected and an error message to that effect displayed on the device's screen.

6.6 Summary

The first section of the chapter provided a brief overview of the development and testing methodologies employed for the development of the two application presented in this document. In this section, after presenting the waterfall model as the most used software development model, testing methodologies were presented in an attempt to determine the best approach for testing the two applications.

The second section of the chapter, after detailing the application structure and the development process, it continued by presenting tools the testing methodology employed for application testing and the tools used.

The third section first presented the development process and some details of the application. It then continued by presenting the testing methodology employed and the tools used in both the development process and in the testing phase.

The last section presented and analysed the testing results obtained. The next chapter will present the conclusions drawn from the work done on both development projects. Recommendations for future work will also be made.

CHAPTER 7

Conclusions and Recommendations

7.1 Introduction

7.2 Conclusions

7.3 Recommendations for Future Work

7.4 Summary

7.1 Introduction

This chapter presents the conclusions obtained as a result of the development process of the two applications designed and tested during the course of this project.

The first section starts by separately presenting the conclusions drawn from each development process, the mobile prototype application and the application implemented in AMT Ireland. The section then concludes by presenting the overall conclusions for the thesis.

The second section will present the recommendations for future work and research directions if this work is to be continued through a follow-on project.

7.2 Conclusions

The conclusions reached as a result of the development process and the analysis made will be presented in two parts. The first section will outline the conclusions resulted from the development of the initial prototype application, conclusions that are mainly technical in nature. The second section will detail the conclusions obtained from the AMT Ireland's implementation.

7.2.1 The mobile prototype application

The initial mobile prototype application was designed mainly to assess the state of the technology at the present time and to determine whether it can be effectively used to design wireless business applications. As such, the user feedback obtained was limited and the conclusions are mainly technical in nature.

1) The first conclusion is the fact that the prototype application proved relatively easy to develop. Although the functioning of WAP involves complicated mechanisms such as Wireless Session Protocol (WSP), Wireless Datagram Protocols (WDP), Wireless Transport Layer Security (WTLS) and WAP gateways, these mechanisms are transparent to the WAP developer. All that is required from a developer in order to be able to design wireless applications is:

- background knowledge of the Internet and its protocols and mechanisms
- the functioning of Web servers
- server-side content delivery mechanisms such as ASP or PHP or some other technology capable of delivering active content using server-side processing
- knowledge about VBscript or JavaScript or Perl or some other scripting that can be used for server-side processing
- some notions of Wireless Markup Language (WML) and its associated scripting language (WMLS)

2) The WAP technology proved able to allow a reasonable amount of functionality for business applications.

3) However, some features advertised in the WAP specifications are not functioning because:

- Either they have been disabled by the service providers (as in the case of the MSISDN identifier being disabled by the service provider for privacy reasons, as mentioned in the previous chapter, in the last section), or
- These features have different behaviours on different mobile devices or they haven't been implemented at all in some others (as the case of the WMLscript function parameter submission mentioned in the previous chapter).

4) The biggest problems related to the WAP technology at the moment are:

- The high latency and low bandwidth of the networks as well as delays in establishing the connections
- The functionality provided is scarce and unable to provide rich content.
- The prices for WAP do not encourage usage at the moment

7.2.2 The AMT Implementation

The second application was designed primarily to assess the impact of the technology in the business sector, specifically in the Small-to-Medium sized Enterprises, Business-to-Business Customer Relationship Management area. Therefore, the main objective was to obtain user feedback and application usage statistics and determine whether the technology rises to the demands of today's business applications requirements.

- 1) Assuming that AMT's Materials Research Lab branch can be regarded as a typical SME, then it can be stated that SMEs are still reluctant to adopt the latest cutting-edge technologies. While still tackling the idea of designing Web applications in order to provide their customers with better ways to interact with the company, some SMEs haven't even resolved the issue of designing a static Web page in order to have a presence on the Internet. AMT did not have a clear idea of what WAP can provide for them and how it could be used to provide a better integration in the supply chain.
- 2) As already stated, the goal of this implementation was to provide an SME with a tighter integration with its supply chain by providing it with the ability to use a state-of-the-art Customer Relationship Management system. This was achieved by employing cutting edge technologies like WAP and WML. Although the main purpose of this implementation was to design this state-of-the-art system by employing WAP in order to make use of mobile devices, it became obvious during the course of the development process that in order to implement such an application, the SME first needed to have a basic electronic data management system set in place. This system would help the SME to keep track of its transactions and its clients/suppliers base by employing an easily accessible and updated database.
- 3) In addition to that, in order to implement a wireless application, the SME also needs to have a Web application set in place (as described in chapter three, section 3.5). This is true since the number of Internet users that employ Web (HTML) browsers as the means to navigate the Internet is larger than the number of users taking advantage of mobile phones (WML browsers).
- 4) As a result, the software application implemented actually comprises of three parts.
 - An integrated data management system, enabling the SME to manage its customer base and orders list, and keep track of the latest changes in the status of orders.

- A value-added service enabling its customers to submit requests for quotation and orders, receive status updates and submit order activation commands using the Internet and the World Wide Web.
- A service enabling its customers to obtain status updates for their orders using wireless devices (mobile phones).

While the initial purpose of the development project was just the development of the latter, the development of the additional two integrated applications made the development process more difficult. The overall application, consisting of all these three combined applications, was harder to develop than it was initially estimated, and led to increases in both time and effort needed for the development process.

5) In addition to that, difficulties in the communication process between the developer (Motto project team members) and the SME lead to delays in the development process and misunderstandings, which in turn lead to additional work being unnecessarily performed. This leads to the conclusion that, when developing this type of applications, face-to-face contact between the developer and the SME needs to be achieved.

6) The multitude of devices present on the market makes the development work difficult. Besides the fact that the mobile phone devices available on the market do not share a common standard from a browser point of view, they also have very different capabilities in terms of resources such as memory, display capability and processing power. The most disturbing issue is the memory capacity of different devices, which sometimes cannot accommodate even reasonably sized WML decks. This makes the development of WAP applications capable of accommodating every device on the market almost impossible.

7) Another conclusion is related to the security issues involved. Indeed, since the system is running on a Web server, it is vulnerable to all sorts of attacks that web servers are vulnerable to, such as:

- Denial of service attacks
- The session ID cookie can be forged using a network analyser tool (packet sniffing software) and a malicious user can gain access to the application posing as a legitimate user. This is only possible if the attacker has access to packets that are exchanged between the server and the legitimate user (i.e. is either on the server's local network or on the client's, or has access to some intermediary network routing the packets between the two) and is able to decipher/decrypt the IP packets and identify the session ID cookie. This means that the attacker cannot authenticate with the application on its own, but rather he will have to wait until a legitimate user authenticates, and then pose

as that user. It also means that the attacker does not necessarily need to know the user's credentials in order to be recognised by the system. [Endler 2001]

- In addition to that, the WAP technology has not been designed as a very secure standard, introducing additional security risks.
- 8) Designing an application log function is useful. This feature was used in the application to make records of all connections and connection attempts. This gives administrators the advantage that, in case of a security breach (a non-authorized connection took place), the IP address of the client machine used for this purpose along with a number of other useful details can be obtained. By using these details, further access attempts from that machine to the web server hosting the application can be denied based on the IP address, and further investigation can be conducted to determine the identity of the perpetrator and the extent of damage that has been inflicted to the database. In addition to that, the log is also useful for providing statistic information regarding application usage.
- 9) The Microsoft Windows platform and the related Web technologies proved to facilitate easy development for such applications. In addition to that the costs for the development are low, since most of the software components used are either integrated into the Windows platform or are provided free of charge by Microsoft.
- 10) On the other hand, the downside of this is that sometimes Microsoft technologies prove unreliable. It was discovered that there are some technical issues related to Microsoft technologies such as the VBscript errors generated by the Web server's engine. These errors were sometimes confusing, not providing enough information as to the cause of the error. When an attempt was made to contact Microsoft Customer Support using the Microsoft Developer Network (MSDN), the answer received was that Web application development is a complex task and that Microsoft cannot afford to provide free help with these issues. Instead, it has been suggested to use a Microsoft paid service.

As a conclusion, it has been determined that, usually, these errors are repetitive, they usually do not happen at random, but rather occur every time a certain piece of code runs that contain a certain type of code design, even if that piece of code is well-designed from a syntactical point of view.

7.3 Recommendations for Future Work

The recommendations for future work in the area of WAP and mobile technologies are as follows (in the order of their importance):

- To conduct an in-depth investigation into the area of portable hand-held devices such as PDAs or other such types of devices. The investigation could be conducted in order to determine whether these devices can be used for B2B applications. These devices allow better functionality, better display capabilities and are able to achieve higher connection speeds (by employing up to four standard GSM channels which amount to over 56kbit per second, which is four times the speed that WAP devices such as mobile phones are capable of). In addition to that, most PDA devices are also able to render content in HTML format, which makes them perfectly suited for use with Web applications.
- To conduct research in the area of GPRS. This new technology promises to offer always-on connectivity and also increased transfer speeds, which might prove useful for designing powerful applications that integrate functions such as email and SMS. This technology also seems to be promising from the perspective of cost, with continuous but small amounts of data transfers, since it was suggested that the projected subscription model will charge for the amount of transferred data, not for the time spent on-line. It might also prove to be a useful integration with the hand-held devices technology mentioned in the previous point.
- To conduct research in the area of Bluetooth and 802.11 Wireless Local Area Network (WLAN) technologies. These emerging new technologies could offer a great potential for building powerful business applications. If this technology will evolve, these types of applications could enable customers to use Bluetooth or WLAN devices in order to make purchases or perform data transactions in the vicinity of local access points.
- To conduct an investigation into the emerging Java Phone technology. This might prove useful in order to integrate features like phone book access and other features into mobile applications.
- To conduct research into the area of portals, specifically mobile portals. The research could be directed towards developing software applications, possibly using the Java programming language, able to achieve automatic data updates for portal databases. The data could either be retrieved using existing Internet delivery mechanisms such as email, or it could be obtained by using requests targeted at the information provider's

Web servers running applications capable of delivering active content. The information, initially stored in the provider's databases, could be delivered as HTML content and then, by employing a analysis and filtering mechanism, it could be retrieved and finally stored into the portal's database by using the same Web application update mechanism between the Java application and the portal's Web server. One example of such useful application could constitute a logistics portal, containing up-to-date logistics information that could be subsequently delivered to users by employing mobile devices.

