



ROSSURYAINDAH BINTI MAAROF

WEK 000199

E-CLINIC SYSTEM (ECS)

Supervisor : Mr. Phang Keat Keong

Moderator : Miss Rafidah Mohd. Noor

ABSTRACT

As we move towards a knowledge-driven economy in 21st century, healthcare providers are nowadays being forced to streamline operation through more efficient and cost effective office procedures. Automating these tasks results in time and materials, providing a positive return on investment.

E-Clinic System (ECS) is windows based application built to assist clinical practitioners with their daily practice. The objective is to automate manual procedures that are currently practiced in a clinic. Prototyping Model is adopted to allow the designer to build a prototype, test and modify it, and through an iterative process to complete the final product.

The development environment for ECS is Microsoft Windows 2000 Professional and Visual Basic 6 is used as its programming languages. While Microsoft Access 2000 and ADO as the Database Management System (DBMS) that is the most efficient to support all the backend processes of ECS.

As a conclusion, ECS is used to assist the clinic management by automating or assisting in repetitive and time-consuming tasks. It is enabling and empowering the tool in managing clinic information.

ACKNOWLEDGEMENT

First of all, I would like to take this opportunity to express my sincere thanks to my supervisor, Mr. Phang Keat Keong for his infinite patience and time. His invaluable assistance, helpful comments and endless understanding are mostly appreciated.

I would also like to acknowledge my moderator, Miss Rafidah Mohd.Noor for her countless suggestion during my viva session. Also to all my lecturers, who in someway or another contribute by teaching the necessary knowledge and skills.

Not to be forgotten, my beloved family, my friends and other individuals for their help, kindness, support and encourages.

Thank you very much.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER 1: OVERVIEW	
1.1 ABOUT THE SYSTEM	1
1.2 PURPOSE OF THE PROJECT	2
1.3 OBJECTIVES	3
1.4 LIMITATION IN IMPLEMENTATION	4
1.5 SCOPE	5
1.6 IMPORTANCE AND BENEFITS	6
1.7 HARDWARE AND SOFTWARE REQUIREMENTS	7
1.8 TIMELINE	9
1.9 SUMMARY OF CHAPTER 1	11
CHAPTER 2: LITERATURE REVIEW	
2.1 INTRODUCTION	12
2.2 THE IMPORTANCE OF LITERATURE REVIEW	12
2.3 CURRENT SYTEM REVIEW	13
2.3.1 Medical Centre System (Non-Computerized System)	13
2.3.2 Medical Centre System (Computerized System)	14
2.3.3 Case Study 1: Klinik Kesihatan UM	15
2.3.4 Case Study 2: Azhar's Clinic at Pantai Dalam	17
2.3.5 Case Study 3: Max-Gold Medical Clinic Software	18
2.4 SOFTWARE ARCHITECTURE	20
2.4.1 Host-Terminal Architecture	20

2.4.2	Client-Server Architecture	20
2.4.3	Tier Architecture	21
2.5	DEVELOPMENT TOOLS AND TECHNOLOGIES CONSIDERATIONS	23
2.5.1	Operating System	23
2.5.2	Programming Languages	28
2.5.3	Database	31
2.5.4	Data Access Technology	33
2.6	SUMMARY OF CHAPTER 2	35

CHAPTER 3: METHODOLOGY AND SYSTEM ANALYSIS

3.1	INTRODUCTION	36
3.2	METHODOLOGY	37
3.2.1	Rapid Prototyping Model	38
3.3	FACTS FINDING TECHNIQUES	41
3.3.1	Discussion and Interview	41
3.3.2	Observation	41
3.3.3	Sampling (Software Packages)	42
3.3.4	Research (Written Materials)	42
3.3.5	Previous Thesis	43
3.4	SYSTEM ANALYSIS	44
3.4.1	Requirement Analysis	45
3.5	SUMMARY OF CHAPTER 3	49

CHAPTER 4: SYSTEM DESIGN

4.1	INTRODUCTION	50
4.2	PROGRAM DESIGN	53
4.3	DATABASE DESIGN	56
4.3.1	Database Dictionary	56
4.4	USER INTERFACE DESIGN	58
4.5	INPUT OR OUTPUT DESIGN	60
4.5.1	Input design	60
4.5.2	Output Design	62
4.6	THE EXPECTED OUTCOME	64

4.7	SUMMARY OF CHAPTER 4	65
-----	----------------------	----

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1	INTRODUCTION	66
5.2	DEVELOPMENT ENVIRONMENT	67
5.2.1	Methodology and technique	68
5.2.3	Programming Languages	68
5.3	ALGORITHMS	69
5.4	CODING	70
5.4.1	VB 6 linking to database	73
5.4.2	VB 6 with ADO password encryption	74
5.5	SUMMARY OF CHAPTER 5	75

CHAPTER 6: SYSTEM TESTING

6.1	INTRODUCTION	76
6.2	UNIT TESTING	80
6.2.1	Examining the code	80
6.2.2	Proving code correct	81
6.2.3	Testing program components	82
6.2.4	Integration testing	83
6.3	SYSTEM TESTING	88
6.3.1	Function testing	92
6.3.2	Performance testing	93
6.3.3	Reliability, availability and maintainability	95
6.3.4	Acceptance testing	96
6.3.5	Installation testing	97
6.4	SUMMARY OF CHAPTER 6	98

CHAPTER 7: SYSTEM EVALUATION

7.1	INTRODUCTION	99
7.1.1	Feature analysis	100
7.1.2	Case Study	100
7.1.3	Survey	101
7.1.4	Formal experiment	101
7.2	PREPARING FOR AN EVALUATION	102

7.3	SELECTING AN EVALUATION TECHNIQUE	103
7.4	EVALUATING	104
7.4.1	Evaluating products	104
7.4.2	Evaluating processes	104
7.5	SYSTEM STRENGTHS	105
7.6	PROBLEMS ENCOUNTERED	106
7.7	SYSTEM LIMITATION	107
7.8	FUTURE ENHANCEMENT	108
7.9	SUMMARY OF CHAPTER 7	109
BIBLIOGRAPHY AND REFERENCE		110
APPENDIX A – USER MANUAL		112
APPENDIX B – SOURCE CODE		117

LIST OF TABLES

TABLE 1.1	The main module	5
TABLE 4.1	Data dictionary of registration	57
TABLE 4.2	Data dictionary of consultation	57
TABLE 5.1	List of ECS Development Environment Tools	67

FIGURE 4.2	The IAS Architecture	58
FIGURE 4.3	ECS Context Diagram	59
FIGURE 4.4	ECS Use Diagram	59
FIGURE 4.5	User Interface ECS Splash Screen	60
FIGURE 4.6	User Interface Login	60
FIGURE 4.7	Input Form Registration	61
FIGURE 4.8	Output Form List of Queue	62
FIGURE 4.9	Output Form MC	63
FIGURE 5.1	Algorithm of the program	64
FIGURE 5.2	Sample of transaction cards	74
FIGURE 6.1	Flowchart of Computerized Registration	83
FIGURE 6.2	Flowchart of testing	84
FIGURE 6.3	Flowchart of testing	85
FIGURE 6.4	Flowchart of testing	86
FIGURE 6.5	Flowchart of testing	87
FIGURE 6.6	Flowchart of testing	87
FIGURE 6.7	Flowchart of testing	88

LIST OF FIGURES

FIGURE 1.1	The Gantt chart	9
FIGURE 2.1	The Two-tier Architecture	22
FIGURE 3.1	The Rapid Prototyping Model	38
FIGURE 4.1	Conceptual and Technical designs	51
FIGURE 4.2	The ECS Architecture	53
FIGURE 4.3	ECS Context Diagram	54
FIGURE 4.4	ECS Zero Diagram	55
FIGURE 4.5	User Interface ECS Splash Screen	59
FIGURE 4.6	User Interface Login	59
FIGURE 4.7	Input Form Registration	61
FIGURE 4.8	Output Form List of Queue	63
FIGURE 4.9	Output Form MC Letter	63
FIGURE 5.1	Algorithm of patient treatment	69
FIGURE 5.2	Sample of frmLogin source code	74
FIGURE 6.1	An example of Component Hierarchies	83
FIGURE 6.2	Bottom-up testing	84
FIGURE 6.3	Top-down testing	85
FIGURE 6.4	Big-bang testing	86
FIGURE 6.5	Sandwich testing	87
FIGURE 6.6	Causes of faults during development	89
FIGURE 6.7	Steps in the testing process	90

CHAPTER 1: OVERVIEW

1.1 ABOUT THE SYSTEM

The project we develop do not exist in vacuum. Often, the hardware and software we put together must interact with users, with other software tasks, with other pieces of hardware, with existing database (with carefully designed sets of data and data relationships), or even with other computer systems. Therefore, it is important to provide a context for any project by knowing the boundaries of the project: what is included in the project and what is not.

What we are really asking is: Where does the project begin and end? The same question applies to any system. A system is a collection of objects and activities, plus a description of the relationships that tie the objects and activities together. Typically, our system definition includes, for each activity, a list of inputs required, actions taken, and outputs produced. Thus, to begin, we must know whether any object or activity is included in the system or not.

We describe a system by naming its parts and then identifying how the component parts are related to one another. This identification is the first step in analyzing the problem presented to us.

E-Clinic System or known as ECS is a full-featured clinic management system designed for the daily operation of a clinic on a network environment. It is windows-based application with point-and-click navigation. It also eliminates all manually handled procedures, streamline administrative workflow and provide complete web-based clinical records. Major functions include: patient registration, diagnosis, dispensary, printing of MC letter, historical patient medical records and many more.

1.2 PURPOSE OF THE PROJECT

With system management of the clinic being computerized, a course is open up to have the complete control over the users of the system, configurations and code tables. This system covered the vital aspects and clinic needs it contributes to save precious time of doctors and clinic staffs as well as lessens the procedural time delays in front office of any clinic.

Nowadays, management clinic is being forced to streamline operations through more efficient and cost effective office procedures. Automating these tasks results in time, material and labor savings, providing a positive return on investment.

E-Clinic System (ECS) is a windows-based application solution specially designed to allow a more efficient and effective control over the clinic's resources, time management and catering to a patient's requirement. It eliminates as many time-consuming tasks as possible, with features such as patient information management, consultation and other aspects of a clinic operation.

The functionalities of this system are:

- i) Keep all the patient records in the specified database for easy management.
- ii) Doctors or staff will be able to retrieve the entire patient records easily and faster when needed.
- iii) To ensure the system is secure in provide of the login system.

1.3 OBJECTIVES

- i) A system with database for storing all the information about the patient, medical records, billing, diagnosis, dispensary and many more. Also the search engine for user to find the information they want faster.
- ii) Save time, consolidated operation, minimize the data entry works; eliminate user manual work and keeping the costs down.
- iii) Easy of maintenance as the entire system will be centralized, so the work is concentrated on only one or two machines. With proper procedures taken, downtime can be minimized or even taken out of the equation all together.
- iv) A computerized system that is easily accessible by the users. Each class of user is given different kinds of access to the system depending on their needs and access privileges.
- v) The security system enables system administrators to track users through logging facilities. It allows administrators to lock out users if they try to exceed their user privileges or hacking the system.

1.4 LIMITATION IN IMPLEMENTATION

During implementation, we must consider the limitation that will affect the system. It is not considerably critical, but we must aware and provide some solution to handle it, such as the end-user of the system, current manual management, to adapt new environment, technical problem and many more.

i) Users Attitude

- The doctors are highly skilled in their field but little knowledge in handling computer.
- The clinical staffs (nurses) have to learn how to use the system, maybe the developer have to training them.

ii) Comfortable with the current management method

- Usually, the clinical staffs are comfortable with the manual management system.
- They do not want to take the risk for new changes by using the new system that being computerized. They lack of confidence to believe that the system will actually help them in daily routine.

iii) Technical Problem

- This is what the user afraid most. When they are using the system, suddenly the server and telecommunication line down.
- Or, when the database crash and not functioning well, it might affect their daily process.

1.5 SCOPE

We can think of the system at which we are looking as having a border or boundary. Some items cross the boundary to enter our system, and others are products of our system and travel out for another system’s use. The scope defines the system boundary, explaining what will be included in the system and what will not be included.

This project is to design a clinic management system for the daily operation of a clinic on a network environment. In the E-Clinic System, the fundamental force is to substitute the traditional system to an integrated networked system. Basically, the system users include the doctors and clinic staffs. The major function of this system includes patient registration, diagnosis, dispensary, printing of MC letter, historical medical records and reports. Basically, this system can divide into four module and services that provided. See Table 1.1 as stated.

MODULE NAME	SERVICES PROVIDED
LOGIN	Login system Link to new user registration page
REGISTRATION	New Patient Registration Patient Registration for treatment
CONSULTATION	Refer the patient to specialist Add new consultant (specialist)
REPORTS	List of patient or queue Printing of MC Letter and receipt

Table 1.1 The Main Module

1.6 IMPORTANCE AND BENEFITS

The importance of this system and the benefits can help us with the workflow. It may increase the productivity, easy of use, save time and cost benefit. Also it helps the staff to give a quality work when serving the patients as it saves energy by using computerized system.

- i) To avoid common manual errors
- ii) The full history of patients and their respective prescriptions
- iii) Automatic Invoice generation. Just point and click mouse
- iv) Fast review of client's account
- v) Maintain all listed clinic details
- vi) Very high end system for low budget
- vii) It is a cost effective integrated solution addressing the major processes and information requirements of a clinic operation.

1.7 **HARDWARE AND SOFTWARE REQUIREMENTS**

Hardware is the physical, touchable, material parts of a computer or other system. The term is used to distinguish these fixed parts of a system from the more changeable software or data components, which it executes, stores, or carries.

Computer hardware typically consists chiefly of electronic devices (CPU, memory, display) with some electromechanical parts (keyboard, printer, disk drives, tape drives, loudspeakers) for input, output, and storage, though completely non-electronic (mechanical, electromechanical, hydraulic, biological) computers have also been conceived of and built.

Software can be split into two main types - system software and application software or application programs. System software is any software required to support the production or execution of application programs but which is not specific to any particular application. Other broad classes of application software include real-time software, business software, scientific and engineering software, embedded software, personal computer software and artificial intelligence software.

Software includes both source code written by humans and executable machine code produced by assemblers or compilers. It does not usually include the data processed by programs unless this is in a format such as multimedia that depends on the use of computers for its presentation.

As stated below, are the hardware and software that maybe require building the system. But, it is not exactly what we will use. It all depends during the implementation.

- i) Computers
- ii) Modem and Networking
- iii) Microsoft Word 2000
- iv) Microsoft SQL Server 2000 or Microsoft Access 2000
- v) Visual Basic.Net or Visual Basic 6.0
- vi) Printer
- vii) At least 16 MB RAM
- viii) At least 10 GB hard disk space
- ix) Windows 9x or Windows NT or Windows 2000 or Windows XP
- x) Pentium II or equivalent

1.8 **TIMELINE**

Scheduling is a very important task in developing a system. The project schedule describes the software development cycle for this project. The most important reason to have scheduling is to have the time management so that the system can be finish on time. It helps to systematically organize the project by partitioning it into various discrete parts that can be done within a period of time. This enables the project developer to organize and rephrase the developer’s steps accordingly.

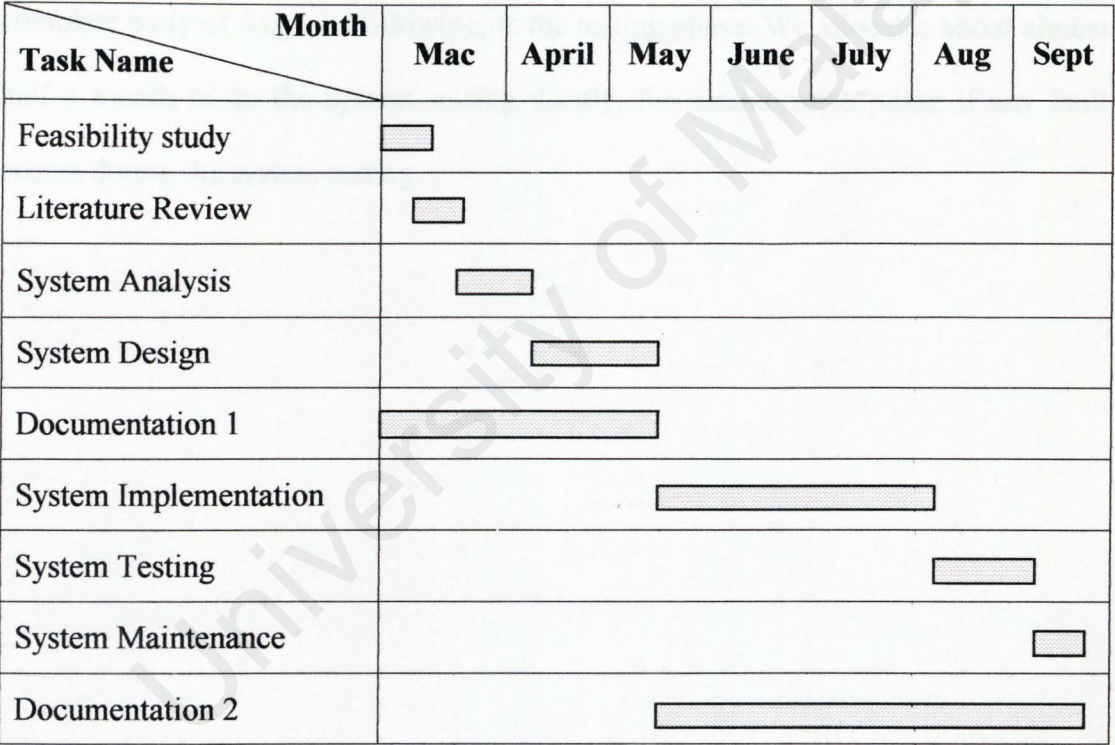


Figure 1.1: Gantt chart represent the Schedule Planned for the Project

Figure 1.1 shows the work breakdown and time allocation for each job that has to be done. As shown in the chart, the development of the project will start of with a conduct on the Feasibility Study. This was done at the beginning of the project.

Requirement Definition be later determined with some help from the supervisor. The Requirement Analysis shortly follows this, where by an in depth study on the algorithms and techniques that may be used for the development of the project will be carried out. A decision will be finalized at the end of this analysis.

System Design will then be done in the month of April. This phrase includes the designing of user interface, system flow and database. Documentation will be generated through out the system development.

After the system design is finalized, the implementation begins. During this phase, a lot of effort, work, hard times and energy to build the system so that it will complete early of August. Following, is the testing phase. We allocates about almost half a month to do the system testing. Lastly, the maintenance phase if any fault occurs during the system testing.

1.9 SUMMARY OF CHAPTER 1

As an introduction to the project, overview of the system gives a general idea on what the project is about. The implementation of this project will produce an electronic way of managing a clinic by developing an integrated window-based application with point-and-click navigation. It is explains the purpose and objective of this project.

This is then followed the scope, which is the parameter of the application section, includes the limitations in implementation. The importance and benefits are the advantages of this system. Thus, the hardware and software that we will use are the tools to help developing the project. Lastly, the project schedule will guide us during the development so that the system will complete within the time given.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

In the process of developing on existing or newly system, there are a lots of activities need to be carried out. Researches should be become the most important and main activity before the development phase start. Research ought to be performed until completely understanding the system is reached. The studies in this project have to review the different between the manual and computerized management information system. It also presents the advantages that will be beneficial from the information system.

