



**Applying Object Oriented Design Methodology
for
e-Government Web Based Applications**

BY

**Nasser N AL-Azmi
(20051220176)**

Supervisor

Prof. Dr. Naim Ajlouni

This Thesis is submitted to the Department of Computer Science, Graduate College of Computing Studies, Amman Arab University for Graduate Studies in partial fulfilment for the requirement of the degree of Master of Science in Computer Science.

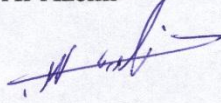
**Department of Computer Science
Graduate College of Computing Studies
Amman Arab University for Graduate Studies
(March – 2008)**

AUTHORIZATION OF DISSEMINATION

I, the undersigned “ Nasser Nassar Al-Azemi ” , Authorize hereby Amman Arab University for Graduate Studies to provide copies of this Thesis to libraries, institutions, agencies, and any other parties upon their request.

Name : Nasser Al-Azemi

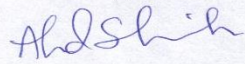

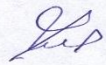
Signature :



Date : 26 – May - 2008

RESOLUTION OF THE EXAMINING COMMITTEE

**This Thesis titled “ Applying Object Oriented Design Methodology for e-Government Web Based Applications ” .
has been defended and approved on 8 / 3 /2008 .**

<u>Examining Committee</u>	<u>Title</u>	<u>Signature</u>
Dr. Ahmad Sharieh	Chair	
Prof. Dr. Naim Ajlouni	Member and Supervisor	
Dr. Sadiq Alhomoz	Member	

Applying Object Oriented Design Methodology for e-Government
Web
Based Applications

Prepared By:
Nasser Al-Azmi

Supervisor:
Prof. Naim M. Ajlouni

ABSTRACT

Despite the sensitive nature of e-government applications, their design and development lack an integrated methodology that participates in delivering high quality and secured applications. Thus, in this thesis, a new integrated use case driven object oriented methodology for the design and development of e-government application is proposed. The application of the proposed methodology has led to two main findings. First, use case specifications fit the external functionalities of e-government applications. The second is that the use case architectural view can successfully lead the development of the design and architectural views of e-government applications, and consequently drive their corresponding software development phases. In addition, it has been found that use case models construct the basis from which a simple and usable graphical

user interface could be developed. This has led to the construction of a usable user friendly Graphical User Interface (GUI) for the anticipated system. Furthermore, the consideration of non-functional requirements in use case models supported the adoption and realization of e-government applications characteristics of excellence in the proposed methodology, and hence, resulting system. These findings showed that the proposed approach supersedes current approaches in that it participated in delivering integrated e-government applications as demonstrated in the Email Access case study. Finally, further work is being carried out to investigate the validity of the proposed methodology by its application on other type(s) of e-government application(s). Also, generalization of the proposed methodology to fit the design and development of other e-commerce applications is planned to be considered in future phases of this research.

Acknowledgement

First of all, I'd like to thank my Supervisor Prof. Dr. Naim Ajlouni for his time, support, and guidance during the lifecycle of this research. Without his ideas, suggestions, and directions it would not have been possible to complete this work.

Finally, I would like to thank all the academic and administration staff members of Amman Arab University for Graduate Studies for their help and support.

Table of Contents

ABSTRACT	IV
ACKNOWLEDGEMENT	VI
TABLE OF CONTENTS.....	VII
LIST OF FIGURES.....	IX
LIST OF ABBREVIATIONS	XI
CHAPTER ONE: INTRODUCTION.....	1
1.1. RESEARCH PROBLEM: DESIGN OF E-GOVERNMENT APPLICATIONS	1
1.2. RESEARCH HYPOTHESIS	6
1.3. RESEARCH APPROACH AND OBJECTIVES	7
1.4STRUCTURE OF THESIS REPORT.....	9
CHAPTER TWO: BACKGROUND AND LITERATURE REVIEW..	11
1.2. INTRODUCTION.....	11
2.2.E-GOVERNMENT EVOLUTION IN DIFFERENT COUNTRIES	11
2.2.1. SINGAPORE	12
2.2.2. GULF COUNTRIES.....	13
2.2.3. INDIA.....	15
2.2.4. CZECH REPUBLIC	16
2.3. DESIGN OF E-GOVERNMENT APPLICATIONS.....	17
2.4. SOFTWARE DEVELOPMENT LIFE CYCLE MODELS	21
2.4.1.WATERFALL MODEL	22
2.4.2.SPIRAL MODEL	26
2.4.3.EXTREME PROGRAMMING	32
2.4.4.RATIONAL UNIFIED PROCESS	38
2.5. THE ADOPTED SOFTWARE DEVELOPMENT LIFE CYCLE.....	41
2.6. RESEARCH METHODOLOGY	48
2.7. SUMMARY AND CONCLUSION	53
CHAPTER THREE: THE PROPOSED METHODOLOGY	55
3.1. INTRODUCTION.....	55
3.2.THE REQUIREMENTS ENGINEERING PHASE	55
3.2.1. REQUIREMENTS ELICITATION	57
3.2.2. REQUIREMENTS ANALYSIS	58
3.2.3. REQUIREMENTS VALIDATION.....	58

3.2.4. REQUIREMENTS CHANGE MANAGEMENT	59
3.3. THE DESIGN PHASE.....	60
3.4. THE IMPLEMENTATION PHASE	63
3.5. THE TESTING PHASE	65
3.6. SUMMARY AND CONCLUSION	67
CHAPTER FOUR: CASE STUDY-EMAIL ACCESS SYSTEM	69
4.1. INTRODUCTION.....	69
4.2. ITERATION ONE.....	69
4.2.1. BUSINESS REQUIREMENTS	71
4.3. ITERATION TWO	76
4.3.1. REQUIREMENTS ANALYSIS	76
4.3.2. SYSTEM-LEVEL NON-FUNCTIONAL REQUIREMENTS	82
4.3.3. SYSTEM DESIGN	85
4.3.4. REQUIREMENTS AND DESIGN VALIDATION	89
4.4. ITERATION THREE.....	90
4.4.1. BLACK BOX TESTING	91
4.4.2. WHITE BOX TESTING	93
4.5. RISK MANAGEMENT	96
4.5.1. TIME LIMITS.....	97
4.5.2. REQUIREMENTS CHANGE	98
4.6. CRITICAL EVALUATION	99
4.6.1. RESEARCH PHASES AND SOFTWARE DEVELOPMENT PROCESS ...	99
4.6.2. THE DEVELOPED E-GOVERNMENT APPLICATION: EMAIL ACCESS	105
4.6.3. RESEARCH QUESTIONS AND OBJECTIVES	106
4.7. SUMMARY AND CONCLUSION	113
CHAPTER FIVE: CONCLUSION AND FUTURE WORK.....	115
5.1. SUMMARY AND CONCLUSION	115
5.2. FUTURE WORK	118
REFERENCES.....	120
APPENDICES	127
ARABIC SUMMARY.....	179

List of Figures

FIGURE 2.1: THE WATERFALL MODEL [41].	24
FIGURE 2.2: THE SPIRAL MODEL [17].	29
FIGURE 2.3: EXTREME PROGRAMMING DEVELOPMENT CYCLE [32].	34
FIGURE 2.4: AN ITERATIVE MODEL GRAPH THAT SHOWS HOW RUP IS STRUCTURED [35].	40
FIGURE 3.1: SOFTWARE SYSTEM'S ARCHITECTURAL VIEWS [31].	62
FIGURE 3.2: STATIC AND DYNAMIC VERIFICATION AND VALIDATION [47].	67
FIGURE 4.1: SYSTEM USE CASE MODEL.	74
FIGURE 4.2: SUBSYSTEM OF EMAIL ACCESS SYSTEM.	75
FIGURE 4.3: SYSTEM ARCHITECTURE.	75
FIGURE 4.4: USE CASE DIAGRAM OF MAINTAIN EMPLOYEE INFO USE CASE.	78
FIGURE 4.5 USE CASE DIAGRAM OF MAINTAIN SUGGESTIONS USE CASE.	78
FIGURE 4.6: CLASS MODEL VERSION 0.1.	85
FIGURE 4.7: CLASS MODEL VERSION 0.2.	86
FIGURE 4.8: CLASS MODEL VERSION 0.3.	87
FIGURE 4.9: CLASS MODEL VERSION 1.0.	87
FIGURE 4.10: EMAIL ACCESS OPTIONS PAGE.	94
FIGURE 4.11: EMAIL ACCESS SECTOR MAINTENANCE PAGE.	95
FIGURE 4.12: EMAIL ACCESS ASSIGN EMPLOYEES TO SECTION PAGE ...	95
FIGURE 4.13: EMAIL ACCESS USER INBOX PAGE.	96

List of Tables

TABLE 1.1: SUMMARY OF E-GOVERNMENT DESIGN APPROACHES.....	9
TABLE 2.1: COMPARISON OF E-GOVERNMENT DESIGN APPROACHES.....	21
TABLE 4.1: SYSTEM FUNCTIONALITIES.	72
TABLE 4.2: E-GOVERNMENT APPROACHES COMPARISON	106

List of Abbreviations

Abbreviation	Full Term
24 x7 ISSS	24 x 7 Info-Security Surveillance Services
DSTA	Defence Science and Technology Agency
EC	European Commission
G2B	Government to Business
G2C	Government to Citizens
G2G	Government to Government
GUI	Graphical User Interface
ICT	Information and Communication Technology
MoSCoW	“Must have”, “Should have”, “Could have” and “Won’t have”
NFR	Non-Functional Requirements
OO	Object-Oriented
RAfEG	Reference Architecture for E-Government
RE	Requirements Engineering
RUP	Rational Unified Process
SDLC	Software Development Life Cycle
SE	Software Engineering
SR	Security Requirements
SW	SoftWare
TDD	Test Driven Development
UC	Use Case

UML	Unified Modelling Language
V & V	Validation and Verification
WfMS	Workflow Management System
XP	eXtreme Programming

Chapter One: Introduction

1.1. Research Problem: Design of e-Government Applications

Electronic Government (e-government) is the application of the Internet and related technologies to provide more convenient and effective government services to individual citizens and businesses. The goal is to provide government services across a different channel. e-Government applications are classified as: Government to Citizens (G2C), Government to Business (G2B), or Government to Government (G2G). Examples of these applications include: Public e-Services deployment, Public e-Procurement Services, E-government Portal design and functionality, and E-Government Document Management Systems.

The e-government application software has to satisfy two key stakeholders, namely, the government agencies and the citizens. E-Government applications can benefit or adversely affect the large population of users depending on the overall usability of system design and the user interface design. Furthermore, there exists a social-technical gap between users and technology, which needs to be bridged. Hence, design of e-government applications involves the

study of application context, mental models, socio-cultural preferences, cognitive and ergonomic requirements of users. Human factors and system usability are truly the ignored imperatives in most e-government projects. As per the World Bank report, approximately 35% of e-governance projects in developing countries are total failures; approximately 50% are partial failures and only 15% can be seen as fully successful. As per NASSCOM report, the Government of India's spending on e-governance has gone up from Rs. 1500 crores in year 2002 to Rs. 2200 crores in 2003-2004. It will be possible to achieve greater returns on these enormous investments if usability-engineering approaches are practiced.

The literature [18,45,50] defined twelve characteristics of excellence for e-government applications:

1. Comprehensive: To the greatest extent possible, citizens should be able to do everything they have to do or want to do with their government through one e-government portal.
2. Integrated: All e-government applications should be integrated with each other, so citizens can avoid the need to provide the same data over and over and governments can save time and money by not needing to re-enter data.

3. Ubiquitous: Access to a jurisdiction's e-government portal and its connected sites and applications should be available to users/citizens from any Internet-capable connection, including PCs, PDAs, smart phones and other Internet appliances.
4. Transparent/Easy to Use: e-government sites should be designed and operated so that even novices of computer users can readily find the information they need, provide the information requested by the government agencies with which they are dealing, and otherwise perform all e-government transactions.
5. Accessible: The design and operation of e-government systems should, from the ground up, take into account the special needs of the disabled, and make it possible for them to use these systems as easily as the non-disabled.
6. Secure: e-government systems need to protect the confidentiality of data provided by citizens, the records created and stored by government, and the content and existence of citizen-government transactions performed over the Internet. Smart cards, with or without biometrics, along with digital certificates, can provide this necessary security.

7. Private: Data about citizen-government transactions, and the content of those transactions, need to be fiercely protected by the government. Under no circumstances should governments unilaterally give, sell, or trade electronic information about their citizens to private entities eager to advertise to them, nor should the government itself be allowed to use this data in any way not allowed by law and explicitly approved by the citizens.
8. Re-engineered: It is not enough to replicate electronically the administrative processes and procedures currently in place. It is necessary to thoroughly re-evaluate the overall mission of the jurisdiction and then design a digital structure that creates a government-citizen interface that simplifies and streamlines each transaction individually and the entire process of government administration generally.
9. Continuously evolving: Based on citizen use patterns and explicitly expressed preferences (in online surveys and online focus groups, as well as in individual e-mails), e-government services need to be continuously upgraded, updated, and

10. modified to suit the citizens' needs, the structure and agenda of the government, and the latest technology in data processing and network design, construction, operation, and access.

11. Fun to use: All else being equal, e-government portals/networks should be entertaining, aesthetically satisfying, and fun to use.

12. Interoperable: An excellent e-government site is one that provides appropriate (and up-to-date) links to other e-government sites, at its own and other levels in the government hierarchy. All e-government sites need to work together seamlessly.

13. Be linked to Internet voting, Smart Initiatives, and Constituent Polling Systems.

Obstacles to excellence in e-government applications are classified into technical and non-technical. Thus, this thesis is concerned with addressing technical obstacles that are related to designing and implementing e-government applications. This is achieved through the

utilization and application of object oriented design methodology enabling an integrated e-government solution. This integrated solution possesses the characteristics of excellence as detailed above.

1.2. Research Hypothesis

This research attempts to investigate the reality of the following hypothesis:

“The use of a software engineering methodology based on Use Case (UC) modelling is appropriate for designing and developing an object oriented web base e-government applications”. The dimensions of Use Case (UC) modelling appropriateness include:

1. Can UC specifications fit the functionalities of e-government applications?
2. Can the UC model of the required e-government application drive the development of a functioning system throughout all the Software Development Life Cycle (SDLC)?
3. To what extent will the adoption of the above twelve characteristics of excellence result in a successful design and development of e-government applications?

1.3. Research Approach and Objectives

UC design and development approach is the most recent object oriented development approach. Recent applications of UC driven development showed its effectiveness and appropriateness for both desktop and web based applications. Among the reasons nominated UC driven development for the analysis, design, and implementation of e-government applications in this research are:

- UC models are central in organising and modelling the behaviour of the software system and subsystems. Each model shows the interaction between the system/subsystem and its environment as a set of UCs (to represent the behaviour/functionality), actors (to represent the context), and the relationships among them (their interaction) [14,19].
- UCs are important to visualise, specify, and document the behaviour of the software system. They make the proposed system, its subsystems, and classes, approachable and understandable by presenting an outside view of how those elements may be used in the context [19,54].
- UCs enable addressing the software architecture in the early stages of software development [14].

- UCs can be utilized in developing Graphical User Interface (GUI)-based applications in general and e-government applications, in particular.
- UCs and the other Unified Modelling Language (UML) [19] diagrams can effectively be integrated with any collection of web based technologies (such as C# and java script programming languages) to propose a new design and implementation methodology for Object-Oriented (OO) e-government applications.

This UC driven approach can be seen as an important new e-government applications design and development approach for the following reasons:

1. In the literature, there doesn't seem to be an integrated e-government application development approach in general, and UC based, in particular. Rather, table 1.1 summarises current e-government applications design approaches in a way showing their main theme compared to out integrated approach.

Table 0.1: Summary of e-Government Design Approaches.

Design Approach	Theme
Beer et al.	Flexible e-government applications
Marchese	Service oriented e-government
Kalloniatis et al.	e-government security
Tian and Tianfield	Possibility of e-government digitalization.

2. Approximately, most of the available e-government applications use no or a traditional software development methodology, while the proposed system aims at adopting a recent OO methodology.

1.4. Structure of Thesis Report

This thesis is composed of five chapters that gradually demonstrate the adopted research approach and its application to the reader.

After the brief overview of the research context and motivations presented in this chapter, the adopted methodology in conducting this research is presented in chapter two. A survey on the work done by the researchers to design, develop, and evaluate e-government applications is presented also in chapter two. In addition, chapter two presents a survey on the different SDLCs followed to develop software

systems. Finally, chapter two contrasts the features of the different SDLCs to select the most suitable one for the newly proposed development methodology of e-government applications.

Chapter three presents the adopted SDLC phases, Requirements Engineering (RE), design, implementation and testing, as part of the newly proposed methodology. The application of the proposed methodology on a selected case study is demonstrated in chapter four.

Chapter four concludes with an evaluation to the appropriateness of the proposed methodology in developing e-government applications. It also evaluates the outcomes of each of its phases. In addition, this chapter discusses (1) the extent to which the aims of this research have been met, and (2) whether the research problem has been solved.

Finally, chapter five presents a summary of the proposed development methodology, the conclusions derived from this research, and concludes with some suggestions for related future research work.

Chapter TWO: Background and Literature review

1.2. Introduction

In this chapter, the evolution of e-government in a number of countries is summarised in section 2.2. Section 2.3 summarises the main research efforts in e-government applications design and development. In section 2.4, the well known and widely used software development life cycles and their defined stages have been summarised. The applications of each development model, their advantages and disadvantages have also been provided. The adopted software development life cycle to develop the anticipated e-government applications design and development methodology and the motivations and reasons behind this adoption are specified in section 2.5. Finally, a summary and the main conclusions of this chapter are provided in section 2.7.

2.2.E-Government Evolution in Different Countries

This section discusses the experience of e-government applications in a number of countries such as: Singapore, gulf countries, India, and Czech Republic, respectively.

2.2.1. Singapore

There have been many areas that the government has used technology in Singapore. These include:

- Health Care and Public Services. Solutions for this purpose have been done by Ramco [44]. The government has many Community Centers, and Clubs spread across the length and breadth of Singapore. The need was to combine all this under one application so that administration and management become easy to handle.
- Defence Science and Technology Agency (DSTA). According to Business Line [28,49] the government has taken e-governance to its DSTA. Lots of government agencies and users use the GeBiz, to submit online bids and to do transactions between the various entities in the government.
- Information and Communication Technology (ICT). According to [28], the ICT looks at enhancing e-governance initiative in Singapore. eCitizen is a portal formed as a forum for eTown Mayors. Some of the services offered by them are as follows,

- E-services for individuals and businesses.
- Connecting Singaporeans living in Singapore or abroad.
- Connecting different government agencies.
- Programs have been put into place to help the public interact and understand the e-governance advantages.
- E-Services@ONE.MOTORING. Transport related system for submitting vehicle registrations and transferring of applications for customers.
- 24 x 7 Info-Security Surveillance Services (24 x7 ISSS). Helps in maintaining the security and protecting networks all over Singapore 24 x 7.

2.2.2. Gulf Countries

E-government has already begun to spread in the Gulf countries [1,2]. This section highlights some of the work that is being undertaken in some of the Gulf countries such as:

- *Kuwait.* According to [6], a UN report calls on improved IT literacy among its citizens so that e-government can be used to its best of its capability.

- *United Arab Emirates.* Work is in process to connect the different departments of the governments [7], and they are looking to increase the number of people using internet connection. Dubai Government [3] is developing its IT applications. The main area that will be looked into is the electronic payment scheme which allows the users to carry out online transactions through the banks. Another area is the judicial departments [3], where in e-governance can be used across the judiciary departments. The main aim is accelerating process involved in the department and improve the performance of staff. On the other hand, Abu Dhabi government has launched online government services [7]. The e-portal that is available connects more than 500 government services which also contain sections for business exchanges and development.
- *Qatar:* Microsoft is helping Qatar in developing its Information Technology [12]. This proposal aims at allowing the country to expand into different areas, not only oil but also all major

- areas of the country, some being, education, healthcare, technology for the study of research and development in these areas.

2.2.3. India

India has very vast population and the governance is doing its best to put forth the best and beneficial policies while entering the 21st century. Karnataka one of the many states of India, which boasts of the IT capital city Bangalore, has in the past few years effectively implemented e-governance strategy [23]. The following are some of the projects that are a part of the e-governance project:

- *Mukya Vahini*. It is a Decision Support System that the Chief Minister uses to oversee all public grievances. It also keeps up-to-date information on the on going proposals, project progress, ministerial decisions, budgetary changes from within the state as well as in relation to the central government.

- *Khajane*. Keeps up-to-date information of the state treasury, it tracks all checks related queries. The system handles money spent on departments, employees, pensioners, physically handicapped people, widows, to name a few.
- *Saarige*. This system keeps track of transport related issues. For instance, vehicle registrations, issue of driving licenses, permits, and tax collection.

The websites are good but have too much of irrelevant contents; there is no up-to-date information, no online support or sitemap to help the user. The websites do not cover all the functionalities of the departments and most of the departments are still not interconnected properly, so there is repeated information across pages.

2.2.4. Czech Republic

This section discusses the e-governance implementation in Czech Republic and Slovakia [33]. The system has spread over different areas, some of them being the Ministry of Finance, Ministry of Interiors, and Ministry of Informatics. Some of their portals have been discussed as follows:

- Public Administration Portal. The system offers information about Land Registry along with Czech and European Commission (EC) legal regulations. This portal is intended to give information on Legislation, contact details of public administrative authorities and help in transactional online
- services. Although the motive behind the website was good, it still is very limited. The navigation menu is too complicated, structure of the web pages is confusing, and there is a lot of irrelevant information. The other issue being that there are no links to the other parts of the e-governance system.
- Ministry of Finance. The system is used for online filing of tax returns. Different kinds of taxes such as VAT, excise duties, income taxes are all available for online processing. The only low point is that electronic signatures still have not been implemented properly. They have only limited application forms being filled, due to its high cost. Therefore, the issue of authorization is still not appropriate. The system is still not are used effectively. Some of the reasons being incompatible laws and misused tax laws.

2.3. Design of e-Government Applications

The raising need for e-government applications leads to many new

approaches in this sector. To fulfill the requirement for a flexible government-to-government (G2G) software system being adaptable for the usage in many sectors of e-government applications, Beer et al. introduced the reference architecture for e-government (RAfEG). The key features of the system are flexibility, security, adaptability and interoperability between authorities. Because security is a critical issue in e-government applications their solution used different types of authentication and authorization methods in addition to supporting secure communication between the interoperating heterogeneous systems. Due to the fact that the electronically supported execution of government procedures is the main aspect of the RAfEG system, an approach where these procedures are modeled as workflows and executed by an underlying workflow management system (WfMS) is the solution they presented. Beer et al. claim that RAfEG system is a new approach because it is able to cope with a wide range of internal official procedures and it is highly adaptable to new procedures within e-government.

Marchese [38] introduced the idea of service oriented architectures for supporting environments in e-government applications due to the major changes in the structure and operation of public administrations in this type of applications.

Kalloniatis et al., as cited in [11], analyzed the current frameworks of e-government applications from the perspective of security requirements engineering due to the increasing need for keeping information secure in modern e-business, e-commerce, and e-government environments. This held because personally identifiable information can be electronically transmitted and disseminated over insecure open networks and the Internet. Security and privacy constitute the basic foundations of a trust framework, which composes a *sine qua non* condition for Information Societies. Requirements engineering (RE) is the principled application of proven methods and tools, which can design this trust framework effectively in all aspects of modern e-Government applications. Kalloniatis et al. described a number of well-known RE frameworks developed for eliciting and managing security requirements (SR). They also presented a comparative analysis of existing frameworks from a number of

complementary viewpoints. Based on the results of this analysis they identified a number of unresolved issues that need to be addressed by research in the security requirements field.

Due to the complexity of designing e-government applications, Tian and Tianfield [51] and Gammeri et al. [25] discussed an object oriented design for a selected e-government application case study.

They firstly analyzed the functions and services of government and the possibility of their digitalization. Secondly they analyzed the overall requirement of e-government systems.

Table 2.1 summarizes the literature of e-government applications design and development. It shows that each study discussed a specific feature of this type of applications and none covered an integrated design and development methodology. Hence, in this thesis an integrated object oriented methodology for the design and development of e-government applications is proposed starting from selecting the overall framework (i.e. SDLC) in the following sections.

Table 0.1: Comparison of e-government design approaches.

Design Approach	Theme	Integrated Approach?
Beer et al.	Flexible e-government applications	No
Marchese	Service oriented e-government	No
Kalloniatis et al.	e-government security	No
Tian and Tianfield	Possibility of e-government digitalization.	No
The proposed approach	Integrated e-government application	Yes

2.4. Software Development Life Cycle Models

In 1970s, software organisations used to develop lifecycle approaches for their organisations. Each approach described the actions and decisions undertaken within the organisation during the different software lifecycle phases. These approaches have been formalised as software development life cycle methodologies. A software life cycle methodology is defined as “a collection of tools, techniques, and methods which provide roles and guidelines for ordering and controlling the actions and decisions of project participants during the software lifecycle” [47]. The different software life cycle methodologies have different orders of different activities to be conducted during the software life cycle. However, the activities defined in these

methodologies are not necessarily distinct. They include specifying, designing, implementing, testing, delivering, putting into operation, and maintaining the software system.

A software development life cycle model, or process model, provides an abstract representation of the activities conducted during the software life cycle and defined by a particular methodology [47]. Many process models have been identified in the literature in order to explain the different software development methodologies. The following subsections summarize the best fit of those SDLCs for the development of e-government applications.

2.4.1. Waterfall Model

This model is sometimes called “*classic life cycle*” or “*linear sequential model*” [41]. It is the first documented and published model and follows a documentation driven paradigm [41]. It is derived from other engineering processes and was developed to cope with the increasing complexity of aerospace products [41]. The waterfall model represents the fundamental process activities as separate and sequential phases, where one phase has to be complete before

moving to the following phase. Moreover, feedback from one phase is provided at the end of this phase, as shown in figure 2.1, and the user is not actively involved in the development process.