- Based on the experience gained during development of this prototype application, it might prove useful that subsequent developments of WAP software should be conducted using IDEs that are specifically designed for the development and deployment of WAP applications. One of these types of development environments is XMLedge from MobileQ (www.mobileq.com). Among other advantages, this development environment offers developers the ability to develop applications regardless of the capabilities of client devices used to display the content. This is achieved by the use of an integrated database containing all the features and characteristics of a number of common mobile devices currently used on the market, and by using these characteristics to provide content personalized for specific devices.

7.4 Summary

This chapter presented the overall conclusions resulted from this project. While the conclusions in this chapter are divided into two sections, they can be summarised as follows:

- At the moment, the use of WAP technology is limited by its restricted capability of providing good functionality, high data transfer speeds and reasonable display capabilities. Besides its inherent drawbacks, some other functionality is not present due to the fact that either the service providers for privacy reasons have disabled it, or due to the lack of standardisation in the area of mobile devices. Also, WAP tends not to be a very good environment for business applications from the security perspective.
- Many European SMEs and specifically the ones in Ireland are not ready yet to adopt new, cutting-edge technologies such as WAP and even have difficulties in employing simpler technologies such as Web applications.

- The development of WAP applications using Microsoft Windows platforms and Microsoft server-side technologies such as IIS, ASP and ADO is relatively easy to accomplish, but the environment is not absolutely reliable.

Recommendations for future work include research onto the areas of:

- Portable hand-held devices such as PDAs.
- GPRS.
- Bluetooth and 802.11 WLAN technologies.
- Emerging Java Phone technology.
- Portals, specifically mobile portals.
- The use of specifically designed WAP IDEs.

Bibliography

1). Professional WAP

By Charles Arehart, Nirmal Chidambaram, Shashikiran Guruprasad, Alex Homer, Ric Howell, Stephan Kasippillai, Rob Machin, Tom Myers, Alexander Nakhimovsky, Luca Passani, Chris Pedley, Richard Taylor, Marco Toschi
Wrox Press, July 2000
ISBN: 1861004044

2). Learning Wml & Wmlscript

By Martin Frost
O'Reilly & Associates, October 2000
ISBN: 1565929470

3). Core Java 2; Volume 1: Fundamentals

Fourth edition
By Cay S. Horstmann, and Gary Cornell
Sun Microsystems Press, August 2001
ISBN 0-13-081933-6

4). Core Java 2; Volume 2: Advanced Features

Fifth edition
By Cay S. Horstmann and Gary Cornell
Sun Microsystems Press, December 2000
ISBN 0-13-092798-4

5). Exploring Java, 2nd Edition

By Patrick Niemeyer, Joshua Peck

O'Reilly & Associates, September 1997

ISBN: 1565922719

6). Testing Computer Software, 2nd Edition

By Cem Kaner, Jack Falk, Hung Quoc Nguyen

John Wiley & Sons, 1999

ISBN 0-471-35846-0

References

- [Accenture 2001] Glover T Ferguson, Rosemary O'Mahony, Liz Padmore, Karl Saynor, Andrew Sinclair, Mark D.B. Smith and Michael Yates: The Unexpected Europe – The surprising success of European eCommerce.
2001
Document available at:
http://www.accenture.com/xd/xd.asp?it=enWeb&xd=ideas\eeurope2001\eeurope2001_home.xml
- [Alfano 1999] Nick Alfano: Wireless Protocols Group. Working Group Summary
Wireless Application Protocol Forum Ltd.
3 Feb 1999
Document available at:
<http://www1.wapforum.org/member/developers/slides/wireless-protocols-group/sld001.htm>
- [Asmussen 2002] Christian Geisler Asmussen: Amazon back in the red - and what it means for European eBusiness.
IDC Doc #VWP000096
Apr 2002

<http://www.idc.com/getdoc.jsp?containerId=VWP000096>

- [Barnard 2001a] Chris Barnard: Monetizing Mobile: July 2001
IDC Newsletter #LM27H
July 2001
- [Barnard 2001b] Chris Barnard, Claudia Lonardi, Giuliana Folco, Pim
Bilderbeek: Mobile in Travel: Prospects for Western Europe.
IDC Doc #LM03H
Dec 2001
- [Barnard et.al. 2001] Chris Barnard, Pim Bilderbeek, Barbara Blesio, Daniele
Bonfanti, Elena Carreri, Tim Sheedy: Banking on mobile:
Prospects for Western Europe.
IDC Report #LM01H
May 2001
- [Bell 2001] Jonathan Bell: Western European Mobile Portals: Opening Up
to New Revenue Opportunities.
Pyramid Research
September 2001
- [Berners-Lee 1990a] Tim Berners-Lee, Robert Cailliau: WorldWideWeb: Proposal
for a HyperText Project.
12 November 1990
Document available at: <http://www.w3.org/Proposal>
- [Berners-Lee 1990b] Tim Berners-Lee, CERN: Information Management: A
Proposal.
March 1989, May 1990
Document available at:
<http://www.w3.org/History/1989/proposal.html>

- [Bigelow 2001] Keith Bigelow: Are Device Independent Wireless Internet Applications Possible?
06/11/2001
Document available at:
http://www.onjava.com/pub/a/onjava/2001/06/11/device_ind.html
- [Biggs 1999] Maggie Biggs: "Enterprise application testing requires more integrated solutions"
CNN.com
March 9, 1999
Document available at:
<http://www.cnn.com/TECH/computing/9903/09/apptest.ent.idg>
- [Blexrud 1999] Chris Blexrud, Andrea Chiarelli, Dan Denault, Dino Esposito, Brian Francis, Mathew Gibbs, Alex Homer, Bill Kropog, Craig McQueen, George Reilly, Simon Robinson, John Schenken, Dean Sonderegger, David Sussman: "Professional Active Server Pages 3.0"
Wrox Press, October 1999
ISBN 1861002610
- [Booth 2000] Bill Booth: The Truth About Electronic Marketing
2000
Document available at:
<http://www.howtoadvice.com/ElectronicMarketing>
- [B2Bdiversity 2000] Glossary and definitions
B2Bdiversity web site:
<http://www.b2bdiversity.com/help/glossary.cfm>
- [Butler 2002] Valerie Butler: Implementation of a Customer Relationship Management System in an SME.
Master of Science Degree thesis

August 2002

Galway-Mayo Institute of Technology, Ireland.

- [Clarke 1997] Roger Clarke: Electronic Publishing: A Specialised Form of Electronic Commerce
Principal, Xamax Consultancy Pty Ltd, Canberra
Version of 8 May 1997
Paper presented at the 10th International Electronic Commerce Conference, Bled, Slovenia, June 1997
Document available at:
<http://www.anu.edu.au/people/Roger.Clarke/EC/Bled97.html>
- [Clarke 1998] Roger Clarke: Electronic Data Interchange (EDI): An Introduction
Principal, Xamax Consultancy Pty Ltd, Canberra
Revision of December 1998
Republished in Business Credit 23-25, October 2001
Document available at:
<http://www.anu.edu.au/people/Roger.Clarke/EC/EDIIntro.html>
- [Clarke 1999a] Roger Clarke: Electronic Commerce Definitions
Revised Definitions of 3 February 1999
Document available at:
<http://www.anu.edu.au/people/Roger.Clarke/EC/ECDefns.html>
- [Clarke 1999b] Roger Clarke: Electronic Trading in Copyright Objects and Its Implications for Universities
Revision of 19 April 1999
Document available at:
<http://www.anu.edu.au/people/Roger.Clarke/EC/ETCU.html>
- [Coffman et.al. 2001a] K. G. Coffman and A. M. Odlyzko: Internet growth: Is there a “Moore’s Law” for data traffic?
AT&T Labs – Research

Revised version, June 4, 2001

Published: *Handbook of Massive Data Sets*, 2001.

Document available at:

<http://www.research.att.com/~amo/doc/recent.html>

- [Coffman et.al. 2001b] K. G. Coffman and A. M. Odlyzko: Growth of the Internet
AT&T Labs – Research
Preliminary version, July 6, 2001
Published: *Optical Fiber Telecommunications IV*, I. P.
Kaminow and T. Li, eds. Academic Press, 2001.
Document available at:
<http://www.research.att.com/~amo/doc/recent.html>

- [Commonwealth 2000] Commonwealth of Australia: COMMONWEALTH
ELECTRONIC PROCUREMENT - implementation strategy
APRIL 2000
IS BN Print: 0 642 75103 x Online: 0 642 75002 s
Document available at:
[http://www.govonline.gov.au/publications/GOL/Eprocurement
Strategy.pdf](http://www.govonline.gov.au/publications/GOL/EprocurementStrategy.pdf)

- [DCI 1999] DCI's Customer Relationship Management Conference and
Exposition - IT events for Business Application.
30.8.1999
<http://www.dci.com/events/crm/>

- [December 1994] John December: Challenges for Web Information Providers.
Computer-Mediated Communication Magazine, Volume 1,
Number 6, Page 8 (October 1, 1994)
The World Wide Web Unleashed (Sams Publishing, 1994).
Document available at:
<http://www.ibiblio.org/cmc/mag/1994/oct/webip.html>

- [Dennis 2001] Sylvia Dennis: Almost 50 percent of online purchases aborted.
Computer User on-line magazine

May 08, 2001

Document available at:

<http://www.computeruser.com/news/01/05/08/news2.html>

[EC 1996]

Commission of the European Communities: Commission Recommendation of 3 April 1996 concerning the definition of small and medium-sized enterprises.