2.2 THE IMPORTANCE OF LITERATURE REVIEW

Literature review is important to a project as it places the project in the context of others, which might have similar characteristic. It also helps to understand the existing features that are offered by a similar system. Besides, literature review also assists to equip the developers with some knowledge of the strength and limitations of several development tools and technologies in the market. This will help the developers to choose the right tools to build up the system.

2.3 CURRENT SYSTEM REVIEW

By reviewing the current system, we can use the idea when building the system to replace an older version, either manual or automated. We want to understand as much as possible about how both the old and new system work.

2.3.1 Medical Centre System (Non-Computerized System)

There are many medical centres still using manual method to manage medical centre's information such as patient and medical reports. All patient information, drug information and treatment involved huge amounts of paper documents.

In the traditional way, patients have to show their registration card. The nurse or clerk finds the patient's record based on their name. After found the patient record, patient will be given one queuing number. That file will be handed to physician.

The physician traditionally record clinical information as hand written progress notes in files. Hand-written information is neither structured nor coded, and therefore cannot be easily used for automated decision support, research and outcome analysis. Then patient's file will be sent to the dispensary department for giving medicine. Pharmacists always check the quantity of remains drug inventory after certain period.

2.3.2 Medical Centre System (Computerized System)

With computer technologies, many systems are being computerized. The use of computers in the industries, businesses or personal uses reduce the overall responsibilities of the human. Computer technologies are also able to assist in the clinic operations. All of the medical records are transfer into database.

When patient coming for treatment, they only need to show their identification card, and clinical staff will try to retrieve the patient's profiles from the clinical database based on patient's identification card number. All the patient profile and data will directly save into the clinical database in forms of electronically transactions.

Once the patient' profiles have been displayed, queuing number will be automatically generated and patient will put into a waiting list. In the other side, doctor can easily retrieve the patient's medical history record while giving treatment. Hence, this will enable doctor to do analysis easily and accurately.

Besides, all the treatment, diagnostic and prescription of medicine to the patient can be recorded into clinical database easily and quickly. Prescription is forwarded to the clinical staff. Base on the information displayed on screen, it gives the appropriate medicine to the patient. This will indeed reduce errors, potential risks and patients' waiting time.

2.3.3 Case study 1: Klinik Kesihatan UM

Although nowadays, the clinic management have been used computerized system, but in University Malaya, the clinic for students still using manual procedure.

Firstly, students have to pre-register by filling the short form. After that, one staff will find student's record in the document room. At the same time, the student's went to the treatment room waiting for his/her name being called. This time, the staff ask student about the purpose for coming and queue number been given. Then the staff gives the student's profile to doctor. And again, the student will wait for his/her turn.

When it's him/her turn, the doctor doing his/her routine by checking the student. After that, the doctor gives student's prescription to the student to handing it to the pharmacy section. Then, the pharmacists prepare the medicine. Again, the student will wait till the pharmacist calling his/her name to get the medicine.

The whole process took about several hours if many patient/student coming for treatment. It waste a lot of student's time as they got their lectures, tutorials, assignment and many more. That is why, many students did not like to come to this clinic even it is free. They rather come to the private clinic that much faster and effective although it is quite expensive.

Strengths

- The setting up expenses for a manual medical center system is low
- No technical skills needed when it is setup.
- No need Internet accesses, no need IT knowledge, and ease of using for old generation workers.

Weakness

- The management system is not efficient and systematic. As the system is time consuming, massy, mistakes or lost may exist in the system, since humans do have errors.
- The staffs are considered very high to maintain the whole system since every piece of work is deal manually.
- The backup system is weak enough in the thread of fires.

2.3.4 Case study 2: Azhar's Clinic at Pantai Dalam

The Azhar's clinic is using partial manual and computer system. For the new registration, the patient's have to fill the form that been given. And if the patient has been treated before, he/she needs to show their identification card. And the clinical staff will try to retrieve the patient's profiles from the clinical database based on patient's identification card number. The queue number was given and the patient has to wait until his/her name being called. In the other side, the doctor retrieve all the patient's medical history record while giving treatment.

After that, the doctor gives the staff the patient's prescription to prepare the medicine. The patient will wait till the staff calling his/her name to get the medicine. The payment also been made at that time.

Strengths

- Data redundancy is under control, since all data is stored in database
- All information can be backup more systematically with tape drive or autoloader.
- The management system becomes more efficient and systematic

Weakness

- The setup expenses are very costly, including IT professional, hardware and software needed.
- There is a risk failure, since the database system may be new to some staffs.
- Technical knowledge is needed to set up the whole system.

2.3.5 Case study 3: Max-Gold Medical Clinic Software

Max-Gold Medical Clinic Software is designed to facilitate end-to-end administrative aspects of clinic management. Now healthcare providers can concentrate more on patients and enhancement of quality of the services, by letting the system takes the responsibility of most manually demanding tasks.

When patient coming for treatment, they only need to show their identification card, and clinical staff will try to retrieve the patient's profiles from the clinical database based on patient's identification card number.

Once the patient's profiles have been displayed, queuing number will be automatically generated and patient will put into a waiting list. In the other side, doctor can easily retrieve the patient's medical history record while giving treatment. Hence, this will enable doctor to do analysis easily and accurately.

Max-Gold Medical Clinic Software is a revolutionary point of care, clinical knowledge management that used by interns and residents. It provides rapid access to current, evidence-based diagnostic, management and treatment recommendations specific to patients' populations and the disease state.

Strengths

- Patient Information Management enables complete and accurate retention of all patient's details and records, total control and easy access of the information.
- Appointments are effectively scheduled with the ability to increase patient convenience through a suitable choice of pre-appointment reminder. At any point of time, the doctor can rearrange their schedule; the appointment management will be automatically updated.

- Doctors will use the Consultation Management module to enter his case notes and consultation fee. The entire fee he entered will automatically be added to bill. Nurses will use the Billing Management module to enter the name of the drug(s) to be issued to patient, and other fee of the bill.
- Drugs and Item Management enable the clinic to have complete control over the inventory of drugs and other items held in storage. It also provides a close view of the expiry date of these items.

Weakness

- No notification or email will be sent to patient automatically when patient had broken the appointment or changing schedule.
- Smart and effect queue management system does not exist in this software which helps to streamline the operations of the clinic.

2.4 SOFTWARE ARCHITECTURE

2.4.1 Host-Terminal Architecture

Host-Terminal Architecture describes a type of network in which a central computer (the “host”) is connected to a number of workstations (“terminal”), and in which the host handles all processing. Terminals are used to input data into the host and to review reports, but the terminals are “dumb” in the sense that the workstations do not participate in processing work. Host-terminal architecture is suited to both large and small networks; the distinguishing characteristic is that a single, central computer handles the processing works.

2.4.2 Client-Server Architecture

Client-Server Architecture describes a network architecture in which each computer handles part of the processing work, but is designated as either a “client” or a “server” with respect to each process. “Servers” are shared, central computers that are dedicated to managing specific tasks for a number of clients. For example, “file servers” are dedicated to storing files, “print servers” are dedicated to printing, and “network servers” are dedicated to routing network traffic. “Clients” are workstations on which users run programs and applications. Clients rely on servers for resources, such as files, devices, and even processing power, but process independently of the servers. Client-server architecture can be used in any sized network; the distinguishing characteristic of client-server architecture is that all computers on the network participate in processing, but that certain computers are dedicated to specific services or tasks and do no other work.

2.4.3 Tier Architecture

Tier Architecture is how different processing work is physically and logically distributed across a computer system and network. This is where developers cross with networks.

2.4.3.1 One-Tier Architecture

In one-tier architecture, the entire application exists on single node. All the applications are installed on individual machines. The types of system are standalone executables and mainframe application types of system.

Advantages

- Simple to build and very natural

Disadvantages

- Facilitates very little reuse and maintenance can be expensive
- Single point of failure
- Scaling system requires buying bigger, costly hardware

2.4.3.2 Two-Tier Architecture

Two-tier architecture also refers to client/server architecture. The user interface runs on the client and the database is stored on the server. The actual application logic can run on either the client or the server. There are two types of client like Fat Client (Server manages data only) and Thin Client (Server manages Data and business logic).

Advantages

- Modifications on server propagated to clients
- Can Distribute processing load

- Better scalability by adding server nodes and clients

Disadvantages

- Client nodes require more computing power
- Development and maintenance more complex

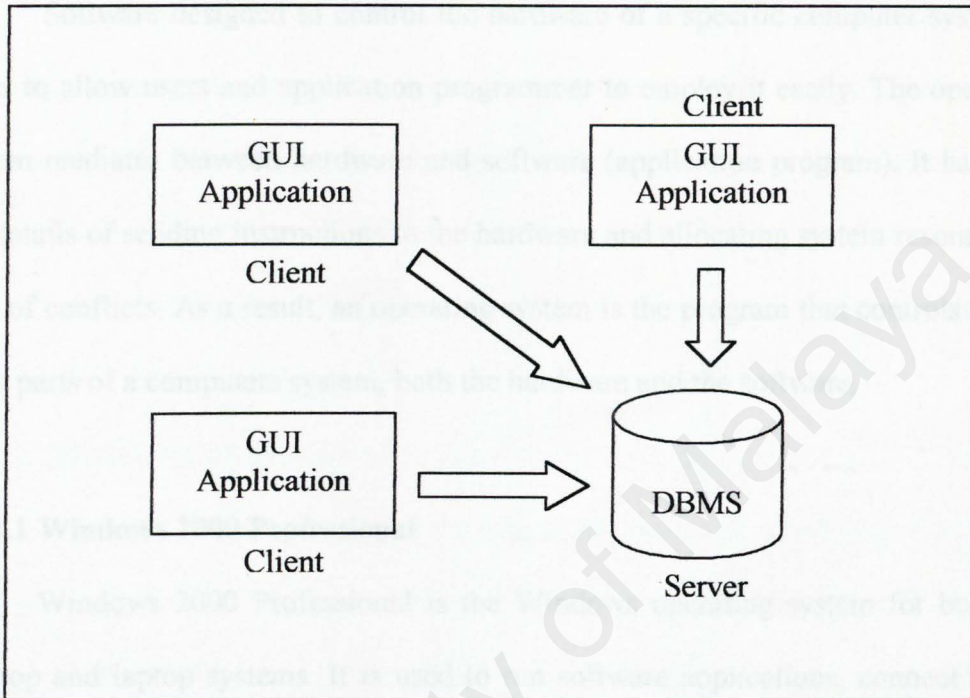


Figure 2.1 The Two-tier Architecture

2.4.3.4 N-Tier Architecture

Allows an unlimited numbers of programs to run simultaneously, send information to one another, use different protocols to communicate, and interact concurrently. This allows for much more powerful applications, providing many different services to many different clients. It supports multiple applications more easily. But, it is required a highest implementation cost and more complex, thus, the more potential for bugs and harder to balance loads.

2.5 DEVELOPMENT TOOLS AND TECHNOLOGIES

CONSIDERATIONS

2.5.1 Operating System

Software designed to control the hardware of a specific computer system in order to allow users and application programmer to employ it easily. The operating system mediates between hardware and software (application program). It handles the details of sending instructions to the hardware and allocating system resources in case of conflicts. As a result, an operating system is the program that controls all the other parts of a computers system, both the hardware and the software.

2.5.1.1 Windows 2000 Professional

Windows 2000 Professional is the Windows operating system for business desktop and laptop systems. It is used to run software applications, connect to the Internet and intranet sites, and access files, printers, and network resources. Built on Windows NT technology and the easy-to-use, familiar Windows 98 user interface, Windows 2000 Professional gives business users increased flexibility. The integrated Web capabilities let user connect to the Internet from anywhere, at any time, giving user company access to host of flexible, cost-effective communications options. In addition, broad peripheral and mobile computer support make Windows 2000 Professional an ideal operating system for a workforce that increasingly relies on notebook computers. The advantages of Windows 2000 Professional are:

- Work how and where user want with new peripheral support and features that extend notebook capabilities.
- Rely on user PC to be up and running with enterprise level quality.

- Work the way user did with Windows 98, only much faster. Combine the ease of Windows 98 with the manageability, reliability, and security of Windows NT, at speeds 30 percent faster than Windows 98 on PCs with 64 MB RAM or more.
- Communicate, share information, and use the Internet quickly and easily. With integrated support for Internet-enable applications, business software developers incorporate the new ways to create and share information made possible by the Internet.

2.5.1.2 Windows 2000 Server

Windows 2000 Server is the multipurpose network operating system for businesses of all sizes. Windows 2000 has the right stuff, both as a workstation and as a server. It is stable, easy to install, and packs in enough new features to make it a must-have upgrade from Windows NT 4.0. Windows 2000's dizzying array of new server tools will keep larger businesses busy deciding which ones to deploy, and the server is sophisticated enough to make it worth the extra effort. The advantages of Windows 2000 Server are:

- Provides services that let user build and deploy servers more quickly, reduce the time it takes to create new Web sites, create virtual directories, manage security settings, and manage security certificates.
- Allows user to configure network more easily. It provides support for Plug and Play network adapters, significantly reducing device configuration time and it provides automated replication and local

caching of DNS and DHCP information so user network is robust and responsive.

- Group Policy, based on the Active Directory helps admin control user access to desktop settings and applications by group rather than by individual user and computer. Windows Management Instrumentation (WMI) provides unified access and event services.
- Windows Script Host (WSH) allows you to automate and integrate common tasks using a variety of scripting environments. Microsoft Management Console (MMC) gives user a common user interface presentation tool where user can integrate all the necessary Windows-based and Web-based administration components needed to fulfill a specific task.

2.5.1.3 Unix

The UNIX operating system was designed to let a number of programmers access the computer at the same time and share its resources. The operating system coordinates the use of the computer's resources, allowing one person, for example, to run a spell check program while another creates a document, lets another edit a document while another creates graphics, and lets another user format a document -- all at the same time, with each user oblivious to the activities of the others.

The operating system controls all of the commands from all of the keyboards and all of the data being generated, and permits each user to believe he or she is the only person working on the computer. This real-time sharing of resources makes UNIX one of the most powerful operating systems ever.

Although programmers for programmers developed UNIX, it provides an environment so powerful and flexible that it is found in businesses, sciences, academia, and industry. Many telecommunications switches and transmission systems also are controlled by administration and maintenance systems based on UNIX.

While initially designed for medium-sized minicomputers, the operating system was soon moved to larger, more powerful mainframe computers. As personal computers grew in popularity, versions of UNIX found their way into these boxes, and a number of companies produce UNIX-based machines for the scientific and programming communities.

2.5.1.4 Linux

Linux is an operating system that was initially created as a hobby by a young student, Linus Torvalds, at the University of Helsinki in Finland. Linus had an interest in Minix, a small UNIX system, and decided to develop a system that exceeded the Minix standards. He began his work in 1991 when he released version 0.02 and worked steadily until 1994 when version 1.0 of the Linux Kernel was released. The current full-featured version is 2.4 (released January 2001) and development continues. Linux is developed under the GNU General Public License and its source code is freely available to everyone.

Linux may be used for a wide variety of purposes including networking, software development, and as an end-user platform. Linux is often considered an excellent, low-cost alternative to other more expensive operating systems.

2.5.1 Due to the very nature of Linux's functionality and availability, it has become quite popular worldwide and a vast number of software programmers have taken Linux's source code and adapted it to meet their individual needs.

2.5.1.5 Windows NT

Microsoft's popular operating system for all types of networks ranging from workgroups to enterprise-level installations. Windows NT is designed to run both 16 and 32 bit applications under an environment subsystem that also enables OS/2 and POSIX applications to be run on the same computer.

DOS and other 16 bit windows applications are made to run inside any number of virtual DOS machines to prevent them from affecting other 32 bit applications. Users can choose to run 16 bit applications in a separate memory space by including the separate option in the run command or by signifying the same in the graphical run utility. A user can also make a shortcut to the program and edit the program information file (PIF) of the shortcut specifying that the program should be run in separate memory space.

The Windows NT architecture also features a Hardware Abstraction Layer, which enables the operating system to run in both the Intel and Alpha hardware platforms. It uses the Network Device Interface Specifications (NDIS) to enable multiple protocols to be bound to different network adapters and media types.

2.5.2 Programming Languages

2.5.2.1 Microsoft.NET

Microsoft .NET is software that connects information, people, systems, and devices. It spans clients, servers, and developer tools, and consists of:

- The .NET Framework programming model that enables developers to build Web-based applications, smart client applications, and XML Web services applications which expose their functionality programmatically over a network using standard protocols such as SOAP and HTTP.
- Developer tools, such as Microsoft Visual Studio® .NET, which provide a rapid application integrated development environment for programming with the .NET Framework.
- A set of servers, including Microsoft Windows® 2000, Microsoft SQL Server™, and Microsoft BizTalk® Server, that integrates, runs, operates, and manages XML Web services and applications.
- Client software, such as Windows XP, Windows CE, and Microsoft Office XP, that helps developers deliver a deep and compelling user experience across a family of devices and existing products.

2.5.2.2 Visual Basic.NET

Microsoft Visual Basic.NET is the newest, most productive version of the Visual Basic tool set that enables developers to address today's pressing application development issues effectively and efficiently. Visual Basic .NET enables you to create rich applications for Microsoft Windows in less time, incorporate data access from a wider range of database scenarios, create components with minimal code, and build Web-based applications using the skills you already have. Visual Basic.NET provides:

- Build robust Windows-based applications
- Resolve deployment and versioning issues seamlessly
- Create web applications with a zero learning curve
- Provides flexible, simple data access. Visual Basic .NET provides support for both the new Microsoft ADO.NET for flexible, highly scalable data access and ActiveX Data Objects (ADO) data binding for connection oriented data access.
- Get on the fast track to building tomorrow's applications today. Visual Basic .NET opens the door for Visual Basic developers and offers a smooth transition to building next-generation applications today.
- Provides a smooth upgrade process for all Visual Basic developers, regardless of their existing code base, skills, or migration plan.

2.5.2.3 Visual Basic 6

Visual Basic 6 (VB 6) is revolutionizing development with multimedia-intensive, object oriented, compiled code for conventional and Internet/Intranet-based applications. In its six versions, Visual Basic has evolved from the simplest programming language for Microsoft Windows to an exceedingly complex development environment, capable of delivering virtually anything from tiny utilities to huge n-tier client/server applications.

VB 6 includes many new features, especially in the database and Internet areas such as ADO, DHTML applications, and WebClasses. A closer look at ADO technology Database programming is important for most Visual Basic developers, and ADO has much to offer in this field. Base a language on one of the world's most widely known languages, Basic. Endow the language with the ability to conveniently build applications for Microsoft Windows® - the world's most widely used platform.

Provide the kinds of heavy duty, high performance capabilities needed for enterprise systems development. Make the language appropriate for implementing Internet-based and World-Wide-Web-based applications, and build in the features people really need such as graphics, graphical user interface components, error handling, multimedia, file processing, database processing, Internet-based client-server networking, World Wide Web documents enhancement with Visual basic Script (VB Script), and prepackaged components

Make the language extensible so that independent software vendors (ISVs) can provide componentry for a vast array of application arenas. These features are precisely what businesses and organizations needs to meet today's information processing requirements.