It has five main stages [41]:

1. *Requirements analysis and definition, or requirements engineering* phase to be more general. This phase includes understanding the problem domain and the required software system. It also includes identifying the services, constraints, goals, stakeholders, and boundaries of the system. The output of this phase is the requirements specification.
2. Software design. During this phase, the system's architecture, data structures, interface representations, and procedural details are identified and constructed based on the requirements specified in the requirements engineering phase.
3. Implementation and unit testing. This phase involves realising the software design, constructed during the design

4. stage, as program units (procedures, classes, etc). It also involves verifying that the developed specifications.program units meet the requirements and design
5. Integration and system testing. The developed program units are integrated into a complete system. The integrated components and the system as a whole are tested to

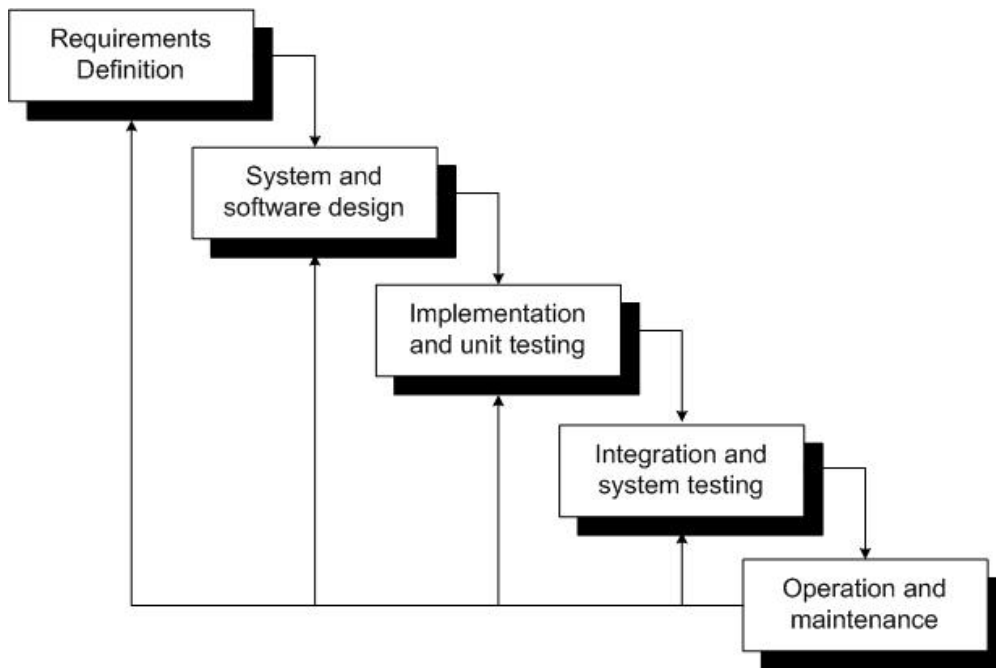


Figure 0.1: The Waterfall Model [41].

ensure that the program units still work correctly after integration with other units and the system meets its specified requirements.

6. Operation and maintenance. After ensuring correct operation of the system and complete implementation of the required services, the system is delivered to the client and installed in the client's operational environment. After using the system, some errors, that were not discovered during the development and testing, might be uncovered. Thus, maintenance of the system might be required. System maintenance involves errors' correction and system improvements.

Although the waterfall model is the most widely used model in software development, it has been criticised by practitioners due to its disadvantages:

1. It has inflexible partitioning of the software development project into distinct phases, which makes it very difficult to accommodate changes or respond to changing customers' requirements after the development process is underway. Hence, it might not be appropriate for developing many business systems as they rarely have stable requirements.

2. It is only appropriate when the system's requirements are well-understood, which is not always the case in real life. Customers often find it difficult to explicitly and completely specify all the requirements at the beginning of the development process.
3. A working version of the system will not be available until late in the development process, which requires customers' patience. In addition, this model does not accommodate prototyping, and the users are not actively involved during the development process. Hence, the developed system might not reflect the customers and users' real needs, and this might not be discovered until the system is delivered.

2.4.2.Spiral Model

The spiral model of the software development process was defined by Boehm [17] in 1988 with the purpose of overcoming the problems associated with the document-driven and code-driven software development lifecycles introduced above. It was developed based on experience with various enhancements of the waterfall model as

applied to large government projects, and has been successfully used to define and develop large applications such as the TRW Software Productivity System [17].

The spiral model has been introduced as a model that serves as a more robust foundation for a software development environment than the previously developed software development models including the waterfall, evolutionary, iterative incremental development, and formal specification models. This has been attributed to the idea that the spiral model includes these models as special cases depending on the software situations, and provides further guidance on how to use some of these models in combination in order to best develop a software system. The main important feature that distinguishes the spiral model from other models is that it creates a risk-driven approach that avoids many of the problems associated with other software development models and makes it adaptable for a full range of software projects as will be discussed later.

As shown in figure 2.2, the spiral model does not represent the development process as a linear sequence of activities with some backtracking among activities. Instead, it presents the development

process activities as a spiral, where each loop (or more) in this spiral represents a phase (feasibility study, requirements engineering, design, code, etc) in the development process [17]. In addition, each loop comprises four main activities:

1. Identification of objectives, alternatives, and constraints.
2. Evaluation of alternatives and risk assessment and reduction.

This activity incorporates evaluating the alternatives with respect to the defined objectives and constraints. This evaluation would frequently identify the project's risks. If some risks are identified, a cost-effective strategy, which might incorporate prototyping, simulation, analytical modelling or any other risk resolution technique, will be identified for resolving these risks.

3. Development and validation. During this activity, any software development model, or combination of models, will be selected to develop the required part of the system during this phase. Selecting the development model depends on the identified risks and risk reduction strategies.

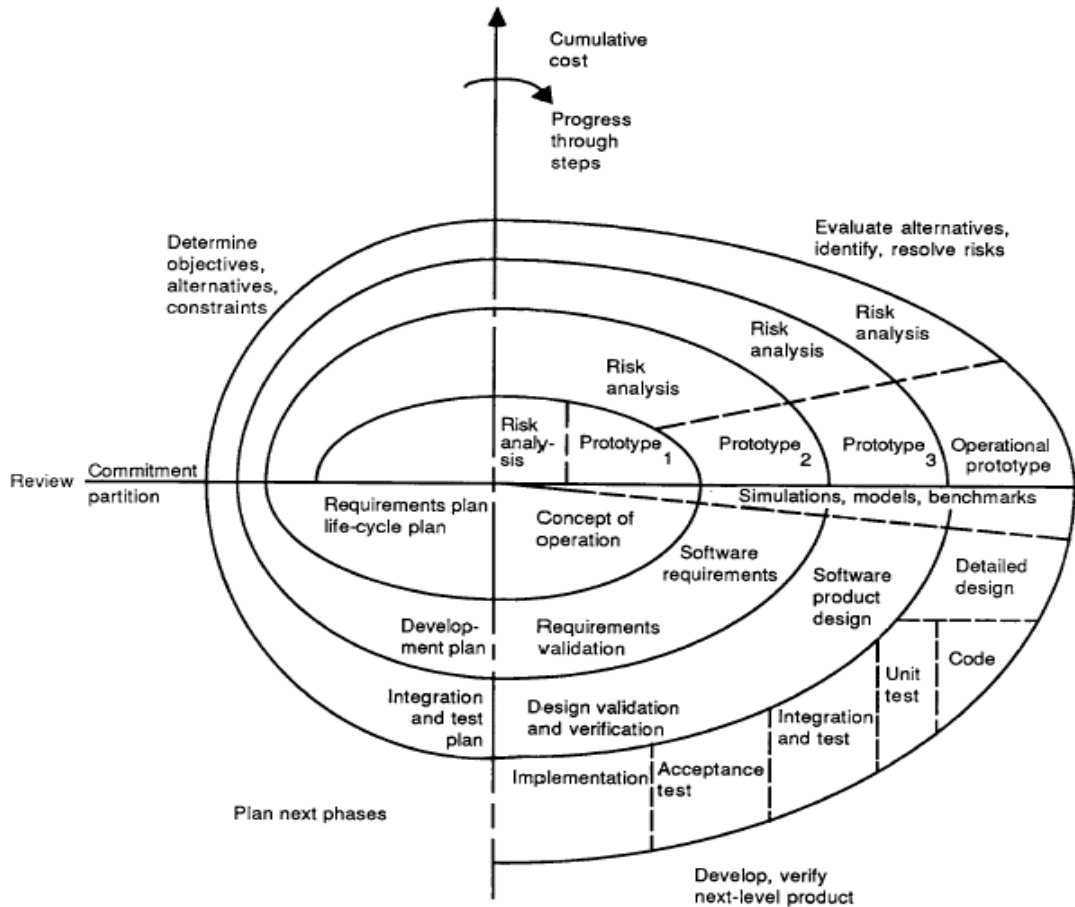


Figure 0.2: The Spiral Model [17].

4. Planning of the next phase. At the end of each cycle in the spiral model, a review that involves the primary people or organisations concerned with the developed system is conducted. This review covers the products developed during this cycle and decides whether to continue with further cycle of the spiral. If a decision of continuing another cycle is undertaken, plans are drawn for the next cycle, and the

5. resources required to carry out these plans are identified.

As the spiral model incorporates the different software development models as special cases, it has many of their features including [47]:

1. It promotes developing specifications that are not necessarily uniform, comprehensive, or formal. In that, great amounts of details are not necessary unless their absence expose the project to risks.
2. It includes prototyping as a risk reduction technique at any development stage.
3. It accommodates rework of earlier stages as new risks need resolution or more alternatives are identified.

Accommodating the different software development models' features in addition to avoiding their difficulties through the risk driven approach is the most important advantage of the spiral model. In addition, it has many additional advantages such as [41]:

1. It pays early attention to options that involve reusing existing software.

2. It includes preparation for the software product's evolution, growth, and change.
3. Identifying all the objectives and constraints during each cycle of the spiral enables incorporating the software quality objectives such as the performance, useability, and reliability into software product development.
4. The risk analysis and validation executed within the spiral cycles enables early elimination of the errors and unattractive alternatives identified within these cycles.
5. It does not involve separate approaches for each of software development and software maintenance. Hence, maintaining a software product requires initiating a new maintenance spiral.

On the other hand, the spiral model suffers from the disadvantage of relying to a great extent on the developers' risk-management experience and their ability to identify and manage the sources of project risks. Hence, if risks have not been correctly identified or evaluated for a software project, this project will proceed to give false

impression about the progress while it is actually heading for failure.

2.4.3. Extreme Programming

EXtreme Programming (XP) is a light-weight software engineering methodology created by Kent Beck and others [32] for developing software projects by small-to-medium-sized co-located teams.

XP is considered the most well-known and widely used method amongst set of methods named Agile methods [32]. Development of these methods has been promoted by the dissatisfaction with the overheads involved in design methods. Thus, all agile methods focus mainly on the system's code instead of design. Moreover, they are based on iterative software development approach and intended to quickly produce working software and evolve it to meet changing requirements [32].

Agile methods are based on five main principles defined to lower the cost of change during the development and increase the users' involvement [32]:

1. Customer's involvement.
2. Incremental development.
3. People not process.
4. Embrace change.
5. Simplicity.

XP is derived from good practices in software development that have been known for a long time. It primarily focuses on five values [32]: *communication, simplicity, feedback, courage, and respect*. These values are used to guide its defined practices that are employed in developing software systems, and evaluate any new practices that might be introduced in the future.

XP works well if used to develop software systems with requirements that change rapidly, involve prototyping or new technology, or small systems that can be managed through informal methods. The environments of these systems should also enable close communications, inexpensive interface changes, and automated testing [32].

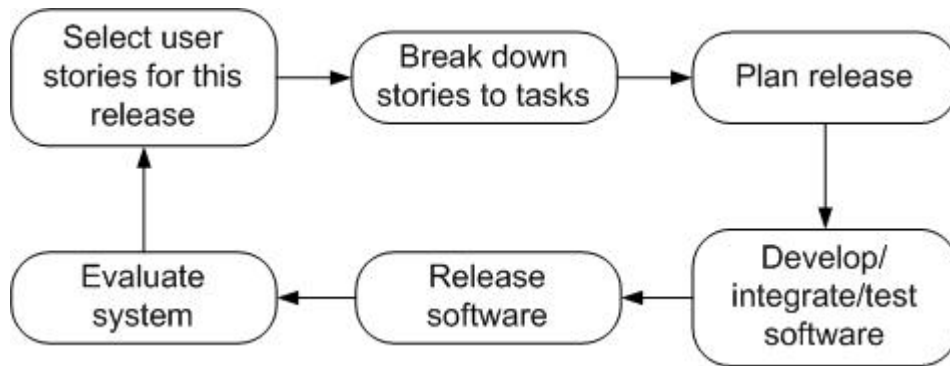


Figure 0.3: Extreme Programming Development Cycle [32].

In order to increase development speed and productivity, facilitate problem domain and system understanding, increase product quality, facilitate changes, and minimise their costs, XP defined twelve practices that define the spine of this methodology and conform to its defined values mentioned above. These practices are defined to interact and reinforce each other as explained below [32]:

1. **Incremental planning.** XP promotes planning for short three-week cycles, where requirements (user stories) of these cycles are written on cards and prioritised by customers, then estimated in terms of effort by developers. These stories are then divided into releases and used to schedule and estimate the development tasks and cost, respectively, depending on stories' priority and available development time, as shown in figure 2.3.

2. **Small Releases.** As in iterative incremental development, systems should be divided and delivered as frequent and incremental increments or releases, where each release should be as small as possible and provide some of the required functionalities.
3. **Metaphor.** It is similar to system architecture that views the whole system and shows how it works to provide the defined goals.
4. **Simple Design.** Simplicity is a key value in XP that aims to design the current requirements of the system regardless of any anticipated future changes.
5. **Testing.** XP defines two types of testing: unit and acceptance (functional). Unit testing should be automated and developed before coding in order to test the code once it is written and provides instant feedback. Acceptance tests are written by customers based on user stories and they used to replace requirements specification.

6. **Refactoring.** Refactoring refers to the continuous redesign of the system in order to improve its internal design and responsiveness to change, but without changing its external behaviour. This includes the elimination of redundant and unused functionality with the intent to make the design simple and easier to understand, modify, and extend.
7. **Pair programming.** Programmers should work in pairs in order to think of better solutions together, check each other's code, and provide support for each other.
8. **Collective ownership.** Any one of the project team can change any part of the system's code at any time. This practice is supposed to encourage the team members to work closely together in order to produce high-quality design, code, and test cases.
9. **Continuous integration.** Once a test is completed, its output is integrated with the whole system.
10. **40-hour week.** This practice is introduced to indicate that the team should only work the normal required hours of work a day.

11. **On-site customer.** XP requires a full time customer's availability with the team during the work on the project. This is required to provide quick feedback once required and answer the team's questions.
12. **Coding standards.** As programmers work in pairs and any one of the team can change the code, standards should be followed in order to make all system parts consistent. This consistency would also facilitate communication among team members.

XP practices highlight some differences between XP and other conventional development methodologies such as [32]:

1. Most of the development models embrace the conventional software engineering practice: "Design for change" as it is believed that spending efforts and time anticipating changes early in the development process might reduce the cost later in the system's life cycle. However, XP is against this as anticipated requirements might change before becoming relevant.

2. As requirements specification defines one of the main bases of software development in conventional development processes, XP defines requirements as automated acceptance tests instead of developing requirements specification.

2.4.4.Rational Unified Process

Rational Unified Process (RUP) is a software engineering process that provides a disciplined approach to enhance team productivity, and deliver high quality software that meets its end users needs within predictable budget and schedule. In order to enhance team productivity and make all team members share common software development language, process, and view, it provides them with a knowledge base of guidelines, templates, and tool mentors for all development activities [35].

Instead of producing large amounts of paper documents as in the waterfall model, the RUP develops and maintains models that represent the software system from different perspectives. Moreover, RUP is supported by tools that automate large parts of it and produce and maintain different artefacts associated with it such as visual

modelling, programming, and testing. The RUP is a configurable process [35] that suits both small development teams and large development organisations. It is based on a simple process architecture that can be customised and extended to suit the needs of various projects and organisations. In addition, it captures the software development best practices that are commonly used by successful organisations in the industry: iterative software development, use of component-based architectures, visual modelling of software, requirements management, change management, and quality management.

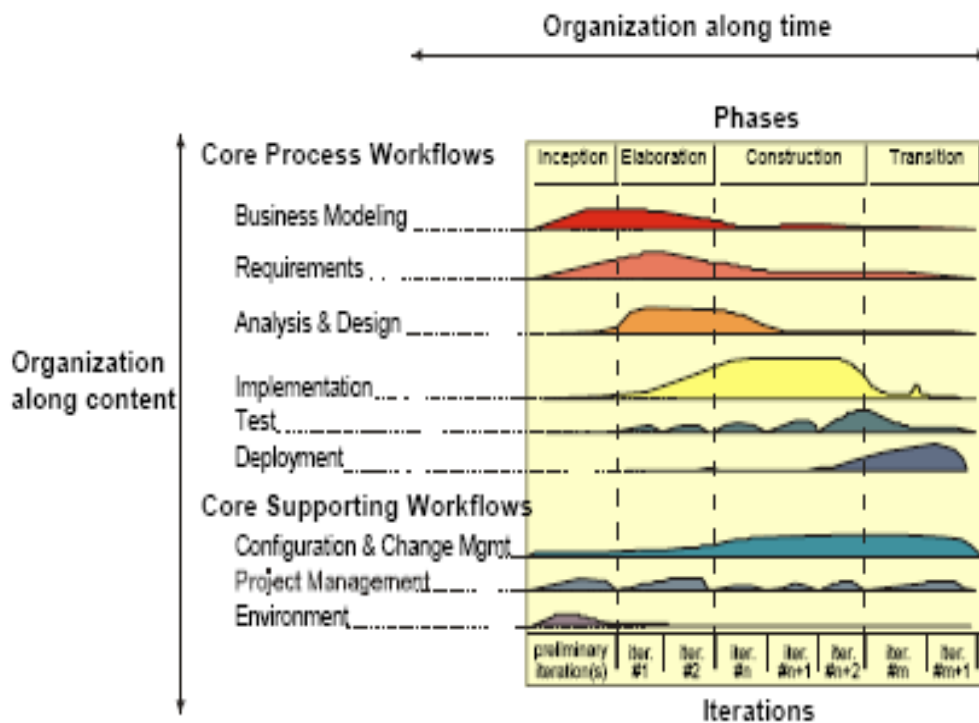


Figure 0.4: An Iterative Model Graph that Shows How RUP is Structured [35].

RUP can be described along two axes [35]: horizontal and vertical as shown in figure 2.4. The horizontal axis represents the time, and shows the dynamic aspect of the development process as it is carried out. This is expressed in terms of phases, cycles, and milestones. On the other hand, the static aspect of the process is represented by the vertical axis and is expressed in terms of artefacts produced or used by the process, process activities, workflows (sequence of activities) that produce deliverables, and workers employed in the process. Along the horizontal axis, the RUP-development cycle is divided into four consecutive phases: inception, elaboration, construction, and transition, where each phase has a specific purpose and ends with the milestone at which the output of that phase is evaluated.

RUP's above technical details shows that it offers many more advantages than the ones specified at the beginning of this section as it 1) places special emphasis on iterations during the different phases to enable the development team to consider any changes in the requirements, and 2) focuses on continuous verification of the developed product in the final phase in order to ensure the release of best quality product within the shortest time.

With respect to all features provided by RUP, it has been criticised by many experts as it has the following disadvantages [47]:

1. It is strongly coupled to Rational tool set and payment to Rational corporation is necessary to get the required tools to use the RUP.
2. It is sometimes difficult for some stakeholders to get used to its iterative nature.
3. It is only a development process and does not cover the maintenance phase of the development life cycle of the software system.

2.5. The Adopted Software Development Life Cycle

Having defined the problem domain to be researched, questions to be answered, research objectives to be achieved, and methodology to be followed, many questions have risen:

- How would the required e-government application be designed and developed?
- How should the planned stages for developing the anticipated application be enacted?

Answers of these questions have been traced to the software development life cycle to be adopted in the application of the defined research methodology to address the research objectives and deliver an operational software system.

The selection of the appropriate SDLC is dependent upon the context of the research to be conducted. Many risks that might delay the planned research schedule, minimise the research productivity, or reduce its validity surrounding the development of e-government applications. These risks have risen from the nature of the research problem and its surrounding environment. The problem domain to be researched (e-government applications) has been completely new to the researcher. Hence, researcher's knowledge of this domain has to be developed from scratch. This fact resulted in being unable to initially define a strict plan with exact time periods and milestones for the research phases and the software development process required within the research. In addition, it created many risks such as: 1) inability to establish good understanding of the research problem and hence master the research, and 2) inability to well understand the

requirements of the system to be developed Moreover, the research time is limited to a strict timeframe which if exceeded would cause rejection of the research.

In addition, the software system (e-government application) that is required to be tackled during this research is dependent to a great extent on external users (citizens) satisfaction. Therefore, their involvement in the research and the software development process has been required. As per the research environment, contacting the client has been quite limited as he cannot be immediately available once required.

The different SDLC models, which are discussed in previous section, have been contrasted and evaluated against their fitness to the research and development context as follows:

1. The waterfall model is not suitable as requirements should be identified and documented early in the development process which is not possible due to the limited understanding of the e-government systems requirements and limited development time. Moreover, requirements changes, which are expected during the software development, are difficult to be

2. accommodated following this model. Furthermore, a working version of the software system will not be available until the later stages of the development, and this would be too risky as client's dissatisfaction or rejection of the delivered system might cause failure of the whole research. In addition, client's involvement which is required during this research is not supported by this model.

3. the extreme programming approach is not suitable as it requires the customer to be closely involved in the development as one of the development team which is impossible in this research. It also focuses on developing systems with minimal formal documentation that are designed to provide only the required functionality without any consideration for future system extensions (new functionality) which does not conform to the research objective of developing a highly maintainable system using a well defined methodology. Moreover, many of its practices that are described in section 2.4.3 cannot be applied within this research such as "pair programming" and "on-site customer".

4. Furthermore, it specifies that the customer should write his requirements “user stories” and the acceptance test cases, which is not acceptable for the customer himself nor in the research environment. In addition, it uses the acceptance tests as a replacement of the requirements specification which is required for testing the research hypothesis, answering its questions, and achieving its objectives. The system’s maintenance also requires the requirements specification as it identifies the core requirements and might define anticipated system changes and expected system evolution. Although some XP practices do not suit the context of the research and software development, it has many practices that can be valuable if adopted in the research such as test-driven development, refactoring to eliminate redundancy and design defects, continuous integration of the developed parts of the system, and continual feedback from both the customer and applied tests during the development.

5. RUP provides features that are suitable for the required system development. However, it requires payments for Rational Corporation in order to use the different support tools required to use the RUP. While the research is not supported financially by any organisation, this payment is not acceptable for the management and development team (i.e. the researcher).
6. The spiral model is found to be the most suitable one to be adopted in developing the required e-government application. This selection is attributed to the fact that the spiral model couples the advantages of both the iterative prototyping nature and the controlled systematic aspects of the waterfall model [20,36]. In addition the risk-driven nature of this model enables early mitigation of risks, which increases the probability of development's success and system's acceptance. This is crucial in developing the required system as many risk elements are expected to rise during this development due to the research nature specified earlier. Following the spiral model would enable continuous but fair engagement of the customer in the development process and hence continuous

7. feedback. Requirements can be better understood through the use of prototyping which might also enable the customer to see a working version of the system early in the development process, and hence, reduce the risk of system rejection. Requirements changes can be easily accommodated during the development and would follow controlled mechanism as tradeoffs will be considered with respect to the expected risks, identified objectives, and constraints. Moreover, the spiral model pays early attention to the option of software reuse as specified in section 2.4.2, which would enable faster software development and hence reduce project's risk.

Both the research hypothesis and its objectives focus on the quality attributes (characteristics of excellence) of the required software system. Hence, the selected SDLC model should enable these attributes to be considered during the development process. The spiral model enables this as it incorporates the software quality objectives early in the development process.

One of the main advantages of the spiral model that supports its selection is that it enables the use of different development models in combination in order to best develop the software system. Hence, attractive and useful practices and features in other models can also be adopted during the development of the anticipated system. These practices include test-driven development, refactoring, and continuous subsystem integration defined in XP. However, this selection of the spiral model should be taken with carefulness during the development as the spiral model highly depends on the developer's risk management ability and experience.

2.6. Research Methodology

Defining the research methodology, the approach of the research process [21], to be followed is a crucial precondition for the success of any research. There are two main paradigms, positivistic and phenomenological, that the research can follow according to the questions it aims to answer. These paradigms look at the world from completely different perspectives.

Positivistic (quantitative) paradigm looks at the world as being external

and objective, and that the observer is independent of the world. It is mainly concerned with establishing causal laws and discovering causal relationships between variables then relating this to theory. Hence, positivistic approaches are concerned with collecting large data volumes, i.e. quantitative data, which should be precise and comparable. It is also concerned with testing hypothesis, and generalising from sample to population [15]. Within this context, many methods can be associated to positivistic methodology such as [21]:

1. Surveys, which are concerned with investigating a sub-sample of a population in order to draw conclusions and predict outcomes regarding the population.
2. Experimental studies: gaining knowledge by experiments through repeated measurements, independent sampling, matched pairing, and single subjects.
3. Cross-sectional studies, which are concerned with making up a sample and a population at one point in time.

On the other hand, phenomenological (qualitative) paradigm doesn't look at the world as being objective. Instead, it assumes that the world

is subjective and socially constructed. Also, it focuses on the meanings and constructions that people attribute to the world based on their experience. It is concerned with generating theories and generalising from one setting to another. For this purpose, it uses small samples of rich and subjective data [21]. There are many phenomenological methods for collecting subjective data, such as [21]:

1. Action Research, which is an approach in which the researcher enters into a situation and participates in the activities carried out within that situation (i.e. group or organisation) in order to bring about change and observe results.
2. Case studies, which are investigation of a certain phenomenon within a particular context.
3. Ethnography, which is an approach in which the researcher collects data by observing people within their societies.