Document: 96/280/EC

Official Journal L 107, 30/04/1996 P. 0004 – 0009

Document available at:

http://europa.eu.int/smartapi/cgi/sga_doc?smartapi!celexapi!prod!CELEXnumdoc&lg=EN&numdoc=31996H0280&model=gui chett

[EC 2001]

Commission of the European Communities: Revised proposal amending Recommendation 96/280/EC concerning the definition of small and medium-sized enterprises.

2001

Document available at:

http://europa.eu.int/comm/enterprise/consultations/sme_definition/documents/com_smes_en.pdf

[EC 2002]

European Commission: Definition of Small and Medium-Sized Enterprises.

22/04/2002.

Document available at:

http://europa.eu.int/comm/enterprise/consultations/sme_definition/

[ECMA 1999]

ECMAScript: A general purpose, cross-platform programming language

3rd Edition.

Standard ECMA-262

ECMAScript Language Specification

December 1999

<http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM>

or <ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf>

[ECP-NL 2000]

Electronic Commerce Platform Nederland: Procedures for
Electronic Purchasing

Version 1.0

15 June 2000

Document available at:

<http://www.ecp.nl/publicatie/publicaties/handbook.PDF>

[Endler 2001]

David Endler: Brute-Force Exploitation of Web Application
Session Ids.

iALERT White Paper

November 1, 2001

[ETSI]

The European Telecommunications Standards Institute

<http://www.etsi.org/>

[FNC 1995]

Federal Networking Council

Internet Monthly Reports, October 1995

IMR collection available at: <ftp://ftp.isi.edu/in-notes/imr/>

[Fooladi 2001]

Pooneh Fooladi, Sophie Janne Mayo: Is the demand for
wireless deployment meeting the hype? Analysis of IDC
demand-side data.

IDC Bulletin #25538

July 2001

[Fujii 2002]

Kenzo Fujii: iMode - The first successful smart phone service
in the world

February 2002

<http://www.fujii.org/biz/csom/imode.html>

- [Gannon 2001] Liam Gannon: The Development of a Mobile Order Entry System
Masters of Engineering Science thesis, May 2001
National University of Ireland, Galway
- [Goldmann 2000] Nahum Goldmann: Developing Extranet Business Communities: A Successful iCommerce model
February 2000
Document available at:
<http://www.csee.umbc.edu/608/spring00/nahum-goldmann-02-18-2000.htm>
- [Gromov 1995] Gregory R. Gromov: History of Internet and WWW: The Roads and Crossroads of Internet History.
1995
Document available at: <http://www.netvalley.com/intval1.html>
- [Groves 2000] Simon Groves: Data gathering using wireless technologies.
XML Europe 2000, Paris, June 2000
Document available at:
<http://www.infoloom.com/gcaconfs/WEB/paris2000/S13-02.HTM>
- [Gupta 2002] Rahul Kumar Gupta: Exploring the World of Application Servers.
May 27, 2002
Document available at:
<http://serverwatch.internet.com/articles/appservers/>
- [Hameleers 2002] Heino Hameleers and Christer Johansson: IP Technology in WCDMA/GSM core networks.
Ericsson Review No. 1, 2002

Document available at:

http://www.ericsson.com/about/publications/review/2002_01/files/2002012.pdf

[Hillebrand 2001]

Rainer Hillebrand, Thomas Wierlemann: Guidelines for the Mobile Internet.

Internet/Intranet Application Style Guide for Mobile Browsers

Ch. 1.4.1. A typical HTTP session.

2001

Document available at:

<http://mobileinternetguide.org/xhtml/ch01s56s57.xhtml>

[Hubbard 1999]

Thomas Hubbard: WAP Architecture. Introduction and Overview

Wireless Application Protocol Forum Ltd.

3 Feb 1999

Document available at:

<http://www1.wapforum.org/member/developers/slides/WAP-Architecture/sld001.htm>

[IDC]

Online Travel Services Set to Boost Mobile Commerce.

IDC Press Release.

23 Jan 2002.

Document available at:

http://www.idc.com/getdoc.jsp?containerId=pr2002_01_23150830

[IETF]

The Internet Engineering Task Force

<http://www.ietf.org/>

[Infonetics 2002a]

“Worldwide VPN Service and Product Expenditures Increase 117% by 2006”

Infonetics Research, Inc. Press Release

June 3, 2002

Document available at:

http://www.infonetics.com/resources/vsus02_vpn.htm

[Infonetics 2002b]

“Application-Layer (SSL) VPN Gateways Begin Shipping;
Market Will Surge to \$871M in 2005”

Infonetics Research, Inc. Press Release

May 29, 2002

Document available at:

http://www.infonetics.com/resources/press_release_vpn_ssl.htm

[IPL 1996]

“An Introduction to Software Testing”

IPL White Papers

IPL Information Processing ltd.

01/08/1996

Document available at: <http://www.iplbath.com/pdf/p0820.pdf>

[ISOC]

The Internet Society

<http://www.isoc.org/>

[Kolsky 2001]

Esteban Kolsky: Customer Service and Loyalty During
Economic Downturns.

Gartner Consulting

Resource ID: 349131

19 November 2001

[Kueter 1999]

Derek Kueter, Robbert Fisher: Business insights in e-
commerce and trusted services

3 March 1999

Published: Future Generation Computer Systems 16 (2000)

373–378

Document available at: <http://www.sciencedirect.com/>

[Lankford 2000]

Published: Information Management & Computer Security,
2000, volume 8, number 1, pages 27 – 30.

MCB University Press [ISSN 0968-5227]

Document available at: <http://www.emerald-library.com>

[Leiner et. al. 2000]

Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff: A Brief History of the Internet.

Internet Society (ISOC)

4 Aug 2000.

Document available at:

<http://www.isoc.org/internet/history/brief.shtml>

[Lorriman 2000]

Greg Lorriman: “Introduction to Multi-tier/N-tier/3-tier Architectures”

2000

Document available at:

<http://www.undu.com/Articles/010131f.html>

[Lyons 2001]

Charles Lyons: Essential Design for Web Professionals.

2001

ISBN 0-13-032161-3

[Mazzi 2001]

Peter Mazzi: 10 Steps to Web-Enable European SMBs.

IDC Bulletin #MM22H

June 2001

[McGarvey 2000]

Robert McGarvey: How To Dotcom

A step-by-step guide to e-commerce

Entrepreneur Press

2000

ISBN 1-891984-18-7

[Microsoft]

Microsoft Corporation’s Web site:

<http://www.microsoft.com>

- [Mobilocity 2001] Fundamentals of M-Business.
White Paper.
Mobilocity, Inc.
May 2001
Document available at:
http://www.mobilocity.com/mi/Mobilocity_Fundamentals2001.pdf
- [Netcraft 2002] Netcraft Web Server Survey
May 16, 2002
Documents available at:
<http://serverwatch.internet.com/netcraft/200007netcraft.html>
<http://www.netcraft.com/survey/>
- [Nordan et.al. 1999] Matthew M. Nordan, Cliff Condon and Abigail Leland:
“Europe's Mobile Internet Opens Up”
A Forrester report
December 1999
- [Odlyzko 2000] Andrew Odlyzko: Internet growth: Myth and reality, use and abuse.
AT&T Labs – Research
Published: *iMP: Information Impacts Magazine*, November 2000.
An updated and slightly revised version has also appeared in *J. Computer Resource Management*, issue 102, Spring 2001, pp. 23-27.
Document available at:
<http://www.research.att.com/~amo/doc/recent.html>
- [Odlyzko-2001] A. M. Odlyzko: Content is not king
Published: *First Monday* 6(2), February 2001.
Document available at:
<http://www.dtc.umn.edu/~odlyzko/doc/recent.html>

- [Oskoboiny 2001] HTML Validation Service
Gerald Oskoboiny
Last updated: 2001/09/19
Application available at: <http://validator.w3.org/>
- [Perman 2000] Stacy Perman: E-tailing Survival Guide: OK, Forget the Whole
Damn Thing
Business 2.0 Magazine
December 2000 Issue
Document available at:
<http://www.business2.com/articles/mag/0,1640,8786.FF.html>
- [Poe 2000] Stephen Poe: WAP: The promise and the reality.
Serverworld Magazine, September 2000 Issue
Document available at:
<http://www.serverworldmagazine.com/hpchronicle/2000/09/wap.shtml>
- [Predescu 2001] Ovidiu Predescu: Using Cocoon to build Web sites for wireless
devices
Hewlett Packard
ApacheCon 2001 conference
Document available at:
<http://www.geocities.com/SiliconValley/Monitor/7464/apachecon-2001/sld002.htm>
- [RFC791] Request for Comments #791
Internet Protocol
DARPA Internet Program
Protocol Specification
Information Sciences Institute
University of Southern California
September 1981

<http://www.ietf.org/rfc/rfc0791.txt?number=791>

[RFC793]

Transmission Control Protocol
DARPA Internet Program
Protocol Specification
Information Sciences Institute
University of Southern California
September 1981

Document available at:

<http://www.ietf.org/rfc/rfc0793.txt?number=793>

[Robinson 2000]

Robin A. Robinson: Customer Relationship Management.
Computer World.
Feb 28, 2000.

