UNIVERSITI
TEKNOLOGI
PETRONAS

Multiclient Validation System with Multiboot Image Support Using Preboot Execution Environment (PXE) Technology

by

Ngee Tze Shik

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
Electrical & Electronics Engineering

MAY 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Multiclient Validation System with Multiboot Image Support Using Preboot Execution Environment (PXE) Technology

by

Ngee Tze Shik

A project dissertation submitted to be the
Electrical and Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,

[illegible]

Mr. Noohul Basheer Zain Ali

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
May 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

NGEE TZE SHIK

ABSTRACT

A software validation is a confirmation by examination and provisions of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled. In practise, most of the software validation are done manually, which is not time efficient and hardly to achieve accurate results, especially involving multiple operating system environments. Therefore, the idea of remote boot validation test using Preboot Execution Environment (PXE) protocol is evolved.

The objective of this project is basically to design, built and test a Multiclient Validation System with Multiboot Image Support by using PXE protocol. The system would eventually allow client machines to perform validation tests automatically in the different operating system environment. This validation system gives a complete routine check on the compatibility and errors of the software to hardware devices.

The scope of this project is planned to ensure the feasibility of the project to be carried out within the given time frame. This project can be broken down to two major stages. The first stage will involve preliminary literature review, design and built a single client validation system to understand the operation of multiboot image support in PXE protocol. Comprehension of system development language like Visual C and JScript is vital since the system coding would be done entirely using those languages. It is envisaged that the major part of the system is designed, built and tested during the first half of the project duration (first semester). The second major stage is to implement the multiclient support feature for the validation system.

ACKNOWLEDGEMENT

The a uthor would like to take this opportunity t o e xpress h is g ratitude a nd t hank several parties who have facilitated him at one stage or another, through the process of designing and building the project.

- The author's family and friends, for giving him the discipline, encouraging him, and bearing with him, during this rather difficult and time consuming project.

- Mr. Zuki, UTP, the coordinator of Final Year Project, for his guidance and dedication in handling and making this Final Year Project a successful one.

- Mr. Noohul Basheer Zain Ali, UTP, for sparing much of his time to supervise the author throughout the length of the project.

- Mr. Seth Nair, Remesh, Intel Technology (M) Sdn. Bhd., for giving the author a golden opportunity participate a multinational company project. His guidance and support are crucial to this project.

- Mr. Musa, UTP, for providing the necessary assistance during several phase of project.

- Universiti Teknologi PETRONAS, for providing the author with the necessary foundation and resources to embark on this project.

ABBREVIATIONS AND NOMENCLATURES

PXE	-	Preboot Execution Environment
PDK	-	Product Development Kit
NIC	-	Network Interface Card
WfM	-	Wired for Management
TCP/IP	-	Transmission Control Protocol / Internet Protocol
DHCP	-	Dynamic Host Configuration Protocol
TFTP	-	Trivial File Transfer Protocol
BIOS	-	Basic Input Output System
BBS	-	BIOS Boot Specification
NSB	-	Network Service Boot
HDD	-	Hard Disk Drive
PXE PDK	-	PXE Product Development Kit
API	-	Application Program Interface
DOS	-	Diskette Operating System
MBR	-	Master Boot Record
OS	-	Operating System
UNIDI	-	Universal Network Interface
WSH	-	Window Script Host
POST	-	Power On Self Test
CHS	-	Cylindrical, Head, Sector
TCO	-	Total Cost Ownership
NBP	-	Network Bootstrap Program

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

APPENDIXES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

The primary goal of software validation is to demonstrate that the completed software e nd p roduct complies with e stablished s oftware a nd s ystem requirement. The correctness and completeness of the system requirements should be addressed as part of the design validation process for the device. Software validation is the confirmation that all software requirements have been met and that all software requirements are traceable to the system requirements. Software validation is a required component of the design validation of a critical device. Whereas the software validation conforms that all software requirements have been met, the design validation goes further to confirm that the c ritical device i tself m eets user needs and intended uses.