The lifecycle of this research passed through different stages that required a hybrid approach utilising both positivistic and phenomenological approaches. The work in these stages has been

iterative, incremental, and risk driven following the spiral model of software development [17]. Therefore, each stage: problem research and understanding, requirements engineering, software design, implementation, and testing have been performed in one or more iterations, and used positivistic, phenomenological or both techniques. The rationale behind adopting the spiral model in this research is discussed in chapter 3.

Interviews, a phenomenological method that involves discussing issues with people [21], have been conducted to gain a detailed understanding of the problem to be researched. This has been achieved by interviewing e-government applications users (citizens) and structuring these interviews in order to utilise the time, record the questions, control the range of topics, and facilitate later analysis [21].

Moreover, the literature has been surveyed in order to collect more information required to better understand the research problem being investigated and its related issues, and to elicit the implemented system's requirements as detailed in following chapters. This survey has been conducted by reading different resources such as books [5,51], journal and conference papers [30,38,49], technical reports

[4,8], and reference manuals [29,39,46,52] that detail the different artefacts of e-government applications: definition, construction, design, experiments, and history. The collected data has been studied, analysed, evaluated and compared to develop better vision of the research problem, its history, and its environment.

Collected data has been used for designing and developing an example e-government application and conducting experimental studies to test the functional requirements and quality attributes (i.e. characteristics of excellence) of the developed e-government application.

Triangulation is the “use of different research approaches, methods and techniques in the same study” [15]. Both data and methodological triangulations [15] have been embraced by the conducted research. Data required to develop the software system has been collected using both positivistic (surveying the literature: books, journal and conference papers, etc) and phenomenological (interviews) approaches. In addition, the used research approach, as described in section 2.6, is a hybrid of both positivistic and phenomenological approaches.

Triangulation in this context has been used to increase research creditability: validity, reliability and accuracy. Data triangulation has been used in order to overcome the potential bias, sterility, and incompleteness that may result from collecting data by one way. On the other hand, methodological triangulation has been used to increase the research productivity and enhance the used qualitative methods [22]. For example, the literature has been surveyed before interviewing citizens in order to have a clear understanding of the required system and focus the requirements elicitation activity. In addition, testing the software system has been done following different approaches such as white and black box testing in order to increase the reliability and accuracy of the results.

2.7. Summary and Conclusion

This chapter summarised the output of the conducted literature review on e-government applications. Further, it summarised the most well known and widely used software development life cycle models that can be followed to develop software systems. In addition, the main advantages and disadvantages of these models as identified in the literature have been presented. Based on the review and evaluation

of the different software development models, the spiral model has been recommended to design and develop e-government applications. The rationale behind selecting the spiral model for the development of this type of applications is also detailed in this chapter. The next chapter describes the software development phases (requirements engineering, design, coding, and testing, respectively) and the activities that needs to be enacted during these phases based on the followed software development model.

Chapter three: The Proposed methodology

3.1. Introduction

In this chapter, the details of the software development phases that should be enacted in the adopted SDLC, and spiral process model [17] as detailed in previous chapter are discussed as part of the proposed OO methodology to develop e-government applications. The Requirements Engineering (RE) phase is discussed in section 3.2. System design activities are detailed in section 3.3. The concepts of the recommended implementation and testing activities are demonstrated in sections 3.4 and 3.5, respectively.

3.2.The Requirements Engineering Phase

The requirements specification is the most important output of the Requirements Engineering (RE) process. This process [48] represents an early phase of the SDLC conducted to elicit the requirements of the software system, analyse these requirements, develop requirements specification, and validate the elicited requirements as discussed in the following subsections. In parallel, requirements' change management activity was undertaken to control the evolution of the system's requirements.

The initial understanding of the problem domain and client's requirements are normally limited in the early stages of software development so that the RE process is recommended to be carried out in an incremental and iterative manner for developing e-government applications following the adopted spiral development model. A review at the end of each iteration should be conducted in order to assess the outcomes of this iteration, and plan the next iteration, if required. During the RE process, requirements should be visualised, specified, and documented as UCs in the system's UC model [34]. This UC model is better being comprehensive enough to specify the services of the anticipated system so as to drive the consequent SDLC phases. UCs (requirements) should be prioritised following the "**Must have**", "**Should have**", "**Could have**" and "**Won't have**" (**MoSCoW**) scheme [37,40].

The bulk of the RE effort occurs early in the SDLC so as to minimise the cost of changing users' requirements at late stages of the development process. However, studies showed that about 20% of users' requirements might be elicited or changed in the later stages of the SDLC due to better understanding of users' objectives and

requirements and more directed feedback [47]. This may result in re-entering the RE phase of the SDLC in order to: ensure that no inconsistencies have been introduced by adding these classes and relations, analyse, document, and validate the emerged requirements.

3.2.1. Requirements Elicitation

This activity is mainly concerned with discovering and identifying the entities of the problem domain under investigation and the relations among them. It is also concerned with gathering both user and system requirements. Thus, different techniques may be used in order to understand the problem domain of the system under investigation, define the problem domain objects and relations among them, define the scope of the system, identify the stakeholders, the services to be provided, and the constraints on both the system and the development process.

The use of UC modelling may help in focusing the requirements gathering, and discovering some missing requirements. It also participates in maintaining better understanding of the problem domain, and defining the initial blueprints of the system architecture and class models that may be refined in the design phase.

3.2.2. Requirements Analysis

Elicited requirements, especially the volatile ones that are related to the customer, should be analysed in order to ensure consistency and completeness. Both functional and non-functional requirements (NFRs) should be analysed during and after the elicitation phase in each RE iteration.

Although most of SDLC models show that designing the software system starts after the RE phase, this might not be true in real life. Designing the required system started during the RE phase, and had its effect on the elicited requirements. Designing the UC model during the requirements engineering phase may help in addressing the architectural aspects of the anticipated e-government application.

Forward and backward traceability should be maintained during this phase. “Backward-from” traceability with the client vision document, and “forward-from” and “backward-to” traceability should be maintained between both system’s requirements and design.

3.2.3. Requirements Validation

This is an important RE process that is enacted to establish that the elicited and documented requirements provide an accurate account of

stakeholders' requirements [24,48]. It is intended to check the requirements for validity, consistency, and completeness. It is also concerned with checking that the requirements can be actually implemented within the specified budget, schedule, and other constraints. The importance of this process comes from the fact that repairing a requirements problem by making a change in the developed system is much more expensive than fixing design or coding error.

Different techniques may be used to validate requirements: (1) conduct formal reviews for the requirements document, and (2) construct system prototypes for discussion with the client.

3.2.4. Requirements Change Management

This process is responsible for managing the requirements changes during the life cycle of the software system [41]. Hence, it is carried out in parallel with all the RE activities specified above, and all the subsequent SDLC phases until the system is delivered and put into operation.

Sommerville's [47] requirements change management process is suggested to be adopted in the proposed methodology. This results in

each requirements change to pass through three stages: problem analysis and change specification, change analysis and costing, and change implementation.

3.3. The Design Phase

The design of a software system is usually documented as a set of graphical models that visualise this system from different perspectives [42]. These models represent a realisation of the requirements elicited in the requirements engineering process that precedes the design phase within the SDLC. Moreover, they can be used to assure that requirements are satisfied, and that the design supports the non-functional requirements (NFRs) before starting the coding phase of the SDLC. This might result in reducing the changes that might emerge during the coding phase, and hence reducing the cost of applying these changes.

Since an object oriented (OO) methodology is proposed in this research, the anticipated e-government application is better designed using a set of Unified Modelling Language (UML) models. UML has been selected to be adopted since it is the most common modelling language for OO systems. It provides different diagrams that can be

used to model the static structure and dynamic behaviour of the software system components. The static structure diagrams include class, object, component, and deployment diagrams, whereas the dynamic behaviour diagrams are: sequence, activity, collaboration, and statechart diagrams [42]. Two levels of systems design need to be considered: architectural high level design and low level class diagrams.

Recent developments in software engineering [9, 31, 34, 53] suggest that software system's architecture consists of several views: logical, process, implementation, deployment, and use case views. The logical view of a system's architecture encompasses the vocabulary of the problem, the collaborations that realise the system use cases, the subsystems that provide the central layering and decomposition of the system, and the interfaces that are exposed by those subsystems and the system as a whole. The process architectural view encompasses the threads and processes that form the system's concurrency and synchronisation mechanisms. The implementation architectural view encompasses the components used to assemble and release the physical system. The deployment architectural view

encompasses the nodes that form the system's hardware topology on which the system executes. And, the use case architectural view encompasses the use cases that describe the behaviour of the system as seen by its end users and other external stakeholders. Activities within a view may require information from other views, and elements from one view may depend on or be driven by those of another. Moreover, the views may need to be ordered so that the information shared between two or more views remains consistent. An exception to this rule occurs with the use case view that drives all other system views as depicted in figure 3.1. However, some systems do not require all views, and others require additional views such as: data and security views.

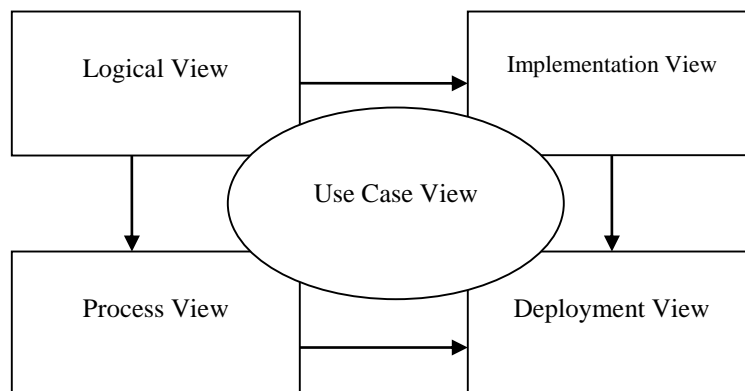


Figure 0.1: Software System's Architectural Views [31].

3.4. The Implementation Phase

A recent approach software systems development is the Test Driven Development (TDD) [10]. This approach implies that the test cases are designed first, and then the code is implemented so as to produce clean running code. Following this approach, the components' interfaces should be designed first, and then the test cases for testing each component should be designed based on these interfaces. In the proposed methodology for developing e-government applications, we recommend adopting TDD to meet the universally defined characteristics of excellence.

Testing each software component should be accomplished according to its defined responsibilities in the system requirements and design. However, some new test cases emerged during the development phase which should be added to the pre-prepared test cases. Finally, among the diverse advantages of test driven development, e-government applications development may benefit from:

1. Test driven development shortened the programming feedback loop.

2. Test driven development facilitated the delivery of high quality source code.

One of the most important features of OO development, in general, and e-government applications development, in particular, is that it facilitates the reuse of already developed and welltested components instead of reinventing the wheel. This reuse can be done in many different ways based on the need and availability of such components. Some components can be integrated with the system and used directly, while others can be extended and customised to fit the context of the new system.

In addition, design patterns [26] have been defined to help the different stakeholders understand the software system design and discuss it easily. Hence, some design patterns may be utilised in designing e-government applications so as to improve its maintainability and understandability and to save the time needed to reinvent new solutions. Examples of useful design patterns in the context of e-government applications include Wrapper and Proxy Pattern. Wrapper pattern may be used to facilitate integrating new online systems with legacy traditional subsystems. Proxy patterns may be utilised to enforce the implementation on some security requirements.

3.5. The Testing Phase

Testing is considered as part of a larger software engineering (SE) activity named software Validation and Verification (V & V). The ultimate goal of the V & V activities is to establish confidence that the SoftWare (SW) system fits for its intended use. However, verification and validation activities are not the same. Verification is concerned with the developers' view of the system. It is intended to show that the SW system is fault-free and conforms to its specification; i.e. meets its specified functional and non-functional requirements. On the other hand, validation is concerned with the clients' view of the system, and is conducted to ensure that the system meets its intended users' requirements. This difference between verification and validation is highlighted by Boehm's definitions [47]:

“Validation: Are we building the right product?”

“Verification: Are we building the product right?”

V & V activities are enacted continuously throughout the software development process, and adopt static and dynamic techniques for checking and analysing the software system. Static techniques can be applied without running/executing the software system, and are

considered the main techniques for detecting errors in requirements specification and design. However, they can not be used to prove that the software meets its requirements, or to check its emergent properties. Static techniques include software *inspections* or *reviews* that are applied to purify the analysis, design, and coding SE activities through checking and analysing their produced documents (output). These documents include the requirements specification, design diagrams, and the source code. Dynamic techniques are used as complementary techniques for the static ones within the V & V of the software system. The main dynamic technique used is *software testing*. It is also considered the main V & V technique. Software testing entails running the software system's implementation with test data, and examining the software's outputs and operational behaviours so as to uncover errors and defects that were made unintentionally as it was designed and constructed. In addition, it checks whether the developed system behaves as required (i.e. meets its functional and non-functional requirements.). Figure 3.2 shows the stages of the software development process where each of

the static and dynamic techniques can be used.

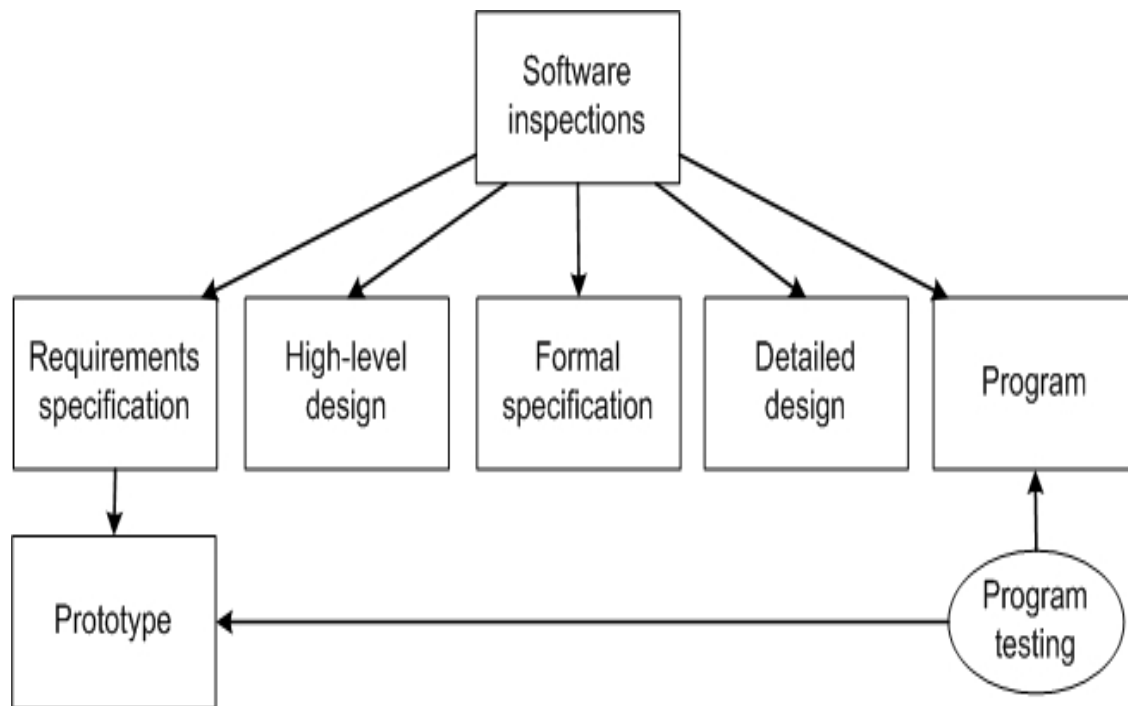


Figure 0.2: Static and Dynamic Verification and Validation [47].

3.6. Summary and Conclusion

The software development phases of the proposed OO methodology to design and develop e-government applications were presented in this chapter. The requirements engineering phase defined the requirements elicitation, analysis, validation and change management workflows. The design phase explained the recommended high level and low level object oriented design concepts to be adopted in the proposed methodology. The implementation and testing phases described the recommended best practices to be adopted in the

proposed methodology to develop e-government applications.

In the next chapter, the application of the proposed methodology on a selected e-government application case study is detailed, followed by a critical evaluation for this application in chapter four.

Chapter four: case study-email access system

4.1. Introduction

In this chapter, the application of the proposed methodology using an “Email Access” e-government application is demonstrated. Three spirals (iterations) were needed to deliver the operational system. The first iteration, as detailed in section 4.2, is the shortest one and was devoted to define the context and boundaries of the system. The initial blue prints of the system architecture started to appear towards the end of the first iteration. The second and third iterations specialized in further analysing, designing, and developing the system as will be detailed in sections 4.3 and 4.4. Validation and verification is a continuous activity that was adopted in all iterations in its both types: static and dynamic when necessary. Risk management of this case study is discussed in section 4.5 followed by a critical evaluation in section 4.6. Finally, section 4.7 summarises the main conclusions of this chapter.

4.2. Iteration One

The first iteration is the shortest iteration among the three. It lasted for two working weeks. The first step in this first iteration researched the

reasons that justify the need for this type of systems: Email Access. It was found that e-government environment needs for a formal communication channel to let all employees communicate by using secure and special application. On the other hand, citizens should be able to use the system to report their complains and suggestions.

Therefore, the respective literature has been surveyed in order to collect more information required to better understand e-government Email Access systems and their related issues, and to elicit the system's requirements as detailed later in this section. This survey has been conducted by analyzing different resources such as books [5,51], journal and conference papers [38,49], technical reports [4,8], and reference manuals [29,39,46,52] that detail the different artefacts of e-government applications, in general, and Email Access, in particular: definition, construction, design, experiments, and history. Furthermore, a number of specialized interviews with domain experts, working in local regional office for an international company [29], in e-government applications have been conducted to support the process or requirements elicitation.

Consequently, three main stakeholders were identified in this context:

- 1- Employees.
- 2- Application administration.
- 3- Outside ministry people (citizens).

Security is of critical nature in such type of applications; hence, every user has a username and a password, and the user can change his password as often as he finds it necessary. Other tools for secured communication channels and backup should also be utilized.

4.2.1. Business Requirements

The purpose of the anticipated system is to send/receive E-mails in an organization and also stores these Emails into the database. To cater for these two main activities, a number of options should be supported: send, forward, reply, reply all, next, previous, set priority, move, and maintain folders (e.g. inbox, outbox, etc).

In addition, an administration subsystem is required to define and maintain organization's structures. This includes the creation/editing/deletion of sectors, departments, and sections. Also, there are different types of "Employee types" in each section: minister, secretary, undersecretary, manager, sections chef, and employees

. Table 4.1 summarizes the list of functionalities required for the anticipated system, and figure 4.1. presents the corresponding use case model that joins the system functionalities with the corresponding actor(s).

Table 0.1: System Functionalities.

No.	Description	Requirement Type
Admin Section		
1.	Admin login, and admin user management	Must
2.	Employee - Add/edit/Delete/View/Search	Must
3.	Employee Category - Add/edit/Delete/View/Search	Must
4.	Sector – Add/Edit/Delete/View	Should
5.	Department – Add/Edit/Delete/View	Should
6.	Section – Add/Edit/Delete/View	Should
User section		
1.	User login	Could
2.	Registration and edit profile	Must
3.	Role management of the users	Should
4.	Forgot password	Could

5.	Sending Emails with multiple upload options and according to the rules specified and role of the user	Should
6.	Maintaining the address book of all users according to the rules	Could
7.	Forward, Reply, Reply all, Next, Previous, Move options	Must
8.	Email options like set priority etc	Could
9	Maintain folders like inbox, outbox, etc / create folder options	Should
	Create message rules and apply	Won't

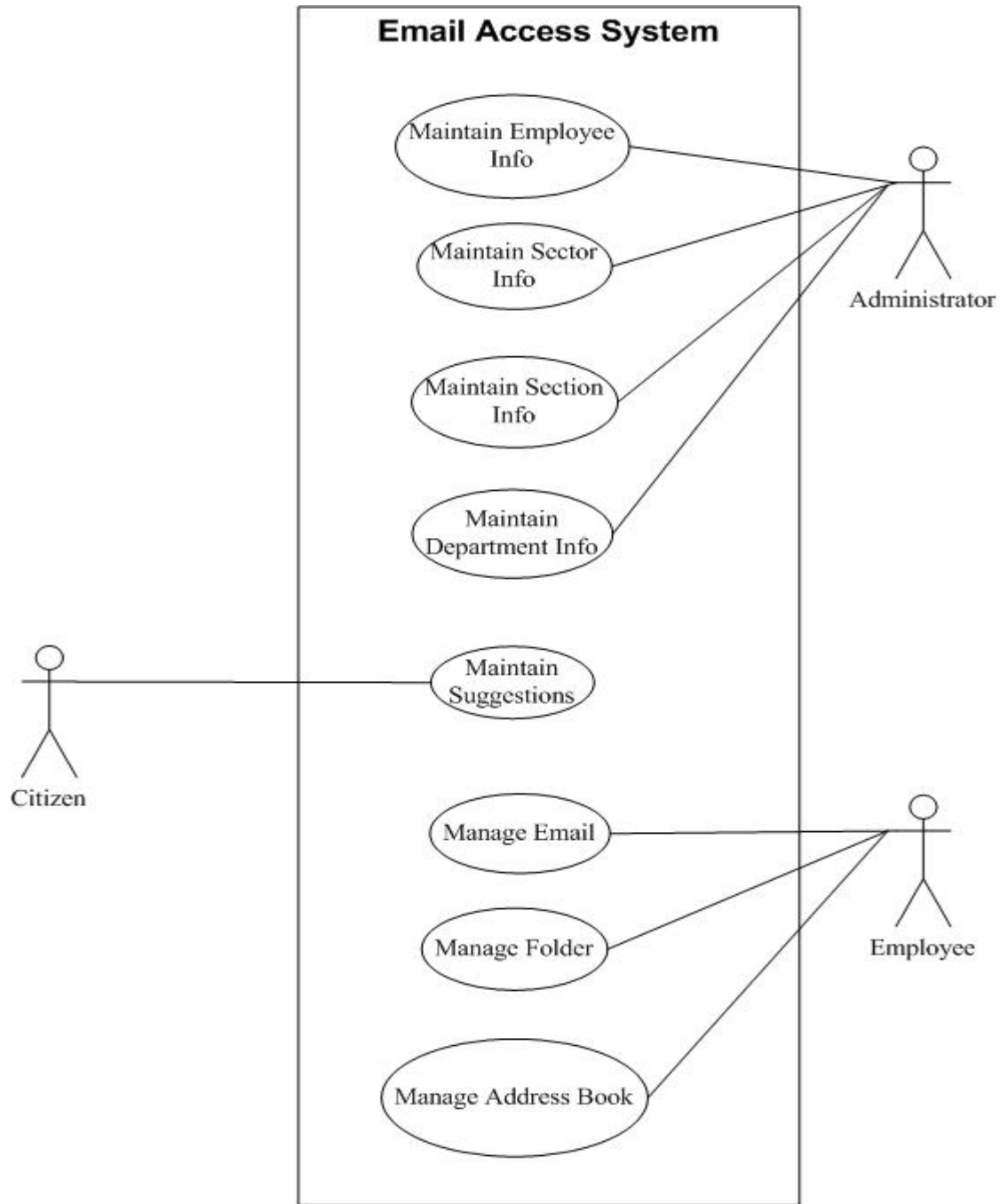


Figure 0.1: System Use Case Model.

The main outcome of identifying the system stakeholders and their corresponding functionalities is the identification of the main

subsystems of the system as presented in figure 4.2. This has been further analyzed to design the object oriented three tiers architecture of the system as depicted in figure 4.3.

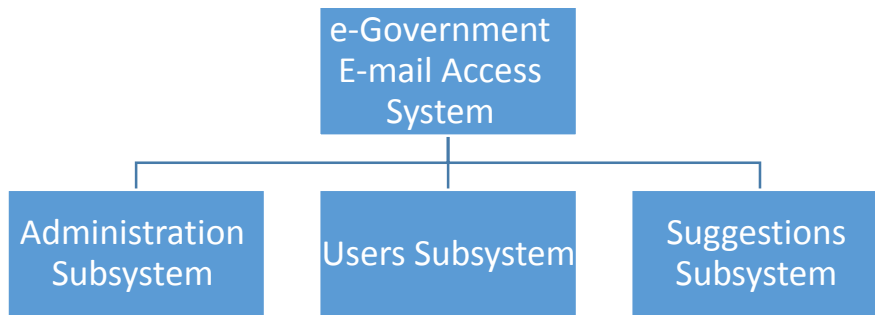


Figure 0.2: Subsystem of Email Access System.

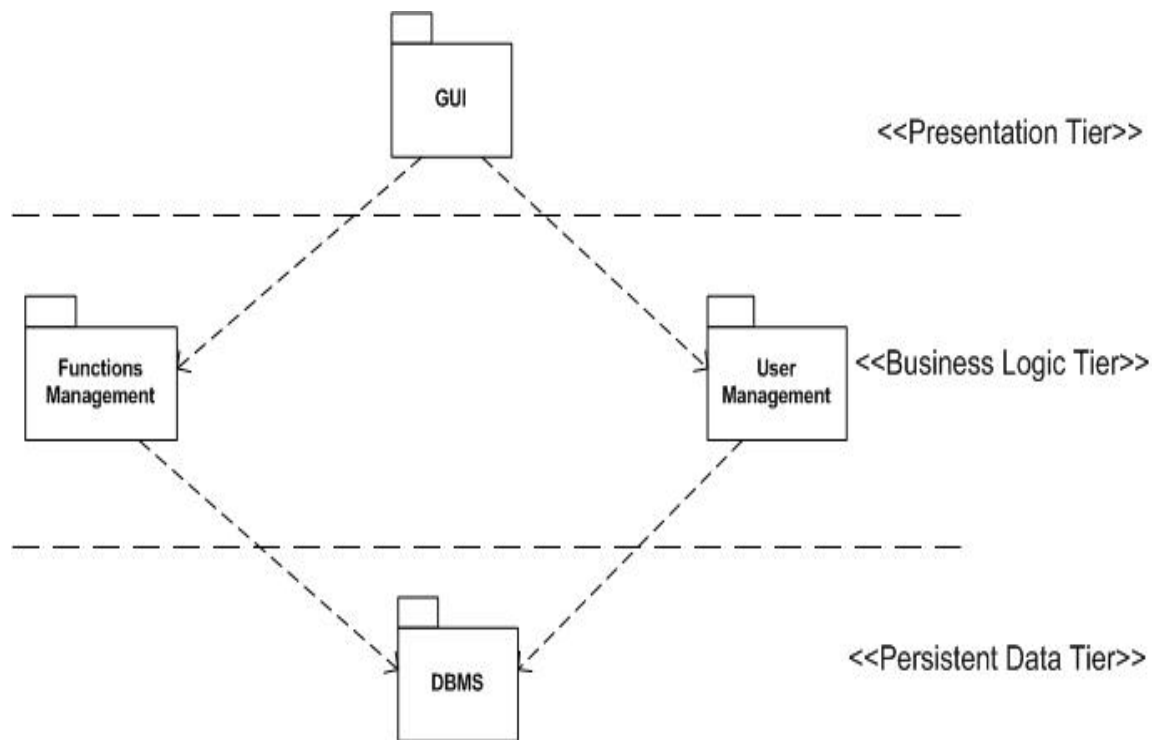


Figure 0.3: System Architecture.