Document available at:

<http://www.computerworld.com/softwaretopics/software/apps/story/0,10801,41519,00.html>

[RTD 2000]

RTD Info: In the lead in the wireless world
Magazine for European research
Issue number 26, May 2000

Document available at:

<http://europa.eu.int/comm/research/rtdinfo/en/26/infosoc.html>

[Rytkönen 2000]

Kimmo Rytkönen: Mobile commerce and WML.
XML Europe 2000, Paris, June 2000

Document available at:

<http://www.gca.org/papers/xml europe2000/papers/s13-01.html>

[Scheier 2001]

Robert L. Scheier: Greasing the Wheels Of Web Commerce
Computer World
APR 02, 2001

Document available at:

puterworld.com/softwaretopics/software/appdev/story/0.10801.59090.00.html

- [Sheedy 2001] Tim Sheedy: The mCommerce Phenomenon – Making Mobile Pay.
IDC Report #HW03H
July 2001
- [SUN] Sun Microsystems Website:
<http://www.javasoft.com>
- [Thompson 1999] Maryann Jones Thompson: Global Spotlight: European Net Adoption Spurs Worldwide Growth.
October 1999
Document available at: http://www.e-gateway.net/studies/st_11.html
- [Tornbohm 2002] Cathy Tornbohm: Managing customer relationships wherever customers go.
Gartner Consulting
May 7, 2002
Document available at:
<http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2863754-1,00.html>
- [Veijalainen 2000] Prof. Jari Veijalainen, Prof. Aphrodite Tsalgatidou: Electronic Commerce Transactions in a Mobile Computing Environment. Presented at the International Conference on Information Society in the 21st Century: Emerging Technologies and new challenges
(IS2000), Nov. 5-8, 2000, Aizu-Wakamatzu City, Japan
Document available at:

1100.pdf

- [VICS 1997] White Paper #1
Developed by the Collaborative Planning, Forecasting, and Replenishment VICS Subcommittee.
Voluntary Interindustry Commerce Standards (VICS) association
Updated December 1997
Document available at:
<http://www.cpfr.org/WhitePapers/19971201.html>
- [Vyas 2001] Charul Vyas: U.S. Mobile Commerce Market Forecast and Analysis, 2001-2005.
IDC Report #26230
December 2001
- [Vyas et.al. 2001] Charul Vyas, Callie Nelsen, Troy Bryant: U.S. Wireless Internet Subscriber Forecast and Analysis, 2000-2005.
IDC Report #25160
October 2001
- [WAPForum] The WAP Forum website:
<http://www.wapforum.org>
- [Whittaker 2000] James A. Whittaker: "What Is Software Testing? And Why Is It So Hard?"
February 2000
Document available at:
<http://www.computer.org/software/so2000/pdf/s1070.pdf>
- [Williams 2000a] Bonnie Shebat Williams and Geoffrey Scott-Baker: White Paper: Traditional EDI and XML/EDI: A Reality Check.
March 2000.
Oracle Corporation, UK

Document available at:

http://www.oracle.com/appsnet/technology/integration/collateral/williams_2.pdf

[Williams 2000b]

Whatis.com: searchCRM.com Definitions

Eric Williams

Oct 25, 2000

Document available at:

http://searchcrm.techtarget.com/sDefinition/0,,sid11_gci213567,00.html

[W3C]

The World Wide Web Consortium

<http://www.w3.org/>

[XML]

The World Wide Web Consortium Issues XML 1.0 as a W3C Recommendation. Key Industry Players, Experts Collaborate to Develop Interoperable Data Format for the Web.

Press Release, 10 February 1998

<http://www.w3.org/Press/1998/XML10-REC>

[Zambelich 2000a]

Keith Zambelich: "Totally Data-Driven Automated Testing - A White Paper"

Automated Testing Specialists, Inc

April 2000

Document available at: http://www.sqa-test.com/w_paper1.html

[Zambelich 2000b]

Keith Zambelich: "Using GUI-based Automated Test Tools to Test Legacy Applications"

December 2000

Omnikron Systems, Inc.

Document available at: http://www.sqa-test.com/w_paper2.html

APPENDIX 1

The Prototype Mobile Application Code

This appendix contains the ASP and WMLscript code that constitutes the application. In addition to this code, other files are also needed for the application to run. These files are as follows:

1). Image files in Wireless Bitmap (WBMP) format. For the application to run properly, these images need to be placed in a folder called *Images*, located in the main application folder. The files are presented in the following table (Table A1).









cimru.wbmp	
ei.wbmp	
gmit.wbmp	
help.wbmp	
motto.wbmp	
news.wbmp	
Nortel.wbmp	
ul.wbmp	

Table A1 – The prototype application images

2). One file named *adovbs.inc* which is the Microsoft ADO constants include file for VBScript. The contents of this file are not listed here.

3). The Microsoft Access database file called *prototyp.mdb*. The file is placed in a folder named *DB*, located in the main application folder. The database structure is described by the following images:

Field Name	Data Type	Description
ordern	AutoNumber	Order Number
user_id	Number	The id of the user that's issuing the order
company	Text	The name of the company on which the order is placed
prod_id	Number	The id number of the product being ordered
pieces	Number	Number of items ordered
sdate	Text	Submission Date
stime	Text	Submission Time
status	Text	Actual status of the item
duedate	Date/Time	Date when the item is due to be delivered

Field Properties	
General	Lookup
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	No

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Figure A1 – The *Orders* table in the prototype application database

Field Name	Data Type	Description
prod_id	Number	The id number for product
prod_name	Text	The product name / description
unit_price	Number	The product price per unit
stock	Number	Number of units in stock

Field Properties	
General	Lookup
Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (Duplicates OK)

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Figure A2 – The *Products* table in the prototype application database

Field Name	Data Type	Description
user_id	Number	User's ID number obtained from the GateWay
username	Text	User's name
email	Text	User's eMail address
company	Text	User's company
password	Text	User's password
validation	Yes/No	Allows users to perform transactions

Field Properties	
General	Lookup
Format	Yes/No
Caption	
Default Value	False
Validation Rule	
Validation Text	
Required	No
Indexed	No

The field description is optional. It helps you describe the field and is also displayed in the status bar when you select this field on a form. Press F1 for help on descriptions.

Figure A3 – The Users table in the prototype application database

The following section presents the ASP code files in alphabetical order.

action.asp

```

<!--#include file="dbcon.asp" -->
<!--#include file="Adovbs.inc" -->
<% Response.ContentType = "text/vnd.wap.wml" %>
<%
Dim SQLquery, userid, username, company, page, loc
Dim SearchRes, User
Set SearchRes = Server.CreateObject("ADODB.Recordset")
Set User = Server.CreateObject("ADODB.Recordset")
'SearchRes.Active.Connection = conn
SearchRes.CursorType = adOpenStatic
User.CursorType = adOpenStatic
%>

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<%
    'userid = Request.Cookies("User-Identity-Forward-msisdn")
    userid = Request.QueryString("userid")
    action = Request.QueryString("action")
    search = Request.QueryString("search")
If search <> "" Then
Else

```

```

        search = "*"
End If
If action = "list" OR search = "*" Then
SQLquery = "SELECT * FROM products WHERE prod_id > 0"
Else
SQLquery = "SELECT * FROM products WHERE prod_id > 0 AND prod_name LIKE
'%" & search & "%'"
End If
'SearchRes.Open SQLquery
Set SearchRes = conn.Execute(SQLquery)

SQLquery = "SELECT username, company, validation FROM users WHERE user_id
= " & userid
Set User = conn.Execute(SQLquery)
If action = "list" Then
page = " Available products "
Else
page = " Product search results "
End If
%>

<card id="list" title="<%=page%>">
  <do type="prev" label="Back">
    <go href="backmain.asp" method="get">
      <postfield name="userid" value="$(userid)"/>
      <postfield name="loc" value="main.asp"/>
    </go>
  </do>
  <p align="left">
    <select name="prodid">
      <option value='0'>Choose a product</option>
    <%
      Do While Not SearchRes.EOF
        Response.Write "<option value='" & SearchRes("prod_id") &
"'>" & SearchRes("prod_name") & "</option>" & vbcrLf
        SearchRes.MoveNext
      loop
    %>
  </select>
  <anchor title="Details"> Details
    <go href="scripts.wmls#nozero('$(prodid:unescape)')"/>
  </anchor>
  <% '<a href="scripts.wmls#nozero('$(prodid:unescape)')">Details</a> %>
<br/>