2.5.3 Database

A file used to store records of information, with each record containing multiple data fields. The most popular type of database is the relational database, in which the records are stored in tables that are related to each other using primary and foreign keys. A primary key is the field in each record that uniquely defines the record. A foreign key is a field in another table that matches the first table's primary key, creating a relationship between the two. An application for creating and managing relational databases is called a relational database management system (RDBMS).

2.5.3.1 MySQL

MySQL is an open source database, which means that it is free. MYSQL is a relational database management system. MySQL is a small, compact, easy to use database server, ideal for small and medium sized applications. By the way, it is available on a variety of UNIX platforms, Linux, Windows NT, Windows 95/98 and Windows 2000.

2.5.3.2 Microsoft SQL Server

SQL Server 2000 is a powerful tool for turning information into opportunity. Industry-leading support for XML, enhanced tools for system management and tuning, and exceptional scalability and reliability make SQL Server 2000 the best choice for the agile enterprise.

2.5.3.3 Microsoft Access 2000

Microsoft Access is the most widespread DBMS (Database Management System) for the Microsoft Windows Environment. Access can be used as an independent database management on a personal computer.

It has a low cost and the application to which it is targeted do not typically require a sophisticated implementation of these services. The system interface exploits the potential of the graphical environment and offers a user-friendly interface, both for user and for the database designer.

Access can be seen as a tool that allows user to avoid from writing SQL code, as it require schemas and simple queries using a graphical representation that is easy to understand. These inputs are translated into suitable SQL commands in a transparent manner.

2.5.4.2 Open Database Connectivity (ODBC)

Open Database Connectivity (ODBC) is a widely accepted application-programming interface (API) for database access. It is based on the Call-Level Interface (CLI) specifications from X/Open and ISO/IEC for database APIs and uses Structured Query Language (SQL) as its database access language.

2.5.4 Data Access Technology

ECS will require data access technology to enable communication and access to its various database. A few of the Microsoft Data access strategies and technology is review and considered.

2.5.4.1 ActiveX Data Object (ADO)

ADO is Microsoft's strategic, high-level interface to all kinds of data. ADO provides consistent, high-performance access to data. ADO is the single data interface the developers need to know for 1- to n-tier client/server and Web-based data-driven solution development.

The ADO programming model represents the best of the existing Microsoft data access programming models. ADO is designed to eventually replace Data Access Objects (DAO) and Remote Data Objects (RDO). Unlike RDO and DAO, which are designed only for accessing relational databases, ADO is more general and can be used to access all sorts of different types of data, including web pages, spreadsheets, and other types of documents.

2.5.4.2 Open Database Connectivity (ODBC)

Open Database Connectivity (ODBC) is a widely accepted application-programming interface (API) for database access. It is based on the Call-Level Interface (CLI) specifications from X/Open and ISO/IEC for database APIs and uses Structured Query Language (SQL) as its database access language.

2.5 ODBC is designed for maximum interoperability – that is, the ability of a single application to access different database management systems (DBMS) with the same source code. Database applications call functions in the ODBC interface, which are implemented in database-specific modules called drivers. The use of drivers isolates applications from database-specific calls in the same way that printer drivers isolate word processing programs from printer-specific commands.

2.5.4.3 OLE DB

OLE DB is Microsoft's strategic low-level interface to data across the organization. OLE DB is an open specification designed to build on the success of ODBC by providing an open standard for accessing all kinds of data.

OLE DB is a set of interfaces for data access and Microsoft's component database architecture that provides universal data integration over an enterprise's network (from mainframe to desktop), regardless of the data type. OLE DB is the fundamental Component Object Model (COM) building block for storing and retrieving records and unifies Microsoft's strategy for database connectivity. It will be used throughout Microsoft's line of applications and data stores.

2.6 SUMMARY

The literature review is done prior to the development of the proposed project. Then it is followed by software architecture and technologies related with the project such as Host Terminal, Client-server architecture and Tier architecture

Further explanation on the existing systems, is the manual and computerized system. Analysis of the current available system gives us a better idea on developing E-Clinic System and improving previous system.

The reviews of development tools and technologies consideration give the benefits to us, as we have to choose the best for better results of ECS implementation.

CHAPTER 3: METHODOLOGY AND SYSTEM ANALYSIS

3.1 INTRODUCTION

Thoroughly planning is a must in project for effective development in order to achieve the project goals. Methodology is an organized, documented set of procedures and guidelines for one or more phases of the software life cycle, such as analysis or design. It is usually presented as a series of steps, with techniques and notation associated with each step.

Process model is very important during the software development process or software life cycle. It can form a common understanding of the activities, resources and constraints involved in software development. When a process model is created, it helps to find inconsistencies, redundancies and omissions in the process. As the problems are noted and corrected, the process becomes more effective and focused on building the final system.

In the development process, several different models have been used, such as Waterfall model, V model, Prototyping model and many more. These models provide guidance on the order in which a project should carry out its major tasks.

3.2 METHODOLOGY

In order to accomplish any given set of tasks effectively one must have a work plan or procedure. Without such a procedure or work plan, activities are performed in a haphazard manner and with little if any coordination. The results are that the various intermediate products rarely fit together into a cohesive whole, and worse yet the finished product rarely meets the initial specifications. In some cases because of a lack of a work plan there are no initial specifications.

The overall work plans for systems development is called systems development life cycles, and the detail plans are called methodologies. Methodology is defined as a system of principles, practices, and procedures applied to a specific branch of knowledge. On the other hand, method is a means or manner of procedures, a regular and systematic way of accomplishing something.

The primary function of a development methodology is to provide discipline to the entire development process. A good development methodology establishes organization-wide standards for requirements gathering, design, programming, and testing. To produce quality software, organizations must select the appropriate methodology and then enforce its use.

3.2.1 Rapid Prototyping Model

The methodology for ECS is Rapid Prototyping Model. Rapid prototyping is an instructional design model that creates a prototype before initiating a full-scale production of the material. A prototype is a miniature version of an end product. It is an executable version of an actual product, incorporating key elements of the final version but is incomplete in terms of functionality, and robustness. Rapid Prototyping Model is a design methodology that allows the designer to build a prototype, test and modify it, and through an iterative process complete the final product. The model can be presented as the following events:

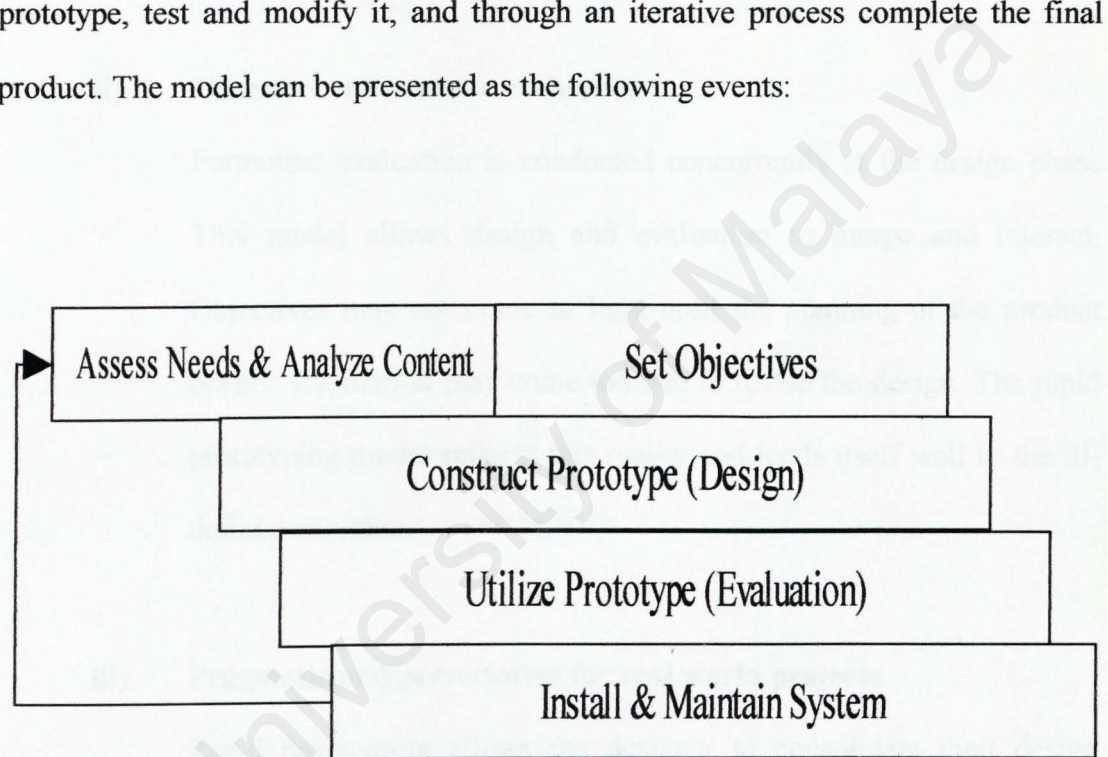


Figure 3.1: The Rapid Prototyping Model

Below is the characteristic of Rapid Prototyping Model:

i) Integral process from needs assessment to evaluation

In the rapid prototyping model, is a sequence of events but these events can occur at the same time. In other words, needs assessment and objective construction can be conducted at the time of creating a prototype. As well, design, evaluation and implementation can be carried out simultaneously.

ii) Concurrent formative evaluation

Formative evaluation is conducted concurrently at the design phase. This model allows design and evaluation to merge and interact. Objectives may not come to light until the planning of the product occurs. Evaluation may come too late to revise the design. The rapid prototyping model reflects this reality and lends itself well to the ill-defined situation.

iii) Pragmatic and prescriptive for real world projects

Rapid prototyping allows the designer to consolidate their design ideas quickly, communicate them with the users directly, and test them to determine the effectiveness of the instruction. Applying rapid prototyping aims at resolving each unique instructional situation.

iv) Efficient and economic production

As this model proposes an integral process towards instructional design, this speeds up the development cycle. This model promotes a

more efficient way of solving an instructional problem. In addition, testing and modifying a prototype can eliminate the risk and cost due to the abandoning of a final product.

v) User-centered methodology

The most praised characteristic of this model is the inclusion of end-users. In other words, by having the users evaluate the prototyped product, the designer have a chance to test his or her design ideas. As a result, building a prototype and conducting evaluation concurrently is a preferred way in the real world.

Rapid prototyping Model is chosen for ECS because:

- To test out a user interface.
- To test the database structure and flow of information in a training system.
- To test the effectiveness and appeal of a particular instructional strategy.
- To develop a model case or practice exercise that can serve as a template for others.
- To get user feedback and reactions to two competing approaches.

3.3 FACTS FINDING TECHNIQUES

In developing ECS project, information and fact-finding are done much kind of techniques such as interview, sampling, research and many more. The objectives of fact finding is to develop a system that is ready to use within requirements.

3.3.1 Discussion And Interview

Most of the information that gathered in this project is based on interviews or conversation with people concern. The interviews were not strictly structured because they were more like conversations. A few conversations have been carried out in order to understand the need of clinical staff and the operational of local clinic. From those sessions, the clearer view of what ECS should provide to produce better healthcare management among clinics in Malaysia. Doctors and clinical staffs have given a lot of precious advices and suggestion to help developing a better healthcare management system.

3.3.2 Observation

Observation has been done almost every time. For this technique, observation was mostly focused on clinics nearby residential such as Petaling Jaya, Lembah Pantai and Pantai Dalam. From the observation, most clinics are still practicing the method of processing paperwork in a costly manner. Due to the lack of automated data capture and keeping data electronically, the existing system involves a lot of inefficient and slow responds time activities. Slow respond times ultimately take their toll on the business of healthcare by severe diminishing productivity.

3.3.3 Sampling (Software Packages)

Sampling techniques carried out using Internet surfing. Internet is a major search engine in finding information required in this project. Some of the web sites that have been visited are:

- <http://www.aiznet.com>
- <http://www.shasoftindia.com>
- <http://www.centrahealth.com>
- <http://www.thatwebsolution.com>
- <http://eclevelandclinic.org>
- <http://ehealth.telus.com>
- <http://www.nec.co.nz>
- <http://www.webhealthcentre.com>

These sites provide some information about the clinical management system and they are enable users to download the trial version of sample clinical management system software. It helps us to understand better the flow processes.

3.3.4 Research (Written Materials)

Research has been done by reading some articles, magazines and other textual mediums in order to gather information related to ECS project and as foundation for the discussion in this project. Reading materials provide a lot of information about the internal structure of clinic and terms that are usually use by doctors.

3.3.5 Previous Thesis

Doing some reference to some previous theses in order to gather more information about the previous clinical management system. The strengths and weakness of previous theses had helped me to better define the structure and functional of each modules in the ECS project. This research also helps to define all the limitation and problems, which will arise while doing the process of development.

But not all the previous theses we could use as reference. It is because of the outdated years. Lately, the evolving of new technology, these theses are not up to the standard. We, as a student must alert about the changing of new methodology and technologies. We have to choose which are still useful and meaningful for our system.

3.4 SYSTEM ANALYSIS

System analysis is important to a newly develop system, where the study of current system is done and including the definition of the user requirements for the new system based on the result from facts finding.

User requirements are the features of system or a description of something that the system is capable of doing in order to fulfill the system's purpose. It describes not only the flow of the information to and from the system but also the constraints on the system's performance.

Requirement elicitation is an especially critical part of the process. It explains the requirement definition of the system. Requirement definition is a complete listing of everything the customer expects the proposed system to do. It represents an understanding between customer and developer of what customer need or wants, as it usually written jointly with developer. Requirement specification restates the technical terms appropriate for the development of a system design. It is written in the requirement analysis.

3.4.1 REQUIREMENT ANALYSIS

Each proposed model of the software development process includes activities aimed at capturing requirements and understanding what the to expect from the system. Thus, our understanding of system intent and function starts with an examination of requirements. When the requirements are defined, we learn how to document them and review them for correctness and completeness in a requirements review. A requirement is a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose.

In general, requirements are partitioned into functional requirement and non-functional requirement. Functional requirements are associated with specific functions, task or behaviors the system must support, while non-functional requirements are constrains on various attributes of these functions or tasks.

3.4.1.1 Functional Requirement

A functional requirement describes an interaction between the system and its environment. For example, to determine functional requirements, we decide what states are acceptable ones for the system to be in. Further, functional requirements describe how the system should behave in given certain stimuli.

These requirements are frequently identified in the terms of physical environment, interfaces, users and human factors; functionality, data, resources and security that are needed to satisfy the system's objectives.

Functional requirement for ECS consists of five main parts

- System Management (Login)
- Patient Information Management (Registration)
- Consultation Management
- Reports (Printing MC Letter or receipts)

Login

This module grants the authorized accesses to administrator by verify their username and password. The authentication is necessary to prevent an unauthorized user to steal, modify or delete the important files or data, which it should be private and confidential.

Also this module let the administrator to set up new user account to gain access to the system, views or edit the existing information of users and change password for security reason.

Registration

Registration handles the new registration of a patient or regular patient seeking treatment. It also allows for complete and accurate retention of all patient's details and record, total control and easy access of this information, which will automatically linked to key clinic processes such as allergy alerts during drug dispensation process.

Patient Details is the sub-section of this module, which is primarily concerned with the storage and retrieval of the patient information, such as name, address and the diagnosis. The key functions allow the user to Add, Modify and Delete either individual elements of the patient's record or the record itself.

Report

The main purpose for this module is to make a list of patient that had come to the clinic. It also enable user to view the patient's particulars, and a complete timeline of medical history. This is also designed to allow for retrieval the list of appointment and printing of MC letter and report.

Consultation

The consultation module is about referring the patient to the specialist, as the clinic's doctor could not help him/her. Usually, the specialists are the people that the clinic's doctors knew or close friends.

This is a module to add, delete or edit the specialist information such as address or telephone number by administrator.

3.4.1.2 Nonfunctional requirements

Besides the constraining requirements, there are other nonfunctional requirements, which must be met. The following section supplements the requirement analysis.

Performance requirement

The server should response in a reasonable time when there are multiple accesses. No compromise to security should be made to any force of retrieving the private and confidential data in the database.

Process requirement

Users are not allowed to do any modification unless he or she has the privilege to do it so. Every user has been given limitation to access the system as a user or administrator.

Attributes

The availability are able to all users unless if they are trying to retrieve the same record. We use the first in to be the first one to get the record. Database maintainability must be scheduled and optimize from time to time. Scheduled task involver organizing and removing or filtering outdated information. The system should be designed as an independent component that is customizable.

3.5 SUMMARY OF CHAPTER 3 SYSTEM DESIGN

As conclusion, the software development process involves activities resources and products. A process model is useful for guiding and its detailed tells us how to design and build system. Process models include organizational, functional, behavioral, and other perspectives, so we can focus on particular aspects of the development process to enhance our understanding or guide our actions. By using a good process, the quality of the products of development can be guaranteed.

The requirement analysis is a software engineering task that bridges the gap between system level software allocation and software design. There are both functional and nonfunctional requirements. The functional requirements explain what the system will do, and the nonfunctional ones constrain the behavior in terms of safety, reliability, schedule and more.

CHAPTER 4: SYSTEM DESIGN

4.1 INTRODUCTION

System design or conceptual design describes the system in language that the customer can understand, rather than in computer jargon and technical terms. [Shari Lawrence Pfleeger, 2001]

System design is a process to convert the conceptual ideas from requirement specification in system analysis into more technical specification. A design specification displays both the physical and logical design of the system. Physical design is the ECS architecture design. For the logical design, specifications are on the system functionality design and prototyped user interface design. The design phase aims to plan a structure approach to solve the problem specified in the system analysis.

Design is really a two-part iterative process. First, we produce a conceptual design or system design that tell the user exactly what the system will do. Then, we translate the conceptual into much more detailed document, the technical design to understand the actual hardware and software needed.

The process is iterative because, in actuality, the designers move back and forth among activities involving understanding the requirements, proposing possible solutions, testing aspects of solution for feasibility, presenting possibilities and documenting the design for the programmers. Sometimes, the design is described in one document, but often there are two, as illustrated in figure 4.1. Thus, the conceptual design concentrates on the system's function, and the technical design describes the form the system will take.