4.3. Iteration Two

The second iteration is longer than the first one that lasted for five working weeks. The main activities of this iteration are: requirements analysis, system design, and testing (requirements and design validation).

4.3.1. Requirements Analysis

In the administration section of the system, all the application related permissions are maintained. The anticipated system has different categories like Sectors, Departments and Sections. Each category contains different Employee Types and the admin has the permissions to Create, Edit, and Delete each category and each employee type. The employee types are: Minister, Secretary, Undersecretary, Manager, Sections chef, Employees.

The administrator creates, edits, or deletes the sectors information. Each sector having different fields like: Sector name, Location and Phone number. Each Sector contains different “Departments” in which the administrator creates, edits, or deletes its information. Each department has different fields like: Sector name, Department name,

Location and Phone number.

Each department contains different “Sections”. In each section, the administrator creates, edits, or deletes the Section’s information. Each section has different fields like: Sector name, Department name, Section name, Location and Phone number.

Each section has employees, in which the administrator creates, edits or deletes. In this context, the Employees Role should be defined (“minister, secretary, undersecretary, manager, section chef, employee). Also sector, department and sections information should be defined. Figure 4.4 depicts the elaborated use case diagram for the maintain employee info use case as an example of how each use case in the system level use case model has been elaborated to drive the development of subsequent phases. On the other hand, figure 4.5 shows the elaborated use case diagram for the “Maintain Sugestions” use case.

When a particular employee interacts with the system, the system ensures that this user is authorized to do so based on his/her username privileges specified in his/her respective user-profile. The system prompts for User ID and Password. The

system verifies that the Username provided corresponds to a valid user profile and that the password matches that of the user-profile's password.

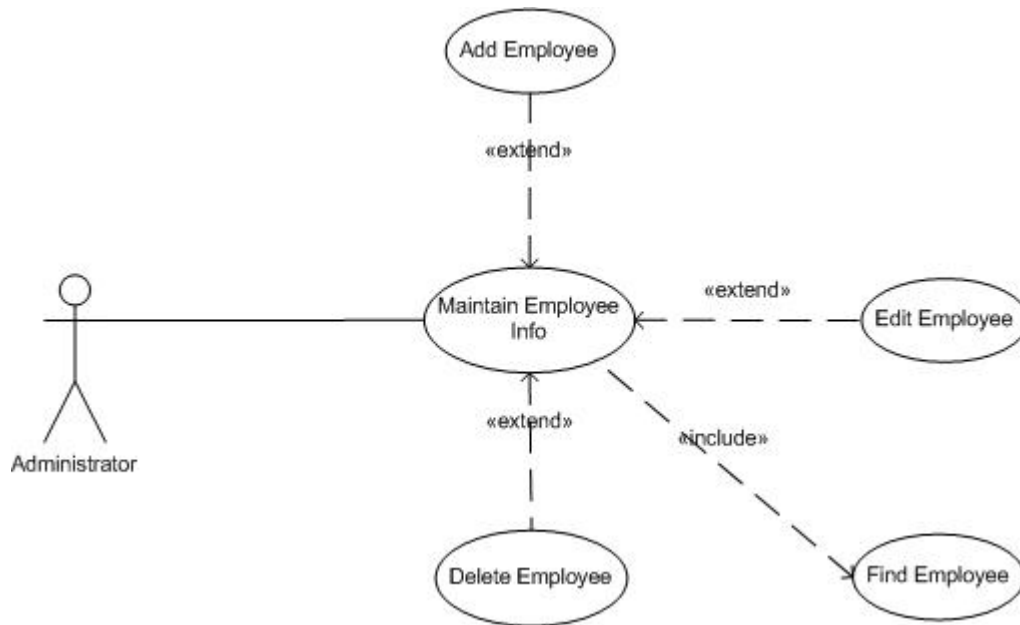


Figure 0.4: Use Case Diagram of Maintain Employee Info Use Case.

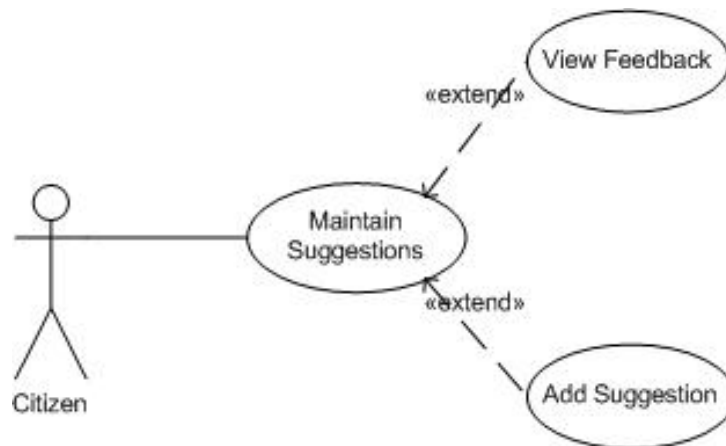


Figure 0.5 Use Case Diagram of Maintain Suggestions Use Case.

Based on user-right privileges stored in the user's user-profile (as actor roles), specific services (use-cases) can be used by this user. The user is properly authenticated for the task he/she will attempt to use.

Else, if the User ID does not exist or the password does not match, then the system denies further operation as follows:

The login screen with empty user ID and password fields is redisplayed for another attempt of login.

In a third failing attempt, the system returns control with inability to use any service.

If the user authorization is unsuccessful then the system does not recognize the user and, subsequently, shall not grant access to the system. Figure 4.6 shows an abstracted sequence diagram for the login process.

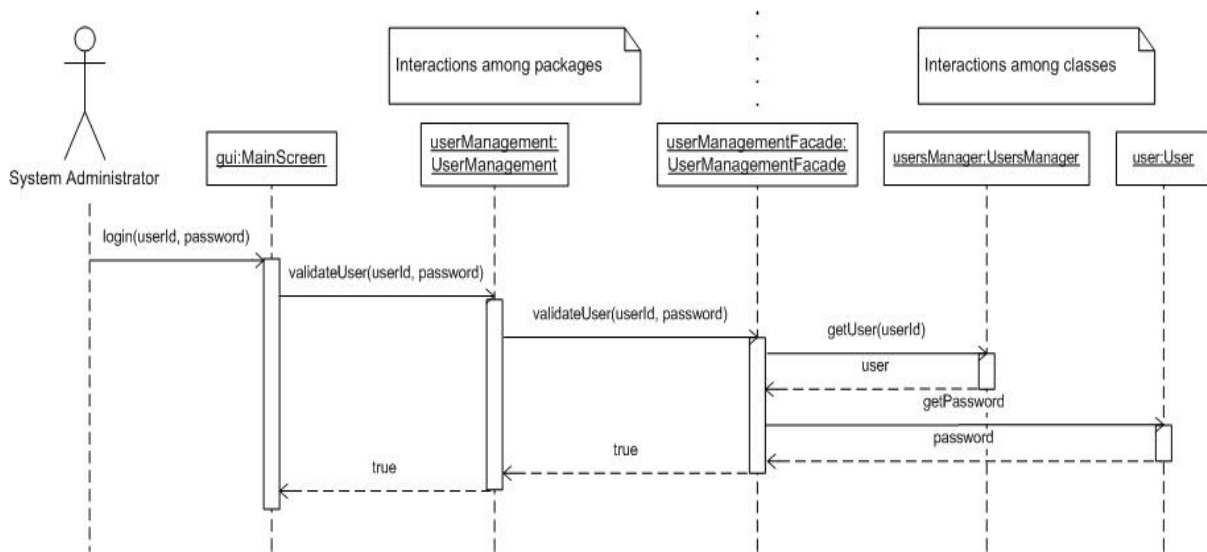


Figure 0.6: Login User Sequence Diagram.

In the “minister/secretary” part of the system, corresponding users have permissions to send emails to all of the users in the organization. Here five options are available: Minister/Secretary, All Undersecretaries, All Managers, All Sections chefs, and All Employees. There are two options to send emails. Those are: (1) “Multiple select in receiver list”, and (2) “Upload attachment”.

“Under secretary” has permissions to send emails to particular users in the organization. Here it consists of four options: secretary , All under secretaries , All managers in his sector , All section chef’s in his sector , All employees in his sector. There are two options to send emails. Those are: (1) “Multiple select in receiver list”, and (2) “Upload

attachment”.

“Managers” have permissions to send Emails to particular users in the organization. Here it consists of four options: under secretary in his sector, all managers in sector, all section chefs in his Department, and All employees in his Department. There are two options to send emails. Those are: (1) “Multiple select in receiver list”, and (2) “Upload attachment”.

“section chef’s” has permissions to send Emails to particular users in the organization. Three options are available: manager of his department, all section chefs in his department and all employees in his section. There are two options to send emails. Those are: (1) “Multiple select in receiver list”, and (2) “Upload attachment”.

“Employees” have permissions to send Emails to particular users in the organization. Two options are available: section chef and all employees in his section. There are two options to send emails. Those are: (1) “Multiple select in receiver list”, and (2) “Upload attachment”.

4.3.2. System-level Non-Functional Requirements

NFRs of a system are defined as constraints on the services offered by the systems. Email Access NFRs are classified based on Sommerville's [47] classification as: product, organisational, and external requirements.

a.Product

Product-related requirements specify the characteristics that Email Access System should possess. The most important product-related NFRs for the client are *usability*, *security*, and *reliability*. However, additional NFRs have been considered such as *portability and maintainability*.

b. Usability

Email Access should provide easy-to-use GUI for its users. Email Access usability should be enhanced by providing informative error messages, online help facilities, and consistent user interface. In addition, the system should be learned by average users within two hours.

c.Security

Security NFR should be included in Email Access to ensure that unauthorised access to the system is not allowed, and the integrity of

the underlying Email Access system. Hence, the user's roles may only be granted/revoked by the system administrator which will explicitly define the access authorities of the different users of the system. In addition, each user should have a distinct username and password to identify his authorities during the logon process to the system.

D. Reliability

The system's reliability NFR represents constraints on the run-time behaviour of the system. This includes the system availability and failure rate. Hence, Email Access should be available whenever requested by the users (i.e. 100% availability and 0% failure rate).

e. Portability

Email Access should be compatible with different operating systems. It should be operated and tested under both Windows 2000 and Windows XP operating systems. Also, different browser types and versions should be supported.

f. Maintainability

The software system should be flexible to change so as to accommodate new requirements. It should also allow changes for improvement such as improving its performance.

g.Organizational

Organisational NFRs are related to the client's and developer's organisations. They are derived from their policies and procedures. These requirements include delivery and implementation requirements.

h. Delivery

The system is required to be delivered with all its related documents (i.e. requirements, design, user guide, coding, and test plan documents) before January 2008.

i. Implementation

The system is required to be implemented using object oriented programming language to support other features such as maintainability.

g.External

External requirements are derived from aspects external to the system and its development environment. For Email Access no such requirements are defined. Nevertheless, interfacing with external systems has been investigated to inform whether these are applicable within the scope of the project.

4.3.3. System Design

The design of the Email Access system passed through a number of internal iterations. Each iteration consists of developing a corresponding class diagram and validating it against the system requirements to confirm the complete and correct realization of them. Figures 4.7 – 4.9 show three versions of the system class model before arriving at the final one shown in figure 4.10.

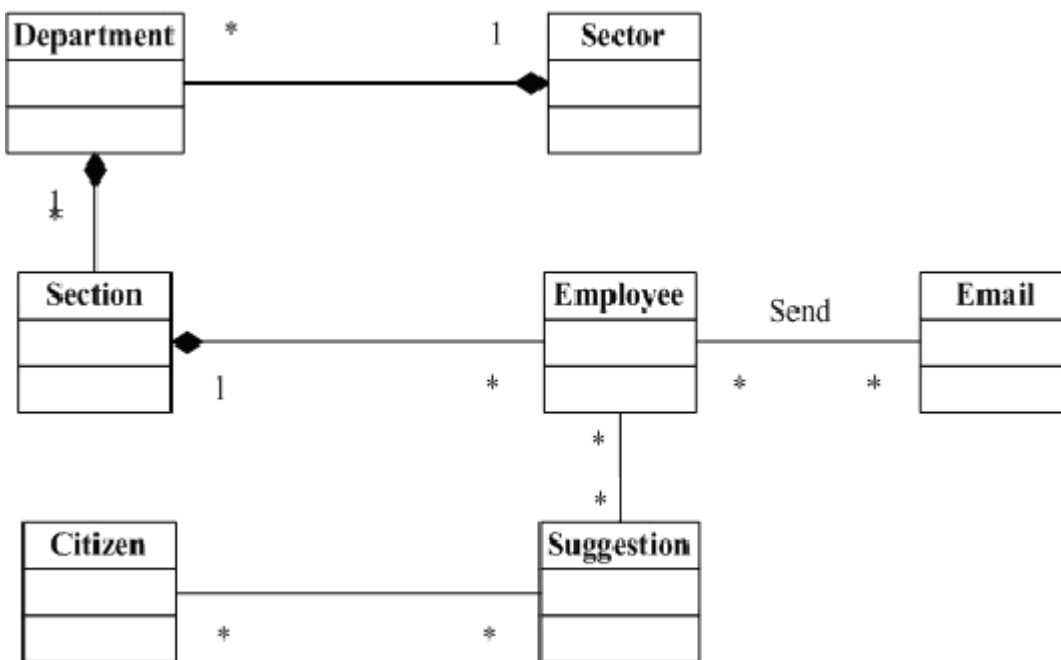


Figure 0.7: Class Model version 0.1.

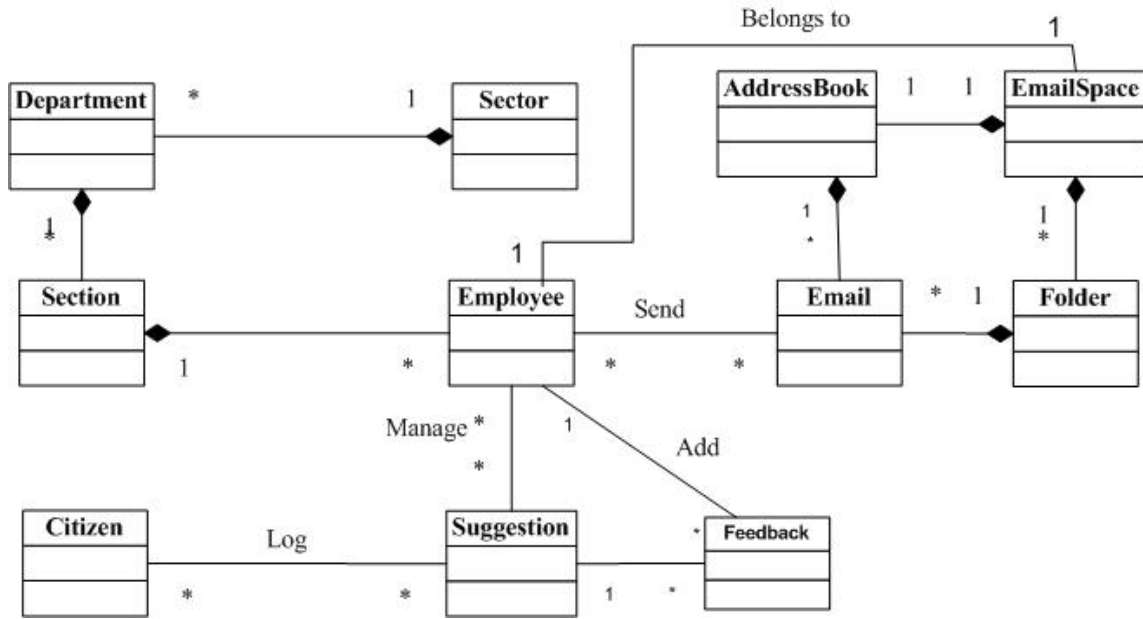


Figure 0.8: Class Model Version 0.2.

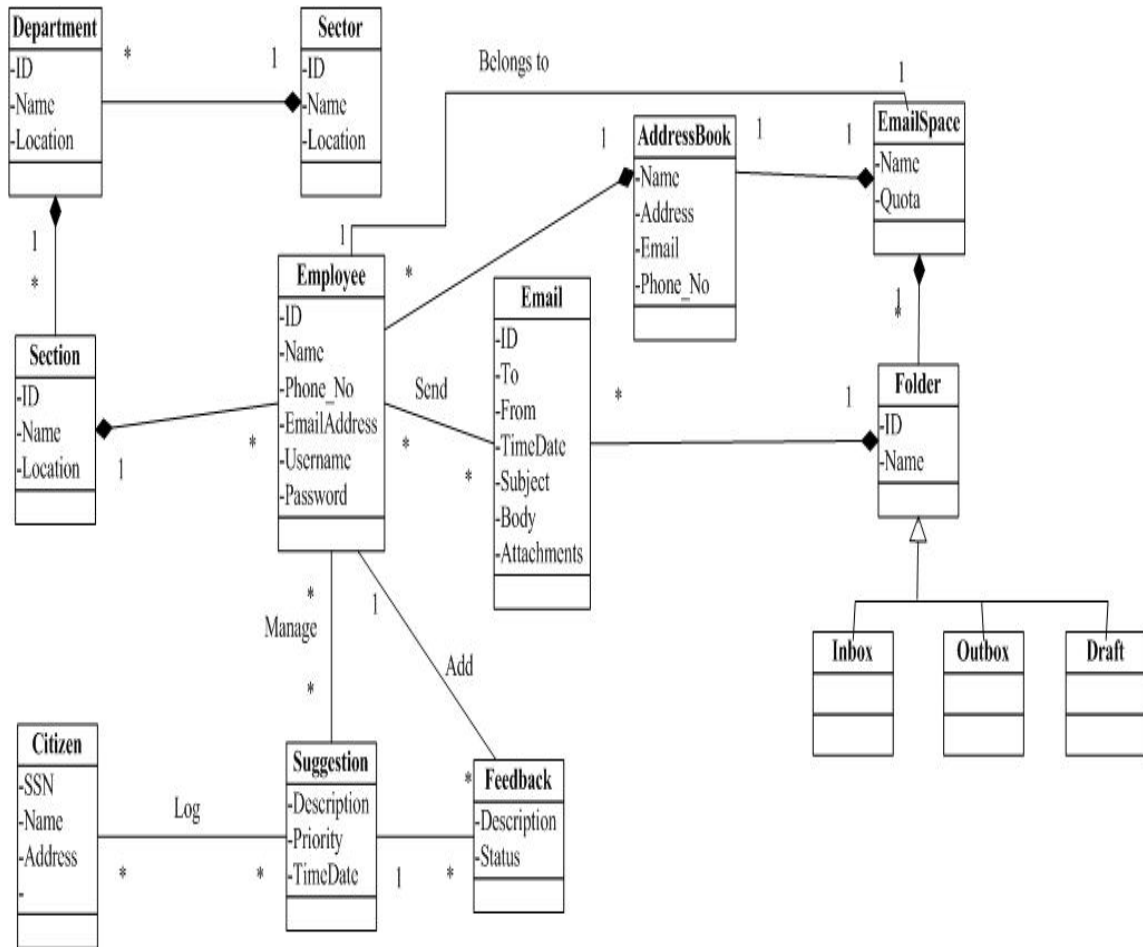


Figure 0.9: Class Model Version 0.3.

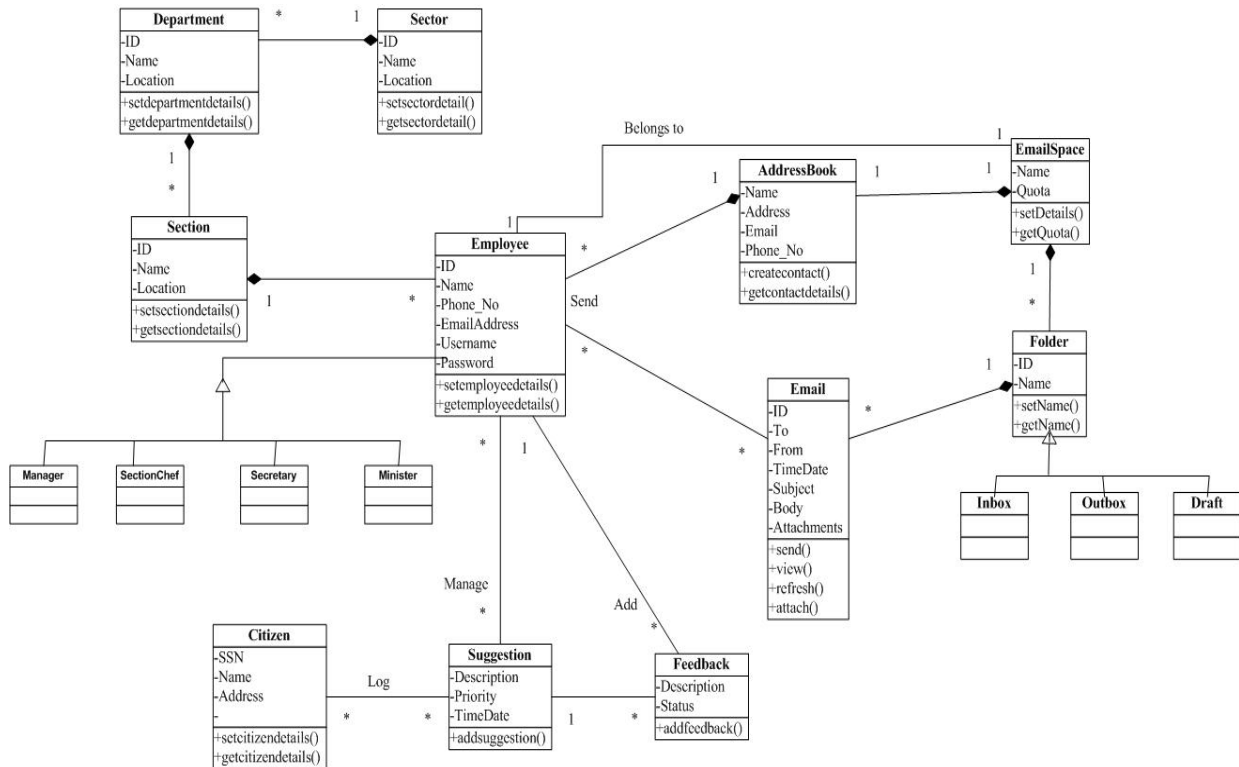


Figure 0.10: Class Model Version 1.0.

A number of object oriented design concepts has been adopted and applied, as can be concluded from figures 4.7-4.10, in Email Access system that participated in achieving system NFRs and e-government application characteristics of excellence. These include:

1. **Information Hiding:** all the implementation details of Email Access components are hidden. Users can use these components through their facades, where these facades provide the services of these components without exposing the internal implementation details. Moreover, Classes are also encapsulated to support this principle. All the classes' attributes are defined as private, and can only be accessed by other classes through their provided (public) methods.
2. **Design for specification,** not for implementation: the classes and components (packages) of Email Access have been designed based on the understanding of the problem domain. Implementation details have been ignored during this stage. As detailed earlier, interactions among classes and components have been accomplished through their provided services/interfaces without any assumptions made about the implementation details. In addition, the classes, attributes, and

3. methods' names are not related to the implementation. Instead, they reflect the purpose of defining and using them (their specification).
4. **Low coupling and high cohesion:** As mentioned earlier, classes and components have been designed to be loosely coupled and highly cohesive. Classes, and their methods, have been designed to do one thing and do it well.

4.3.4. Requirements and Design Validation

Validating the requirements and design of Email Access system has been conducted using different techniques. Requirements and design artefacts have been reviewed for anomalies and omissions. This has been done by conducting informal reviews to check the requirements for verifiability, comprehensibility, traceability, and adaptability. The developed UC model helped in focusing this process and in developing the test cases for testing the documented requirements and their corresponding realizing design, and checking the system's verifiability before coding the system.

4.4. Iteration Three

In this iteration, the development of Email Access system is concluded. Two main activities (development and testing) were conducted within this 6 weeks iteration. However, requirements and design phases were re-visited to handle several cases of emerging functionalities.

A number of integrated technologies were used in the development of Email Access system. These include:

1. Front End Development: C# Programming Language, Java Script.
2. Middle Tier: C# Programming Language.
3. Back End Development: Microsoft SQL Server.
4. Operating System: Windows 2000 Server, Windows XP.
5. Browsers Compatibility: Microsoft internet explorer, Netscape, and Opera.

Regardless of how structured the development phase is, none can guarantee the correctness of the software system, nor provide full testing coverage, i.e. test all data inputs and all execution paths. Hence, two types of testing were adopted in the development of Email

Access system: (1) “testing to specification” which is also called *Black Box Testing*, and (2) “testing to code” which is also called White Box Testing. Both of the above techniques were used to uncover different classes of errors, and hence should not be used as alternatives to each other.