```

back.asp

```
<%
    'set the MIME type
        Response.ContentType = "text/vnd.wap.wml"
        Response.buffer = True
        Dim userid
        userid = Request.QueryString("userid")
        loc = Request.QueryString("loc")
    %>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="card1" title=" Please wait... ">
        <onevent type="ontimer">
            <go href="<%=loc%>" method="get">
                <postfield name="userid" value="<%=userid%>" />
            </go>
        </onevent>
        <timer value="1"/>
        <p align="center">
            <br/>
            <br/>
            Loading...
            <br/>
        </p>
    </card>
</wml>
```

backmain.asp

```
<%
    'set the MIME type
        Response.ContentType = "text/vnd.wap.wml"
        Response.buffer = True
        Dim userid, loc
        userid = Request.QueryString("userid")
        loc = Request.QueryString("loc")
    %>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="card1" title=" Please wait... ">
        <onevent type="ontimer">
```

```
<go href="<%=loc%" method="get">
  <postfield name="userid" value="<%=userid%"/>
</go>
</onevent>
<timer value="1"/>
  <p align="center">
    <br/>
    <br/>
    Loading...
    <br/>
  </p>
</card>
</wml>
<%
'SearchRes.Close
'Set SearchRes = Nothing
%>
```

dbcon.asp

```
<%
'Option Explicit
Dim conn
Set conn = Server.CreateObject("ADODB.Connection")
conn.open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("DB/prototyp.mdb")
%>
```

discon.asp

```
<%
conn.close
Set conn = Nothing
%>
```

go.asp

```
<%
'set the MIME type
Response.ContentType = "text/vnd.wap.wml"
Response.buffer = True
Dim userid
userid = Request.QueryString("userid")
loc = Request.QueryString("loc")
%>
```

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1" title=" Please wait... ">
    <onevent type="ontimer">
      <go href="<%=loc%>" method="get">
        <postfield name="userid" value="<%=userid%>" />
      </go>
    </onevent>
    <timer value="1" />
    <p align="center">
      <br />
      <br />
      Loading...
      <br />
    </p>
  </card>
</wml>
<%
SearchRes.Close
Set SearchRes = Nothing
%>
<!--#include file="discon.asp" -->
```

help.asp

```
<%
  'set the MIME type
  Response.ContentType = "text/vnd.wap.wml"
  Dim userid
  userid = Request.QueryString("userid")
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="Help" title="MOTTO Help Page">
    <do type="accept" label="Home page">
      <go href="main.asp" method="get">
        <postfield name="userid" value="$(userid)" />
      </go>
    </do>
    <p align="center">
      <br />
    </p>
  </card>
</wml>
```


Appendix 1

```
SQLquery = "SELECT username, company FROM users WHERE user_id = " &
userid
Set ReqUser = conn.Execute(SQLquery)
If ReqUser.EOF then
username = ""
company = ""
NewUser = "1"
Else
username = ReqUser("username")
company = ReqUser("company")
NewUser = "0"
End If
ReqUser.Close
Set ReqUser = Nothing
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="flip1" title="MOTTO project">
    <onevent type="ontimer">
      <go href="#flip2"/>
    </onevent>
    <timer value="20"/>
    <do type="options" label="Next >">
      <go href="#Welcome"/>
    </do>
    <do type="prev" label=" Back">
      <go href="initiate.asp#card1"/>
    </do>
    <p align="center">
      <br/>
      Sponsored by
      <br/>
      <big>
      Nortel Networks
      </big>
    </p>
  </card>
  <card id="flip2" title="MOTTO project">
    <onevent type="ontimer">
      <go href="#flip1"/>
    </onevent>
    <timer value="20"/>
    <do type="options" label="Next >">
```

```
<go href="#Welcome"/>
</do>
  <do type="prev" label=" Back">
    <go href="initiate.asp#card1"/>
  </do>
<p align="center">
  <br/>
  Managed by
  <br/>
  <big>
  GMT / NUIG
  </big>
</p>
</card>
  <card id="Welcome" title="MOTTO WAP Site">
    <do type="prev" label=" Back">
      <prev/>
    </do>
<% If NewUser = "1" Then %>
  <do type="accept" label="Register">
    <go href="#Register"/>
  </do>
  <p align="center">
    <strong>
      Welcome!<br/>
    </strong>
    <small>
      Since you're a new user
      you should register.<br/>
    </small>
  </p>
<% Else %>
  <do type="accept" label="Login">
    <go href="#Login"/>
  </do>
  <p align="center">
    Hello
    <% If username <> "" Then %>
      <%=username%>
    </p>
    <% Else %>
      there ! <br/>
    </p>
    <p align="left">
      <small>
```

```
        You should consider registering your name into the
database.<br/>
        </small>
        <% If company <> "" Then %>
        <% Else %>
        <small>
        You should also consider registering your company name
into the database.<br/>
        </small>
        <% End If %>
    </p>
<% End if %>
<% If username <> "" Then %>
    <% If company <> "" Then %>
        <p align="center">
        from <%=company%> !<br/>
        </p>
    <% Else %>
        <p align="left">
        <small>
        You should consider registering your company name into
the database.<br/>
        </small>
        </p>
    <% End if %>
<% End if %>
<p align="center">
<small>
Press Login to go
to the Login page.<br/>
</small>
</p>
<% End If%>
</card>
<card id="Register" title="Registration">
    <do type="prev" label=" Back">
        <prev/>
    </do>
    <do type="accept" label="Submit">
        <go href="register.asp" method="get">
            <postfield name="userid" value="$(userid)"/>
            <postfield name="username" value="$(username)"/>
            <postfield name="email" value="$(email)"/>
            <postfield name="company" value="$(company)"/>
            <postfield name="password" value="$(password)"/>
```

```
        <postfield name="rpassword" value="\$(rpassword)"/>
    </go>
</do>
<p align="left">
<fieldset title="Registration info">
    User Name <input title="User Name:" emptyok="false" type="text"
value="" name="username" size="32"/><br/>
    eMail address <input title="eMail address:" emptyok="true"
type="text" value="" name="email" size="32"/><br/>
    Company <input title="Company name:" emptyok="true" type="text"
value="" name="company" size="20"/><br/>
    Password <input title="Password:" emptyok="true"
type="password" value="" name="password" size="20"/><br/>
    Reenter Password: <input title="Reenter Password:"
emptyok="true" type="password" value="<%=rpassword%" name="rpassword"
size="20"/><br/>
</fieldset>
<br/>
<anchor title="Submit"> Submit
    <go href="Register.asp" method="get">
        <postfield name="userid" value="\$(userid)"/>
        <postfield name="username" value="\$(username)"/>
        <postfield name="email" value="\$(email)"/>
        <postfield name="company" value="\$(company)"/>
        <postfield name="password" value="\$(password)"/>
        <postfield name="rpassword" value="\$(rpassword)"/>
    </go>
</anchor>
</p>
</card>
<card id="Login" title="Login">
    <do type="prev" label=" Back">
        <prev/>
    </do>
    <do type="accept" label="Submit">
        <go href="scripts.wmls#check(\$(retry:unesc))"/>
    </do>
    <p align="left">
        <fieldset title="Input data">
            User Name: <input title="User Name:" emptyok="true" type="text"
value="<%=username%" name="username" size="20"/><br/>
            Password: <input title="Password:" emptyok="true" type="password"
value="<%=password%" name="password" size="20"/><br/>
        </fieldset>
    </p>
</br/>
```

```
        </p>
        <p align="center">
        <small>
After three unsuccessful attempts you will be invalidated.
        </small>
        <br/>
        </p>
    </card>
<card id="Submit" title=" Please wait... ">
    <onevent type="ontimer">
        <go href="login.asp" method="get">
            <postfield name="userid" value="{userid}"/>
            <postfield name="username" value="{username}"/>
            <postfield name="password" value="{password}"/>
            <postfield name="retry" value="{retry}"/>
        </go>
    </onevent>
    <timer value="1"/>
<p align="center">
Loading...
</p>
</card>
</wml>
<!--#include file="discon.asp" -->
```

info.asp

```
<%
    'Response.buffer = True
    'set the MIME type
    Response.ContentType = "text/vnd.wap.wml"
    Dim userid
    'requesting userid from client device
    'userid = Request("userid")
    userid = Trim(Request("userid"))
    'userid = 646467464
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="Info" title="Info - Motto project">
<do type="accept" label="Home page">
    <go href="main.asp" method="get">
        <postfield name="userid" value="{userid}"/>
```



```
<%
'Response.buffer = True
'Server.ScriptTimeout = 1
'set the MIME type
    Response.ContentType = "text/vnd.wap.wml"
    Dim userid, faID, SQLquery, RID, unique
Set RID = Server.CreateObject("ADODB.Recordset")
RID.CursorType = adOpenStatic
SQLquery = "SELECT users.user_id FROM users"
Set RID = conn.Execute(SQLquery)
    'requesting userid from client device
    'userid = Request("userid")
    'userid = Trim(Request("userid"))
    'userid = 646467464
unique = 0
faID = 0
    Do While unique = 0
        faID = faID + 1
        unique = 1
        Do While (Not RID.EOF And unique = 1)
            If faID <> RID("user_id") Then
                RID.MoveNext
            Else
                unique = 0
            End If
        loop
        RID.MoveFirst
    loop
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="card1" title="Id corellation">
        <onevent type="onenterforward">
            <refresh>
                <setvar name="retry" value="0"/>
            </refresh>
        </onevent>
        <do type="accept" label="Next">
            <go href="index.asp" method="get">
                <postfield name="userid" value="$(userid)"/>
                <postfield name="retry" value="0"/>
            </go>
        </do>
    </card>
</wml>
```

```

        <p align="left">
        <br/>
        Please input ID #
        <br/>
        <input emptyok="false" type="text" value="<%=userid%>"
name="userid" size="9" format="*N" title="User ID"/><br/>
        <small>
        If you are not registered, the first available User ID number is
        <b>&nbsp;<a href="#use"><%=faID%></a></b>
        </small>
        </p>
    </card>
<card id="use" title="Please wait.">
    <onevent type="onenterforward">
        <refresh>
            <setvar name="userid" value="<%=faID%>"/>
        </refresh>
    </onevent>
    <onevent type="ontimer">
        <go href="index.asp" method="get">
            <postfield name="userid" value="$(userid)"/>
            <postfield name="retry" value="0"/>
        </go>
    </onevent>
    <timer value="1"/>
    <p align="center">
        <big>
        Loading...
        </big>
    </p>
</card>
</wml>
<%
RID.Close
Set RID = Nothing
%>
<!--#include file="discon.asp" -->

```

login.asp