1.1 Background of Study

This project is generally a subset of a software validation life cycle for validating computer graphic driver. This project will provides evidences that all graphic drivers' requirements have been implemented correctly and completely and are traceable to system requirements. Compatibility and functionality of the graphic driver in different operating system environment and computer architecture will be tested. A conclusion that graphic driver is validated is high dependent upon comprehensive software testing, i nspections, analyses, a nd other v erification tasks performed at each stage of the software validation life cycle. In this project, testing of software functionality in a simulated use environment is typically control by an embedded technology that store in server and network interface card (NIC) namely, Preboot Execution Environment (PXE). Besides monitoring validation testing, PXE also responsible for setting the active partitions in hard disk and provide multiclient support when multiple testing platforms are connected to the server for validation purpose. Having done this, the consistency, completeness and correctness of the graphic driver could be studied and subsequently used as the evidence to conform that the validation output meets all of the specific requirements.

1.2 Problem Statement

1.2.1 Problem Identification

Eventually, software validation is a matter of developing a "level confidence" that the system meets all requirements and user expectations for the software automated functions and features of the system. There is a common challenge in developing the level confidence in software validation process, where most of the validation processes are manually performed. The level of confidence will vary depending on the validation engineer's full concentration and observation during the validation process, especially in detecting and analysing the defects found in specifications documents, estimates of defects remaining, testing coverage, and others. These unnecessary human errors will affect the acceptable level of confidence indirectly before the software is being released.

Situation becomes worse when multiple operating system environments are involved. Extra time and resources need to spend on setting up the numerous of testing platforms with different computer architecture and operating systems. Because of its complexity, the validation process for software is even more tightly controlled than for hardware, in order to prevent problems that cannot be easily detected later in the development process. For these and other reasons, software engineering needs an even greater level of managerial skill and control than hardware engineering.

In order to solve all the problems that has been mentioned earlier, a client-server interaction system need to be designed to detect the defects automatically during the validation process automatically. PXE protocol is used as the vital solution, which acts as the communication link between server and testing platforms. Therefore it is a need to understand the PXE protocol in terms of its operation and requirements on individual testing platforms. The knowledge of the behaviour and pattern of PXE protocol in response to individual testing platform would subsequently enable inter-connection of various testing platform to the server without any protocol conflict.

Once the multiple testing platforms are connected to the PXE server, validation tests will be executed in first operating system environment automatically. After the tests are being performed, PXE protocol will switch the testing platform into the next active partition, where different operating system will be loaded. The same validation test is repeated in the newly loaded operating system environment until all the operating systems have been tested out. This multiclient validation system is able to report errors and defects that have been detected along the validation process. This feature will enhance the compatibility and stability of the designed software.

1.2.2 Significant of the project

Software validation includes confirmation of conformance to all software specifications and confirmation that all software requirements are traceable to the system specifications. Confirmation is an important part of the design validation to ensure that all aspects of the device conform to user needs and intended uses. All of the confirmation can be achieved with the output of this project. Towards the end of the semester, a Multiclient Validation System prototype that could handle multiboot images support on different testing platforms will be completed. The core of the Multiclient Validation System prototype would be a Windows NT4 server with PXE protocol, which is basically the required communication protocol that automatically executes the validation tests and active partition changing process. This system can increase the usability and reliability of the device, resulting in decreased failure rates, fewer recalls and corrective actions, less risk to users, and reduced liability to manufacturers. It can also reduce long-term costs by making it easier and less costly to reliable modify software and revalidate software changes.

1.3 Objective and Scope of Study

1.3.1 Relevancy of the project

The final year project would require students to put what they have learned in their past 4 years in UTP into a practice. First and foremost is the network communication, and followed by software programming. The details pertaining to the computer network would be picked up later in the course of Data and Computer Communication and Computer System Architecture. Nevertheless, the early exposure would prove advantageous and the courses later would serve to further enhance students' understanding and knowledge on computer networking. Examples of knowledge that student would require and have picked up in their university includes: Data and Computer Networks, Software Engineering, Introduction to C++ Programming as well as Computer System Architecture.

Then there are also JScript, Batch Script, C programming and network communication interface on hands experience gained towards the end of the project. Basically, most of the courses are not yet to be offered in UTP but it would be good to be able to pick up these skills. For scripting and computer network interface, student would basically need to start from the basic as there is no prior experience in that area.