4.4.1. Black Box Testing

Black box testing technique refers to the tests that are conducted to specification. This technique tests the functionality of the software system through its provided interface. It is conducted to ensure that the software system provides all the functions specified in the specification, with little consideration for the internal logical structure of the software. In addition to testing that none of the specified functionalities is missing. Black box testing can be conducted to test the non-functional requirements that are constrained by the clients, such as performance, reliability, and security.

The idea of testing the software system through its provided interface, and without knowing the system’s internal structure, enables the users to test the system instead of the developers themselves. Studies showed that testing the software system by its users or any other

independent test group, instead of the system's developers, can give quite more confidence that the software system fits for its purpose. This is based on the idea that software testing should be destructive, and developers might be unable to destroy what they had built. They would usually design and execute tests that will show that software system works as required, instead of discovering the inadvertent errors in the system.

For the implemented system, *Email Access*, black box testing had been applied to the different system components, and then to the system as a whole. It had been conducted to test whether the system and its components correctly provide all the specified functionalities, correctly use the different data structures, and to find any initialisation, termination, or interface errors.

Black box testing had been applied to the implemented software system with the purpose of discovering the existence or absence of classes of errors, instead of uncovering errors that are only related to the specific tests performed. In order to achieve this goal, some supporting methods had been used such as *equivalence partitioning* and *boundary value analysis* methods.

4.4.2. White Box Testing

White box testing considers the internal structure of the software components to derive the test cases. In this approach, the tester should study the code so as to decide the data inputs for the test cases. These test cases should be designed to guarantee the execution of each independent path in the software component at least once, exercise both the true and false sides of all logical decisions, execute every loop within its operational bounds and at its boundaries, and exercise every data structure used in the software component to ensure its validity.

In order to determine how much code, of the developed software system, had to be exercised by the white box testing, both *operation coverage* and *path coverage* methods had been applied.

Operation coverage method had been applied to the methods/operations of the system classes, and the services provided by the system and its subcomponents. It had been applied to ensure that each of these methods and services had been exercised at least once. On the other hand, path coverage method had been applied for finding and exercising the independent execution paths in the code of

the software system, and more specifically in the code of methods and services provided by the system classes and components respectively. For small methods and services, every independent execution path had been defined and exercised at least once. But, for the large methods, this was quite difficult. Thus, the most critical and frequently used paths had been defined and exercised instead.

Figures 4.11 – 4.14 represent screenshots from the developed and tested Email Access system, Samples of source code of these screens are shown in Appendix A.

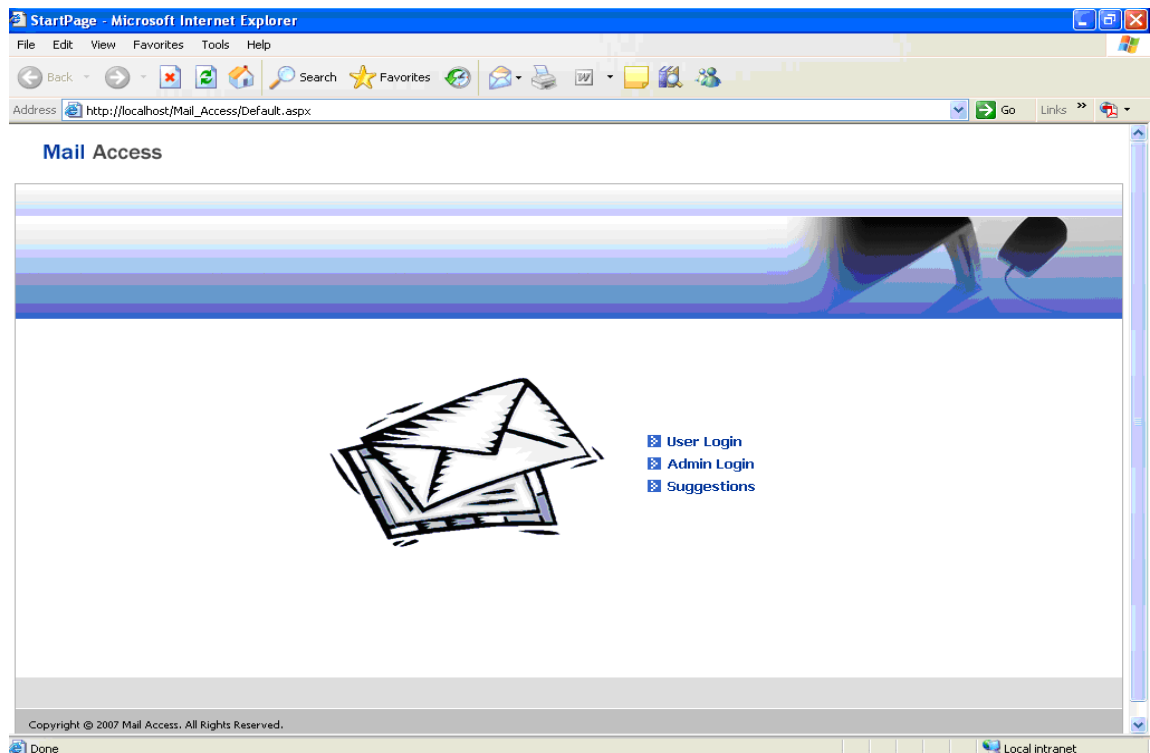


Figure 0.11: Email Access Options Page.

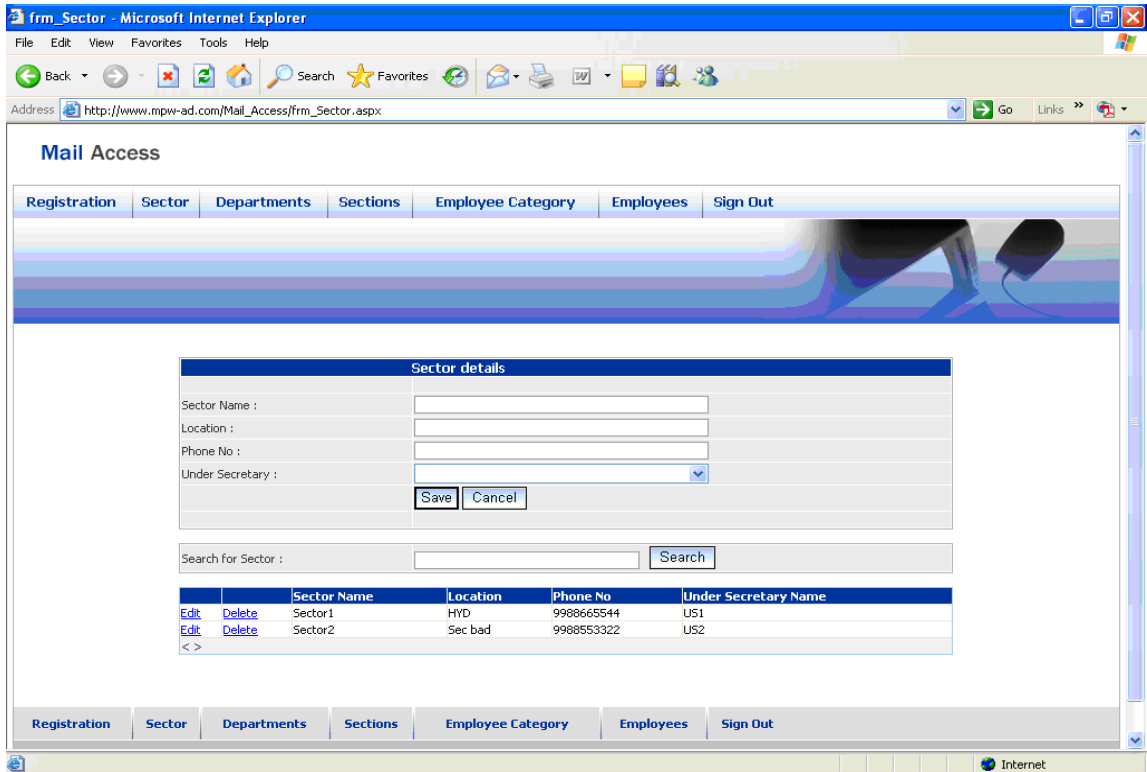


Figure 0.12: Email Access Sector Maintenance Page.

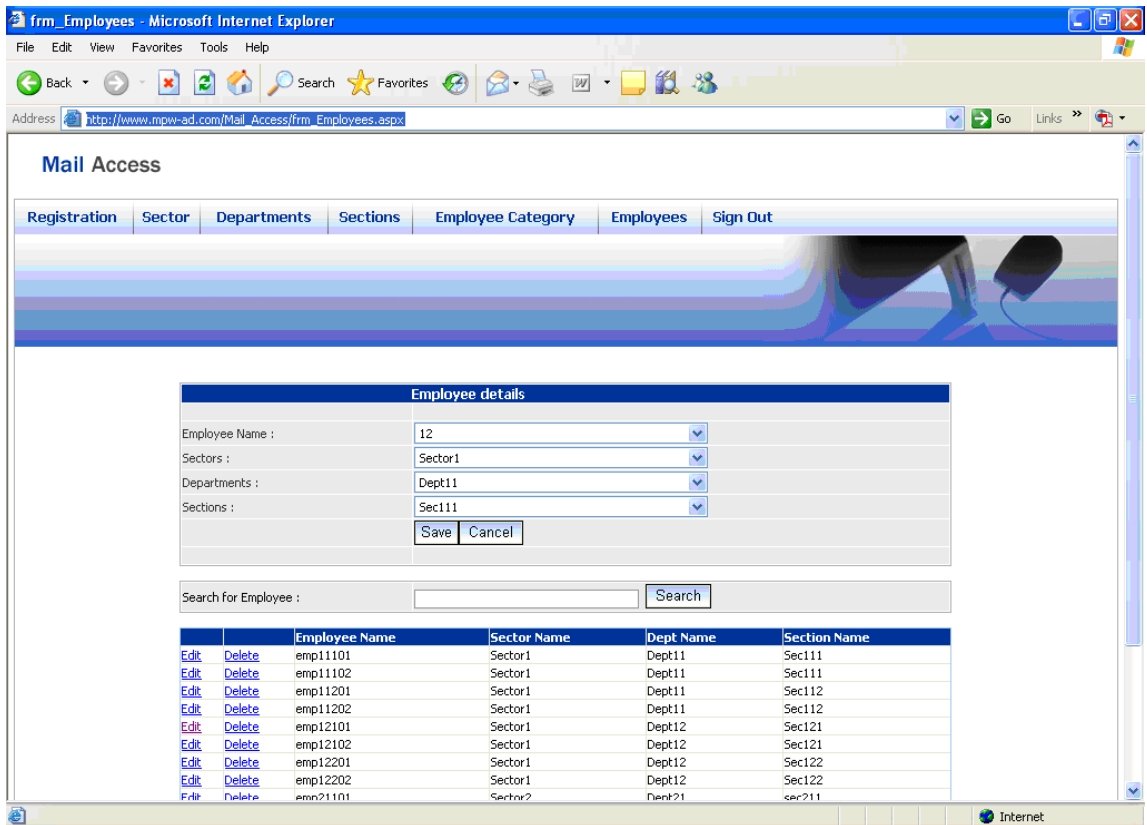


Figure 0.13: Email Access Assign Employees to Section Page .

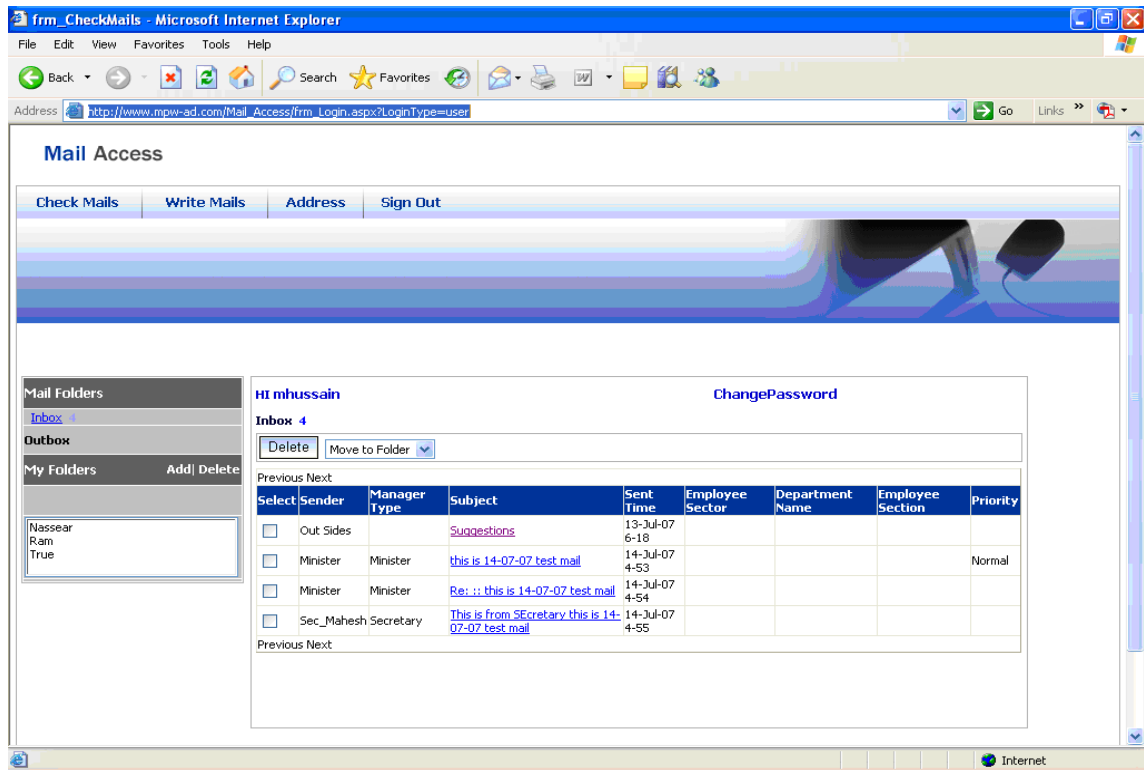


Figure 0.14: Email Access User Inbox Page.

4.5. Risk Management

Any research and software development is candidate to have several areas of concerns and risks. Accordingly, a list of most candidate risks throughout the research lifecycle has been prepared. In addition, a contingency plan for resolving each risk has been prepared in order to initiate appropriate corrective actions once the risk possibility increases.

In this research, two main risks have been identified. Details of the identified risks (time limits and requirements change) and the

corresponding contingency plan(s) are discussed in sections 4.5.1 and 4.5.2, respectively. These risks and others if emerged during the research together with their contingency plans are reviewed and managed regularly in each iteration.

4.5.1. Time Limits

During the research, the start of many activities will be dependent upon finishing other activities. Thus, if finishing one activity is delayed, then all subsequent activities will be delayed correspondingly. The worst case would happen if the delayed activity lies in the critical path of the research plan. If this is the case, then, the research will not be finished on time, which is firmly restricted to January, 2008.

Risk Avoidance

Each activity is assigned a period of time more than what is actually required. In addition, the research has been planned to be finalised on October 2007. However, if any delay occurs in the research, and the specified end date of the initial plan is exceeded, the remaining activities would be planned to be enacted and executed before January 2008.

Risk Minimisation

The research progress will be reviewed every two to three weeks and at the end of each iteration to ensure being on schedule. If lateness is encountered, then suitable actions would be undertaken according to the situation. Example of possible action is to increase the working time on the activity to be finished earlier (i.e. working overtime).

4.5.2. Requirements Change

Changing requirements is a problem that faces most of the development approaches especially the ones that develop the software incrementally. In addition, due to the dynamicity nature of web based applications, in general, and e-government applications, in particular, changing requirements is a real risk in this research project. Reasonable changes, which can be applied within the planned period, might be accepted. However, if this exceeded the expectation, it might confuse the development and delay the project which might result in project failure.

Risk Minimisation

The spiral model is adopted in the proposed development process. Hence, risks will be evaluated with each iteration. If requirements have

been changed, the effect of this change and the risks that might rise will be evaluated accordingly. Based on this evaluation, changes might be accepted (if their risks are low), delayed for later iterations (if risks can be managed), or rejected (if their risks are high/serious).

4.6. Critical Evaluation

Given that the anticipated object oriented design methodology is developed and tested as detailed in previous chapters, this section evaluates the proposed methodology within the framework of the adopted software development process, spiral model, *and hence*, the extent to which the research aims and objectives have been addressed. Section 4.6.1 evaluates the adopted SDLC in relation to the conducted research and development. A brief evaluation of the developed software system, Email Access, is presented in section 4.6.2. The extent to which the research questions, aims, and questions has been addressed is discussed in section 4.6.3.

4.6.1. Research Phases and Software Development Process

This research passed through different stages during its lifetime. However, two main stages dominate:

1. Investigating the problem domain of e-government applications design methodologies and proposing a new object oriented design methodology for this type of applications.
2. The application of the proposed methodology on a selected Email Access system to evaluate the proposed methodology and participating in addressing the research questions and objectives.

During the first stage, an extensive literature survey has been conducted to understand the problem domain. This survey could not find a comprehensive UC model that realizes the functionality of e-government applications. Thus, utilizing a use case based object oriented approach has been adopted in this research.

The development of the anticipated Email Access has been accomplished following a hybrid software development life cycle. Mainly, the spiral model has been followed, but other principles and practices of other software development life cycles such as XP and iterative and incremental development have been adopted. These practices include test-driven development, refactoring, continuous subsystem integration, and development of small releases.

Following the spiral model, the development of Email Access has been conducted iteratively and incrementally which is known to be a good approach for developing web-based applications, in general, and e-government applications, in particular that are characterised by high volatility and complexity. Each development phase, i.e. RE, design, implementation, and testing, has been enacted within one or more iterations until the defined purpose and objectives of this phase are achieved, evaluated, and agreed. This has led to good understanding of the system to be developed and sharing the same vision of the system between the different stakeholders of the system.

The fact that the spiral model combines the controlled systematic aspects of the waterfall model with its iterative approach facilitated the definition of static phases of the development process (RE, design, implementation, and testing), and hence, going over the problems introduced by the evolutionary development approaches that lack this feature. Having these phases in the development process helped in developing and reviewing the requirements specification which is strongly required in this research as a basis for the subsequent development phases and ignored or sacrificed in many development

approaches as discussed in chapter three. It also helped in producing good design of the software system which is based upon the produced requirements document. In addition, it facilitated the documentation of the undertaken design decisions for subsequent phases and future reference. Moreover, the prototyping approach, which is comprised within the spiral model, added the advantages of the prototyping to the development process. By that, it enabled the development team to gain good understanding of the stakeholder's requirements that have led to minimal requirements changes in the later stages of the development. Furthermore, it helped in reducing the complexity of understanding and developing the different constructs of the required e-government application as it facilitated communicating this understanding with the citizens and employees using tangible prototypes instead of only discussing ideas.

In addition, the definition of objectives, alternatives, and constraints at the beginning of each cycle within the spiral paid early attention to the reuse option of some patterns and existing software components, COTS, in the design and implementation of the Email Access instead of reinventing the wheel. This saved the time required to develop the

reused parts from scratch and testing them to ensure their correctness as the reused parts have been thoroughly tested before being published. The definition of objectives also enabled considering the required quality attributes, i.e. e-government characteristics of excellence, in the development process. Thus useability, portability, reliability, implementation, and maintainability NFRs have been considered and supported in the different development phases as discussed in previous chapters.

The adopted practices such as test-driven development, refactoring, continuous subsystem integration, and the development of small working releases during the development had positive effects on both the system and the development process. The test-driven development approach enabled effective unit testing of the developed software system and quick programming feedback. This is because the developed tests have been applied to small parts of the unit code instead of the whole implemented unit, and re-applied regularly to each small part of the functionality added to the code. Applying the tests in this way facilitated finding and then fixing bugs in the unit code as the found bugs were more likely to exist in the new code

. Refactoring helped in producing loosely coupled and highly cohesive design of the system's components and classes. It enabled fixing defects and amending parts of the design without changing the interface of the components and hence without affecting other parts of the design and implementation that are dependent upon the modified parts. Developing small releases enabled having working parts of the software system before the end of the project. This enabled the client and the development team to use a working version of the system that implements parts of the required functionality before delivering the whole system. The developed releases also minimised the risk of project failure as the most important functionalities have been delivered within the earlier releases. Continuous subsystem integration helped in early testing that the developed components of the system correctly interface with each other to provide the expected functionality. In this case, bugs were discovered and fixed earlier and not accumulated until the last stages of the development process. Also, the risk driven development of the spiral model had a positive effect on the development process as it enabled early mitigation of some of the main risks in the project.

4.6.2. The Developed e-Government Application: Email Access

The conducted research aimed to develop an e-government application using an object oriented design methodology. The implemented software system, Email Access, has been designed using UML language and developed as an OO software system in which related classes have been clustered into components that provide the defined functionality as specified in both the UC model and system design. Both system classes and components have been designed to be loosely coupled and highly cohesive in order to support the maintainability NFR. This is because highly cohesive and loosely coupled classes and components enables: (1) reusing them independently from each other, (2) replacing one or more components with others that provide the same interface, and (3) maintaining these components with no effects, or minor effects in the worst cases, on other system classes and components. In addition, new components that provide new functionality can be added to the system and use its components through their provided interfaces in order to extend its provided functionality.

Table 4.2 summarizes the characteristics of the proposed approach compared to those of existing e-government design approaches.

Table 0.2: E-government Approaches Comparison .

	E-government Design Approach				
	Bee r et al.	Marches e	Kalloniati s et al.	Tian and Tianfiel d	Our Approac h
Comprehensive				Yes	Yes
Integrated		Yes	Yes		Yes
Ubiquitous	Yes	Yes		Yes	Yes
Usability					Yes
Accessibility	Yes		Yes	Yes	Yes
Security			Yes		Yes
Privacy			Yes		Yes
Support Re-engineering		Yes			Yes
Support Evolution		Yes		Yes	Yes
Interoperable	Yes	Yes	Yes		Yes

4.6.3. Research Questions and Objectives

This section discusses the extent to which the defined problem has been solved by validating the research hypothesis and evaluating the extent to which the research questions have been answered. In addition, it discusses the extent to which the research aims and objectives have been met.

This research aimed at answering the following questions and used their answers to validate the hypothesis addressed in section 1.2:

- 1. Can UC specifications fit the functionalities of e-government applications?** This has been answered by the conducted research following the definition of the UC as a “*sequence of transactions performed by a system that yields a measurable result of values for a particular actor*” [19]. Based on this definition, the UCs have been successfully used to model all the functions/services that the system is required to provide to its end users. Moreover, their specification defined the interactions between the end users (human actors) and the developed software system. Furthermore, the UCs have also been successfully used to model some anticipated system functions/services which are supposed to drive future system extensions.
- 2. Can the UC model of the required e-government application drive the development of a functioning system throughout the Software Development Life Cycle?** The conducted research proved that UC modelling

can lead the development of Email Access throughout the development life cycle. During the initial stages of the development, UCs have been used to elicit, specify, and agree the systems functional requirements. The developed UC model also helped in designing/developing the architectural view of the system as early as possible during the development life cycle. It helped in identifying and validating the main architectural components of the system, their responsibilities, and the relations among them. Moreover, it helped in making early decisions of reusing COTS within the architecture. Furthermore, the GUI of the developed system has been designed based on the UC model as the different screens designed to provide the services defined by the embodied UCs, and traceability with the UC model has been used to validate these screens. The different classes that build up the software system have been identified in order to fulfil the functionality specified by the UCs. From the UCs' scenarios, the main classes and relations among them have been identified. After producing the detailed system design,

the UC model has been used to validate this design and ensure that it provides all the required functionality. During the implementation phase, the system has been implemented based on the produced design. In addition, the priorities of the UCs helped in planning this phase as the most important functionality has been implemented first. Test-driven development has been adopted during the implementation phase. In this case, UCs have been used to plan and implement these tests. Furthermore, the UC model has been used as the basis for planning the acceptance testing as the test cases have been designed to test all the required functionality specified by the embodied UCs. As specified with the test-driven development, UCs have been used to identify the test data required to test the different functionalities of the software system.

3. To what extent will the adoption of 12 characteristics of excellence result in a successful design and development of e-government applications? A high quality e-government application resulted from the adoption of these characteristics of excellence. This includes, usability,

4. correctness, and extensibility. Further details are provided later in this section in the context of discussing the achievement of research aims and objectives.

In addition to answering the previous research questions, this research aimed to achieve the following objectives:

1. **Using UCs as a basis for developing Graphical User Interface (GUI)-based e-government application.** This has been achieved as mentioned while answering the first research question. UC model helped in driving the different development phases of Email Access. In addition, it served as the basis for designing the GUI of the developed Email Access as each screen of the GUI has been designed to provide one of the services specified by the UCs. Consequently, the developed Email Access and its GUI have been validated based on the system's UC model.
2. **Using the different Unified Modelling Language (UML) diagrams and a collection of web-based technologies to propose a new design and implementation methodology**

OO e-government applications. The conducted research and development achieved this aim as it used the different UML models to visualise and document the different artefacts of the anticipated system including: functional requirements, architectural design, and detailed logical design. The Email Access has been designed as an OO system that is composed of classes whose objects interact with each other through messages to provide the required services. The system has been designed and implemented to be highly maintainable as it is built up of loosely coupled and highly cohesive classes that are clustered into packages, which define highly cohesive and loosely coupled subsystems/system components. These packages are fitted in three-tier architecture: data, business, and user tiers in which the different layers have relations with the directly adjacent layers only. This loose coupling among the different classes, components, and layers in addition to the high cohesion are supposed to make the system highly maintainable as these different constructs can be reused, maintained, or replaced by others that provide the same functionality and

interface without affecting, or with minor effects on, other constructs. Usability of the system has been achieved by addressing the main concerns of the system's client and users as detailed in the UC model, and then providing these needs by the implemented system. Moreover, a simple GUI is provided by the system to enable the end users to use the system throughout GUI components such as: menus, lists, text fields, and buttons. Furthermore, an online help is provided to the end users to assist them while using the system. Consistency between the different screens is considered in order to avoid confusing the users while using the system and to enable quick and easy learning and use of the system.