```

<!--#include file="dbcon.asp" -->
<% Response.ContentType = "text/vnd.wap.wml" %>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

```



```
<%
Dim SQLQuery, userid, username, company
Dim LoginR, password, email

'userid = Request.Cookies("User-Identity-Forward-msisdn")
userid = Request.QueryString("userid")
username = Request.QueryString("username")
password = Request.QueryString("password")
retry = Request.QueryString("retry")
SQLQuery = "SELECT username, password FROM users WHERE user_id = " &
userid
Set LoginR = conn.Execute(SQLQuery)
%>
<wml>
  <% If username <> LoginR("username") OR password <>
LoginR("password") Then %>
  <card id="Error" title="Error">
  <do type="prev" label="Back">
    <go href="index.asp#Login" method="get">
      <postfield name="retry" value="<%=retry%>" />
      <postfield name="userid" value="$(userid)" />
    </go>
  </do>
  <p align="center">
  <br/>
  Invalid username or password
  <br/>
  <small>
  Make sure you've typed them correctly
  </small>
  </p>
</card>
  <% If retry = 3 Then
  SQLQuery = "UPDATE users SET users.validation = False WHERE
users.user_id = " & userid & ";"
  conn.execute(SQLQuery)
  %>
  <% End if %>
  <% Else %>
  <card id="OK" title="Success">
  <onevent type="ontimer">
  <go href="main.asp" method="get">
    <postfield name="userid" value="$(userid)" />
  </go>
  </onevent>
```

```
<timer value="10"/>
<do type="options" label="Ok ">
    <go href="main.asp" method="get">
        <postfield name="userid" value="$(userid)"/>
    </go>
</do>
<do type="prev" label=" Back">
    <prev/>
</do>
<p align="center">
    <em>
        Logon successful.
    </em>
    <br/>
    <br/>
    Please wait.<br/>
    Loading...
<%
    '<anchor title="Main page"> Main page
        '<go href="main.asp" method="get">
            '<postfield name="userid" value="$(userid)"/>
        '</go>
    '</anchor>
%>
    </p>
</card>
<% End if %>
</wml>
<%
    LoginR.Close
    Set LoginR = Nothing %>
<!--#include file="discon.asp" -->
```

main.asp

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<%
Dim userid, search
    userid = Request.QueryString("userid")
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="Main" title=" MOTTO Home page ">
<p align="center">
<br/>
```

```
</p>
<p align="left">
<br/>
<anchor title="List"> &gt;&nbsp;&nbsp;&nbsp;List all products
    <go href="action.asp" method="get">
        <postfield name="userid" value="\$(userid)"/>
        <postfield name="action" value="list"/>
        <postfield name="search" value="*" />
    </go>
</anchor><br/>
<anchor title="Search"> &gt;&nbsp;&nbsp;&nbsp;Search products
    <go href="#Search"/>
</anchor><br/>
<anchor title="Status"> &gt;&nbsp;&nbsp;&nbsp;Order status
    <go href="status.asp" method="get">
        <postfield name="userid" value="\$(userid)"/>
    </go>
</anchor><br/>
<anchor title="News"> &gt;&nbsp;&nbsp;&nbsp;News
    <go href="news.asp" method="get">
        <postfield name="userid" value="\$(userid)"/>
    </go>
</anchor><br/>
<anchor title="Info"> &gt;&nbsp;&nbsp;&nbsp;Info
    <go href="info.asp" method="get">
        <postfield name="userid" value="\$(userid)"/>
    </go>
</anchor><br/>
<anchor title="Settings"> &gt;&nbsp;&nbsp;&nbsp;Settings
    <go href="settings.asp" method="get">
        <postfield name="userid" value="\$(userid)"/>
    </go>
</anchor><br/>
</p>
<p align="right">
<a href="Help.asp">Help</a>
<br/>
<br/>
</p>
</card>
<card id="Search" title="Product Search">
<do type="edit" label="Search">
    <go href="action.asp" method="get">
        <postfield name="userid" value="\$(userid)"/>
        <postfield name="action" value="search"/>
```



```

    <p align="center">
        You have either submitted a zero value or the quantity entered
exceeds the stock.<br/>
        <anchor title="Back"> Back ot Order Form
            <go href="#Order"/>
        </anchor>
    </p>
</card>
<card id="Submit" title=" Please wait... ">
    <onevent type="ontimer">
        <go href="submit.asp" method="get">
            <postfield name="userid" value="<%=userid%>"/>
            <postfield name="company" value="<%=company%>"/>
            <postfield name="prodid" value="<%=prodid%>"/>
            <postfield name="qty" value="$(qty)"/>
            <postfield name="stock" value="<%=QProduct("stock")%>"/>
        </go>
    </onevent>
    <timer value="1"/>
<p align="center">
Loading...
</p>
</card>
</wml>
<!--#include file="discon.asp" -->

```

register.asp

```

<!--#include file="dbcon.asp" -->
<% Response.ContentType = "text/vnd.wap.wml" %>
<%
'Response.buffer = True
Dim userid, username, email, company, password, rpassword
Dim SQLquery
userid = Request.QueryString("userid")
username = Request.QueryString("username")
email = Request.QueryString("email")
company = Request.QueryString("company")
password = Request.QueryString("password")
rpassword = Request.QueryString("rpassword")
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

```



```
<wml>
<% If password <> rpassword Then %>
  <card id="Error" title="Error">
    <do type="prev" label=" Back">
      <prev/>
    </do>
    <p align="center">
      Password does not match.
      <br/>
      Please reenter password.
      <br/>
    </p>
  </card>
<% Else
  SQLquery = "INSERT INTO users VALUES (" & userid & ", '" & username
& "', '" & email & "', '" & company & "', '" & password & "', False);"
  'SQLquery = "INSERT INTO users (user_id, username, email, company,
password, validation) VALUES (14, 'JohnDoe', 'John@x.com', 'Sharp', 'ok',
false);"
  'SQLquery = "INSERT INTO users (user_id, username, email, company,
password, validation) VALUES (10, 'John Doe', 'John@x.com', 'Sharp',
'ok', False);"
  'SQLquery = "INSERT INTO users VALUES ()"
  conn.execute(SQLquery)
%>
<card id="card1" title="Registration">
  <do type="options" label="Home page">
    <go href="main.asp" method="get">
      <postfield name="userid" value="$(userid)"/>
    </go>
  </do>
  <p align="center">
    Your registration was succesful <%=username%>
  <br/>
  </p>
</card>
<% End if %>
</wml>
<!--#include file="discon.asp" -->
```

second.asp

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<?xml version="1.0"?>
```

Appendix 1

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1" title="Card #1">
    <do type="prev" label=" Back">
      <prev/>
    </do>
    <p align="center">
      <!-- Card implementation here. -->
      <big><b>OK</b></big><br/>
      1-st card
    </p>
  </card>
  <card id="card2" title="Card #2">
    <do type="prev" label=" Back">
      <prev/>
    </do>
    <p align="center">
      <big><b>OK</b></big><br/>
      2-nd card
    </p>
  </card>
</wml>
```

setsub.asp

```
<!--#include file="dbcon.asp" -->
<% Response.ContentType = "text/vnd.wap.wml" %>
<%
'Response.buffer = True
Dim userid, username, email, company, password
Dim SQLquery
userid = Request.QueryString("userid")
username = Request.QueryString("username")
email = Request.QueryString("email")
company = Request.QueryString("company")
password = Request.QueryString("password")
SQLquery = "UPDATE users SET users.username = '" & username & "' WHERE
users.user_id = " & userid
conn.execute(SQLquery)
SQLquery = "UPDATE users SET users.email = '" & email & "' WHERE
users.user_id = " & userid
conn.execute(SQLquery)
SQLquery = "UPDATE users SET users.company = '" & company & "' WHERE
users.user_id = " & userid
```

```
conn.execute(SQLquery)
SQLquery = "UPDATE users SET users.password = '" & password & "' WHERE
users.user_id = " & userid
conn.execute(SQLquery)
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="Submit" title=" Complete ">
    <onevent type="ontimer">
      <go href="main.asp" method="get">
        <postfield name="userid" value="<%=userid%>" />
      </go>
    </onevent>
    <timer value="600"/>
    <do type="accept" label="Home page">
      <go href="main.asp" method="get">
        <postfield name="userid" value="<%=userid%>" />
      </go>
    </do>
  <p align="center">
  <br/>
  <b>
  Your settings were successfully updated.
  </b>
  </p>
</card>
</wml>
<!--#include file="discon.asp" -->
```

settings.asp

```
<!--#include file="dbcon.asp" -->
<% Response.ContentType = "text/vnd.wap.wml" %>
<%
'Response.buffer = True
Dim userid, username, email, company, password, rpassword
Dim SQLquery, ReqUser
userid = Request.QueryString("userid")
SQLquery = "SELECT username, email, company, password FROM users WHERE
user_id = " & userid
Set ReqUser = conn.Execute(SQLquery)
username = ReqUser("username")
email = ReqUser("email")
```

```
company = ReqUser("company")
password = ReqUser("password")
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="Settings" title=" Modify settings ">
  <do type="prev" label=" Back">
    <prev/>
  </do>
  <p align="left">
    <fieldset title="Settings">
      User Name:&nbsp;<input title="User Name:" emptyok="false"
type="text" value="<%=username%>" name="username" size="32"/><br/>
      eMail address:&nbsp;<input title="eMail address:"
emptyok="true" type="text" value="<%=email%>" name="email"
size="32"/><br/>
      Company:&nbsp;<input title="Company name:" emptyok="true"
type="text" value="<%=company%>" name="company" size="20"/><br/>
      Password:&nbsp;<input title="Password:" emptyok="true"
type="password" value="<%=password%>" name="password" size="20"/><br/>
      Reenter Password:&nbsp;<input title="Reenter Password:"
emptyok="true" type="password" value="<%=password%>" name="rpassword"
size="20"/><br/>
    </fieldset>
  <br/>
  <anchor title="Submit"> Submit
    <go href="scripts.wmls#compare()"/>
  </anchor>
</p>
</card>
<card id="Submit" title=" Please wait... ">
  <onevent type="ontimer">
    <go href="setsub.asp" method="get">
      <postfield name="userid" value="<%=userid%>"/>
      <postfield name="username" value="$(username)"/>
      <postfield name="email" value="$(email)"/>
      <postfield name="company" value="$(company)"/>
      <postfield name="password" value="$(password)"/>
    </go>
  </onevent>
  <timer value="1"/>
<p align="center">
<br/>
```

Loading...