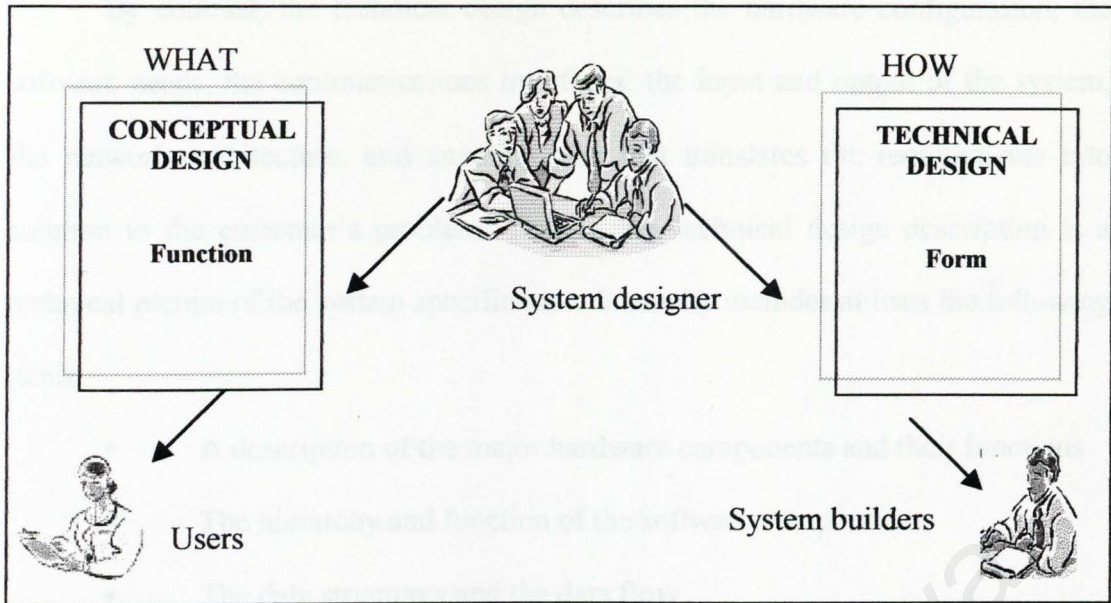


Figure 4.1: Conceptual and Technical designs

System is defined by its boundary, entities, attributes and relationships. The conceptual design describes each of these system aspects, answering questions such as the following:

- Where will data come from?
- What will happen to the data in the system?
- What will the system look like to users?
- What choices will be offered to users?
- What is the timing of events?
- What will the reports and screens look like?

By contrast, the technical design describes the hardware configuration, the software needs, the communications interfaces, the input and output of the system, the network architecture, and anything else that translates the requirements into solution to the customer's problem. That is, the technical design description is a technical picture of the system specification. It usually includes at least the following items:

- A description of the major hardware components and their functions
- The hierarchy and function of the software components
- The data structures and the data flow

As a result, the system design includes the following issues:

- Program design
- Database design
- User Interface design
- Input or Output design

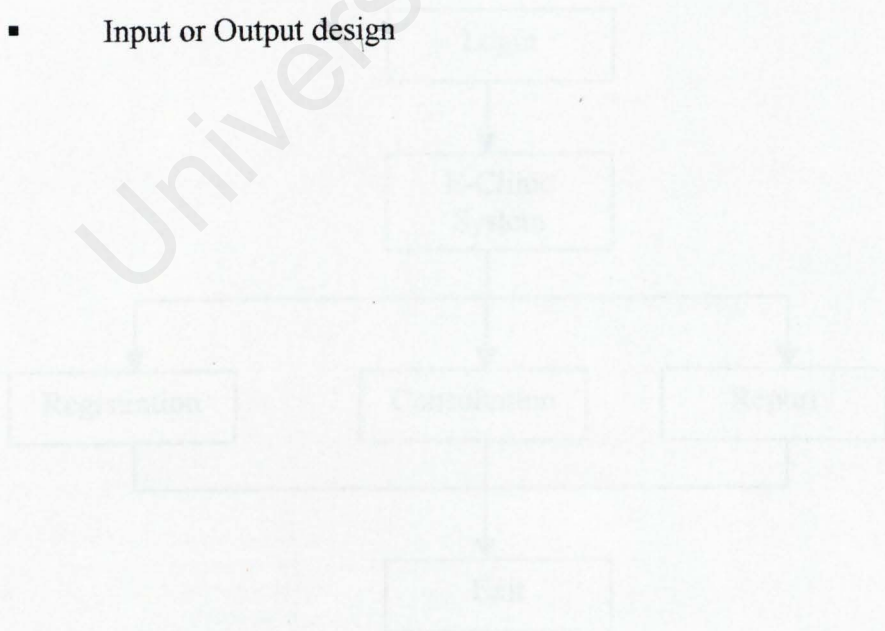


Figure 4.1: The EC's Architecture

4.2 PROGRAM DESIGN

Program design is the design from which coders will implement the system. The program design efforts begin with the objects and classes from the system design and modify to include more items:

- Nonfunctional requirements, such as performance and input or output constraints
- User interface requirements
- Data structures and management details

During program design, we must make more detailed decisions about the data. We must also specify the features of each object's interface. Once we have defined the interfaces, we can classify them by type, and build a hierarchy of interface types where some interface inherit properties from other interfaces. When a particular object is instantiated, the compiler allocates storage for the internal data needed by the object. It then associates operation with data.

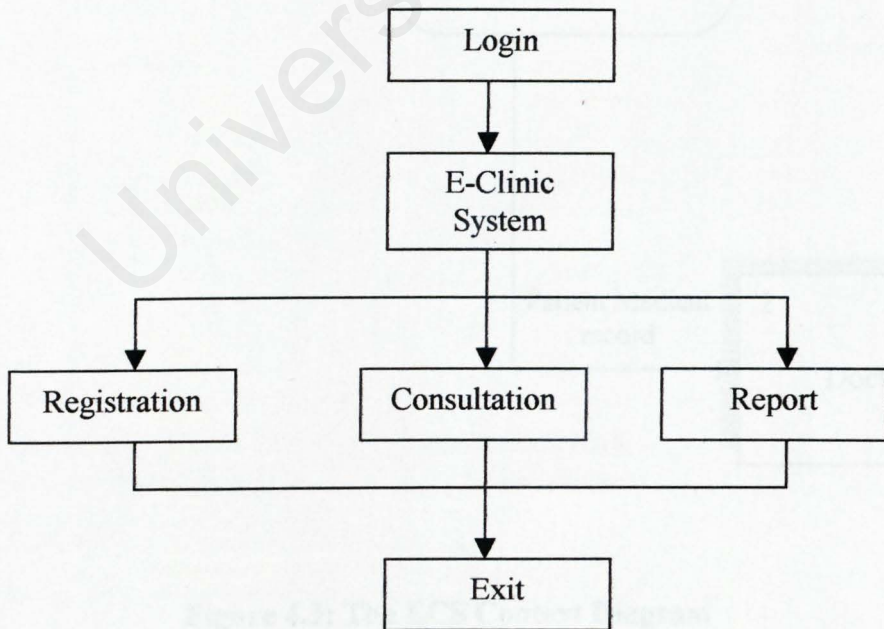


Figure 4.2: The ECS Architecture

Firstly, we must determine the E-Clinic System flow management. The context diagram shows us about the scope and limitation for the information system. It is the highest level of data flow diagram and was the first to draw when we like to prepare the data flow for the environment system.

As stated, we can see the context diagram of ECS in figure 4.3

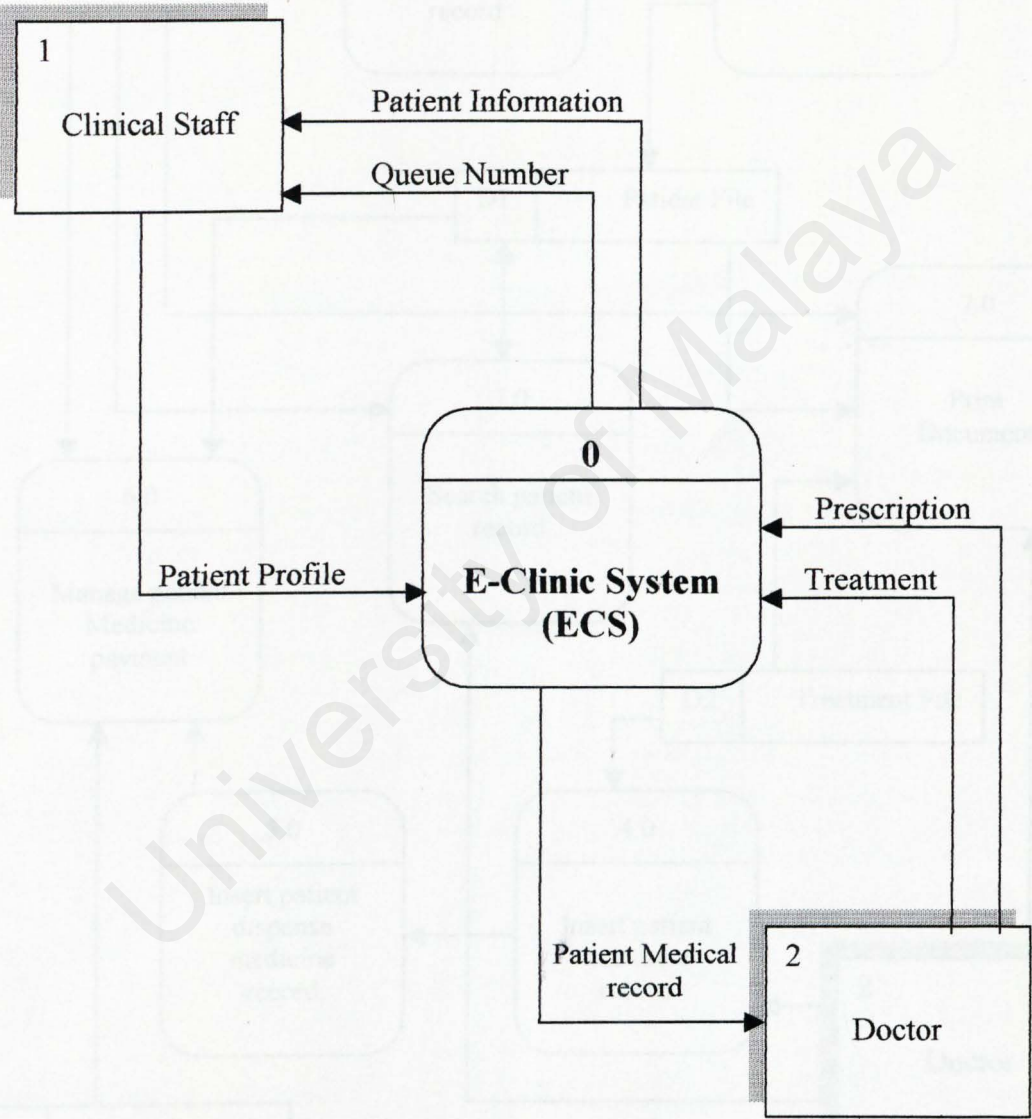


Figure 4.3: The ECS Context Diagram

While, Zero Diagram is second level of data flow diagram and it shows us the detail about the Context Diagram. We can see it through the diagram that stated in figure 4.4.

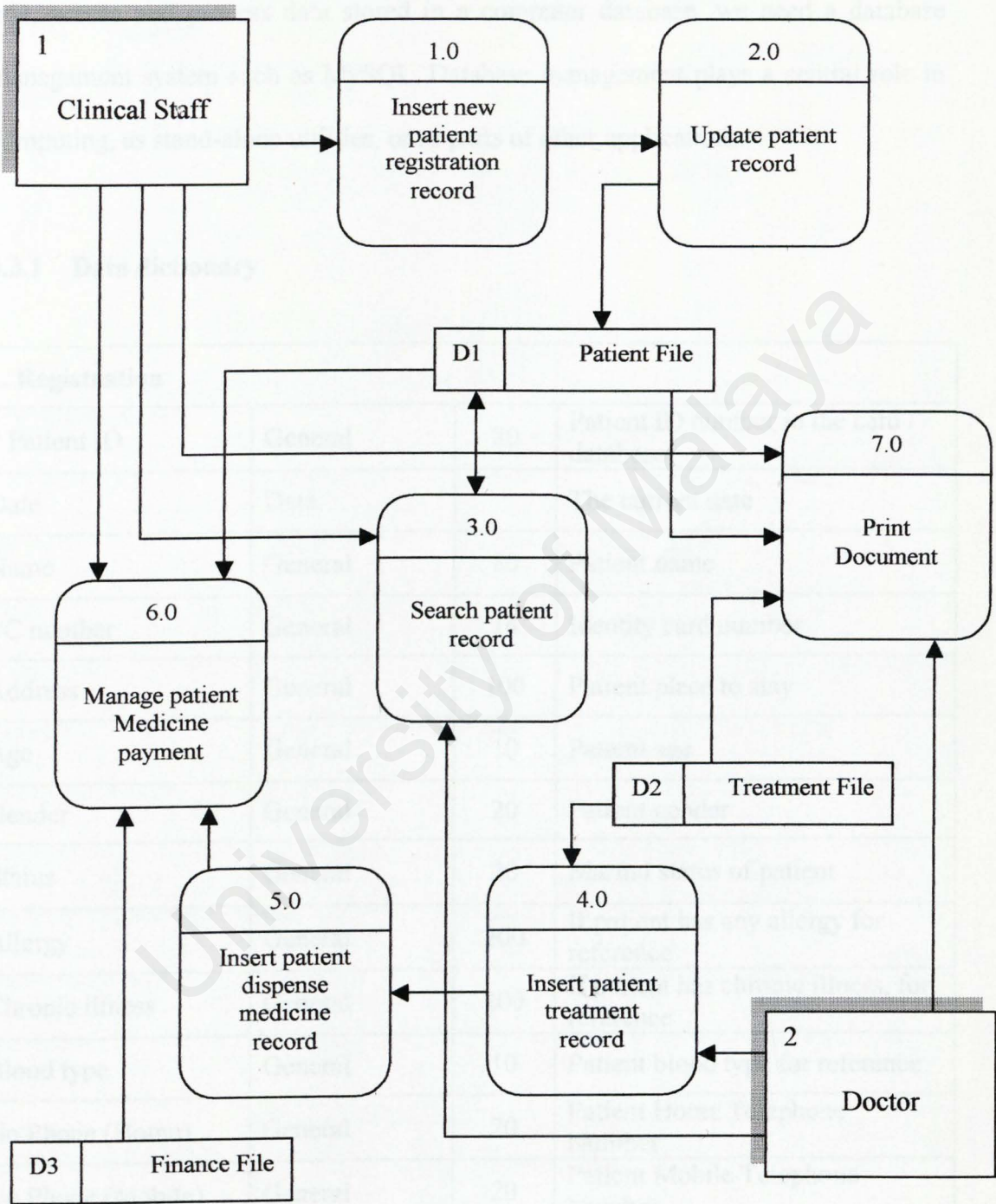


Figure 4.4: The ECS Zero Diagram

4.3 DATABASE DESIGN

A database is a collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access and process data stored in a computer database, we need a database management system such as MySQL. Database management plays a central role in computing, as stand-alone utilities, or as parts of other applications.

4.3.1 Data dictionary

1. Registration			
* Patient ID	General	20	Patient ID number in the card / database
Date	Date		The current date
Name	General	80	Patient name
I/C number	General	14	Identity card number
Address	General	100	Patient place to stay
Age	General	10	Patient age
Gender	General	20	Patient gender
Status	General	20	Marital status of patient
Allergy	General	200	If patient has any allergy for reference
Chronic illness	General	200	If patient has chronic illness, for reference
Blood type	General	10	Patient blood type for reference
No Phone (Home)	General	20	Patient Home Telephone Number
No Phone (Mobile)	General	20	Patient Mobile Telephone Number
No Phone (Office)	General	20	Patient Office Telephone Number

Relative Name	General	80	Patient close relative / friends/ if anything happens
No. Contact (Home)	General	20	Emergency contact person home telephone. Number
No Phone (Office)	General	20	Emergency contact person home telephone. Number
No Phone (Mobile)	General	20	Emergency contact person mobile telephone. Number
Address	General	100	Emergency contact person address

Table 4.1: Data Dictionary of Registration

2. Consultation			
* Patient ID	General	20	Patient ID number in the card / database
Name	General	80	Patient name
Allergy	General	200	If patient has any allergy for reference
Treatment	General	200	Type of treatment patient receive
Remarks	General	200	Additional notes patient
Specialist's name	General	80	Specialist name
Address	General	100	Specialist address
No Phone (Office)	General	20	Specialist office telephone. Number
No Phone (Fax)	General	20	Specialist fax number
Remarks	General	100	Additional notes for specialist

Table 4.2: Data Dictionary of Consultation

4.4 USER INTERFACE DESIGN

User interfaces can be tricky things to design, because different people have different styles of perceiving, understanding, and working. For example, one user may use a word processing package by pressing on function keys, whereas another relies mostly on the mouse. Similarly, users differ in the sequence in which they perform actions; in their preferences for commands, dials, and windows; and in the degree to which they use help screens and manuals. The issues involved in interface design are:

- **Metaphors:** the fundamental terms, images and concepts that can be recognized and learned.
- **A mental mode:** the organization and representation of data, functions, tasks and roles
- **The navigation rules for the model:** how to move among data, functions, activities and roles
- **Look:** the characteristics of the system's appearance that convey information to the user
- **Feel:** the interaction techniques that provide an appealing experience for the user

In order to design comfortable, effective interfaces, we must consider two key issues: culture and preference.

Cultural issues: to determine interface preferences must take into account both cultural differences and group dynamics for the population of likely users.

User preferences: some aspect of design depends on user preferences, either alone or as members of a group of workers.

These are the example of user interface design for the ECS.



Figure 4.5: User Interface ECS Splash Screen

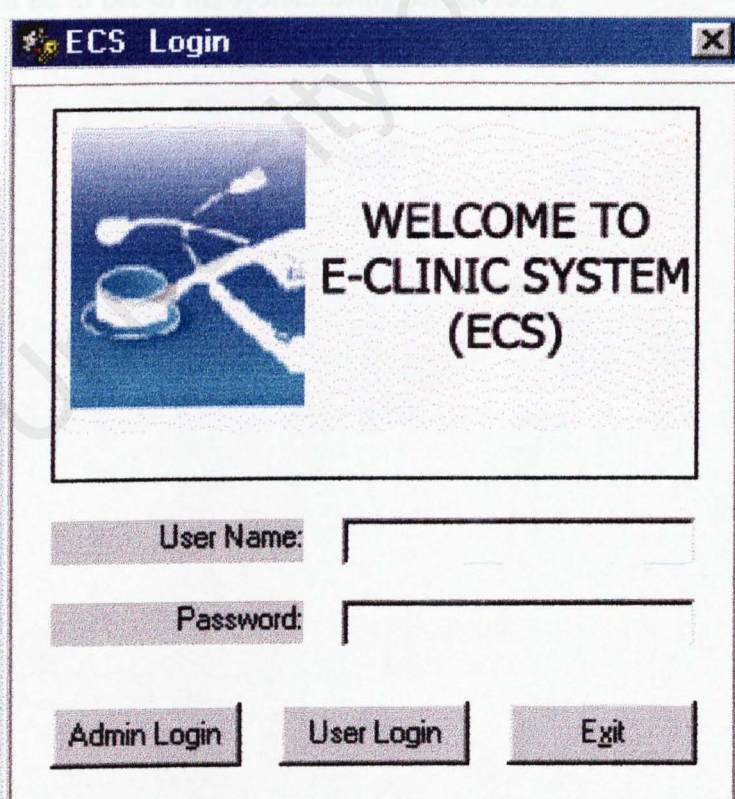


Figure 4.6: User Interface Login

4.5 INPUT OR OUTPUT DESIGN

4.5.1 Input design

The input design is how to specify the best method to input the data in the system. Output will not be able to view if no input. To get the best output, it depends on the best input. There are many input devices to consider for insert or input data. For example, keyboard, mouse, touch-screen, point-of-sale and many more.

The method to input data in the system usually use these two methods. The first, using a screen form and the second, using printed form. A screen form is an input data that view in the computer screen. It is design for the purpose to input data in the system directly using some devices. While, printed form is a conventional method form that already prints on a piece of paper. Then, the information in the printed form will be to put in the system using screen form.