After answering the research questions and achieving the defined aims and objectives, the research proved the correctness of the proposed hypothesis as it proved that UC modelling can drive the development of an OO e-government application that adopts the internationally recognized characteristics of excellence. In addition, the adoption of a software engineering approach helped in developing the required system in a systematic way, managing this development

throughout its life cycle, addressing the concerns of the different stakeholders, satisfying the functional requirements, conforming to the NFRs, achieving good testing, and documenting the different aspects of the system.

4.7. Summary and Conclusion

In this chapter, the application of the proposed object oriented methodology to design and develop e-government applications is demonstrated using an Email Access system. Three iterations were needed to deliver an operational system.

The first iteration devoted to define the context and boundaries of the Email Access system. Further analysis to system requirements and building of system design were conducted in the second iteration. The third iteration specialized in implementing and testing the Email-Access system with minor revisiting to system requirements and design to cope with the continuously changing requirements.

The risks surrounding the life cycle of this research project were also discussed in this chapter along with their corresponding avoidance and minimization plans. Risk monitoring and controlling activities were explained in the framework of the adopted spiral software development process.

An evaluation of the proposed object oriented design methodology of e-government applications is also provided. This evaluation showed that the adoption of a use case based approach within the framework of a spiral model was successful as it helped in developing an Email Access that provides all the required functionality within the required characteristics of excellence. In addition, an evaluation of the outcomes of the different research phases, including the SDLC phases, is provided and the achievements of each phase with respect to the defined phase's objectives have been highlighted. Moreover, the developed software system has been evaluated. A discussion of the extent to which the defined aims and objectives of this research has been achieved, and the problem addressed and solved is presented.

Finally, the next chapter summarises the main conclusions of this research and concludes with suggestions for future work.

Chapter five: Conclusion and future work

5.1. Summary and Conclusion

This thesis reports on a software engineering methodology followed to develop an object-oriented e-government application. It describes the use-case driven approach undertaken to elicit and document requirements, design, implement, and test the developed Email Access system. The different development phases with the main outcomes of each phase are detailed in chapters three and four. However, the research questions and main aims and objectives that motivated this research and how they were tackled in this research are described below.

This research investigated the appropriateness of using a software engineering approach based on use-case modelling to develop an object oriented e-government applications by attempting to answer the following research questions:

1. Can UC specifications fit the functionalities of e-government applications?

2. Can the UC model of the required e-government application drive the development of a functioning system throughout the Software Development Life Cycle?
3. To what extent will the adoption of characteristics of excellence result in a successful design and development of e-government applications?

The outcomes of the conducted research proved that the UC specification can fit the functionality of e-government applications. And, e-government application's UC model can drive all subsequent development activities including design, implementation, and testing. Moreover, it showed that adopting the object-oriented development techniques positively affects some of the quality attributes of the resulted system such as correctness, maintainability, and efficiency as detailed in chapter four.

The main research aims and objectives have been achieved during the research as UCs have been used as a basis for developing Email Access, which is an OO e-government application that provides simple GUI for its end users. In addition, the different UML diagrams

have been utilised to model Email Access from different perspectives.

The main outcomes of this research that participated in answering the underlying research questions, investigating the reality of the research hypothesis, and achieving the research objectives could be summarised as follows:

- The development of a UC model that specifies the functionality of Email Access e-government application to drive its development. This UC model addresses current powerful functionalities of Email Access and can be extended to specify any future functionality that might be acquired by the different stakeholders.
- The development of new architectural style to present e-government applications in a three-tier architecture. The developed architecture within this research separates the components that are concerned with data storage, from those concerned with providing the business processes, and both from the GUI component(s). This architecture is supposed to enhance the maintainability and other NFRs, characteristics of excellence, of the resulted system.

- The proposal and application, Email Access system, of new object oriented design and development methodology for e-government applications.

Although the aims and objectives of this research were achieved, it is believed that its outcomes can form a basis for future research programs as outlined in the following section.

5.2. Future Work

This section provides some directions for future research programs to extend this research and improve its outcomes.

First, the developed use case model was developed in such a general way to be utilized in future extensions for Email Access system. In addition, it constructs the basis for re-development and re-engineering of Email Access to cope with emerging technologies.

The applicability and appropriateness of the proposed object oriented methodology to develop e-government applications need further assessment. Hence, the application of this proposed methodology on another case study will be considered in prospective phases of this research.

Finally, the fitness of the proposed methodology in the design and development of e-commerce applications needs to be investigated. This is due to the high intersections between the characteristics of excellence of both paradigms. Software metrics and measurement [13,16,27,43] are to be employed in this investigation for better management and informed evaluation. The main expected outcome of this investigation is a generalized methodology for the design and development of object oriented business applications.

References

- [1] Al Hakeem. M, (2007). *Saudi Telecom Sector to See Rapid Growth* [online]. Web page: Gulf News. Available from: www.Gulfnews.com [Accessed 17/7/2007].
- [2] Al-Qadhi. M, (2003). *UNDP Launches ICT Project in Yemen in Regional E-government workshop* Yemen p.13.
- [3] AME Info, (2007). *DC World, Dubai E-Government Sign MoU on Cooperation* [online]. UAE: Available from: <http://www.ameinfo.com/113342.html> [Accessed 15/7/2007].
- [4] APCICT, Incheon Republic of Korea, (2007). *Planning, Designing, Implementing, and Managing E-Government: Key Issues, Case Studies, and Lessons Learned* [online]. Republic of Korea: UNITED NATIONS. Available from: http://unapcict.org/event/eGov_tp_2007/E_Govt%20Welcome%20Letter%20-%20Director%20APCICT.pdf [Accessed 10/6/2007].
- [5] Apperly, H., Hoffman, R., Latchem, S., Maybank, B., Piper, D., Simons, C., and McGibbon, B., (2003). *Service-And Component-Based Development*. Addison Wesley.
- [6] Arab Times, (2007). *E-Governance Readiness in Gulf Progressing: UN Report, ICDL Calls on Govts to Assess Needs of IT Literacy* [online]. Kuwait: Arab Times. Available from: <http://www.egovnews.org/?p=265> [Accessed 17/6/2007].
- [7] Arabian Business, (2007). *Online Government Services Launched in Abu Dhabi* [online]. Web Page: Available from: www.ArabianBusiness.com [Accessed 15/7/2007].
- [8] Backus, M., (2001). *E-Governance and Developing Countries, Introduction and Examples* [online]. Research Report: Available from: <http://www.iicd.org/articles/IICDnews.import1857> [Accessed 17/7/2007].

- [9] Bass, L., Clements, P. and Kazman, R., (1998). *Software Architecture in Practice*. Reading, Mass. ; Harlow : Addison-Wesley.
- [10] Beck, K., (2003). *Test-Driven Development : by Example*. Boston : Addison-Wesley.
- [11] Beer, D., Kunis, R., and Runger, G., (2006). A Component Based Software Architecture for E-Government Applications *in Proceedings of First International Conference on Availability, Reliability and Security (ARES'06)* IEEE Computer Society, pp.1004-1011 .
- [12] Bennett, J., *Microsoft CEO: Gulf's IT Market 'the Fastest Growing Globally* [online]. Arabian Business. Available from: www.ArabianBusiness.com [Accessed 17/7/2007].
- [13] Bennett, K. and Rajlich, V., (2000). Software Maintenance and Evolution:a Roadmap *in International Conference on Software Engineering* New York, NY, USA IEEE-CS : Computer Society , pp.73-87.
- [14] Bittner, K. and Spence, I., (2003). *Use Case Modeling*. Boston ; London : Addison-Wesley.
- [15] Blaxter, L., Hughes, C., and M. Tight, (1996). *How to Research*. Buckingham. Englang: Open University Press.
- [16] Boehm, B., (1981). *Software Engineering Economics*. Englewood Cliffs ; London : Prentice-Hall.
- [17] Boehm, B., (1988). A Spiral Model of Software Development and Enhancement *ACM SIGSOFT Software Engineering Notes*, 11 (4), pp. 14-24.
- [18] Boersma, P., (2004). E-Government and User Centered Design [online]. Peter Boersma. Available from: <http://www.peterboersma.com/blog/2004/09/e-government-and-user-centered-design.html> [Accessed 17/6/2007].

- [19] Booch, G., Rumbaugh, J., and Jacobson, I., (1999). *The Unified Modeling Language User Guide*. Addison Wesley.
- [20] Center for Technology in Government University at Albany/SUNY, (1998). A Survey of System Development Process Models. Technical Report. CTG.MFA -003.
- [21] Collis, J. and Hussey, R., (2003). *Business Research: A Practical Guide for Undergraduate and Postgraduate Students*. 2nd edition. New York, US: Palgrave: Macmillan Business, London.
- [22] Creswell, J., (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. USA: Sage.
- [23] Department of Information Technology & Biotechnology, (1998). *E-Governance Strategy for Karnataka* [online]. India: Government of Karnataka. Available from: <http://www.bangaloreit.in/html/govtinformation/policies.htm> [Accessed 15/6/2007].
- [24] El Emam, K. and Madhavji, N., (1995). Measuring the Success of Requirements Engineering Processes in *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering, Mar 27-29 1995 York, Engl* IEEE, Los Alamitos, CA, USA, pp.204-211.
- [25] Gammeri, S., Di Cerbo, F., and Scotto, M., (2005). Open Source for E-Government Application Integration: a PHP-Based Solution in *Proceedings of the First International Conference on Open Source Systems* Genova pp.204-208.
- [26] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., (1995). *Design Patterns : Elements of Reusable Object-Oriented Software*. Reading, Mass. : Addison-Wesley.
- [27] Hughes, B., (2000). *Practical Software Measurement*. New York ; London : McGraw-Hill.

- [28] Infocomm Development Authority of Singapore , (2007). *Singapore E-Government* [online]. Singapore: Ministry of Finance. Available from: <http://www.igov.gov.sg/> [Accessed 16/6/2007].
- [29] Integrant Inc. (2003). Integrant Business Solutions Products [online]. US: Available from: <http://www.integrantinc.com/products.aspx> [Accessed 4/11/2004].
- [30] Irani, Z., Elliman T., and Themistocleous M., (2005). Evaluating Egovernment: Earning From the Experiences of Two UK Local Authorities' *Information Systems Journal*, 15 pp. 61-82.
- [31] Jacobson, I., Booch, G. and Rumbaugh, J., (1999). *The Unified Software Development Process*. Reading, Mass ; Harlow, England : Addison Wesley Longman.
- [32] Kent, B., (2000). *Extreme Programming EXplained : Embrace Change*. Reading, MA : Addison-Wesley.
- [33] Klazar, S., Nemec, J., Pribil , J., and Sumpikova, M., *E-Governance and Its Application in Area the of Programming Public Expenditures: the Case for the Czech Republic and Slovakia* [online]. Czech Republic : The Grant Agency of the Czech Republic Project. Available from: <http://unpan1.un.org/intradoc/groups/public/documents/NISPAce/ UNPAN020448.pdf> [Accessed 16/8/2007].
- [34] Kruchten, P., (1995). Architectural Blueprints - The 4 + 1 View Model of Software Architecture *IEEE Software*, 12 (6), pp. 42-50.
- [35] Kruchten, P., (2002). *The Rational Unified Process : an Introduction*. Swedish edition. Boston, Mass. ; London : Addison-Wesley.
- [36] Level A Software, (2003). Software Life Cycle Models [online]. Level A Software. Available from: http://www.levela.com/software_life_cycles_swdoc.htm [Accessed 28/1/2004].

- [37] Maciaszek, L., Liong, B., and Bills, S., (2005). *Practical Software Engineering: A Case Study Approach*. England: Addison Wesley.
- [38] Marchese, M., (2003). Service Oriented Architectures for Supporting Environments in EGovernment Applications *in Proceedings of 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)* Orlando, Florida, USA IEEE Computer Society, p.106.
- [39] NetSuite Inc, (2004). NETCRM [online]. USA: Available from: <http://www.netsuite.com/portal/home.shtml> [Accessed 16/9/2004].
- [40] Nuseibeh, B. and Easterbrook, S., (2000). Requirements Engineering: a Roadmap *in International Conference on Software Engineering Poceedings of the Conference on The Future of Software Engineering* ACM Press New York, NY, USA , pp.35-46.
- [41] Pressman, R., (2001). *Software Engineering: A Practitioner's Approach*. 5th edition. India: McGraw-Hill.
- [42] Priestly, M., (2000). *Practical Object-Oriented Design With UML*. McGraw-Hill.
- [43] Putnam, L. and Myers, W., (1992). *Measures for Excellence : Reliable Software on Time, Within Budget*. Englewood Cliffs, N.J. : Prentice Hall.
- [44] Ramco Systems Ltd., *Delivering EGovernance to Singapore: Case Study EGovernance* [online]. India: Ramco Systems Ltd. Available from: <http://www.ramco.ch/images/download/Singapore%20Government%20Projects.pdf> [Accessed 16/6/2007].
- [45] Riedl, R., (2002). *DESIGN PRINCIPLES FOR E-GOVERNMENT SERVICES*. Institut für Informatik: Universität Zürich, Winterthurerstrasse 190, CH-8057 Zürich.

- [46] Sferyx Internet Based Systems, (2001). Sferyx Tools Library [online]. Italy: Available from: <http://www.sferyx.com/english/b2bcomponents/index.htm> [Accessed 9/92004].
- [47] Sommerville, I., (2001). *Software Engineering*. 6th ed. Harlow, England ; New York : Addison-Wesley.
- [48] Sommerville, I. and Kotonya, G., (1998). *Requirements Engineering: Processes and Techniques*. Chichester, New York: J. Wiley.
- [49] Sriramesh. K, (2006). *E-Government in a Corporatist, Communitarian Society: the Case of Singapore* *New Media & Society*, 8 (5), pp. 707-730.
- [50] Strassman, M., (2001). Objectives for E-Government [online]. voxpolitics.com: voxpolitics.com. Available from: <http://www.voxpolitics.com/news/voxfpub/story265.shtml> [Accessed 1/6/2007].
- [51] Tian, J. and Tianfield, H., (2003). *Object-Oriented Design of E-Government System: A Case Study in Lecture Notes in Computer Science*. Germany: Springer Berlin / Heidelberg.
- [52] Varatek Software, Inc., (2002). B-Liner 2002 [online]. Varatek Software. Available from: <http://www.varatek.com/index.html> [Accessed 1-6-2005].
- [53] Wirfs-Brock, R., Wilkerson, B., and Wiener, L., (1990). *Designing Object-Oriented Software*. Englewood Cliffs, N.J. : Prentice-Hall.

- [54] Zundorf, A., (2001). From Use Cases to Code-Rigorous Software Development With UML *in Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001, 12-19 May 2001* Toronto, Ont., Canada IEEE Comput. Soc, pp.711-712.

Appendices

Appendix A: System Source Code Samples

A.1: Check Email

```
public class frm_CheckMails : System.Web.UI.Page
{
    protected System.Web.UI.WebControls.HyperLink
HyperLink2;
    protected System.Web.UI.WebControls.HyperLink
HyperLink3;
    protected System.Web.UI.WebControls.Label Label1;
    protected System.Web.UI.WebControls.Label Label2;

    Data data_Layer=new Data();
    protected System.Web.UI.WebControls.DataGrid
Datagrid1;
    static OleDbConnection Ocon;
    static DataTable dtCheckMail,dtExistOutdieMails;
    protected System.Web.UI.WebControls.Label lblMails;
    protected System.Web.UI.WebControls.ListBox
lstPersonallInfo;
    protected System.Web.UI.WebControls.LinkButton
InkAddPersonalFolder;
    static DataTable dtExistMailids;
    protected System.Web.UI.WebControls.Label lblMsg;
    protected System.Web.UI.WebControls.DropDownList
DropDownList1;
    protected System.Web.UI.WebControls.Button
btn_Delete;
    protected System.Web.UI.WebControls.LinkButton
InkDeleteFolder;
    //String EmpID;
    static String Subjet,sendDatetime;
    static string EmpNo,str_EmpName;
    protected System.Web.UI.WebControls.Label lblname;
    protected System.Web.UI.WebControls.LinkButton
Ink_ChangePassword;
```

```

        protected System.Web.UI.WebControls.LinkButton
InkInbox;
        protected System.Web.UI.WebControls.ListBox
lstOfficialInfo;
        protected System.Web.UI.WebControls.Label Label3;
        protected System.Web.UI.WebControls.Label
lblUsername;
        protected System.Web.UI.WebControls.Label lblCount;
        //DataTable dTCheckMail=new DataTable();
        Int16 int_MailID;
        private void Page_Load(object sender,
System.EventArgs e)
        {

            try
            {
                if((string)Session["LoginUser"]==" ||
Session["LoginUser"].ToString()==String.Empty)
                {

                    Response.Redirect("default.aspx");
                }
            }
            catch (Exception ex)
            {
                Response.Redirect("default.aspx");
            }

            //Session["LoginUser"]="Emp005";
            String str_emp=Session["LoginUser"].ToString();
            EmpNo=Session["LoginUser"].ToString();
            lblMsg.Text="";
            if (!IsPostBack)
            {
                if (lstOfficialInfo.Items.Count>0)
                {
                    int int_Folders;
                    for
(int_Folders=0;int_Folders<lstPersonalInfo.Items.Count;int_Folders+
+)

```

```

        if
(IstOfficialbfo.Items[int_Folders].Selected==true)
        {
            IstOfficialbfo.Items[int_Folders].Selected=false;
        }
    }
    str_EmpName=GetUserName(EmpNo);

    data_Layer.Set_Str_Conn="Provider=Microsoft.Jet.Oledb.4.0;
    Data Source=" + Server.MapPath ("Data")+
    @"\\Email.mdb";
    Ocon=new
OleDbConnection(data_Layer.Set_Str_Conn);
    string EmpId=Session["UserID"].ToString();
    lblUsername.Text=EmpId;

    InkDeleteFolder.Attributes.Add("onclick", "javascript:return
    validateListBox();");

    IstPersonallInfo.Items.Clear();
        DropDownList1.Items.Clear();
        GetUserFilders(EmpNo);
        // Put user code to initialize the page here
        Label3.Text="Inbox";
        Buil_Grid();
    }

}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    //
    // CODEGEN: This call is required by the ASP.NET
Web Form Designer.
    //
    InitializeComponent();
    base.OnInit(e);
}

```

```

    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    private void InitializeComponent()
    {
        this.InkInbox.Click += new
System.EventHandler(this.InkInbox_Click);
        this.InkAddPersonalFolder.Click +=
        new
System.EventHandler(this.InkAddPersonalFolder_Click);
        this.InkDeleteFolder.Click += new
System.EventHandler(this.InkDeleteFolder_Click);
        this.lstPersonalInfo.SelectedIndexChanged +=
        new
System.EventHandler(this.lstPersonalInfo_SelectedIndexChanged);
        this.Ink_ChangePassword.Click += new
System.EventHandler(this.Ink_ChangePassword_Click);
        this.btn_Delete.Click += new
System.EventHandler(this.btn_Delete_Click);
        this.DropDownList1.SelectedIndexChanged +=
        new
System.EventHandler(this.DropDownList1_SelectedIndexChanged);
        this.Datagrid1.PageIndexChanged +=
        new
System.Web.UI.WebControls.DataGridPageChangedEventHandler(t
his.Datagrid1_PageIndexChanged);
        this.Load += new
System.EventHandler(this.Page_Load);
    }
    #endregion

    private void Buil_Grid()
    {
        try
        {

```

```
String str_getEmpName="SELECT s.mail_id as
[MailID],s.Emp_No as [Employee Number],
s.Emp_Name as [Sender], s.Catg_Name as
[Manager Type], s.Subject as [Subject],
s.Sent_Time as [Sent Time], sctr.Sector_Name as
[Employee Sector], dept.dept_Name as
[Department Name], Sect.Section_Name as
[Employee Section],s.Priority as [Priority]";
```

```
str_getEmpName += "FROM";
str_getEmpName += "[select
sctr.Sector_Name , s.mail_id, s.Emp_No,
s.Emp_Name,s.Catg_Name,s.Subject,
s.Sent_Time,s.Priority, dept.dept_Name";
```

```
str_getEmpName += " from";
```

```
str_getEmpName += " (select s.mail_id,
s.Emp_No, s.Emp_Name,s.Catg_Name,s.Subject,
s.Sent_Time,s.Priority, dept.dept_Name";
str_getEmpName += " from ";
str_getEmpName += " (select emp.emp_no,
Emp.Emp_Name, Catg.Catg_Name,Mails.Subject,
ms.mail_id, Mails.Sent_Time,Mails.Priority ";
str_getEmpName += " from tbl_EmpDetails
as Emp, Tbl_MilSent as Ms, Tbl_EmpCategory
Catg, tbl_Mails Mails";
str_getEmpName += " where ";
str_getEmpName += "
Emp.Emp_No=Ms.SentFrom_EmpID and MS.SentTo_EmpID="" +
EmpNo +"
and Ms.Folder=0";
str_getEmpName += " and Emp.Catg_Id =
Catg.Catg_Id";
str_getEmpName += " and Mails.Mail_Id=
Ms.Mail_Id) as s";
str_getEmpName += " left join";
str_getEmpName += " (select ed.Emp_No,
d.Dept_Name from tbl_departments d,
Tbl_EmpDetails ed ";
```

```

        str_getEmpName += " where
d.Dept_ID=ed.Dept_ID) as dept";
        str_getEmpName += " on";
        str_getEmpName += " s.Emp_No =
dept.Emp_No) as s";
        str_getEmpName += " left join ";
        str_getEmpName += " (select n.Emp_No,
se.Sector_Name from tbl_sectors se,
        Tbl_EmpDetails n ";
        str_getEmpName += " where
se.Sector_ID=n.Sector_ID ) as sctr";
        str_getEmpName += " on";
        str_getEmpName += " s.Emp_No=
sctr.emp_No]. AS S LEFT JOIN [Select n.Emp_No,
        sec.Section_Name from tbl_sections sec,
Tbl_EmpDetails n ";
        str_getEmpName += " where
sec.Section_ID=n.Section_ID ]. AS Sect ON s.Emp_No =
        Sect.Emp_No";
        str_getEmpName += " union select ' ' as
[MailId]," as [Employee Number],'Out Sides'
        as [Sender],' ' as [Manager Type],'Suggestions' as
[Subject], Sent_Time AS [Sent
        Time],' ' as [Priority] , 'AS [Employee Sector], ' ' as
[Department Name], ' ' as
        [Employee Section] from Tbl_SuggestionSent
WHERE Sent_To_EmpId="" + EmpNo + """;

```

```

        OleDbDataAdapter daCheckMail=new
OleDbDataAdapter(str_getEmpName,Ocon);
        dtCheckMail=new DataTable();
        daCheckMail.Fill(dtCheckMail);
        lblMails.Text="";
        if (dtCheckMail.Rows.Count>0)
        {

```



```

lblMails.Text=(dTCheckMail.Rows.Count).ToString ();
lblCount.Visible=true;

lblCount.Text=(dTCheckMail.Rows.Count).ToString ();
Datagrid1.DataSource=dTCheckMail;
Datagrid1.DataBind();

LinkButton InkSubject=new
LinkButton();

int m;

for(m=0;m<Datagrid1.Items.Count;m++)
{

InkSubject=(LinkButton)Datagrid1.Items[m].Cells[4].FindControl("Ink
Sub");

string
str_nbsp=Datagrid1.Items[m].Cells[5].Text;
if
(Datagrid1.Items[m].Cells[5].Text.Trim()=="&nbsp;")
{

InkSubject.Text="None";
String
SenderId,Toadd,Sub,senderName;

SenderId=Datagrid1.Items[m].Cells[1].Text;

senderName=Datagrid1.Items[m].Cells[2].Text;

Toadd=Datagrid1.Items[m].Cells[5].Text;

Sub=Datagrid1.Items[m].Cells[6].Text;

InkSubject.Attributes.Add("OnClick","javascript:mailopen(""+
SenderId +", ""+ Toadd +", ""+ Sub +", "" + str_EmpName + "", "" +
senderName + "");");

```

```

    }
    else
    {

        InkSubject.Text=Datagrid1.Items[m].Cells[5].Text;
        String
SenderID,Toadd,Sub,senderName;

        SenderID=Datagrid1.Items[m].Cells[1].Text;

        senderName=Datagrid1.Items[m].Cells[2].Text;

        Toadd=Datagrid1.Items[m].Cells[5].Text;

        Sub=Datagrid1.Items[m].Cells[6].Text;

        InkSubject.Attributes.Add("OnClick","javascript:mailopen(""+
SenderID +"," + Toadd +"," + Sub +"," + str_EmpName + "," +
senderName + ");");
    }
}
else
{
        DataTable dtMAilnull = new
DataTable();

        dtMAilnull.Columns.Add("Sender",typeof(String));
        dtMAilnull.Columns.Add("Manager
Type",typeof(String));

        dtMAilnull.Columns.Add("Subject",typeof(String));
        dtMAilnull.Columns.Add("Sent
Time",typeof(String));

        dtMAilnull.Columns.Add("Employee
Sector",typeof(String));

        dtMAilnull.Columns.Add("Department
Name",typeof(String));

        dtMAilnull.Columns.Add("Employee
Section",typeof(String));

```

```

dtMailsnull.Columns.Add("Priority",typeof(String));
        Datagrid1.DataSource=dtMailsnull;
        Datagrid1.DataBind();
        lblCount.Visible=false;
    }
}
catch (Exception ex)
{
    if(lstPersonallInfo.SelectedIndex>=0)
    {
        if (Datagrid1.CurrentPageIndex - 1 >=
0)
        {
            Datagrid1.CurrentPageIndex =
Datagrid1.CurrentPageIndex - 1;
        }
        Buil_Grid();
    }
    else
    {
        if(lstPersonallInfo.SelectedIndex>=0)
        {
            if (Datagrid1.CurrentPageIndex -
1 >= 0)
            {
                Datagrid1.CurrentPageIndex = Datagrid1.CurrentPageIndex -
1;
            }
            Buil_Grid();
        }
    }
}
}

public DataTable GetMaillds(String senderSub,String
sentsub)