</p>

</card>

</wml>

<!--#include file="discon.asp" -->

stat_det.asp

<!--#include file="dbcon.asp" -->

<%

Response.ContentType = "text/vnd.wap.wml"

Dim SQLquery, userid, ordern

Dim OrdRes, Prod

userid = Request.QueryString("userid")

ordern = Request.QueryString("ordern")

SQLquery = "SELECT * FROM orders WHERE orders.ordern = " & ordern & ";"

Set OrdRes = conn.Execute(SQLquery)

SQLquery = "SELECT * FROM products WHERE products.prod_id = " &

OrdRes("prod_id") & ";"

Set Prod = conn.Execute(SQLquery)

%>

<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

<card id="Details" title="Order #<%=ordern%>">

<do type="prev" label=" Back">

<prev/>

</do>

<do type="options" label="Main Page">

<go href="main.asp" method="get">

<postfield name="userid" value="\$(userid)"/>

</go>

</do>

<p align="left">

<small>

<%

If OrdRes.EOF Then

Response.Write "No details about this order."

Else

%>

User ID: <%=userid%>.

Company: <%=OrdRes("company")%>.

Product ID: <%=OrdRes("prod_id")%>.

Product name: <%=Prod("prod_name")%>.


```

Unit price: <%=Prod("unit_price")%>.<br/>
Quantity ordered: <%=OrdRes("pieces")%>.<br/>
Submit date: <%=OrdRes("sdate")%>.<br/>
Submit time: <%=OrdRes("stime")%>.<br/>
Status: <%=OrdRes("status")%>.<br/>
Due date: <%=OrdRes("duedate")%>.<br/>

```

```

<%
    End If
%>
</small>
</p>
</card>
</wml>
<%
OrdRes.Close
Set OrdRes = Nothing
Prod.Close
Set Prod = Nothing
%>
<!--#include file="discon.asp" -->

```

status.asp

```

<!--#include file="dbcon.asp" -->
<!--#include file="Adovbs.inc" -->
<%
Response.ContentType = "text/vnd.wap.wml"
Dim SQLquery, userid, ordern
Dim OrdRes
userid = Request.QueryString("userid")
Set OrdRes = Server.CreateObject("ADODB.Recordset")
OrdRes.CursorType = adOpenStatic
SQLquery = "SELECT * FROM orders WHERE orders.user_id = " & userid & ";"
Set OrdRes = conn.Execute(SQLquery)
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="Orders" title="Orders">
    <do type="prev" label="Back">
        <go href="main.asp" method="get">
            <postfield name="userid" value="$ (userid)"/>
        </go>
    </do>

```

```
<p align="center">
<small>
Order ID - Submission date
</small>
</p>
<p align="center">
<small>
<%
If OrdRes.EOF Then
    Response.Write "You have no submitted orders"
Else
    Do While Not OrdRes.EOF
        Response.Write "<anchor title='Details'> " &
OrdRes("ordern") & "&nbsp; - &nbsp;";" & OrdRes("sdate") & vbcrLf
        Response.Write "<go href='stat_det.asp' method='get'>" &
vbcrLf
        Response.Write "<postfield name='userid' value='" & userid
& "'/>" & vbcrLf
        Response.Write "<postfield name='ordern' value='" &
OrdRes("ordern") & "'/>" & vbcrLf
        Response.Write "</go>" & vbcrLf
        Response.Write "</anchor><br/>" & vbcrLf
        OrdRes.MoveNext
    loop
End If
%>
<br/>
</small>
Choose an order for details.
<br/>
</p>
</card>
</wml>
<%
OrdRes.Close
Set OrdRes = Nothing
%>
<!--#include file="discon.asp" -->
```

submit.asp

```
<!--#include file="dbcon.asp" -->
<!--#include file="Aovbs.inc" -->
<%
    'Response.buffer = True
```

```
'set the MIME type
  Response.ContentType = "text/vnd.wap.wml"
  Dim SQLquery, userid, company, prodid, qty, stock, Order_n, sdate,
stime, SubOrd, corder
  userid = Request.QueryString("userid")
  company = Request.QueryString("company")
  prodid = Request.QueryString("prodid")
  qty = Request.QueryString("qty")
  stock = Request.QueryString("stock")
SQLquery = "SELECT max([orders].[order_n]) FROM orders;"
Set Order_n = conn.execute(SQLquery)
corder = (Order_n("Expr1000") + 1)
SQLquery = "INSERT INTO orders VALUES (" & corder & ", " & userid & ", '"
& company & "', " & prodid & ", " & qty & ", '" & date & "', '" &
hour(time) & ":" & minute(time) & "', 'Submitted', 01/01/2001);"
conn.execute(SQLquery)
'SQLquery = "INSERT INTO orders VALUES (" & Order_n("Expr1000")+1 & ", "
& userid & ", '" & company & "', " & prodid & ", " & qty & ", 01/01/01,
'2:03', 'Submitted', 01/01/01);"
'conn.execute(SQLquery)
SQLquery = "UPDATE products SET products.stock = " & (stock-qty) & "
WHERE products.prod_id = " & prodid
conn.execute(SQLquery)
'SQLquery = "SELECT (orders.sdate, orders.stime) FROM orders WHERE
(orders.order_n = " & corder & ");"
'Set SubOrd = conn.execute(SQLquery)
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="GoodBye" title="Submission confirmation">
  <do type="accept" label="Home page">
    <go href="main.asp" method="get">
      <postfield name="userid" value="$(userid)"/>
    </go>
  </do>
<p align="center">
<b>
Your order has been submitted.<br/>
</b>
<small>Thank you for shopping at</small><br/>
<br/>
We hope you had an enjoyable experience.<br/>
```



```
<small>
Your order number is: &nbsp;   <%=corder%><br/>
Submission date and time is:<%=Now%>
</small>
</p>
</card>
</wml>
<!--#include file="discon.asp" -->
```

scripts.wmls (containing the WMLscript code)

```
// scripts.wmls
extern function nozero(spId)
{
var pid;
pid = Lang.parseInt(spId);
if (pid > 0)
{WMLBrowser.go("#chkid");}
else
{WMLBrowser.go("#list");}
}

extern function id(sv)
{
if (sv == "True")
{WMLBrowser.go("#Order");}
else
{WMLBrowser.go("#NoOrder");}
}

extern function qty()
{
var sq = WMLBrowser.getVar("qty");
var stock = WMLBrowser.getVar("stock");
var q;
var s;
q = Lang.parseInt(sq);
s = Lang.parseInt(stock);
if ((q < s) && (q >= 0))
{WMLBrowser.go("#Submit");}
else
{WMLBrowser.go("#Error");}
}

extern function compare()
```

```
{
var pass = WMLBrowser.getVar("password");
var rpass = WMLBrowser.getVar("rpassword");
if (pass == rpass)
{WMLBrowser.go("#Submit");}
else
{WMLBrowser.go("#Settings");}
}

extern function check(tr)
{
var t;
t=Lang.parseInt(tr);
t=t+1;
WMLBrowser.setVar("retry",t);
WMLBrowser.refresh();
WMLBrowser.go("#Submit");
}
```

APPENDIX 2

The AMT Implementation Mobile Application Code

This appendix contains the main parts of the ASP and WMLscript code contained in the mobile application of the AMT Ireland implementation. In addition to this code, other files are also needed for the application to run, such as the database files (*AMT.mdb* and *Log.mdb* located in a folder named *DB* contained in the main application folder) and the *adovbs.inc* file (the Microsoft ADO constants include file for VBScript). The Web application code (containing both the administrative view and the customer view) cannot be presented here due to space considerations (the whole Web view of the application contains in excess of 6500 lines of code). Also, considering the fact that the implemented mobile application is actually a downsized version of the initial prototype application presented in appendix one, some of the files for this application (such as *about.asp*, *help.asp*, *settings.asp*, *setsub.asp*) have been left out because they are similar if not identical to the ones already presented.