Without considering its visual or in command shape, screen form is design to receive input data within the specification needed by user. This is the example of input screen design as shown below in figure 4.7.

Personnel Information			
Patient ID:			
Name:			
I/C Number:		Age:	
Gender:		Blood Type:	
Address:			
Phone No (Home)		Phone No (Mobile)	

Medical History	
Allergy:	
Chronic Illness:	
Remarks:	

Main Menu	Add	Edit	Delete	Search	Exit
-----------	-----	------	--------	--------	------

Figure 4.7: The ECS Input Form Design - Registration

4.5.2 Output Design

Output is the result of the process and is generated from input. Output is a medium to present information to the user. The usual devices for output are printer, monitor screen, plotter and many more. Output can be divided into three main types such as output screen, printed output and others output.

Output screen can be obtained by using devices based on computer. The computer screen is the mediator of visual display through CRT Terminal or monitor. Output screen provides the facility for user system to access information easily and faster.

Other than output screen, the method that usually use is printed output or printing paper. In this time, the cost of using paper is still cheap and also it's last out. Printed output is acquired by using printer like impact printer or laser printer. Usually, the printed output format is internal or external.

As shown in figure 4.8 and 4.9, are the samples of output for ECS.

Registration

LIST OF PATIENT QUEUE

No	Patient ID	Name	Time came	Remarks
1	P01/102	Lisa Goh	10.02 a.m	Emergency, see doctor 10.03 a.m
2	P03/001	Ali bin Abu	9.55 a.m	
3	P01/083	Salleh bin Mat	10.10 a.m	

Figure 4.8: Screen Output Design – List of Queue

MC LETTER

Hereby this, I have examined this patient, _____
and I/C Number _____. (He / she) is under my treatment from _____
_____ to _____.

I certified that (he / she) is suffering from _____.

I am the opinion that (he / she) **is not** (physically / mentally) incapable to go to
(work / school) from _____ till _____.

Thank you.

Yours truly,

(Dr. Azim Ali)
Clinic Azim

Figure 4.9: Printed Output Design – MC Letter

4.6 THE EXPECTED OUTCOME

When the design is complete, the review process is done in three steps, corresponding to the steps of the design process. Firstly, a preliminary design review to examine the conceptual design. Then, in a critical design review, the technical design to check its detail before proceeding with implementation. Finally, the program design review as a set of design descriptions for the actual components to be coded and tested.

E-Clinic System will operational as we expected within the six modules in this system. Each module contains their own characteristics to register the new patient, make a scheduler for next appointment, payment of medicines, referring to a specialist if the clinic could not handle the case and many more. We expected that the E-Clinic System would successfully be developed during the phase of implementation later.

4.7 SUMMARY OF CHAPTER 4

In this phase, we have seen that design begin at a high level, with important decisions about system architecture based on system requirements, desirable design attributes, and the long term intended use of the system (such as reuse or modification).

System design is to determine the user requirement and specifically what they expected from the system. When the phase implementation begins, we can avoid fatal error or mistakes and develop the system easily and successfully.

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 INTRODUCTION

After the system-designing phase on how the system should be functioning, the next process will include the system implementation phase. This phase is an important element especially when integration of system is needed between subsystems. We have to consider about the issues of setting up the development environment, which include software and hardware requirements.

Now, we must focus on implementing the solution as software. That is, we must write the programs that implement the design. First, we may not address the entire platform and programming environment; structures and relationships that are easy to describe with charts and tables are not always straightforward to write as code. Second, we must write our code in a way that is understandable not only to us when we revisit for testing. Third, we must take advantage of the characteristics of the data structures and the programming language's construct while still creating code that is easily reusable.

The most critical standard is the need for a direct correspondence between the program design components and the program code components. The entire design process is of little value if the design's modularity is not carried forward into the code. Design characteristics, such as low coupling, high cohesion, and well-defined interfaces, should also be program characteristics, so that the algorithms, functions, interfaces, and data structures can be traced easily from design to code and back again.

5.2 DEVELOPMENT ENVIRONMENT

Clearly, there are many ways to implement a design, and many languages and tools are available. In order to optimize the development process, suitable development environment is major factor to be considered. The software configuration of E-Clinic System is describes in the following table.

Development tools	Function	Description
Microsoft Windows 2000 Professional	Development Platform	Operating System
Microsoft Visual Studio 6.0	Programming Tool	System Development Tool
Microsoft Access 2000	Database Programming	Records will be store in this database
Microsoft Visual Basic 6.0	Graphical User Interface (GUI)	To design the user interface
Adobe Photoshop 7.0	Variety of picture format	To edit picture format

Table 5.1: List of ECS Development Environment Tools

5.2.1 The Methodology and Technique

By using the rapid prototyping model, it is a miniature version of an end product. We develop the prototype, test and modify it, and through an iterative process to complete the final product. Then, we build several models, some of them have to throw away and the other we fix and repair it to be useful for further progress repetitively.

5.2.2 Programming Languages

After doing some researches between May till June of Visual Basic.Net, we conclude that VB.Net is a new programming that just been launched early 2003. So, as we look for the reference for ECS development, all the materials seem hard to find. It is difficult to us if we have just limited of things to refer to.

In the nick of time, we think that if we pursue of using VB.Net, it is difficult task to do. We consider that we will use Visual Basic 6.0 as this programming languages is widely use and commonly known. And the references are easily to find and to retrieve from.

5.3 ALGORITHMS

The program design often specifies a class of algorithms to be used in coding the component developers is writing. There are several ways developer used to organize the codes.

- i) Keeping the program simple and tidy to make sure that the codes are readable.
- ii) Using data structures to determine program structures
 - Coding such as function that needed to call for few times may put out of the main program. Besides, variables that may need to use in several forms, should put in general module (public).
- iii) Always give comment on code, additional comments are useful wherever helpful information can be added to a component.
 - Comment block in VB 6 may looks like [`'Add new record]`

Before writing code, algorithms had been prepared to make sure that the flows of the system are smooth and easy to understand. For example, the flows and algorithms for a patient who comes for doctor as followed:

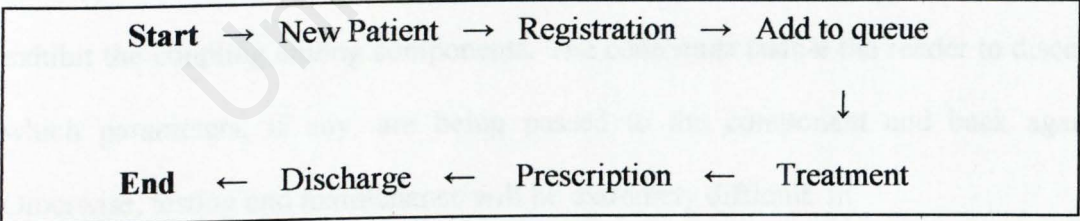


Figure 5.1: Algorithm of patient treatment

5.4 CODING

Programming involves a great deal of creativity. The design is to guide to the function or purpose of each component, but we have great flexibility in implementing the design as code. The program design often specifies a class of algorithms to be used in coding the component we are writing. We have the flexibility in converting the algorithm to code, subject to the constraints of the implementation language and hardware.

Each program component involves at least three major aspects: control structures, algorithms, and data structures. We examine each more closely.

Control structures

Many of the control structures for a component are suggested by the architecture and design, and we want to preserve them as the design is translated to code. It is important for our program structure to reflect the design's control structure.

In writing code, keep in mind that generality is a virtue; do not make code more specialized than it needs to be, and use parameter names and comments to exhibit the coupling among components. The code must enable the reader to discern which parameters, if any, are being passed to the component and back again. Otherwise, testing and maintenance will be extremely difficult. In

Data structures

In writing programs, we should format and store data so that data management and manipulation are straightforward. There are several techniques that use the structure of the data to suggest how the program should be organized.

Keeping the program simple. The program's design may specify some of the data structures to be used in implementing functions. Often, these structures are chosen because they fit into an overall scheme that promotes information hiding and control of components interfaces. Data manipulation within a component can influence our choice of data structures in a similar way.

Using a data structure to determine a program structure. In general, data structures can influence the organization and flow of a program. A data structure is said to be recursive if identifying an initial element and then generating successive elements as a function of those previously define it.

General guidelines

Several overall strategies are useful in preserving the design quality in coding

- i) Localizing input and output - those parts of program that read input or generates output are highly specialized and must reflect characteristics of the underlying hardware and software.
- ii) Including pseudo code – the design usually lays out a framework for each program component. Since the design is an outline of what is to be done in a program component, it is useful to move in stages from the specified design to the code, rather than to translate the design immediately into code.
- iii) Revising and rewriting, not patching – when writing code, we often write a rough draft, then carefully revise and rewrite until we are satisfied with the result.
- iv) Reuse – there are two kinds of reuse: producer reuse, where we are creating components designed to be reused in subsequent applications, and consumer reuse, where we are using components that were originally developed for other projects.

5.4.1 VB 6 linking to database

As a management information system, one of the most important parts is linking to database. The vast majority of applications today have to perform some sort of database access. Whether you are building Windows applications and want to perform simple database access, or you are creating middle-tier components to handle database access, understanding how to connect to a database, retrieve information, and manipulate data are critical to understanding.

All the new database-related capacities in Visual Basic 6 are based on Microsoft ActiveX Data Objects (ADO), a technology that lets you access any database or data source, as long as someone has written an OLE DB provider that connects to that source. Data access methods differ greatly in the number of layers that sit between your application and the database you're connecting to.

Visual Basic 6 includes several tools and facilities for creating ADO applications quickly and effectively. ADO is the high-level interface to OLE DB. ADO builds on OLE DB to provide functions that aren't available directly in OLE DB or that would make stringent demands on the coding abilities of a programmer. ADO can make asynchronous queries and connections and optimistic batch updates.

The single most important feature of ADO is probably its extensibility. Instead of being a complex and monolithic object hierarchy as DAO and RDO are, ADO consists of fewer objects that can be combined in more ways. New features can be added to ADO in the form of special OLE DB providers, such as the MSDataShape provider, which offers hierarchical Recordset objects to other providers. Microsoft also is making new features available in ADO in the form of separate libraries that link dynamically to the core ADO library.

5.4.2 VB 6 with ADO password encryption

Characters type by the user is displayed as asterisks (*) to prevent characters from being seen on the screen. Once an access code is entered, access is either granted or denied. We create a simple frmLogin form, which asks the end user for his or her name and password and refuses to unload if the password isn't the correct one. This simple form has only two public properties, *UserName* and *Password*, which are set to the contents of the txtUserName and txtPassword controls, respectively, in the *Unload* event procedure. This is the sample source code of the frmLogin form module:

```
Private Sub CommandUserLogin_Click()  
' Try to see if the user wrote a correct username  
' by trying to extract it from the Database.  
' If it exists - it's correct, otherwise, it's wrong.  
  
Adodc1.RecordSource = "SELECT * FROM Admin Where  
UserID=" & TextUserName.Text & ""  
Adodc1.Refresh  
  
If Adodc1.Recordset.RecordCount = 0 Then 'user doesn't exist!  
    MsgBox "Invalid Username! Try again.", vbExclamation, "ECS"  
    TextUserName = ""  
    TextPassword = ""  
    TextUserName.SetFocus  
    Exit Sub ' so that it won't move on the checking the pass.  
End If  
  
'check the password  
'extracted the right fields from the DataBase by checking the  
username.  
'Just compare to the password:  
  
If Adodc1.Recordset.Fields("Password") = TextPassword.Text Then  
'Password Correct  
    MMenu3.Show  
    MMenu3.Caption = "Welcome User..."  
    Unload Me
```

Figure 5.2: Sample of frmLogin source code

5.5 SUMMARY OF CHAPTER 5

As we know, system implementation is not as easy as designing. It is different because building something from a scratch (design) to represent it in the real thing. Our use of common design techniques and strategies makes our code easier to test, maintain and reuse.

Implementing the requirements to real world is difficult task. We develop the system with the Visual Basic 6 for the user interface and MS Access 2000 for the database. The operating system we use Windows 2000 and for the graphic picture or icon, we use Adobe Photoshop 7 for editing.

Lastly, we do the programming languages for the code of ECS system using VB script with the algorithm is been set, and we just follow it. Doing the coding is the longest time to take compare to all as we must be careful and make sure there are no mistakes or the modules will not functioning well.

CHAPTER 6: SYSTEM TESTING

6.1 INTRODUCTION

Once we have coded the program components, it is time to test them. There are many types of testing, testing approaches that lead to delivering a quality system to the customers. Testing is not the first place where fault finding occurs; we have seen how requirements and design reviews help us ferret out problems early in development. But, testing is focused on finding faults, and there are many ways we can make our testing efforts more efficient and effective.

After coding the program component, we usually examine the code to spot faults and eliminate them right away. When no obvious faults exist, we then test our program to see if we can isolate more faults by creating conditions where the code does not react as planned. Thus, it is important to know what kind of faults we are seeking.

An algorithm fault occurs when a component's algorithm or logic does not produce proper output for a given input because something is wrong with the processing steps. These faults sometimes easy to spot just by reading through the program (called desk checking) or by submitting input data from each of the different classes of data that we expect the program to receive during its regular working.

Typical algorithm faults include: -

- Branching too soon
- Branching too late
- Testing for the wrong condition
- Forgetting to initialize variables or set loop invariants
- Forgetting to test for a particular condition (such as when division by zero might occur)
- Comparing variables of inappropriate data types

When checking for algorithm faults, we may also look for syntax faults.

Here, we want to be sure that we have properly used the constructs of the programming languages. Sometime, the presence of a seemingly trivial fault can lead to disastrous results. Fortunately, compilers catch many of our syntax faults for us.

Computation and precision faults occur when a formula's implementation is wrong or does not compute the result to the required degree of accuracy. For instance, combining integer and fixed- or floating-point variables in an expression may produce unexpected results. Sometimes, improper use of floating-point data, unexpected truncation, or ordering of operation may result in less-than-acceptable precision.

The requirements specification usually details the number of users and devices and the need for communication in a system. By using this information, the designer often tailors the system characteristics to handle no more than a maximum load described by the requirements. These characteristics are carried through to the program design as limits on the length of queues, the size of buffers, the dimensions of tables, and so on. **Stress or overload faults** occur when these data structures are filled past their specified capacity.

Similarly, **capacity or boundary faults** occur when the system's performance becomes unacceptable as system activity reaches its specified limit. Example, if the requirements specify that a system must handle 32 devices, the programs must be tested to monitor system performance when all 32 devices are active. By testing and documenting the system's reaction to overloading its stated capacity, it can help with maintenance later to understand the implication of increasing system capacity in the future.

In developing real-time systems, a critical consideration is the coordination of several processes executing simultaneously or in a carefully defined sequence. **Timing or coordination faults** occur when the code coordinating these events is inadequate.

Throughput or performance faults occur when the system does not perform at the speed prescribed by the requirements. These are timing problems of a different sort; time constraints are placed on the system's performance by the customer's requirements, rather than by the need for coordination.

As we saw during design and programming, we take great care to ensure that the system can recover from a variety of failures. **Recovery faults** can occur when a failure is encountered and the system does not behave as desired or required.

For many systems, some of the hardware and related system software are prescribed in the requirements, and the components are designed according to the specification of those reused or purchased programs. However, **hardware and system software faults** can arise when the supplied hardware and system software do not actually work according to the documented operating conditions and procedures.

Finally, the code should be reviewed to confirm that organizational standards and procedures have been followed. **Standard and procedure faults** may not always affect the running of the programs, but they may foster an environment where faults are created as the system is tested and modified. By failing to follow the required standards, one programmer may make it difficult for another to understand the code's logic or to find the data description needed for solving problem.

6.2 UNIT TESTING

If our goal is to find faults in components, how do we begin? The process is similar to the one we use when testing a program assigned in class. First, we examine the code by reading through it, trying to spot algorithm, data and syntax faults. You may even compare the code with the specifications and with design to make sure we have considered all relevant cases. Next, we compile the code and eliminate remaining syntax faults. Finally, we develop test cases to show the input is properly converted to the desired output. Unit testing follows exactly these steps, and we examine them one at a time.

6.2.1 Examining the code

Because the design description helps you to code and document each program component, our program reflects the interpretation of the design. The documentation explains in words and pictures what the program is supposed to do in code.

- Code review – ask an objective group of experts to review both code and documentation for misunderstandings, inconsistencies, and other faults.
- Code walkthrough – present code and accompanying documentation to the review team, and the team comments on their correctness.
- Code inspection – the review team checks the code and documentation against a prepared list of concerns.

6.2.2 Proving code correct

Suppose when our component has been coded, examined and reviewed. The next step in testing is to subject the code to scrutiny in a more structured way to establish its correctness. For the purpose of unit testing, a program is correct if it implements the function and data properly as indicated in the design, and if it interfaces properly with other components

- Formal proof techniques – convert the code to its logical counterpart in a series of steps
- Other proof techniques – the logical proof technique ignores the structure and syntax of the programming language in which the test program is implemented. The technique proves that the component's design is correct but not necessarily its implementation.
- Automated theorem proving – some software engineers have tried to automate the process of proving programs correct by developing tools that read as input; the input data and conditions, the output data and conditions, the lines of code for the component to be tested.

6.2.2 Proving code correct

Suppose when our component has been coded, examined and reviewed. The next step in testing is to subject the code to scrutiny in a more structured way to establish its correctness. For the purpose of unit testing, a program is correct if it implements the function and data properly as indicated in the design, and if it interfaces properly with other components

- Formal proof techniques – convert the code to its logical counterpart in a series of steps
- Other proof techniques – the logical proof technique ignores the structure and syntax of the programming language in which the test program is implemented. The technique proves that the component's design is correct but not necessarily its implementation.
- Automated theorem proving – some software engineers have tried to automate the process of proving programs correct by developing tools that read as input; the input data and conditions, the output data and conditions, the lines of code for the component to be tested.

6.2.3 Testing program components

Proving code correct is a goal to which software engineers aspire; consequently, much related research is done to develop methods and automated tools.

- Testing versus proving – in proving a program correct, the test team or programmer considers only the code and its input and output conditions
- Choosing test cases – we choose input data and conditions allow the components to manipulate the data, and observe the output
- Test thoroughness – to perform a test, we decide how to demonstrate in a convincing way that the test data exhibit all possible behaviors. To test code thoroughly, we can choose test cases using at least one of several approaches based on data manipulated by the code: statement testing, branch testing, path testing, definition-use path testing, all-predicate-uses / some-predicate-uses testing

6.2.4 Integration Testing

When we are satisfied that individual components are working correctly and meet our objectives, we combine them into a working system. This integration is planned and coordinated so that when a failure occurs, we have some idea of what caused it. In addition, the order in which components are tested affects our choice of test cases and tools. Our test strategy explains why and how components are combined to test the working system. This strategy affects not only the integration timing and coding order, but also the cost and thoroughness of the testing.

The system is again viewed as hierarchy of components, where each component belongs to a layer of the design. We can begin from the top and work our way down as we test, work from the bottom up, or use a combination of these two approaches.