```

```

        Ocon=new
        OleDbConnection(data_Layer.Set_Str_Conn);
        string str_query="Select
        Mail_ID,Sent_From_EmpID from Tbl_Mails where
        Subject='"+senderSub+"' and Sent_Time='"+sentsub+"'";
        OleDbDataAdapter daMailID=new
        OleDbDataAdapter(str_query,Ocon);
        DataTable dtMailid=new DataTable();
        daMailID.Fill(dtMailid);
        return dtMailid;
    }
    public void DeleteMails(DataTable dt)
    {
        int k;
        if(dt.Rows.Count>0)
        {
            for(k=0;k<dt.Rows.Count;k++)
            {
                string str_query="Delete from
                Tbl_MilSent where Mail_ID="+dt.Rows[k].ItemArray[0]+" and
                SentFrom_EmpID="+dt.Rows[k].ItemArray[1]+" and
                SentTo_EmpID="+dt.Rows[k].ItemArray[2]+"";
                OleDbCommand DelMails=new
                OleDbCommand(str_query,Ocon);
                Ocon.Open();
                DelMails.ExecuteNonQuery();
                Ocon.Close();
            }
        }
    }

    private void InkDeleteFolder_Click(object sender,
    System.EventArgs e)
    {
        if (IstPersonallInfo.SelectedIndex == -1)
        {

```

```

        lblMsg.Text="Select Folder to Delete !!";
        return;
    }
    Ocon=new
OleDbConnection(data_Layer.Set_Str_Conn);

    DeleteFolderMails(lstPersonallInfo.Selected.Value,EmpNo);
        OleDbCommand DelFoder=new
OleDbCommand("Delete from Tbl_Folder where Folder_Name="" +
lstPersonallInfo.SelectedItem.Text + "" and Emp_No="" + EmpNo +
"",Ocon);
        Ocon.Open();
        DelFoder.ExecuteNonQuery();
        Ocon.Close();
        lblMsg.Visible=true;
        lblMsg.Text=lstPersonallInfo.SelectedItem.Text + "
Folder Deleted Successfully";
        GetUserFilders(EmpNo);
        DataTable dtMAilnull = new DataTable();
        dtMAilnull.Columns.Add("Sender",typeof(String));
        dtMAilnull.Columns.Add("Manager
Type",typeof(String));
        dtMAilnull.Columns.Add("Subject",typeof(String));
        dtMAilnull.Columns.Add("Sent Time",typeof(String));
        dtMAilnull.Columns.Add("Employee
Sector",typeof(String));
        dtMAilnull.Columns.Add("Department
Name",typeof(String));
        dtMAilnull.Columns.Add("Employee
Section",typeof(String));
        dtMAilnull.Columns.Add("Priority",typeof(String));
        Datagrid1.DataSource=dtMAilnull;
        Datagrid1.DataBind();
    }

    private void Datagrid1_PageIndexChanged(object
source,
System.Web.UI.WebControls.DataGridPageChangedEventArgs e)

```

```

        try
        {
            Datagrid1.CurrentPageIndex=e.NewPageIndex;
            if (IstPersonallInfo.SelectedIndex>=0)
            {
                Buil_GridFolderMails();
            }
            else
            {
                Buil_Grid();
            }
        }
        catch (Exception ex)
        {
        }
    }

    public void GetUserFilders(String EmpIID)
    {
        IstPersonallInfo.Items.Clear();
        DropDownList1.Items.Clear();
        Ocon=new
OleDbConnection(data_Layer.Set_Str_Conn);
        OleDbDataAdapter daGetFolders=new
OleDbDataAdapter("Select distinct Folder_Name,Folder_Id from
tbl_Folder where Emp_No='"+EmpNo+"'",Ocon);
        DataTable dtFolders=new DataTable();
        daGetFolders.Fill(dtFolders);

        DropDownList1.DataSource=dtFolders;

        DropDownList1.DataTextField="Folder_Name";
        DropDownList1.DataValueField="Folder_Id";
        DropDownList1.DataBind();

        DropDownList1.Items.Add("Move to Folder");
        DropDownList1.SelectedIndex
=DropDownList1.Items.Count - 1;
    }
}

```

```

        IstPersonallInfo.DataSource=dtFolders;

        IstPersonallInfo.DataTextField="Folder_Name";
        IstPersonallInfo.DataValueField="Folder_Id";
        IstPersonallInfo.DataBind();

    }

    private void InkAddPersonalFolder_Click(object sender,
System.EventArgs e)
    {
        string sScript ="<script language='javascript'" +

        "window.open('frm_AddFolder.aspx?','mywin','status=no,width
=400,height=360, menubar=no,
resizable=no,status=no,toolbar=no,location=no,center:Yes')" +
        "</script>";
        Response.Write(sScript);

    }

    public DataTable GetMailIDs(String subject , String
sentTime)
    {
        Ocon=new
OleDbConnection(data_Layer.Set_Str_Conn);
        OleDbDataAdapter daGetMaillds=new
OleDbDataAdapter("Select Mail_ID from tbl_mails where
Subject='"+subject+"' and Sent_Time='"+sentTime+"'",Ocon);
        DataTable dtGetMaillds=new DataTable();
        daGetMaillds.Fill(dtGetMaillds);
        return dtGetMaillds;
    }

    public void mailsMovetoFolder(DataTable dtMailMove)
    {
        int mf;
        Ocon=new

```

```

OleDbConnection(data_Layer.Set_Str_Conn);
    if (dtMailMove.Rows.Count>0)
    {
        for(mf=0;mf<dtMailMove.Rows.Count;mf++)
        {
            OleDbCommand cmdMoveMail=new
OleDbCommand("Insert into
Tbl_MailMoveFolder(Folder_Id,Emp_No,Mail_ID)
values("+dtMailMove.Rows[mf].ItemArray[1]+"','"+dtMailMove.Rows[
mf].ItemArray[0]+"','"+dtMailMove.Rows[mf].ItemArray[2]+"")",Ocon);
            Ocon.Open();
            cmdMoveMail.ExecuteNonQuery();
            Ocon.Close();

            OleDbCommand UpdateMails=new
OleDbCommand("Update Tbl_MilSent set Folder=1 where
Mail_ID="+dtMailMove.Rows[mf].ItemArray[2]+" and
SentTo_EmpID='"+dtMailMove.Rows[mf].ItemArray[0]+""",Ocon);
            Ocon.Open();
            UpdateMails.ExecuteNonQuery();
            Ocon.Close();

        }
    }
}

public int getFolderId(String folderName)
{
    Ocon=new
OleDbConnection(data_Layer.Set_Str_Conn);
    OleDbDataAdapter dagetFolderId=new
OleDbDataAdapter("Select Folder_Id from tbl_Folder where
Folder_Name='"+folderName+""",Ocon);
    DataTable dtgetFolderId=new DataTable();
    dagetFolderId.Fill(dtgetFolderId);
    if (dtgetFolderId.Rows.Count>0)
    {

        return
(int)dtgetFolderId.Rows[0].ItemArray[0];
    }
}

```



```

        else
        {
            return 0;
        }
    }
    public String GetUserName(String employeeID)
    {
        Ocon=new
        OleDbConnection(data_Layer.Set_Str_Conn);
        OleDbDataAdapter daGetUserName=new
        OleDbDataAdapter("Select Emp_Name from Tbl_EmpDetails where
        Emp_No=" + EmpNo + """,Ocon);
        DataTable dtGetUserName=new DataTable();
        daGetUserName.Fill(dtGetUserName);
        if (dtGetUserName.Rows.Count>0)
        {
            return
            dtGetUserName.Rows[0].ItemArray[0].ToString();
        }
        else
        {
            return "";
        }
    }

    private void btn_Delete_Click(object sender,
    System.EventArgs e)
    {
        int i,j;

        DataTable
        dtDeleteOutsideMails=new DataTable();
        DataTable dtDelete=new
        DataTable();
    }

```

```

dtDeleteOutsideMails.Columns.Add("EmplId",typeof(String));
dtDeleteOutsideMails.Columns.Add("Sentteime",typeof(string))
;
//DataTable
dtExistMailids=new DataTable();

dtDelete.Columns.Add("Mailid",typeof(int));

dtDelete.Columns.Add("Sender",typeof(String));
dtDelete.Columns.Add("Receiver",typeof(String));
DataRow dr;
CheckBox
chkMailDelete=new CheckBox();
foreach(DataGridItem GridItem in Datagrid1.Items)
{

chkMailDelete=(CheckBox)GridItem.Cells[0].FindControl("chk
MailsDelete");

if (chkMailDelete.Checked == true)
{

Subjet=GridItem.Cells[5].Text;

if (Subjet=="Suggestions")
{

if (chkMailDelete.Checked == true && Subjet=="Suggestions")
{

DataRow dr_delete;

dtExistOutdieMails=new DataTable();

```

```
dtExistOutdieMails=GetOutsideMailIds(EmpNo.ToString(),GridItem.Cells[6].Text.ToString());
```

```
if (dtExistOutdieMails.Rows.Count>0)
```

```
{
```

```
for(j=0;j<dtExistOutdieMails.Rows.Count;j++)
```

```
{
```

```
dr_delete=dtDeleteOutsideMails.NewRow();
```

```
dr_delete[0]=dtExistOutdieMails.Rows[j].ItemArray[0].ToString();
```

```
dr_delete[1]=dtExistOutdieMails.Rows[j].ItemArray[1].ToString();
```

```
dtDeleteOutsideMails.Rows.Add(dr_delete);
```

```
}
```

```
}
```

```
}
```

```
GetOutsideMailIds(dtDeleteOutsideMails);
```

```
lblMsg.Text=dtDeleteOutsideMails.Rows.Count + "Deleted Successfully";
```

```
}
```

```
else
```

```

        {

            sendDatetime=GridItem.Cells[6].Text;

            dtExistMailids=new DataTable();

            dtExistMailids=GetMailIds(Subjet.ToString(),sendDatetime.ToS
tring());

            if (dtExistMailids.Rows.Count>0)

                {

                    for(j=0;j<dtExistMailids.Rows.Count;j++)

                        {

                            dr=dtDelete.NewRow();

                            dr[0]=dtExistMailids.Rows[j].ItemArray[0];

                            dr[1]=dtExistMailids.Rows[j].ItemArray[1];
                            dr[2]=EmpNo;

                                dtDelete.Rows.Add(dr);

                                    }

                                        }

                                            }

DeleteMails(dtDelete);

IbIMsg.Text=dtDelete.Rows.Count + "Deleted Successfully";

        }
    }

```

```

        if(lstPersonallInfo.SelectedIndex>=0)
        {
            Buil_GridFolderMails();
        }
        else
        {
            Buil_Grid();
        }
    }
    public void GetOutsideMaillds(DataTable dt)
    {
        int k;
        if(dt.Rows.Count>0)
        {
            for(k=0;k<dt.Rows.Count;k++)
            {
                string str_query="Delete from
Tbl_SuggestionSent where
Sent_To_Empld='"+dt.Rows[k].ItemArray[0]+" and
Sent_Time='"+dt.Rows[k].ItemArray[1]+"";
                OleDbCommand DelMails=new
OleDbCommand(str_query,Ocon);
                Ocon.Open();
                DelMails.ExecuteNonQuery();
                Ocon.Close();
            }
        }
    }
    public DataTable GetOutsideMaillds(String User,String
sentTime)
    {
        Ocon=new
OleDbConnection(data_Layer.Set_Str_Conn);
        string str_query="Select
Sent_To_Empld,Sent_Time from Tbl_SuggestionSent where
Sent_To_Empld='"+User+"' and Sent_Time='"+sentTime+"";
        OleDbDataAdapter daMailID=new
OleDbDataAdapter(str_query,Ocon);

```

```

        DataTable dtMailid=new DataTable();
        daMailID.Fill(dtMailid);
        return dtMailid;
    }

    private void
    DropDownList1_SelectedIndexChanged(object sender,
    System.EventArgs e)
    {
        int mailFold;
        string folderId,MailID;
        DataTable dtMailMoveFolder = new
    DataTable();
        dtMailMoveFolder.Columns.Add("FolderId",
    typeof(String));
        dtMailMoveFolder.Columns.Add("EmpID",
    typeof(String));
        dtMailMoveFolder.Columns.Add("MailID",
    typeof(int));
        DataRow drMfolder;
        CheckBox MailMove = new CheckBox();

        if
    (DropDownList1.SelectedItem.Text.ToString()!="Move to Folder")
        {
            foreach (DataGridItem griditem in
    Datagrid1.Items)
            {
                MailMove =
    (CheckBox)griditem.Cells[0].FindControl("chkMailsDelete");

                if (MailMove.Checked == true)
                {

                    String str_subject,
                    str_sentTime;

                    str_subject =

```

```

griditem.Cells[5].Text;
griditem.Cells[6].Text;

int_MaillID=Convert.ToInt16(griditem.Cells[11].Text);
DataTable dtMaillds = new
DataTable();
dtMaillds =
GetMaillds(str_subject, str_sentTime);
folderId =
getFolderId(DropDownList1.SelectedItem.Text).ToString();

drMfolder =
dtMailMoveFolder.NewRow();
drMfolder[0] =
EmpNo;
drMfolder[1] =
folderId;
drMfolder[2] = int_MaillID;

MailID=Convert.ToString(int_MaillID);
dtMailMoveFolder.Rows.Add(drMfolder);
}
}
}

int int_Ist=lstPersonallInfo.SelectedIndex;
if (lstPersonallInfo.SelectedIndex== -1)
{
int Int_Folder;

for(Int_Folder=0;Int_Folder<dtMailMoveFolder.Rows.Count;Int
_Folder++)
{
if

```

```

(GetFolderMailids(Convert.ToInt16(DropDownList1.SelectedValue),d
tMailMoveFolder.Rows[Int_Folder].ItemArray[0].ToString(),Convert.T
oInt16(dtMailMoveFolder.Rows[Int_Folder].ItemArray[2]))==false)
    {

        mailsMovetoFolder(dtMailMoveFolder);
                                lblMsg.Visible = true;
                                lblMsg.Text =
dtMailMoveFolder.Rows.Count + " Moved to " +
DropDownList1.SelectedItem.Text + " Folder";

        Buil_Grid();
                                }

else
    {

if
(UpdateFolder(Convert.ToInt16(DropDownList1.SelectedValue),dtM
ailMoveFolder.Rows[Int_Folder].ItemArray[0].ToString(),Convert.ToI
nt16(dtMailMoveFolder.Rows[Int_Folder].ItemArray[2]))==true)
    {
                                }
    }
}

else
{

                                /******Close******/

if
(lstPersonalInfo.SelectedItem.Text==DropDownList1.SelectedItem.T
ext)
    {
                                lblMsg.Visible=true;
                                lblMsg.Text="Select another Folder";
                                return;

```



```

    }
    int Int_Folder;

    for(Int_Folder=0;Int_Folder<dtMailMoveFolder.Rows.Count;Int
    _Folder++)
        {

if
(GetFolderMailids(Convert.ToInt16(1stPersonallInfo.SelectedValue),dt
MailMoveFolder.Rows[Int_Folder].ItemArray[0].ToString(),Convert.T
oInt16(dtMailMoveFolder.Rows[Int_Folder].ItemArray[2]))==false)
        {

            mailsMovetoFolder(dtMailMoveFolder);
                                lblMsg.Visible = true;
                                lblMsg.Text =
dtMailMoveFolder.Rows.Count + " Moved to " +
DropDownList1.SelectedItem.Text + " Folder";
                                GetUserFilders(EmpNo);
                                Buil_Grid();
        }

else
        {

if
(UpdateFolder(Convert.ToInt16(DropDownList1.SelectedValue),dtM
ailMoveFolder.Rows[Int_Folder].ItemArray[0].ToString(),Convert.ToI
nt16(dtMailMoveFolder.Rows[Int_Folder].ItemArray[2]))==true)
        {
                                lblMsg.Visible = true;
                                lblMsg.Text =
dtMailMoveFolder.Rows.Count + " Moved to " +
DropDownList1.SelectedItem.Text + " Folder from" +
1stPersonallInfo.SelectedItem.Text;

                                Buil_GridFolderMails();
        }
        }
}

```

```

        }
        DropDownList1.SelectedIndex
        =DropDownList1.Items.Count - 1;

    }

    private void Ink_ChangePassword_Click(object sender,
System.EventArgs e)
    {
        Server.Transfer("frm_ChangePassword.aspx");
    }

    private void Buil_GridFolderMails()
    {
        try
        {
            String str_getEmpName="Select
Rslt.SentFrom_EmpID as [Employee Number] ,Rslt.Emp_Name as
[Sender], Rslt.Catg_Name as [Manager Type],Rslt.Subject as
[Subject],Rslt.Sent_Time as [Sent Time], Rslt.Sector_Name as
[Employee Sector],Rslt.Dept_Name as [Department
Name],Scns.Section_Name as [Employee Section],rslt.Priority as
[Priority],Rslt.Mail_Id as [MailID]";

            str_getEmpName += " from";

            str_getEmpName += " (Select
Rslt.SentFrom_EmpID , Rslt.Mail_Id,
Rslt.Subject,Rslt.Sent_Time,rslt.Priority,Rslt.Emp_Name,
Rslt.Section_Id,Rslt.Catg_Name,Sctr.Sector_Name,Dept.Dept_Nam
e";

            str_getEmpName += " from ";
            str_getEmpName += " (Select
Rslt.SentFrom_EmpID ,Rslt.Mail_Id,

```

```
Rslt.Subject,Rslt.Sent_Time,rslt.Priority,Rslt.Emp_Name,  
Rslt.Dept_Id,Rslt.Section_Id,Rslt.Catg_Name,Sctr.Sector_Name";
```

```
str_getEmpName += " from";
```

```
str_getEmpName += " (Select  
Rslt.SentFrom_EmpID ,Rslt.Mail_Id,  
Rslt.Subject,Rslt.Sent_Time,rslt.Priority,Rslt.Emp_Name,  
Rslt.Sector_Id,Rslt.Dept_Id,Rslt.Section_Id,Catg.Catg_Name";  
str_getEmpName += " from";
```

```
str_getEmpName += " (Select  
Rslt.SentFrom_EmpID , Rslt.Mail_Id,  
Rslt.Subject,Rslt.Sent_Time,rslt.Priority,Emp.Emp_Name,  
Emp.Catg_Id,Emp.Sector_Id,Emp.Dept_Id,emp.Section_Id";  
str_getEmpName += " from";
```

```
str_getEmpName += " (Select  
Rslt.SentFrom_EmpID, Rslt.Mail_Id,  
Rslt.Subject,Rslt.Sent_Time,rslt.Priority ";  
str_getEmpName += " from";  
str_getEmpName += " (Select  
Sent.SentFrom_EmpID, Sent.SentTo_EmpID,MInfo.Mail_Id,  
MInfo.Subject, MInfo.Sent_Time,MInfo.Priority ";  
str_getEmpName += " from tbl_MilSent as  
Sent";
```

```
str_getEmpName += " right join";
```

```
str_getEmpName += " (select Mails.Mail_Id,  
Mails.Subject, Mails.Sent_Time,Mails.Priority";  
str_getEmpName += " from ";  
str_getEmpName += " tbl_Mails Mails";  
str_getEmpName += " where Mail_Id in";  
str_getEmpName += " (select Mail_Id from  
Tbl_MailMoveFolder where Folder_Id=" +  
IstPersonallInfo.SelectedValue + ")";
```

```
str_getEmpName += " as Minfo";
```

```

str_getEmpName += " on ";

str_getEmpName += " Sent.Mail_ID=
Minfo.Mail_Id)";

str_getEmpName += " as Rslt";

str_getEmpName += " where
Rslt.SentTo_EmpID=" + EmpNo + ") as rslt ";

str_getEmpName += " left join";

str_getEmpName += " tbl_EmpDetails as
Emp";

str_getEmpName += " on ";
str_getEmpName += " Rslt.SentFrom_EmpID
= Emp.Emp_NO) as rslt";

str_getEmpName += " left join";

str_getEmpName += " Tbl_EmpCategory as
Catg";

str_getEmpName += " on ";

str_getEmpName += " rslt.Catg_ID=
Catg.Catg_Id";

str_getEmpName += " ) as Rslt";

str_getEmpName += " left join";
str_getEmpName += " Tbl_Sectors as Sctr";
str_getEmpName += " on";

str_getEmpName += " Rslt.Sector_Id =
Sctr.Sector_ID) as Rslt left join Tbl_Departments as Dept on
Rslt.Dept_Id = Dept.Dept_ID) as Rslt left join Tbl_Sections as scns
on Rslt.Section_Id =Scns.Section_Id";

```

```

OleDbDataAdapter daCheckMail=new
OleDbDataAdapter(str_getEmpName,Ocon);

```

```

        dTCheckMail=new DataTable();
        daCheckMail.Fill(dTCheckMail);

        if (dTCheckMail.Rows.Count>0)
        {
lblCount.Visible=true;

        lblCount.Text=dTCheckMail.Rows.Count.ToString();
        Datagrid1.Visible=true;
        Datagrid1.DataSource=dTCheckMail;
        Datagrid1.DataBind();

        LinkButton InkSubject=new LinkButton();
        int m;
        for(m=0;m<Datagrid1.Items.Count;m++)
        {

            InkSubject=(LinkButton)Datagrid1.Items[m].Cells[4].FindControl("InkSub");

string str_nbsp=Datagrid1.Items[m].Cells[5].Text;

            if (Datagrid1.Items[m].Cells[5].Text.Trim()=="&nbsp;")
            {

                    InkSubject.Text="None";
                    String
SenderID,Toadd,Sub,senderName;

                SenderID=Datagrid1.Items[m].Cells[1].Text;

                senderName=Datagrid1.Items[m].Cells[2].Text;

                Toadd=Datagrid1.Items[m].Cells[5].Text;

                Sub=Datagrid1.Items[m].Cells[6].Text;

```

```
InkSubject.Attributes.Add("OnClick","javascript:mailopen(""+  
SenderID +", ""+ Toadd +", ""+ Sub +", "" + str_EmpName + "", "" +  
senderName + "");");
```

```
}
```

```
else
```

```
{
```

```
InkSubject.Text=Datagrid1.Items[m].Cells[5].Text;  
String
```

```
SenderID,Toadd,Sub,senderName;
```

```
SenderID=Datagrid1.Items[m].Cells[1].Text;
```

```
senderName=Datagrid1.Items[m].Cells[2].Text;
```

```
Toadd=Datagrid1.Items[m].Cells[5].Text;
```

```
Sub=Datagrid1.Items[m].Cells[6].Text;
```

```
InkSubject.Attributes.Add("OnClick","javascript:mailopen(""+  
SenderID +", ""+ Toadd +", ""+ Sub +", "" + str_EmpName + "", "" +  
senderName + "");");
```

```
}
```

```
}
```

```
}
```

```
else
```

```
{
```

```
DataTable dtMAilsnul = new DataTable();
```

```
dtMAilsnul.Columns.Add("Sender",typeof(String));
```

```
dtMAilsnul.Columns.Add("Manager
```

```
Type",typeof(String));
```

```
dtMAilsnul.Columns.Add("Subject",typeof(String));
```

```

        dtMAilssql.Columns.Add("Sent
Time",typeof(String));
        dtMAilssql.Columns.Add("Employee
Sector",typeof(String));
        dtMAilssql.Columns.Add("Department
Name",typeof(String));
        dtMAilssql.Columns.Add("Employee
Section",typeof(String));

        dtMAilssql.Columns.Add("Priority",typeof(String));
        Datagrid1.DataSource=dtMAilssql;
        Datagrid1.DataBind();
        lblCount.Visible=false;
    }
}
catch (Exception ex)
{
    if(lstPersonallInfo.SelectedIndex>=0)
    {
        if (Datagrid1.CurrentPageIndex - 1 >=
0)
        {
            Datagrid1.CurrentPageIndex =
Datagrid1.CurrentPageIndex - 1;
        }
        Buil_GridFolderMails();
    }
}
else
{
    if (Datagrid1.CurrentPageIndex - 1 >= 0)
    {
        Datagrid1.CurrentPageIndex = Datagrid1.CurrentPageIndex -
1;
    }
    Buil_Grid();
}
}

```

```

    }
}
public bool ExistMailCheck(String FolderID,int Mailid)
{
    OleDbCommand cmdExistMailCheck=new
OleDbCommand("Select * from Tbl_MailMoveFolder where
Folder_Id=" + FolderID + " and Mail_ID=" + Mailid + """,Ocon);
    Ocon.Open();
    if (cmdExistMailCheck.ExecuteNonQuery(>0)
    {
        return true;
    }
    else
    {
        return false;
    }

    Ocon.Close();
}

private void InkInbox_Click(object sender,
System.EventArgs e)
{
    IstPersonallInfo.Items.Clear();
    GetUserFilders(EmpNo);
    Label3.Text="Inbox";

    Buil_Grid();
}
public bool GetFolderMailids(int folderID,String emplId,int
mailId)
{
    DataTable dt=new DataTable();
    dt=data_Layer.Get_Result ("Select * from
Tbl_MailMoveFolder where Folder_Id=" + folderID + " and
Emp_No=" + emplId + " and Mail_ID="+mailId+""",Ocon);

```



```

if (dt.Rows.Count >0)
    {
return true;
    }

else
    {

return false;
    }

}

public bool UpdateFolder(int folderID,String emplId,int
mailId)
{
    OleDbCommand cmdUpdateFolder=new
OleDbCommand("Update Tbl_MailMoveFolder set
Folder_Id="+folderID+" where Emp_No=" + emplId + " and
Mail_ID="+mailId+",Ocon);
    Ocon.Open();
    cmdUpdateFolder.ExecuteNonQuery();
    Ocon.Close();
    return true;
}

public void DeleteFolderMails(String FolderId,String
EmplID)
{

    OleDbDataAdapter da_SelectFolderMails=new
OleDbDataAdapter("select Mail_ID from tbl_mailMoveFolder where
Folder_Id="+FolderId+" and Emp_No=" + EmplID+"",Ocon);
    DataTable dt_SelectFolderMails=new DataTable();
    da_SelectFolderMails.Fill(dt_SelectFolderMails);
    int int_DelMail;
    if(dt_SelectFolderMails.Rows.Count>0)