The following section presents the remaining ASP and WMLscript code files in alphabetical order.

dbcon.asp

```
<%  
    'Option Explicit  
    Dim conn  
    Set conn = Server.CreateObject("ADODB.Connection")  
    conn.open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &  
Server.MapPath("../DB/AMT.mdb")  
%>
```

discon.asp

```
<%  
    conn.close  
    Set conn = Nothing  
%>
```

in.asp

```
<% Response.ContentType = "text/vnd.wap.wml" %>
```

```
<%Response.Buffer = true%>
<%Session.Timeout=15%>
<%Application("Path") = Server.MapPath(".")%>
<%
Session("remote_addr") = Request.ServerVariables("remote_addr")
Session("remote_host") = Request.ServerVariables("remote_host")
Session("http_user_agent") = Request.ServerVariables("http_user_agent")
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="Initial" title=" Please wait... ">
    <onevent type="ontimer">
        <go href="index.asp"/>
    </onevent>
    <timer value="1"/>
<p align="center">
Starting application...
</p>
</card>
</wml>
```

index.asp

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<%Response.Buffer = true%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="flip1" title="&nbsp; Welcome to the " newcontext="true">
        <onevent type="onenterforward">
            <refresh>
                <setvar name="retry" value="0"/>
            </refresh>
        </onevent>
        <onevent type="ontimer">
            <go href="#flip2"/>
        </onevent>
        <timer value="50"/>
        <do type="options" label="Login">
            <go href="#Login"/>
        </do>
        <p align="center">
```

```

        
        <br/>
        WAP site.
    </p>
</card>
<card id="flip2" title="&nbsp; Built by the ">
    <onevent type="ontimer">
        <go href="#flip1"/>
    </onevent>
    <timer value="30"/>
    <do type="options" label="Login">
        <go href="#Login"/>
    </do>
    <p align="center">
        
        <br/>
        <big>
        PROJECT
        </big>
    </p>
</card>
<card id="Login" title="Login">
    <do type="prev" label=" Back">
        <prev/>
    </do>
    <do type="accept" label="Submit">
        <go href="scripts.wmls#check({retry:unescape})"/>
    </do>
    <p align="left">
        <fieldset title="Input data">
User Name: <input title="User Name:" emptyok="false" type="text"
value="<%=username%>" name="username" size="20"/><br/>
Password: <input title="Password:" emptyok="true" type="password"
value="<%=password%>" name="password" size="20"/><br/>
        </fieldset>
        <br/>
    </p>
</card>
<card id="Submit" title=" Please wait... ">
    <onevent type="ontimer">
        <go href="login.asp" method="get">
            <postfield name="username" value="{username}"/>
            <postfield name="password" value="{password}"/>
            <postfield name="retry" value="{retry}"/>
        </go>

```

```
</onevent>
  <timer value="1"/>
<p align="center">
Validating...
</p>
</card>
</wml>
```

logdbcon.asp

```
<%
  'Option Explicit
  Dim Lconn
  Set Lconn = Server.CreateObject("ADODB.Connection")
  Lconn.open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("../") & "\DB\Log.mdb"
%>
```

logdbdiscon.asp

```
<%
  Lconn.close
  Set Lconn = Nothing
%>
```

login.asp

```
<!--#include file="dbcon.asp" -->
<!--#include file="logdbcon.asp" -->
<% Response.ContentType = "text/vnd.wap.wml" %>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<%
Dim SQLquery, userid, username, company, Eid, EventID, Status
Dim LoginR, password, email, retry, Red
Red = 0
  username = Request.QueryString("username")
  password = Request.QueryString("password")
  retry = Request.QueryString("retry")
  SQLquery = "SELECT Users.UserID, Users.Username, Users.FullName,
Users.Company, Users.Password FROM Users WHERE Users.Username = '" &
username & "'"
  Set LoginR = conn.Execute(SQLquery)
  If LoginR.EOF Then
```

```

        Red = 1
Status = "Failed"
UserID = 0
Details = "Logon failed due to incorrect username"
        elseif (password <> LoginR("Password")) Then Red = 1
Status = "Failed"
UserID = LoginR("UserID")
Details = "Logon failed due to incorrect password"
        Else %>
<wml>
    <card id="OK" title="&nbsp; Logon successful ">
        <onevent type="onenterforward">
            <refresh>
                <setvar name="userid" value="<%=LoginR("UserID")%>" />
            </refresh>
        </onevent>
        <onevent type="ontimer">
            <go href="main.asp" method="get">
                <postfield name="userid" value="$(userid)" />
            </go>
        </onevent>
        <timer value="40" />
        <do type="options" label="Ok >">
            <go href="main.asp" method="get">
                <postfield name="userid" value="$(userid)" />
            </go>
        </do>
        <p align="center">
            <em>
                <br/>
                Hello <%=LoginR("FullName")%> from <%=LoginR("Company")%>.
            </em>
        </p>
    </card>
<%
Status = "Success"
UserID = LoginR("UserID")
Details = "Logon successful"
End if %>
</wml>
<%
SQLquery = "SELECT MAX(EventID) FROM Events"
EID = Lconn.Execute(SQLquery)
If VarType(EID("Expr1000")) <> 3 Then
EventID = 0

```

```
Else
EventID = EID("Expr1000") + 1
End If
If Request.QueryString("username") <> "" Then
    username = Request.QueryString("username")
Else
    username = "-"
End If
If Request.QueryString("password") <> "" Then
    password = Request.QueryString("password")
Else
    password = "-"
End If
If Request.QueryString("retry") <> "" Then
If Request.QueryString("retry") = "1" Then
Details = Details & " (first attempt)."
```

```
ElseIf Request.QueryString("retry") = "2" Then
Details = Details & " (second attempt)."
```

```
ElseIf Request.QueryString("retry") = "3" Then
Details = Details & " (third attempt)."
```

```
ElseIf Request.QueryString("retry") = "4" Then
Details = Details & " (fourth attempt)."
```

```
ElseIf Request.QueryString("retry") = "5" Then
Details = Details & " (fifth attempt)."
```

```
ElseIf Request.QueryString("retry") = "6" Then
Details = Details & " (sixth attempt)."
```

```
ElseIf Request.QueryString("retry") = "7" Then
Details = Details & " (seventh attempt)."
```

```
ElseIf Request.QueryString("retry") = "8" Then
Details = Details & " (eighth attempt)."
```

```
ElseIf Request.QueryString("retry") = "9" Then
Details = Details & " (ninth attempt)."
```

```
Else
Details = Details & " (subsequent attempt)."
```

```
End If
End If
SQLquery = "INSERT INTO Events VALUES (" & EventID & ", '" & Status & "',
" & UserID & ", '" & username & "', '" & password & "', '" & Now() & "',
'Wireless', '" & Request.ServerVariables("remote_addr") & "', '" &
Request.ServerVariables("remote_host") & "', '" &
Request.ServerVariables("http_user_agent") & "', '- ', '- ', '- ', '- ', '- ',
'- ', '- ', '- ', '" & Details & "')
```

```
Lconn.execute(SQLquery)
Session("EventID") = EventID
Session("Wireless") = 1
```



```
Session("UName") = username
Session("UID") = UserID
Set EID = Nothing
Set LoginR = Nothing %>
<!--#include file="logdbdiscon.asp" -->
<!--#include file="discon.asp" -->
<%
If Red = 1 Then
Response.Redirect("LoginError.asp?retry=" & retry)
End If
%>
```

LoginError.asp

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<%Response.Buffer = true%>
<% Dim retry
retry = Request.QueryString("retry")
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="Error" title="Authentication Error">
    <do type="prev" label="Back to login">
      <go href="index.asp#Login" method="get">
        <postfield name="retry" value="<%=retry%>" />
      </go>
    </do>
    <p align="center">
      Invalid username or password
      <br/>
      <small>
        Make sure you've typed them correctly.
        <br/>
        Note: the password is case-sensitive.
      </small>
    </p>
  </card>
</wml>
```

main.asp

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<%
Dim userid
```

```
        userid = Request.QueryString("userid")
%>
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="Main" title="&nbsp; Main page ">
    <p align="center">
    <br/>
    </p>
    <p align="left">
    <anchor title="Status"> &gt;&nbsp; Order status
        <go href="status.asp" method="get">
            <postfield name="userid" value="$(userid)"/>
        </go>
    </anchor><br/>
    <anchor title="Info"> &gt;&nbsp; Info
        <go href="About.asp" method="get">
            <postfield name="userid" value="$(userid)"/>
        </go>
    </anchor><br/>
    <anchor title="Settings"> &gt;&nbsp; Settings
        <go href="settings.asp" method="get">
            <postfield name="userid" value="$(userid)"/>
        </go>
    </anchor><br/>
    </p>
    <p align="right">
    <anchor title="Help"> &gt;&nbsp; Help
        <go href="Help.asp" method="post">
            <postfield name="userid" value="<%=userid%>"/>
        </go>
    </anchor>
    </p>
    </card>
</wml>
```

scripts.wmls

```
extern function nozero(spId)
{
var pid;
pid = Lang.parseInt(spId);
if (pid > 0)
{WMLBrowser.go("#chkid");}
```

```
else
{WMLBrowser.go("#list");}
}

extern function id(sv)
{
if (sv == "True")
{WMLBrowser.go("#Order");}
else
{WMLBrowser.go("#NoOrder");}
}

extern function qty()
{
var sq = WMLBrowser.getVar("qty");
var stock = WMLBrowser.getVar("stock");
var q;
var s;
q = Lang.parseInt(sq);
s = Lang.parseInt(stock);
if ((q < s) && (q >= 0))
{WMLBrowser.go("#Submit");}
else
{WMLBrowser.go("#Error");}
}

extern function compare()
{
var pass = WMLBrowser.getVar("password");
var rpass = WMLBrowser.getVar("rpassword");
if (pass == rpass)
{WMLBrowser.go("#Submit");}
else
{WMLBrowser.go("#Error");}
}

extern function check(tr)
{
var t;
t=Lang.parseInt(tr);
t=t+1;
WMLBrowser.setVar("retry",t);
WMLBrowser.refresh();
WMLBrowser.go("#Submit");
}
```