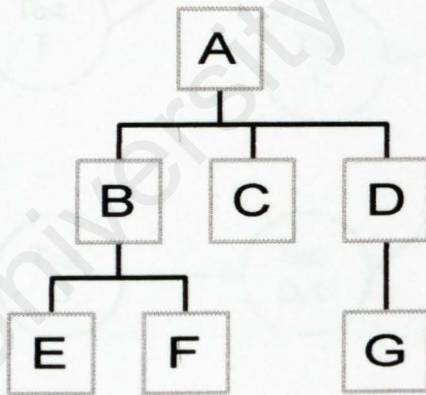


Figure 6.1: An example of component hierarchy

6.2.4.1 Bottom-up Integration

One popular approach for merging components to test the larger system is called bottom-up testing. When this method is used, each component at the lowest level of the system hierarchy is tested individually first. Then, the next components to be tested are those call the previously tested ones. This approach is followed repeatedly until all components are included in the testing. The bottom-up method is useful when many of the low-level components are general-purpose utility routines that are invoked often by others, when the design is object oriented or when the system is integrating a large number of stand-alone reused components.

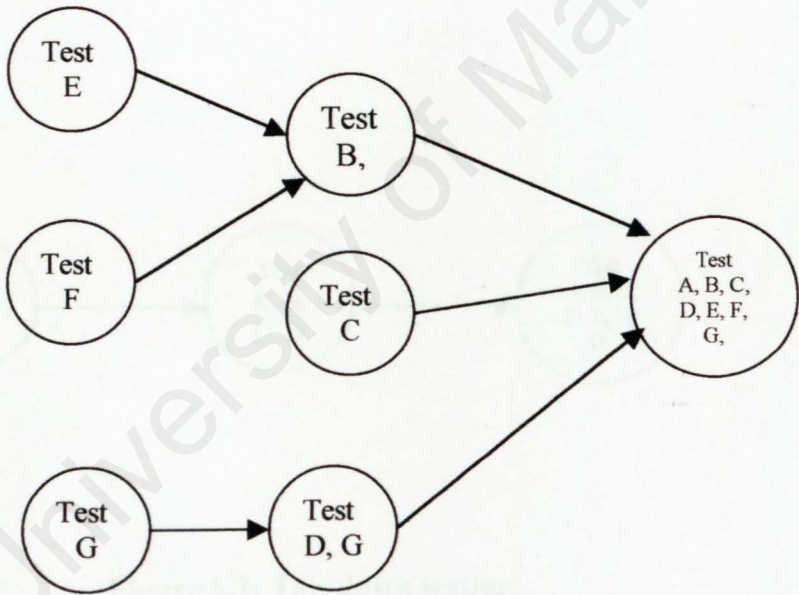


Figure 6.2: Bottom-up testing

6.2.4.2 Top-down Integration

Many developers prefer to use top-down approach, which in many ways is the reverse of bottom-up. The top level, usually one controlling component, is tested by itself. Then, all components called by the tested component(s) are combined and tested as a larger unit. This approach is reapplied until all components are incorporated.

A component being tested may call another that is not yet tested, so we write a stub, a special-purpose program to simulate the activity of the missing component. The stub answers the calling sequence and passes back output data that lets the testing process continue.

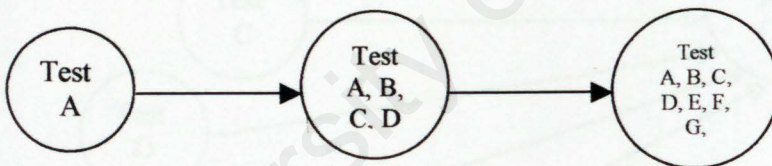


Figure 6.3: Top-down testing

6.2.4.3 Big-bang integration

When all components are tested in isolation, it is tempting to mix them together as the final system and see if it works the first time, Myers (1979) call this big-bang testing. Many programmers use the big-bang approach for small systems, but it is not practical for large ones.

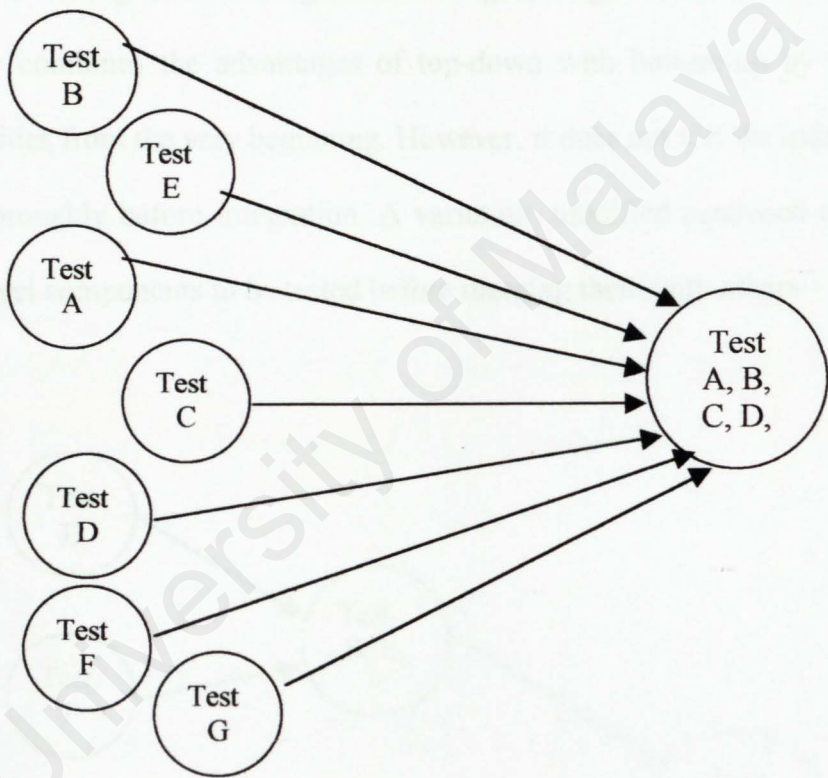


Figure 6.4: Big-bang testing

6.2.4.4 Sandwich integration

Myers (1979) combines a top-down strategy with a bottom-up one to form a sandwich testing approach. The system is viewed as three layers, just like a sandwich: the target layer in the middle, the level above the target, and the levels below the target. A top-down approach is used in the top layer and a bottom-up one in the lower layer. Testing converges on the target layer, chosen on the basis of system characteristics and the structure of the component hierarchy.

Sandwich testing allows integration testing to begin early in the testing process. It also combines the advantages of top-down with bottom-up by testing control and utilities from the very beginning. However, it does not test the individual components thoroughly before integration. A variation, modified sandwich testing, allows upper-level components to be tested before merging them with others.

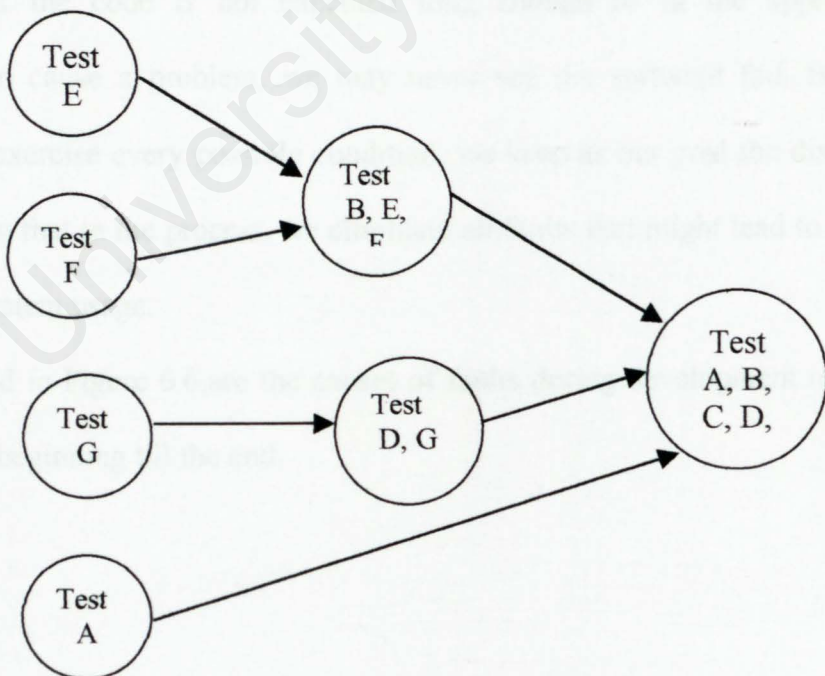


Figure 6.5: Sandwich testing

6.3 SYSTEM TESTING

Testing the system is very different from unit and integration testing. When we unit our components, we have complete control over the testing process. We create our own test data, design our own test cases, and run the test our self.

The objective of unit and integration testing was to ensure that the code implemented the design properly; that is we wrote the code as expected. In system testing, we have very different objective: to ensure that the system does what the customer wants it to do. To understand how to meet this objective, we first must understand where faults in the system come from.

Sources of software faults

Recall that a software fault causes a failure only when accompanied by the right conditions. That is, a fault may exist in the code, but if the code is never executed, or if the code is not executed long enough or in the appropriate configuration to cause a problem, we may never see the software fail. Because testing cannot exercise every possible condition, we keep as our goal the discovery of faults, hoping that in the process, we eliminate all faults that might lead to failure during actual system usage.

As stated in Figure 6.6, are the causes of faults during development in every phase from the beginning till the end.

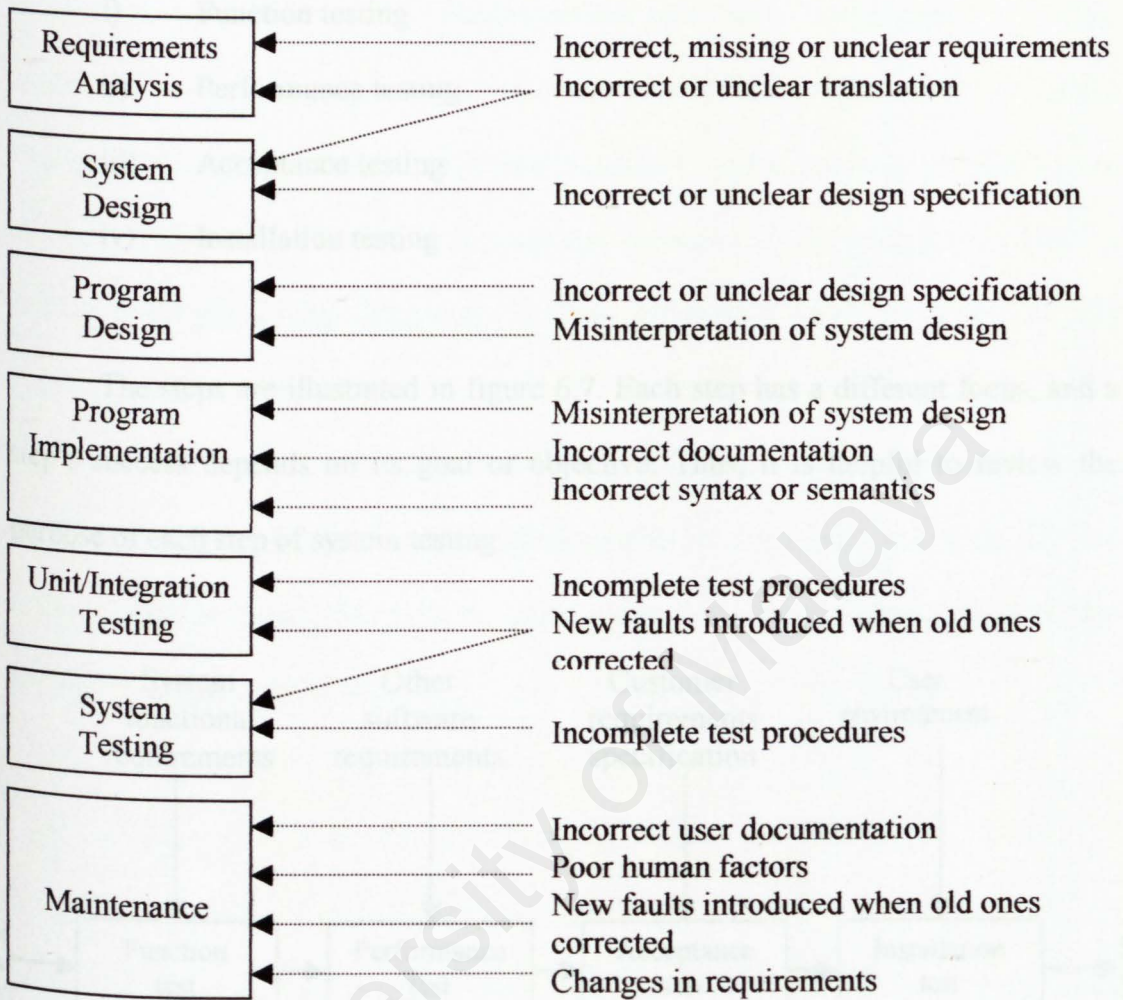


Figure 6.6: Causes of faults during development

System testing process

There are several steps in testing a system

- i) Function testing
- ii) Performance testing
- iii) Acceptance testing
- iv) Installation testing

The steps are illustrated in figure 6.7. Each step has a different focus, and a step's success depends on its goal or objective. Thus, it is helpful to review the purpose of each step of system testing.

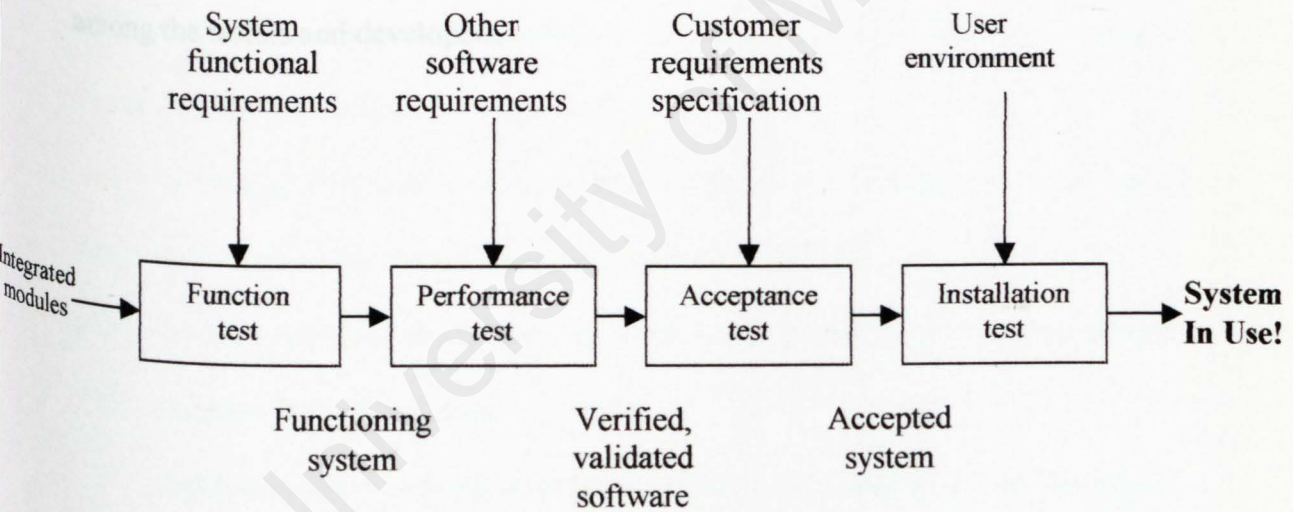


Figure 6.7: Steps in the testing process

Configuration management

We often test a system in stages or pieces, based on spins (as before) or on subsystems, functions, or other decomposition that make testing easier to handle. However, system testing must also take into account the several different system configurations that are being developed. A system configuration is a collection of system components delivered to a particular customer. The configuration may be further distinguished by those who running on certain kinds of chips or with particular devices available.

Developing and testing these different configurations required configuration management, the control of system differences to minimize risk and error. During testing, configuration management is especially important, coordinating efforts among the testers and developers.

6.3.1 Function testing

System testing begins with function testing. Whereas previous test concentrated on components and their interactions, this first step ignores system structure and focused on functionality. Our approach from now on is more closed box than open. We need not know which component is being executed; rather, we must know what the system is supposed to do. Thus, function testing is based on the system's functional requirements.

Purpose and roles

Each function can be associated with those systems components that accomplish it. For some function, the parts may comprise the entire system. The set of actions associated with a function is called a thread, so function testing is sometimes called thread testing.

Logically, it should be easier to find the cause of a problem in a small set of components than in a large set. Thus, ease of testing calls for choosing carefully, the order in which functions are tested. Functions may be defined in a nested manner, just as spins are defined in levels.

Function testing is performed in a carefully controlled situation. Moreover, since we are testing one function at a time, function testing can actually begin before the entire system is constructed, if need be.

Function testing compares the system's actual performance with its requirements, so the test cases for function testing are developed from the requirements document.

6.3.2 Performance testing

Once we determine that the system performs the functions required by the requirements, we turn to the way in which those functions are performed. Thus, functional testing addresses the functional requirements, and performance testing addresses the nonfunctional requirements.

Purpose and roles

System performance is measured against the performance objectives set by the customer as expressed in the nonfunctional requirements. Performance testing is designed and administered, and the results are provided to the customer and performance testing usually involves hardware as well as software.

Types of performance test

Performance testing is based on the requirements, so the types are determined by the kinds of nonfunctional requirements specified.

- i) Stress test – evaluate the system when stressed to its limits over a short period of time
- ii) Volume test – address the handling of large amounts of data in the system.
- iii) Configuration test – analyze the various software and hardware configurations specified in the requirements.
- iv) Compatibility test – are needed when a system interfaces with other systems, find out whether the interface function perform according to the requirements

- v) Regression test – are required when the system being tested is replacing an existing system.
- vi) Security test – ensure that the security requirements are met.
- vii) Timing test – evaluate the requirements dealing with time respond to a user and time to perform a function.
- viii) Environmental test – look at the system's ability to perform at the installation site.
- ix) Quality test – evaluate the system's reliability, maintainability, and availability, include calculation of mean time to failure or to repair
- x) Recovery test – address response to the presence of faults or to the loss of data, power, devices, or services.
- xi) Maintenance test – address the need for diagnostic tools and procedures to help in finding the sources of problems
- xii) Documentation test – ensure that we have written the required documents
- xiii) Human factor test – investigate requirements dealing with the user interface to the system.

6.3.3 Reliability, availability and maintainability

One of the most critical issues in performance testing is assuring the system's reliability, availability, and maintainability. Because each of these system characteristics cannot always be measured directly before delivery, this assurance is especially difficult; we must use indirect measures to estimate the system's likely characteristics.

Reliability involves behavior over a period of time, but availability describes something at a given point in time.

We want our software to function consistently and correctly over long periods of time, to be available when we need it, and to be repaired quickly and easily if it does fail. We say formally that software reliability is the probability that a system will operate without failure under given conditions for a given time interval.

Similarly, software availability is the probability that a system is operating successfully according to specification at a given point of time. More formally, it is the probability that a system is functioning completely at a given instant in time, assuming that the required external resources are also available.

Likewise, software maintainability is the probability that, for a given condition of use, a maintenance activity can be carried out within a stated time interval and using stated procedures and resources. It is very different from hardware maintenance; hardware usually requires the system to be unavailable as maintenance is being carried out, but software maintenance can sometimes be done while the system still up and running.

Because reliability, availability, and maintainability are defined in terms of failures, they must be measured once the system is complete and working. We often assign a severity level to each failure, to capture its impact on the system.

6.3.4 Acceptance testing

When function and performance testing are complete, we are convinced that the system meets all requirements specified during the initial stages of software development. The next step is to ask the customers and users if they concur.