```

```

        {
            for(int_DelMail=0;int_DelMail<dt_SelectFolderMails.Rows.Count;int_DelMail++)
            {
                OleDbCommand cmd_DeleteFolderMails=new
                OleDbCommand("Delete from Tbl_MailMoveFolder where
                Mail_ID="+dt_SelectFolderMails.Rows[int_DelMail].ItemArray[0]+"
                and Emp_No="+EmpID+""",Ocon);
                Ocon.Open();

                cmd_DeleteFolderMails.ExecuteNonQuery();
                Ocon.Close();
            }
        }

        private void
        lstPersonallInfo_SelectedIndexChanged(object sender,
        System.EventArgs e)
        {
            Label3.Text=lstPersonallInfo.SelectedItem.Text;

            Datagrid1.DataSource=null;
            Datagrid1.DataBind();
            //lstPersonallInfo.SelectedItem.Text=null;
            Buil_GridFolderMails();
        }

        public int GetMailsCount(int FolderID,String EmpID)
        {
            OleDbDataAdapter daGetMailsCount=new
            OleDbDataAdapter("Select count(Mail_ID) from tbl_MailMoveFolder
            where Folder_Id="+FolderID+" and Emp_No=" + EmpID+ """,Ocon);
            DataTable dtGetMailsCount=new DataTable();
            daGetMailsCount.Fill(dtGetMailsCount);
            if (dtGetMailsCount.Rows.Count>0)

```

```

return Convert.ToInt16(dtGetMailsCount.Rows[0].ItemArray[0]);
        }

else

return 0;
    }

}

```

A.2: Default

```

public class _default : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.LinkButton
Ink_User;
        protected System.Web.UI.WebControls.LinkButton
Ink_Admin;
        protected System.Web.UI.WebControls.LinkButton
LinkButton4;
        TextBox User =new TextBox();
        private void Page_Load(object sender,
System.EventArgs e)
        {
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET
Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }
    }

```

```

/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.Ink_User.Click += new
System.EventHandler(this.Ink_User_Click);
    this.Ink_Admin.Click += new
System.EventHandler(this.Ink_Admin_Click);
    this.LinkButton4.Click += new
System.EventHandler(this.LinkButton4_Click);
    this.Load += new
System.EventHandler(this.Page_Load);

}
#endregion

private void Ink_User_Click(object sender,
System.EventArgs e)
{
    User.Text= "user";
    Response.Redirect("frm_Login.aspx?LoginType="
+ User.Text);
}

private void Ink_Admin_Click(object sender,
System.EventArgs e)
{
    User.Text= "Admin";
    Response.Redirect("frm_Login.aspx?LoginType="
+ User.Text);
}

private void LinkButton4_Click(object sender,
System.EventArgs e)
{
    Response.Redirect("Sugesstion.aspx");
}
}

```

A.3: Employees

```
public class frm_Employees : System.Web.UI.Page
{
    protected System.Web.UI.WebControls.Label Label6;
    protected System.Web.UI.WebControls.Label Label1;
    protected System.Web.UI.WebControls.Label Label2;
    protected System.Web.UI.WebControls.Label Label8;
    protected System.Web.UI.WebControls.DropDownList
ddl_Section;
    protected System.Web.UI.WebControls.Button
btn_Submit;
    protected System.Web.UI.WebControls.Button
btn_Cancel;
    protected System.Web.UI.WebControls.Label lbl_Save;
    protected System.Web.UI.WebControls.Label Label7;
    protected System.Web.UI.WebControls.TextBox
txt_Search;
    protected System.Web.UI.WebControls.DropDownList
ddl_Emp;
    protected System.Web.UI.WebControls.DropDownList
ddl_Sector;
    protected System.Web.UI.WebControls.DropDownList
ddl_Dept;
    protected System.Web.UI.WebControls.Label Label3;
    protected System.Web.UI.WebControls.DataGrid
dg_Details;

    /* Global variables */
    DataTable Datatable = new DataTable ();
    DataSet Dataset=new DataSet();
    static OleDbConnection _Connection;
    string str_sector=string.Empty;
    protected System.Web.UI.WebControls.Button
btn_Search;
    Data data_Layer= new Data();
    protected
System.Web.UI.WebControls.RequiredFieldValidator
RequiredFieldValidator1;
```

```

        protected
        System.Web.UI.WebControls.RequiredFieldValidator
        RequiredFieldValidator2;
        protected
        System.Web.UI.WebControls.RequiredFieldValidator
        RequiredFieldValidator3;
        protected
        System.Web.UI.WebControls.RequiredFieldValidator
        RequiredFieldValidator4;
        protected
        System.Web.UI.WebControls.ValidationSummary
        ValidationSummary1;
        OleDbCommand _Command;
        OleDbDataReader _DataReader;

        private void Page_Load(object sender,
        System.EventArgs e)
        {
            if (Session["Admin"].ToString()=="")
            {
                Server.Transfer("default.aspx");
            }

            string StrAdmin=Session["Admin"].ToString();
            // Put user code to initialize the page here
            if(!IsPostBack)
            {

                data_Layer.Set_Str_Conn="provider=Microsoft.Jet.OLEDB.4.0
;data source="+Server.MapPath("Data")+ @"\Email.mdb";
                Fill_Grid();
                Fill_Employee();
                Fill_Sectors();
                Fill_Departments();
                Fill_Sections();
            }
        }
        public void Fill_Employee()

```

```

        OleDbDataAdapter _Adapter=new
OleDbDataAdapter("SELECT Emp_Name from Tbl_EmpDetails
where Sector_Id=0 and Dept_Id=0 and Section_Id=0 and Catg_Id=6
order by Emp_Name",_Connection);
        _Adapter.Fill(Dataset,"emp");
        ddl_Emp.DataSource=Dataset.Tables["emp"];
        ddl_Emp.DataTextField="Emp_Name";
        ddl_Emp.DataBind();
    }
    public void Fill_Sectors()
    {
        OleDbDataAdapter _Adapter1=new
OleDbDataAdapter("SELECT Sector_Name,Sector_Id from
Tbl_Sectors order by Sector_Name",_Connection);
        _Adapter1.Fill(Dataset,"Sector");
        ddl_Sector.DataSource=Dataset.Tables["Sector"];
        ddl_Sector.DataTextField="Sector_Name";
        ddl_Sector.DataValueField="Sector_Id";
        ddl_Sector.DataBind();
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        //
        // CODEGEN: This call is required by the ASP.NET
Web Form Designer.
        //
        InitializeComponent();
        base.OnInit(e);
    }

    private void InitializeComponent()
    {
        this.ddl_Sector.SelectedIndexChanged += new
System.EventHandler(this.ddl_Sector_SelectedIndexChanged);
        this.ddl_Dept.SelectedIndexChanged += new
System.EventHandler(this.ddl_Dept_SelectedIndexChanged);
    }

```

```

        this.btn_Submit.Click += new
System.EventHandler(this.btn_Submit_Click);
        this.btn_Cancel.Click += new
System.EventHandler(this.btn_Cancel_Click);
        this.btn_Search.Click += new
System.EventHandler(this.btn_Search_Click);
        this.dg_Details.PageIndexChanged += new
System.Web.UI.WebControls.DataGridPageChangedEventHandler(t
his.dg_Details_PageIndexChanged);
        this.dg_Details.EditCommand += new
System.Web.UI.WebControls.DataGridCommandEventHandler(this.
dg_Details_EditCommand);
        this.dg_Details.DeleteCommand += new
System.Web.UI.WebControls.DataGridCommandEventHandler(this.
dg_Details_DeleteCommand);
        this.dg_Details.SelectedIndexChanged += new
System.EventHandler(this.dg_Details_SelectedIndexChanged);
        this.Load += new
System.EventHandler(this.Page_Load);

```

```

    }
    #endregion

```

```

public void Fill_Departments()
{
    if(ddl_Sector.SelectedValue=="")
    {
        return;
    }
    OleDbDataAdapter _Adapter2=new
OleDbDataAdapter("SELECT Dept_Name,Dept_Id from
Tbl_Departments where Sector_Id="+ddl_Sector.SelectedValue+"
order by Dept_Name",_Connection);
    _Adapter2.Fill(Dataset,"dept");

    ddl_Dept.DataSource=Dataset.Tables["dept"];
    ddl_Dept.DataTextField="Dept_Name";
    ddl_Dept.DataValueField="Dept_Id";
    ddl_Dept.DataBind();

```



```

    }

    public void Fill_Sections()
    {
        if(ddl_Dept.SelectedValue=="")
        {
            return;
        }
        OleDbDataAdapter _Adapter3=new
OleDbDataAdapter("SELECT Section_Name,Section_Id from
Tbl_Sections where Dept_Id="+ddl_Dept.SelectedValue+" order by
Section_Name",_Connection);
        _Adapter3.Fill(Dataset,"Section");

        ddl_Section.DataSource=Dataset.Tables["Section"];
        ddl_Section.DataTextField="Section_Name";
        ddl_Section.DataValueField="Section_Id";
        ddl_Section.DataBind();
    }
    public void Fill_Grid()
    {
        try
        {
            _Connection=new
OleDbConnection(data_Layer.Set_Str_Conn);
            OleDbDataAdapter _Adapter4=new
OleDbDataAdapter("SELECT sc.Sector_Name, dpt.Dept_Name,
se.Section_Name, ed.Emp_Name,ed.Emp_No FROM Tbl_Sectors
AS sc, Tbl_EmpDetails AS ed, Tbl_Departments AS dpt,
Tbl_Sections AS se WHERE sc.Sector_Id=ed.Sector_Id and
dpt.Dept_Id=ed.Dept_Id and se.Section_Id=ed.Section_Id and
ed.Catg_Id=6",_Connection);
            _Adapter4.Fill(Datatable);
            if(Datatable.Rows.Count>0)
            {
                dg_Details.DataSource=Datatable;
                dg_Details.DataBind();
            }
        }
    }
}

```

```

else
    {
        DataTable dt=new DataTable();
        dt.Columns.Add("Employee Name");
        dt.Columns.Add("Sector Name");
        dt.Columns.Add("Dept Name");
        dt.Columns.Add("Section Name");
        dg_Details.DataSource=dt;
        dg_Details.DataBind();
    }
}

catch(Exception ex)
{
    lbl_Save.Text=ex.Message;
}
}

private void btn_Search_Click(object sender, System.EventArgs e)
{
    try
    {
        if(txt_Search.Text.IndexOf("",0)<0)
        {
            lbl_Save.Text="";
            OleDbDataAdapter _Adapter5=new
OleDbDataAdapter("SELECT sc.Sector_Name, dpt.Dept_Name,
se.Section_Name, ed.Emp_Name,ed.Emp_No FROM Tbl_Sectors
AS sc, Tbl_EmpDetails AS ed, Tbl_Departments AS dpt,
Tbl_Sections AS se WHERE sc.Sector_Id=ed.Sector_Id and
dpt.Dept_Id=ed.Dept_Id and se.Section_Id=ed.Section_Id and
ed.Catg_Id=6 and ed.Emp_Name like
"+txt_Search.Text+"%",_Connection);
            _Adapter5.Fill(Datatable);
            dg_Details.DataSource=Datatable;
            dg_Details.DataBind();
        }
    }
}

```

```

else
    {
        lbl_Save.Text="Enter valid data to
search";
        lbl_Save.ForeColor=Color.Red;
    }
}

catch(Exception ex)
{

if(dg_Details.CurrentPageIndex-1>0)
    {
        dg_Details.CurrentPageIndex=dg_Details.CurrentPageIndex-
1;
    }
    OleDbDataAdapter _Adapter5=new
OleDbDataAdapter("SELECT sc.Sector_Name, dpt.Dept_Name,
se.Section_Name, ed.Emp_Name,ed.Emp_No FROM Tbl_Sectors
AS sc, Tbl_EmpDetails AS ed, Tbl_Departments AS dpt,
Tbl_Sections AS se WHERE sc.Sector_Id=ed.Sector_Id and
dpt.Dept_Id=ed.Dept_Id and se.Section_Id=ed.Section_Id and
ed.Catg_Id=6 and ed.Emp_Name like
"+txt_Search.Text+"%",_Connection);
    _Adapter5.Fill(Datatable);
    dg_Details.DataSource=Datatable;
    dg_Details.DataBind();
}
}

private void ddl_Sector_SelectedIndexChanged(object sender,
System.EventArgs e)
{
    OleDbDataAdapter _Adapter2=new
OleDbDataAdapter("SELECT Dept_Name,Dept_Id from
Tbl_Departments where Sector_Id="+ddl_Sector.SelectedValue+"
order by Dept_Name",_Connection);

```

```

        _Adapter2.Fill(Dataset,"dept");
        ddl_Dept.DataSource=Dataset.Tables["dept"];
        ddl_Dept.DataTextField="Dept_Name";
        ddl_Dept.DataValueField="Dept_Id";
        ddl_Dept.DataBind();

        OleDbDataAdapter _Adapter3=new
OleDbDataAdapter("SELECT Section_Name,Section_Id from
Tbl_Sections where Dept_Id="+ddl_Dept.SelectedValue+" order by
Section_Name",_Connection);
        _Adapter3.Fill(Dataset,"Section");

        ddl_Section.DataSource=Dataset.Tables["Section"];
        ddl_Section.DataTextField="Section_Name";
        ddl_Section.DataValueField="Section_Id";
        ddl_Section.DataBind();
    }

private void ddl_Dept_SelectedIndexChanged(object sender,
System.EventArgs e)
    {
        OleDbDataAdapter _Adapter3=new
OleDbDataAdapter("SELECT Section_Name,Section_Id from
Tbl_Sections where Dept_Id="+ddl_Dept.SelectedValue+" order by
Section_Name",_Connection);
        _Adapter3.Fill(Dataset,"Section");

        ddl_Section.DataSource=Dataset.Tables["Section"];
        ddl_Section.DataTextField="Section_Name";
        ddl_Section.DataValueField="Section_Id";
        ddl_Section.DataBind();
    }

private void btn_Submit_Click(object sender, System.EventArgs e)
    {
        try

```

```

        _Connection=new
OleDbConnection(data_Layer.Set_Str_Conn);

        if(btn_Submit.Text=="Save")
        {
            _Connection.Open();
            _Command=new
OleDbCommand("SELECT Emp_No from Tbl_EmpDetails where
Catg_Id=5 and Sector_Id="+ddl_Sector.SelectedValue+" and
Dept_Id="+ddl_Dept.SelectedValue+" and
Section_Id="+ddl_Section.SelectedValue+"",_Connection);

            _DataReader=_Command.ExecuteReader();
            _DataReader.Read();

string str_emp=_DataReader[0].ToString();
            _DataReader.Close();

            _Command=new
OleDbCommand("SELECT Emp_No from Tbl_EmpDetails where
Emp_Name="+ddl_Emp.SelectedValue+" and Sector_Id=0 and
Dept_Id=0 and Section_Id=0",_Connection);

            _DataReader=_Command.ExecuteReader();
            _DataReader.Read();

string str_empno=_DataReader[0].ToString();
            _DataReader.Close();

            _Command=new
OleDbCommand("UPDATE Tbl_EmpDetails set
Sector_Id="+ddl_Sector.SelectedValue+",Dept_Id="+ddl_Dept.Selec
tedValue+",Section_Id="+ddl_Section.SelectedValue+",Mgr_Id="+str
_emp+" where Emp_No="+str_empno+"",_Connection);
            _Command.ExecuteNonQuery();
            _Connection.Close();
            Fill_Grid();
            Fill_Employee();
            lbl_Save.Text="Record inserted

```

```

successfully";
        }

else if(btn_Submit.Text=="Update")
    {
        _Connection.Open();
        _Command=new
OleDbCommand("SELECT Emp_No from Tbl_EmpDetails where
Emp_Name='"+ddl_Emp.SelectedValue+"'",_Connection);

        _DataReader=_Command.ExecuteReader();
        _DataReader.Read();

string str_empno=_DataReader[0].ToString();
        _DataReader.Close();

        _Command=new
OleDbCommand("UPDATE Tbl_EmpDetails set
Sector_Id='"+ddl_Sector.SelectedValue+",Dept_Id='"+ddl_Dept.Selec
tedValue+",Section_Id='"+ddl_Section.SelectedValue+" where
Emp_No='"+str_empno+"'",_Connection);
        _Command.ExecuteNonQuery();
        _Connection.Close();

        Fill_Grid();
        Fill_Employee();
        Fill_Sectors();
        Fill_Departments();
        Fill_Sections();
        lbl_Save.Text="Record updated

successfully";

        btn_Submit.Text="Save";

    }
}

catch(Exception ex)
{
    lbl_Save.Text=ex.Message;
}

```

```

private void dg_Details_DeleteCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
    {
        try
        {
            string EmpName;
            lbl_Save.Text="";
            EmpName=e.Item.Cells[6].Text;
            string ename=e.Item.Cells[2].Text;
            string strDelete="UPDATE Tbl_EmpDetails
set Sector_Id=0,Dept_Id=0,Section_Id=0 where
Emp_No='"+EmpName+"'";
            _Command=new
OleDbCommand(strDelete,_Connection);
            _Connection.Open();
            _Command.ExecuteNonQuery();
            _Connection.Close();

            dg_Details.EditItemIndex=-1;
            Fill_Grid();
            Fill_Employee();
            lbl_Save.Text="Record deleted Successfully";

        }
        catch(Exception ex)
        {
            lbl_Save.Text=ex.Message;
        }
    }

```

```

private void dg_Details_EditCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
    {
//        try
//        {

            lbl_Save.Text="";
            string str_CmbValue="";
            Fill_Employee();

```

```

Fill_Sectors();
ddl_Emp.Items.Insert(0,e.Item.Cells[2].Text);
foreach ( ListItem obl_Item in ddl_Sector.Items )
{
    if (obl_Item.Text.Trim ().ToUpper () ==
e.Item.Cells[3].Text.Trim ().ToUpper ())
    {
        str_CmbValue =obl_Item.Value ;
        ddl_Sector.Items.Remove (obl_Item);
        break;
    }
}
ddl_Sector.Items.Insert(0,e.Item.Cells[3].Text);
ddl_Sector.Items[0].Text = e.Item.Cells[3].Text;
ddl_Sector.Items[0].Value =str_CmbValue;
Fill_Departments();

foreach ( ListItem obj_Item in ddl_Dept.Items )
{

if (obj_Item.Text.Trim ().ToUpper () == e.Item.Cells[4].Text.Trim
().ToUpper ())
    {
        str_CmbValue =obj_Item.Value ;
        ddl_Dept.Items.Remove (obj_Item);
        break;
    }
}
ddl_Dept.Items.Insert(0,e.Item.Cells[4].Text);
ddl_Dept.Items[0].Text = e.Item.Cells[4].Text;
ddl_Dept.Items[0].Value =str_CmbValue;
Fill_Sections();

foreach ( ListItem obk_Item in ddl_Section.Items )
{

if (obk_Item.Text.Trim ().ToUpper () == e.Item.Cells[5].Text.Trim
().ToUpper ())

```



```

        str_CmbValue =obk_Item.Value ;
        ddl_Section.Items.Remove (obk_Item);
        break;
    }
}
ddl_Section.Items.Insert(0,e.Item.Cells[5].Text);
ddl_Section.Items[0].Text = e.Item.Cells[5].Text;
ddl_Section.Items[0].Value =str_CmbValue;

if(btn_Submit.Text=="Save")
{
    btn_Submit.Text="Update";
}

private void dg_Details_PageIndexChanged(object
source,
System.Web.UI.WebControls.DataGridPageChangedEventArgs e)
{
    try
    {
        dg_Details.CurrentPageIndex=e.NewPageIndex;
        Fill_Grid();
    }
    catch(Exception ex)
    {
        lbl_Save.Text=ex.Message;
    }
}

private void btn_Cancel_Click(object sender, System.EventArgs e)
{
    lbl_Save.Text="";
    ddl_Emp.Enabled=true;
    btn_Submit.Text="Save";
    Fill_Employee();
}

```

```

private void dg_Details_SelectedIndexChanged(object sender,
System.EventArgs e)
    {
        }
    }

```

A.4: Login

```

public class frm_Login : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Label Label1;
        protected System.Web.UI.WebControls.Label Label2;
        protected
System.Web.UI.WebControls.RequiredFieldValidator rfv_UserName;
        protected
System.Web.UI.WebControls.RequiredFieldValidator rfv_Password;
        protected System.Web.UI.WebControls.TextBox
txt_UserName;
        protected System.Web.UI.WebControls.Button
btn_SignIn;
        protected System.Web.UI.WebControls.TextBox
txt_Pswd;
        protected System.Web.UI.WebControls.Label lbl_Signin;
        protected System.Web.UI.WebControls.Label lblType;

        Data data_Layer= new Data();
        protected System.Web.UI.WebControls.HyperLink
HypHome;
        OleDbConnection _Connection;
        string _UserType="";

        private void Page_Load(object sender,
System.EventArgs e)
        {
            _UserType=Convert.ToString(Request.QueryString["LoginTyp
e"]);

```

```

if (Page.IsPostBack == false)
    {

if(_UserType==" " || _UserType==null || _UserType==string.Empty)
    {
        Response.Redirect("default.aspx");
    }

        data_Layer.Set_Str_Conn =
"Provider=Microsoft.Jet.Oledb.4.0; Data Source=" + Server.MapPath
("Data")+ @"\Email.mdb";

if (_UserType.ToUpper() == "User".ToUpper())
    {
        lblType.Text="User Login";
    }

else if(_UserType.ToUpper() == "Admin".ToUpper())
    {
        lblType.Text="Admin Login";
    }
    }
}

#region Web Form Designer generated code

override protected void OnInit(EventArgs e)
    {
        //
        // CODEGEN: This call is required by the ASP.NET
Web Form Designer.
        //
        InitializeComponent();
        base.OnInit(e);
    }
private void InitializeComponent()

```

```

        this.btn_SignIn.Click += new
System.EventHandler(this.btn_SignIn_Click);
        this.Load += new
System.EventHandler(this.Page_Load);

    }
#endregion

```

```

private void btn_SignIn_Click(object sender, System.EventArgs e)
{

```

```

        _Connection=new
OleDbConnection(data_Layer.Set_Str_Conn);
        _Connection.Open();
        OleDbDataAdapter _Adapter;
        DataSet _Dataset=new DataSet();

```

```

        _Adapter =new OleDbDataAdapter("SELECT
User_Name,Pwd,User_Type from Tbl_EmpDetails where
User_Name='"+txt_UserName.Text+"' and Pwd='" +txt_Pswd.Text
+ "'",_Connection);
        _Adapter.Fill(_Dataset);

```

```

if ( _Dataset.Tables[0].Rows.Count>0 )
{

```

```

    Session["UserID"] = txt_UserName.Text;
    Data.str_MailAddress ="";
    Data.str_MailEmps ="";

```

```

    OleDbDataAdapter _EmpidAdapter;
    DataSet _EmpidDataset=new DataSet();

```

```

        _EmpidAdapter=new
OleDbDataAdapter("SELECT Emp_No from Tbl_EmpDetails where
User_Name='" + txt_UserName.Text + "'",_Connection);

```

```

        _EmpidAdapter.Fill(_EmpidDataset);

if (_EmpidDataset.Tables[0].Rows.Count>0)
    {
        Session["LoginUser"]=_EmpidDataset.Tables[0].Rows[0].ItemA
rray[0].ToString();
    }

if (_UserType.ToUpper() == "User".ToUpper())
    {
        _Connection.Close ();
        Server.Transfer
("frm_CheckMails.aspx");
    }

else if (_UserType.ToUpper() == "Admin".ToUpper())
    {

if ( _Dataset.Tables[0].Rows[0].ItemArray[2].ToString().ToUpper()
== "General".ToUpper())
    {
        lbl_Signin.Text="You dont have
Administrator Previliges";
    }

else if(
_Dataset.Tables[0].Rows[0].ItemArray[2].ToString().ToUpper() ==
"Admin".ToUpper())
    {

        Session["Admin"]=txt_UserName.Text;

```

```

        _Connection.Close ();
        Server.Transfer
("frm_Registration.aspx");
    }
}

else
{
    lbl_Signin.Text = "Invalid
Username/Password";

}
_Connection.Close ();
}

private void lblHome_Click(object sender, System.EventArgs e)
{
    Server.Transfer("default.aspx");
}
}

```

Arabic Summary

توظيف نمذجة الكيانات الموجهة في تطبيقات الحكومة الإلكترونية عبر الانترنت

ملخص

اعداد الطالب: ناصر العازمي

إشراف : أ.د. نعيم العجلوني

هذه الرسالة تبحث في اقتراح طريقة جديدة تعتمد على استخدام حالات الاستعمال والكيانات الموجهة لنمذجة وتصميم تطبيقات الحكومة الإلكترونية المستخدمة عبر الانترنت . النتائج العملية لتطبيق الطريقة المقترحة تنقسم الى قسمين .

أولا : توثيق حالات الإستعمال تصلح للإستخدام في تطبيقات الحكومة الإلكترونية .

ثانيا : مقاطع هيكلية النظام المتعلقة بحالات الاستعمال تصلح لقيادة المراحل المختلفة لبناء وتطوير برمجيات تطبيقات الحكومة الإلكترونية هذا بالإضافة إلى أنه قد وجد أن استخدام حالات الإستعمال غالبا ما يقود لإنشاء واجهات مستخدم جرافيكية ذات طابع مقبول ومألوف للمستخدم ومن ناحية اخرى امكانية توثيق متطلبات البرمجيات غير العملية في توثيق حالات الاستعمال أدى الى النجاح في تطبيق مواصفات الجودة العالمية الخاصة بتطبيقات الحكومة الإلكترونية في الطريقة المقترحة أولا والنظام التطبيقي ثانيا .

كل هذه النتائج أسهمت في تدعيم صلاحية الطريقة المقترحة لنمذجة وتصميم تطبيقات الحكومة الإلكترونية والتي تم توضيحها من خلال التطبيق العملي على نظام البريد الإلكتروني للحكومة الإلكترونية .

أخيرا ، البحث العملي ما زال مستمرا لتقييم صلاحية الطريقة المقترحة على مجالات اخرى من تطبيقات الحكومة الإلكترونية والتجارة الإلكترونية لأغراض تعميم غايات استعمالها .