Until now, we as developers have designed the test cases and administrated all tests. Now the customer leads testing and defines the cases to be tested. The purpose of acceptance testing is to enable the customers and users to determine if the system we built really meets their needs and expectations. Thus, acceptance tests are written, conducted, and evaluated by the customers, with assistance from the developers only when the customer request an answer to a technical question.

Usually, those customer employees who were involved in requirements definitions play a large part in acceptance testing, because they understand what kind of system the customer intended to have built.

Acceptance testing uncovers more than requirements discrepancies, also allows customers to determine what they really want, whether specified in the requirements documents or not... after acceptance testing, the customer tells us which requirements are not satisfied and which must be deleted, revised, or added because of changing needs. Configuration management identifies these changes and record consequent modifications to design, implementation and testing.

6.3.5 Installation testing

The final round of testing involves installing the system at user sites. If acceptance testing has been performed on-site, installation testing may not be needed. However, if acceptance testing conditions were not the same as actual site conditions, additional testing is necessary. To begin installation testing, we configure the system to the user environment. We attach the proper number and kind of devices to the main processor and establish communications with other systems. We allocate files and assign access to appropriate functions and data.

Installation test requires us to work with the customer to determine what tests are needed on-site. Regression tests may be administered to verify that the system has been installed properly and works "in the field" as it did when tested previously. The test cases assure the customer that the system is complete and that all necessary files and devices are present. The test focuses on two things; completeness of the installed system and verification of any functional or nonfunctional characteristics that may be affected by site conditions.

When the system customer is satisfied with the result, testing is complete and the system is formally delivered.

6.4 SUMMARY OF CHAPTER 6

There are many techniques that we can use to test our code components individually or integrated. It is important for us to understand the difference between a fault (a problem in the requirements, design, code, documentation, or test cases) and a failure (a problem in the functioning of the system). Testing looks for faults, sometimes by forcing code to fail and then seeking the root cause.

The goal of testing is to find faults, not to prove correctness. Indeed, the absence of faults does not guarantee correctness.

The major issues in software testing, including those related to reliability and safety. During requirements analysis, we should think about system function that will capture state information and data that will help you we find the root cause if the software fails. During design, we should use fault-tree analysis, failure modes and effect analysis, and other techniques to help us avoid failures or moderate their effects. During design and code reviews, we can build a safety case to convince that our software is highly reliable and will lead to a safe system. And during testing, we can make take great care to consider all possible test cases, to automate where appropriate, and to ensure that your design addresses all possible hazards.

CHAPTER 7: SYSTEM EVALUATION

7.1 INTRODUCTION

We as software developer use a large variety of methods and tools to elicit and specify requirements, design and implement system, and test and maintain them as it evolves. Also we are keen to evaluate our products and the ways in which we produce them.

The evaluation techniques we use are similar to those in other disciplines: we measure key aspects of our products, processes, and resources and use this information to determine whether we have meet goals for productivity, performance, quality, and other desirable attributes.

We can think of an evaluation technique as being in one of four categories:

- i) **Feature analysis**
- ii) **Case study**
- iii) **Survey**
- iv) **Formal experiment**

7.1.1 Feature analysis

The simplest type of assessment is a feature analysis, used to rate and rank the attributes of various products so we can tell which tool to buy or method to use. Feature analysis is necessarily very subjective, and the ratings reflect the raters' biases. It is useful for narrowing down which tools to buy, but it does not really evaluate behavior in terms of cause and effect. It is not at all useful in determining which design technique is most effective at helping us build complete and consistent designs; in this case, we want to run controlled studies so we can understand cause and effect.

We list four key attributes that our tools should have:

1. Good user interface
2. Handles object-oriented design
3. Checks for consistency
4. Handles use cases

7.1.2 Case study

Both case studies and formal experiments are usually retrospective. We decide in advance what we want to investigate and then plan how to capture data to support the investigation. In a case study, we identify key factors that may effect and activity's outcome and then document them: inputs, constraints, resources and outputs. By contrast, a formal experiment is a rigorous, controlled investigation, where an activity's key factors are identified and manipulated to do document their effects on the outcome.

7.1.3 Survey

A survey is retrospective study to try to document relationships and outcomes in a given situation. Surveys are often done in the social sciences, where attitudes are polled to determine how a population feels about a particular set of issues, or a demographer surveys a population to determine trends and relationships. We record data to determine how project participants reacted to a particular method, tool, or technique, or to determine trends or relationships. We can also capture information related to products or projects, to document the size of components, number of faults, effort expended, and so on.

When performing a survey, we usually have no control over the situation at hand. Because a survey is retrospective study, we record information about situation and compare it with similar ones, but we cannot manipulate variables; for that, we need case studies and experiments.

7.1.4 Formal experiment

In a formal experiment, values of independent variables are manipulated, and we observe changes in dependent variables to determine how changes in the input affect changes in the output. We may examine the effect of a tool or technique on a product quality or programming productivity; or we may try to discover the relationship between preparation time and inspections effectiveness.

In a formal experiment, several methods are used to reduce bias and eliminate confounding factors so cause and effect can be evaluated with some confidence. Formal experiments are designed carefully, so that the instances we observe are as representative as possible.

7.2 PREPARING FOR AN EVALUATION

No matter what kind of evaluation we choose to do, there are several key steps to making sure we are focused and can identify the appropriate variables.

Setting the hypothesis

We begin by deciding what we wish to investigate; expressed as hypothesis we want to test. That is we must specify exactly what it is that we want to know. The hypothesis is the tentative theory or supposition that we think explains the behavior we want to explore.

Maintaining control over variables

Once we have an explicit hypothesis, we must decide what variables can affect its truth. Then, for each variable identified, we decide how much control we have over it.

Making the investigation meaningful

It is important to remember that we cannot control everything; software is not like biology or chemistry experiments. We must take into account the limitations and lack of control when deciding whether study results apply to a new situation.

7.3 SELECTING AN EVALUATION TECHNIQUE

Since formal experiments require a great deal of control, they tend to be small, involving small numbers of people or events. Case studies usually look at a typical project, rather than trying to capture information about all possible cases; these can be thought of as “research in the typical”. And surveys try to poll what is happening broadly over large groups of projects.

Several general guidelines can help us decide whether to perform a survey, case study, or a formal experiment. As we have seen, control is a key element in our decision. If we have a high level of control over the variables that can affect the outcome, then we consider an experiment. If we do not have that control, a case study is the preferred technique. But the level of control satisfies the technical concerns; we must also address practical concerns. It may be possible but very difficult to control the variables, either because of the high cost of doing so or the degree of risk involved.

As E-Clinic System, semi safety-critical system, it may entail a high degree of risk in experimentation, and a case study may be more feasible. Case study may be preferable, as the process changes caused by the independent variables are wide ranging, requiring the effects to be measured at a high level and across too many dependent variables to control and measure.

7.4 EVALUATING

Evaluation always involves measurement. We capture information to distinguish different values of dependent and independent variables, and we manipulate the information to increase our understanding. In addition, measurement helps us to separate typical from unusual situation, or to define baseline and set goals.

We validate a prediction system in a given environment by establishing its accuracy by empirical means; that is we compare the model's performance with known data in the given environment. We state a hypothesis about the prediction, and then we look at a data to see whether the hypothesis is supported or refuted.

7.4.1 EVALUATING PRODUCTS

We have seen that software development produces a large number of artifacts: requirements, design, code, components, test cases, and more. In each case, we can examine a product to determine if it has desirable attributes. That is, we can ask whether system has certain properties, such as completeness, consistency, reliability, or maintainability.

7.4.2 EVALUATING PROCESSES

A process intended to improve our software in some way. The effort needed to enact a process varies. Some process involve the entire software development life cycle, whereas other focus on a small group of activities. We want our processes to be effective and efficient.

7.5 SYSTEM STRENGTHS

E-Clinic System (ECS) is a full-featured clinic management system designed for the daily operation of a clinic. The system strengths are the advantages and speciality of the ECS that will benefit the user as stated below: -

1. Different Level of User Access
 - The users are dividing to two types, as administrator and user.
 - As administrator, he/she can access and update all four modules
 - As user, he/she can access only certain modules.
2. Easy-to-use application
 - Just point and click navigation mouse that will help users to do their daily work.
3. Efficient data management
 - The data that are store in the database will be process as record and save it in the database management system (DBMS).
4. Easy data retrieval
 - All the records in the database using DBMS can be retrieved easily
5. Organized and user friendly interface
 - The system is easy to use as the interface is easy to understand with the help of icon and toolbar for operation.

7.6 PROBLEMS ENCOUNTERED

After the development and testing of ECS, there are a lot of problems that encounter. We try to fix and repair it but the problems still occur. As first time developer, we have no experience to handle and resolve the problems.

1. Difficulties in determining the scope of the system

- The scope defines the system boundary, explaining what will be included in the system and what will not be included.
- Some items may cross the boundary of system, and others are products of our system that travel out for another system's use.
- The scopes are difficult and complex to determine which field and opportunities for the system.

2. Understanding on current system procedure

- To understand the current system procedures and enhance it for the new system are not easy.
- Maybe the current procedures are much more complex to implement it for ECS.
- Usually, the users have lot of things to adapt for new environment, as they are comfortable with the old one.

3. Consideration on relationship of subsystem

- Each module has it relationship with other module. We have to consider each relationship so it can link with each other in the database.

7.7 SYSTEM LIMITATION

Every system has its own limit. We have to consider the limitation that will affect the system. It is not considerably critical, but we must aware and provide some solution to handle it

1. Lack of functional module

- Actually, there are five modules in the proposed system. But, at last the finance module, we have to eliminate it.
- It is because of the relation between Finance module and Inventory of medicine stock control module.
- The inventory module is other system that not include in the ECS.

2. Database Management

- Records are one of asset that is very important. So, we have to make sure that the records are managed efficiently. So, the database administrator responsible to the management of data.
- The administrator has to decide how to collect and save the data including of the backup data or records. But, if the administrator is unable to come/work, so there is nobody able to backup data.

3. Time constraints

- As first developer, we have a lot of things to realize how to develop a system within the time specified. It is not easy for us to manage all if we have no experience at all. This will affect the system that we build as it is not will be perfect.

7.8 FUTURE ENHANCEMENT

For the future enhancement, we will try to make sure that all limitation and problems encountered will be resolved. It will make ECS more availability of its functionality.

Cover more functionality

The functionality of ECS is just point and click navigation. We will try to cover more functionality such as add more command button, mouse right-click navigation, more graphic elements and many more.

Module enhancement

As in the limitation of ECS that there is lack of functional module, we will try to cover the problem by just adding Inventory of Medicine Stock Control in the system. Or, we build and develop that module for future enhancement.

7.9 SUMMARY OF CHAPTER 7

We have look at ways to evaluate products and processes of the ECS system.

We begin to review several approaches to evaluation, including feature analysis, surveys, case studies, and formal experiments. We see that measurement is essential and important for the evaluation.

After we analysis the ECS strengths, limitation and the problems that encounter during developing the system, we will try to overcome all of it for future enhancement later if we have the time. ECS is the first version, and maybe some other time it will be second version and many more.

6. A. Lohr, J. K. C. (2001). *Network Management: A Practical Perspective*. Addison-Wesley.
7. R. P. (2001). *Software Engineering: A Practitioner's Approach*. McGraw-Hill International, 3rd Edition.
8. M. Flynn, J. L. and M. J. (2001). *Operating Systems*. 2nd Edition. PWS Publishing.
9. L. P. (2001). *Introduction to Management Systems Project Management*. Irwin/McGraw-Hill.
10. C. S. (2001). *Project Basic II: A step-by-step guide*. Prentice Hall.
11. T. R. N. (2001). *Project Basic II: A step-by-step guide*. Prentice Hall.
12. P. R. (2001). *Project Basic II: A step-by-step guide*. Prentice Hall.
13. <http://MSDN.microsoft.com>
14. www.cnet.com
15. www.bbc.com
16. www.abc.com
17. www.abc.com

BIBLIOGRAPHY

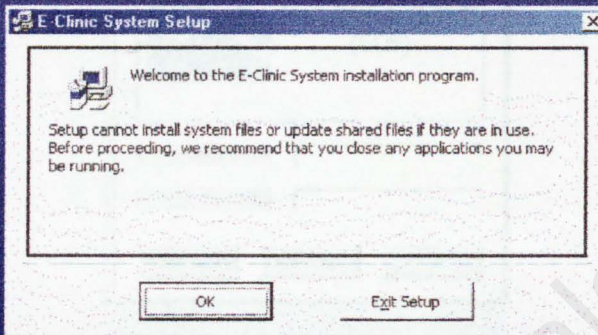
1. Pfleeger, Shari Lawrence (2001). *Software Engineering Theory and Practice Second Edition*. Prentice Hall International Inc.
2. Mohamad Noorman Masrek, Safawi Abdul Rahman and Kamarulariffin Abdul Jalil (2001). *Analisis & Rekabentuk Sistem Maklumat*. Malaysia. McGraw-Hill Sdn. Bhd.
3. Dr. Abdullah Embong. (2000). *Sistem Pangkalan Data: Konsep Asas, Rekabentuk dan Pelaksanaan*. Tradisi Ilmu Sdn. Bhd.
4. Bray, L.K (2002). *An Introduction to Requirements Engineering*. Addison Wesley.
5. MacLaszek, L (2002). *Requirement Analysis and System Design: Development Information System with UML*. Addison Wesley.
6. A.Leindwand, K.Conroy (2001). *Network Management A Practical Perspective*. Addison Wesley.
7. S. Pressman, Roger. *Software Engineering, A Practioner's Approach*. McGraw- Hill International. 3rd Edition
8. M.Flynn, Ida and McHoe, Ann McIver. *Understanding Operating Systems*. 2nd Edition. PWS Publishing Company.
9. L.Olson, David. *Introduction to Management Systems Project Management*. Irwin McGraw-Hill.
10. Chooi See, Chua (2000). *Visual Basic 6 – A step-by-step guide*. Federal Publication Sdn. Bhd.
11. T.R. Nieto, H.M. Deitel, P.J. Deitel (2000). *Visual Basic 6, How to program*. Prentice Hall.
12. Ferret, Robert and Sally, Preston and John, Preston (2000). *Essentials Access 2000 Intermediate*. Prentice Hall.
13. <http://MSDN.Microsoft.com>
14. www.dpro.com
15. www.stickyminds.com
16. www.comptechdoc.org
17. www.webhealthcentre.com

18. www.nec.com
19. www.hhheart.com
20. www.eclevelandclinic.org
21. www.ehealth.telus.com
22. www.secure.gsm.com
23. www.encoreservices.com
24. www.visualbasic.about.com
25. www.mvps.org/vbnet
26. www.dotnetforum.net
27. www.synfusion.com/FAQ/WinForms/default.asp
28. www.icshapecode.net
29. www.centrahealth.com
30. www.jhhs.org

APPENDIX A – USER MANUAL

- 1) Firstly, in the folder of ECS, choose setup.exe to install E-Clinic System. Just follow the instruction until the setup is complete.

E-Clinic System Setup



- 2) Then, search the ECS system that we install in the start menu. We will see as stated below.

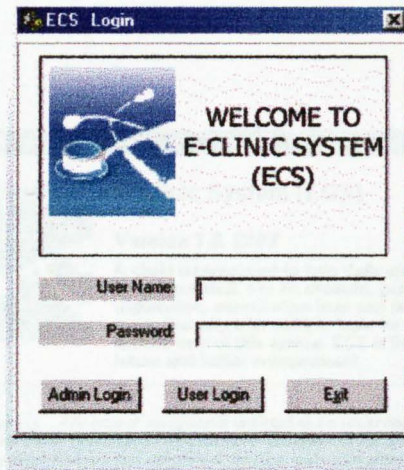


**WELCOME TO
E-CLINIC SYSTEM
(ECS)**

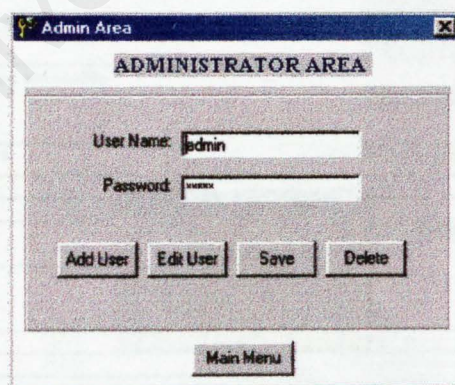
By. Rossuryaindah binti Maarof

WEK 000199

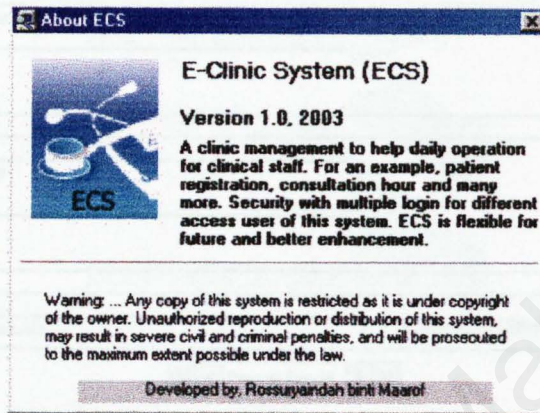
3) Then, we have to log in as user or administrator.



4) If we log in as administrator, we can add, edit or update for user of the system



- 5) As administrator, we can access to all module in the system, while as user access only for certain module.
- 6) If we want to know about the system, we can click the menu 'About'.



- 7) There are many modules we can add, edit, delete, print or search records.

Personel Information:-

Patient Name:	Surya Omar	PatientID:	P010
I/C No. /Birth Certificate:	800604-01-5912	Gender:	Female
Address:	21 Jln Utama, tmn Gembira	Age:	23
Contact No (Home):	07-3348054	(Handset):	019-7639775

Relative Information:-

Relative:	Omar Ali	Contact Number:	07-3348054
Address:			

Registration :-

Symptoms:	fever	Add to Queue
Allergy:		Medical History

Entry by:-

Staff Name:	Ramah Alip	Staff ID:	S002
-------------	------------	-----------	------

25/09/2003 11:05 PM

Patient Registration [X]

File Edit

Personal Information:-

Patient Name: Surya Omar PatientID: P010

I/C No. /Birth Certificate: 800604-01-5013 Gender: Female

Address: 21 Jln U Enter Patient ID to Search OK Cancel

Contact No (Home): 07-3348 Age: 23

Relative Information:-

Relative: Omar Ali Number: 07-3348054

Address:

Registration :-

Symptoms: fever Add to Queue

Allergy: Medical History

Entry by:-

Staff Name: Rsmah Alip Staff ID: S002

Patients Record

25/09/2003 11:05 PM

Patient Registration [X]

File Edit

Print

Printer Name: Canon BJC-1000SP Properties...

Status: Default printer: Ready

Type: Canon BJC-1000SP

Where: LPT1:

Comment: ☐ Print to file

Print range

☒ All

☐ Pages from: to:

☐ Selection:

Copies

Number of copies: 1

☐ Collate

OK Cancel

PatientID: P010

Gender: Female

Age: 23

(Handset): 019-7639775

Contact Number: 07-3348054

Registration :-

Symptoms: fever Add to Queue

Allergy: Medical History

Entry by:-

Staff Name: Rsmah Alip Staff ID: S002

Patients Record

25/09/2003 11:05 PM

8) There are also a report module such as List of Queue and MC Letter.

Report - List of Queue

Patient Name:

Priority Status:

Time Arrival:

Add Patient List

Save List

Information

1 - Serious

2 - Half Serious

3 - Half Non-Serious

4 - Non Serious

Patients List:

Salina Ahmad	1	11:11
Chua Zhe Lei	1	11:11

Patient Priority List

				Treatment Start	Treatment End
Salina Ahmad	1	11:11	Dr Lau Ming Shi	11:11	11:41
Chua Zhe Lei	1	11:11	Dr Idrizah Salleh	11:11	11:41

Alert


For every 10 patient in the list. Please, print the document ! (For future reference later).

Print List

Main Menu

MC Letter

Zoom 75%



Klinik Maimunah

Lot 123, Landmark Mall, Putrajaya

Phone : 03 - 7777 8888

Fax : 03 - 8888 7777

MEDICAL CERTIFICATE LETTER (MC)

Hereby this, I have examined this patient

and I/C Number _____ (He / she) is under my treatment

_____ to _____

I certified that (he / she) is suffering from _____

I am in opinion that (he / she) is not (physically / mentally) incapable to

(work / school) from _____ till _____

Thank you.

Yours truly,

Pages: 1

116

APPENDIX B – SOURCE CODE

1) Registration form

```
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal  
hWnd As Long, ByVal wParam As Long, ByVal lParam As  
Any) As Long  
Const EM_UNDO = &HC7  
Private Declare Function OSWinHelp% Lib "user32" Alias "WinHelpA" (ByVal  
hWnd&, ByVal HelpFile$, ByVal wCommand%, dwData As Any)  
  
Private Sub CommandMedHist_Click()  
    Unload Me  
    MedicalHist.Show  
End Sub  
  
Private Sub CommandQueue_Click()  
    Unload Me  
    FormQ.Show  
End Sub  
  
Private Sub delete_Click()  
    MsgBox ("Delete this record?"), vbYesNo + vbExclamation, "ECS"  
    Adodc1.Recordset.delete  
    Adodc1.Recordset.MoveNext  
    TextPatientName.SetFocus  
End Sub  
  
Private Sub Form_Load()  
    Me.Top = (Screen.Height - Me.Height) / 2  
    Me.Left = (Screen.Width - Me.Width) / 2  
End Sub  
  
Private Sub mmenu_Click()  
    Unload Me  
    MMenu2.Show  
End Sub  
  
Private Sub new_Click()  
    Adodc1.Recordset.AddNew  
    TextPatientName.SetFocus  
End Sub  
  
Private Sub print_Click()  
    On Error Resume Next  
    With CommonDialog1  
        ' Prepare to print using the Printer object.  
        .PrinterDefault = True  
        ' Disable printing to file and individual page printing.
```

```

.Flags = cdlPDDisablePrintToFile Or cdlPDNoPageNums
If TextPatientName.SelLength = 0 Then
    ' Hide Selection button if there is no selected text.
    .Flags = .Flags Or cdlPDNoSelection
Else
    ' Else enable the Selection button and make it the default
    ' choice.
    .Flags = .Flags Or cdlPDSelection
End If
' We need to know whether the user decided to print.
.CancelError = True
.ShowPrinter
If Err = 0 Then
    If .Flags And cdlPDSelection Then
        Printer.Print TextPatientName.SelText
    Else
        Printer.Print TextPatientName.Text
    End If
End If
End With

End Sub

Private Sub save_Click()
    MsgBox ("Save this record?"), vbYesNo + vbQuestion, "ECS"
    Adodc1.Recordset.save
End Sub

Private Sub search_Click()
    sstr = InputBox("Enter Patient ID to Search")
    txtPatientID.SetFocus
    If sstr = "" Then
        Exit Sub
    Else
        lblStatus = "Search Results for " & sstr
        Adodc1.Recordset.Find "PatientID=" & sstr & ""
    End If
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)

On Error Resume Next
Select Case Button.Key

    Case "New"
        Adodc1.Recordset.AddNew
        TextPatientName.SetFocus

    Case "Save"
        MsgBox ("Save this record?"), vbYesNo + vbQuestion, "ECS"

```


Adodc1.Recordset.save

Case "Delete"

```
MsgBox ("Delete this record?"), vbYesNo + vbExclamation, "ECS"
Adodc1.Recordset.delete
Adodc1.Recordset.MoveNext
TextPatientName.SetFocus
```

Case "Print"

```
On Error Resume Next
With CommonDialog1
' Prepare to print using the Printer object.
.PrinterDefault = True
' Disable printing to file and individual page printing.
.Flags = cdIPDDisablePrintToFile Or cdIPDNoPageNums
If TextPatientName.SelLength = 0 Then
' Hide Selection button if there is no selected text.
.Flags = .Flags Or cdIPDNoSelection
Else
' Else enable the Selection button and make it the default
' choice.
.Flags = .Flags Or cdIPDSelection
End If
' We need to know whether the user decided to print.
.CancelError = True
.ShowPrinter
If Err = 0 Then
If .Flags And cdIPDSelection Then
Printer.Print TextPatientName.SelText
Else
Printer.Print TextPatientName.Text
End If
End If
End With
```

Case "Search"

```
sstr = InputBox("Enter Patient ID to Search")
txtPatientID.SetFocus
If sstr = "" Then
Exit Sub
Else
lblStatus = "Search Results for " & sstr
Adodc1.Recordset.Find "PatientID=" & sstr & ""
End If
```

End Select

End Sub

2) List of Queue

We assume that a doctor spends

30 minutes with a patient whose priority is 1

25 minutes with a patient whose priority is 2

15 minutes with a patient whose priority is 3

10 minutes with a patient whose priority is 4

Option Explicit

Private Type Patients 'Define Patient Type

SN As String 'name

PS As Integer 'Priority Status

TA As Double 'Time Arrival

End Type

Dim ERPats() As Patients 'Dynamic Patient Array

Dim SortTA As Variant 'Dynamic Index Sort Array on Time Arrival

Dim pAdd As Long 'Main index counter

Dim TimeInMins As Single 'Conversion of hours+minutes to minutes

Dim FileNum As Integer 'Next free file number

Dim appPath As String 'Application Path

Dim DataName As String 'Data FileName

Dim ErrMsg(4) As String 'Array of Error messages

Dim ListChanged As Boolean 'Assign True when the Patients Array changes

Private Sub CommandMMenu_Click()

Unload Me

MMenu2.Show

End Sub

Private Sub CommandPrint_Click()

On Error Resume Next

With CommonDialog1

' Prepare to print using the Printer object.

.PrinterDefault = True

' Disable printing to file and individual page printing.

.Flags = cdlPDDisablePrintToFile Or cdlPDNoPageNums

If TxtName.SelLength = 0 Then

' Hide Selection button if there is no selected text.

.Flags = .Flags Or cdlPDNoSelection

Else

' Else enable the Selection button and make it the default


```

' choice.
.Flags = .Flags Or cdIPDSelection
End If
' We need to know whether the user decided to print.
.CancelError = True
.ShowPrinter
If Err = 0 Then
    If .Flags And cdIPDSelection Then
        Printer.Print TxtName.SelText
    Else
        Printer.Print TxtName.Text
    End If
End If
End With

End Sub

Private Sub Form_Load()
    'Centre the form
    Me.Top = (Screen.Height - Me.Height) / 2
    Me.Left = (Screen.Width - Me.Width) / 2
    'Error messages
    ErrMsg(1) = "Name": ErrMsg(2) = "Priority": ErrMsg(3) = "Hour": ErrMsg(4) =
"Minute"
    'Deal with 'No data file Present' Scenario
    On Error Resume Next
    If FileLen(appPath + DataName) Then Call LoadFile
End Sub

Private Sub mnuFileExit_Click()
    Unload Me 'Close the application
    MMenu2.Show
End Sub

Private Sub txtName_KeyPress(KeyAscii As Integer)
    'On Return Key move focus to next TextBox
    If KeyAscii = 13 Then txtPriority.SetFocus
End Sub

Private Sub txtName_LostFocus()
    'On Focus lost ensure first character of name is a Capital
    TxtName = UCase(Mid(TxtName, 1, 1)) + Mid(TxtName, 2, Len(TxtName))
End Sub

Private Sub txtPriority_KeyPress(KeyAscii As Integer)
    'On Return Key move focus to next TextBox
    If KeyAscii = 13 Then TxtHrs.SetFocus
End Sub

```

```
Private Sub txtHrs_KeyPress(KeyAscii As Integer)
```

```
    'On Return Key move focus to next TextBox
```

```
    If KeyAscii = 13 Then txtMins.SetFocus
```

```
End Sub
```

```
Private Sub txtMins_KeyPress(KeyAscii As Integer)
```

```
    'On Return Key Add to Lists
```

```
    If KeyAscii = 13 Then Call cmdAdd_Click
```

```
End Sub
```

```
Private Sub cmdAdd_Click()
```

```
    Dim anyErrors As Integer 'Return value from Errors function
```

```
    anyErrors = Errors 'Ascertain an error value
```

```
    If anyErrors > 0 Then
```

```
        'Notify user of error on exit the sub
```

```
        MsgBox ErrMsg(anyErrors) + " Field Incorrect", vbCritical, "ECS"
```

```
        Exit Sub
```

```
    End If
```

```
    TimeInMins = (Val(TxtHrs) * 60) + Val(txtMins) 'Calculate Arrival Time in minutes
```

```
    pAdd = pAdd + 1 'Increment main index counter
```

```
    ReDim Preserve ERPats(pAdd) 'Increase array size and preserve the existing index
```

```
    With ERPats(pAdd) 'Assign the text fields to the properties of the Type
```

```
        .SN = TxtName 'name
```

```
        .PS = Val(txtPriority) 'Priority Status
```

```
        .TA = TimeInMins 'Arrival Time (In Minutes)
```

```
    End With
```

```
    cmdSave.Enabled = True 'Allow user to Save changes to the file
```

```
    TxtName = "": txtPriority = "": TxtHrs = "": txtMins = "" 'Set textboxes to Null
```

```
    TxtName.SetFocus 'Set focus to Surname textbox ready for next addition
```

```
    Call PopPatientList 'Update Patient List
```

```
    Call PopPriorityList 'Update Priority List
```

```
    ListChanged = True 'Set to record that a change has taken place
```

```
End Sub
```

```
Private Sub cmdSave_Click()
```

```
    Dim loopER As Long 'Main Index loop counter
```

```
    FileNum = FreeFile 'Get next File number
```

```
    'Save the contents of the Patient Array
```

```
    Open appPath + DataName For Output As FileNum
```

```
    For loopER = 1 To pAdd
```

```
        Write #FileNum, ERPats(loopER).SN, ERPats(loopER).PS,
```

```
ERPats(loopER).TA
```

```
    Next loopER
```

```
    Close FileNum
```

```
    cmdSave.Enabled = False 'All changes saved to file so disable control
```

```
    ListChanged = False 'Set to record that no change has taken place
```


End Sub

Private Function Errors() As Integer

 'User Input Error Capture Function

 If TxtName = "" Then Errors = 1: TxtName.SetFocus: Exit Function

 If Val(txtPriority) < 1 Or Val(txtPriority) > 4 Then

 Errors = 2: txtPriority = "": txtPriority.SetFocus: Exit Function

 End If

 Dim IsNumber As Boolean 'Check if number

 IsNumber = IsNumeric(TxtHrs)

 'If it's not a number or number is out of range then raise Error

 If Not IsNumber Or Val(TxtHrs) < 0 Or Val(TxtHrs) > 23 Then

 Errors = 3: TxtHrs = "": TxtHrs.SetFocus: Exit Function

 End If

 IsNumber = IsNumeric(txtMins) 'Check if number

 'If its not a number or number is out of range then raise Error

 If Not IsNumber Or Val(txtMins) < 0 Or Val(txtMins) > 59 Then

 Errors = 4: txtMins = "": txtMins.SetFocus: Exit Function

 End If

End Function

Private Sub LoadFile()

 FileNum = FreeFile 'Get next File number

 'Load the contents of the data file into the Patient Array

 Open appPath + DataName For Input As FileNum

 While Not EOF(FileNum)

 pAdd = pAdd + 1 'Increment main index array counter

 ReDim Preserve ERPats(pAdd) 'Increase array and preserve the existing

index

 Input #FileNum, ERPats(pAdd).SN, ERPats(pAdd).PS, ERPats(pAdd).TA

 Wend

 Close FileNum

 Call PopPatientList 'Update the Patient List

 Call PopPriorityList 'Update the Priority List

End Sub

Private Sub PopPatientList()

 Dim loopER As Long 'Main Index loop counter

 Dim joinData As String 'Concatenation of list Data

 Dim TabSN As String 'name Padding spaces

 lstPList.Clear 'Clear the Patient List

 For loopER = 1 To UBound(ERPats)

 TabSN = String(60 - Len(ERPats(loopER).SN), Chr\$(32)) 'name + padding

 joinData = ERPats(loopER).SN + TabSN + Format(ERPats(loopER).PS) + _

 vbTab + Format(Int(ERPats(loopER).TA / 60), "0#") + ":" _

 + Format(ERPats(loopER).TA Mod 60, "0#")

 lstPList.AddItem joinData

 Next loopER

 'Highlight the last item in the list

 If lstPList.ListCount - 1 > 0 Then lstPList.Selected(lstPList.ListCount - 1) = True

End Sub

Private Sub PopPriorityList()

SortTA = ArriveTimeSort() 'Acquire the Sort Time Arrival Index

Dim loopER As Long 'Main Index loop counter

Dim joinData As String 'Concatenation of list Data

Dim DrA As String 'First Doctor

Dim DrB As String 'Second Doctor

Dim dr1 As Single 'First Doctor's Patient/Time duration

Dim dr2 As Single 'Second Doctor's Patient/Time duration

Dim timeP(4) As Integer 'Priority Time Array

Dim TabSN As String 'Surname Padding spaces

lstPQ.Clear 'Clear the Priority List

DrA = "Dr Lau Ming Shi": DrB = "Dr Idrizah Salleh" 'Assign Doctor's names

timeP(1) = 30: timeP(2) = 25: timeP(3) = 15: timeP(4) = 10 'Assign Priority Times

dr1 = 330: dr2 = 360 'Doctor's Start Times

For loopER = 1 To UBound(ERPats) 'Loop through the main index array

TabSN = String(30 - Len(ERPats(SortTA(loopER)).SN), Chr\$(32)) 'Surname + padding

If dr1 <= dr2 Then 'Doctor Lau becomes free before or at the same time as Dr Nur

If ERPats(SortTA(loopER)).TA > dr1 Then dr1 =

ERPats(SortTA(loopER)).TA

joinData = ERPats(SortTA(loopER)).SN + vbTab + vbTab +

Format(ERPats(SortTA(loopER)).PS) + _

vbTab + Format(Int(ERPats(SortTA(loopER)).TA / 60), "0#") + ":" + _

+ Format(ERPats(SortTA(loopER)).TA Mod 60, "0#") + _

vbTab + vbTab + DrA + vbTab + vbTab + vbTab + Format(Int(dr1 / 60),

"0#") + ":" + _

Format(dr1 Mod 60, "0#") + vbTab + vbTab

dr1 = dr1 + timeP(ERPats(SortTA(loopER)).PS) 'Record Time spent with

Patient

joinData = joinData + Format(Int(dr1 / 60), "0#") + ":" + Format(dr1 Mod 60,

"0#")

lstPQ.AddItem joinData 'Add to Priority List

Else 'Doctor Nur becomes free before Dr Lau

If ERPats(SortTA(loopER)).TA > dr2 Then dr2 =

ERPats(SortTA(loopER)).TA

joinData = ERPats(SortTA(loopER)).SN + vbTab + vbTab +

Format(ERPats(SortTA(loopER)).PS) + _

vbTab + Format(Int(ERPats(SortTA(loopER)).TA / 60), "0#") + ":" + _

+ Format(ERPats(SortTA(loopER)).TA Mod 60, "0#") + _

vbTab + vbTab + DrB + vbTab + vbTab + vbTab + Format(Int(dr2 / 60),

"0#") + ":" + _

Format(dr2 Mod 60, "0#") + vbTab + vbTab

dr2 = dr2 + timeP(ERPats(SortTA(loopER)).PS) 'Record Time spent with

Patient

joinData = joinData + Format(Int(dr2 / 60), "0#") + ":" + Format(dr2 Mod 60,

"0#")

lstPQ.AddItem joinData 'Add to Priority List


```

    End If
    Next loopER
End Sub

Private Function ArriveTimeSort() As Variant
    'Returns an index of pointers To the array
    Dim OuterLoop As Long    'Outside Loop
    Dim InnerLoop As Long    'Inside Loop
    Dim tempIndex() As Long  'Temporary dynamic array
    Dim LB As Long           'Lower bounds of Patient Array
    Dim UB As Long           'Upper bounds of Patient Array

    LB = LBound(ERPats) + 1 'Assign Lower Bound Base 1
    UB = UBound(ERPats)     'Assign Upper Bound

    ReDim tempIndex(UB)      'Increase array size
    For InnerLoop = LB To UB
        tempIndex(InnerLoop) = InnerLoop 'Assign values to temporary array index
    Next

    For OuterLoop = UB To LB Step -1 'Step backwards through array
        For InnerLoop = LB + 1 To OuterLoop 'Step forwards through array
            'Compare Time Arrivals and force higher times to the OuterLoop array index
            If ERPats(tempIndex(InnerLoop - 1)).TA >
ERPats(tempIndex(OuterLoop)).TA Then
                Swap tempIndex(InnerLoop - 1), tempIndex(OuterLoop)
            End If
        Next InnerLoop
    Next OuterLoop

    ArriveTimeSort = tempIndex() 'Return Sorted index of pointers
End Function

Private Sub Swap(ByRef firstValue As Long, ByRef secondValue As Long)
    'Swaps the First value with the Second Value using an intermediate variable
    Dim tmpValue As Variant
    tmpValue = firstValue
    firstValue = secondValue
    secondValue = tmpValue
End Sub

Private Sub lstPList_DblClick()
    Dim response As Integer 'Return value indicating which button the user clicked.
    Dim tempIndex As Integer 'Record the index item of the Patient ListBox
    'Ask User for confirmation of a delete operation
    response = MsgBox("Delete Patient?", vbExclamation + vbYesNo, "ECS")
    If response = 6 Then

```

```

tempIndex = lstPList.ListIndex + 1 'Add one because listbox index array is base
0
'Reposition the array contents from the position of the deleted item
For response = tempIndex To UBound(ERPats) - 1
    ERPats(response) = ERPats(response + 1)
Next response
'Tidy up the array and release the data held in the deleted index position
ERPats(response).SN = "": ERPats(response).PS = Empty:
ERPats(response).TA = Empty
ReDim Preserve ERPats(response - 1) 'Decrease the Patient Array
pAdd = pAdd - 1 'Decrease the Patient counter
Call PopPatientList 'Re-populate the Patient list
'Highlight the next indexed item in the Patient list
If UBound(ERPats) > tempIndex - 1 Then lstPList.Selected(tempIndex - 1) =
True
    Call PopPriorityList 'Re-populate the Priority list
    cmdSave.Enabled = True 'Allow user to Save changes to the file
    ListChanged = True 'Set to record that a change has taken place
End If
End Sub

```