At the completion of this project, student is expected to achieve the following objectives:

I. To develop a multiclient validation system with PXE protocol in multiboot image support.

II. To develop a communication interface, which handle the file sending and receiving processes in between the PXE server and testing platforms.

III. To develop real life applications that can handle the software validation and partition changing application.

1.3.2 Feasibility of the project within the Scope and Timeframe

With reference to the Gantt chart created for this project, the suggested time allocation should be sufficient. The preliminary portion of the project focuses on the acquirement of knowledge and skills which are vital to kick start the research. The later stages of the project would target primarily on building up the skills through practical work and familiarization of the tools and software to be used in the design stage. Overall, the progress as projected in the Gantt chart seems realistic and achievable taking into consideration the amount of work to be done in the specified timeframe.

Towards the completion of the multiclient support sub-system is the main objective for this project. Additional work required to improve the project is included under "Chapter 5 Results and Discussion".

CHAPTER 2

LITERATURE REVIEW AND THEORY

2. LITERATURE REVIEW & THEORY

Almost every corporate PC purchased since 1998 is "Wired for Management" (WfM) compliant. WfM is an industry standard, initiated by Intel to improve the manageability of client PC systems, and is part of the Intel and Microsoft PC98 specification. The Preboot Execution Environment (PXE) is part of the WfM specification. PXE is defined on a foundation of industry-standard Internet protocols and services that are widely deployed in the industry, namely Transfer Control Protocol / Internet Protocol (TCPIP), Dynamic Host Configuration Protocol (DHCP), and Trivial File Transfer Protocol (TFTP). These standardize the form of the interactions between clients and servers.

2.1 Overview of Multiclient Validation System

The Multiclient Validation System is a robust network connection of testing platforms and server that serve as one of sophisticated tools for validation engineers to accomplish their validities goals in software based graphic driver design. The unique Intel remote-boot solution provided by PXE protocol allows a testing platform to boot up its multiple operating systems automatically from the different active partition in its local hard disk. This is a cost-effective, robust and easy-to-manage network solution to develop a “level confidence” that the software meets all requirements and user expectations for the software automated functions and features of the system.

In this system, each testing platform contains a Network Boot Read Only Memory (ROM), which integrating PXE protocol into the Basic Input/Output System (BIOS). PXE have the functionality of enables the start-up and c onfiguration between t he server and the testing platforms. This is commonly known as network boot process (or remote boot process, please refer **Appendix A**). With features specifically designed to ensure the c onsistency, completeness and correctness of the designed graphic driver is being produced and tested in multiple operating systems environment, the Multiclient Validation System is a flexible, reliable and affordable means of bringing powerful computer assisted tool in the laboratory.

2.1.1 How the Remote Boot Utility Works

The main utility that used to remotely boot the testing platform from a server is the Intel® PXE Remote Boot Utility. It was initially designed by Intel, with input from several other vendors including 3Com, HP, Dell, Compaq, and Phoenix Technologies. PXE works with a network interface card (NIC) in the PC, and makes the NIC a boot device. The PXE vision is to "Make the network interface a standard, industry-accepted PC boot device." This means adding the NIC to the traditional list of standard boot devices, such as floppy drives, hard disks, and CD-ROMs, that load the operating system or set up programs on the PC. It allows the client PC to "network boot." Booting from the network opens up a vast array of management and support features.

PXE boots the client PC from the network by transferring a "boot image file" from a server. This file can be the operating system for the client PC or a Preboot agent that performs client management tasks. Since PXE is not operating system specific, the image file can load any OS. It provides support for network booting, of embedded and other operating systems. Because PXE works with the NIC, it requires a PXE-enabled NIC. Most currently available NICs do support PXE, including those from 3Com, Intel, Digital, RealTek, and SMC.

PXE is available either as a boot ROM chip that add to the NIC, or as part of the system BIOS if the network interface is on the motherboard. PXE is specific to a type of NIC; a boot ROM for one type will not work on another type of NIC. A client PC may have PXE installed, but it may not be enabled by the BIOS. Most PCs support the BIOS Boot Specification (BBS), or other methods that let user order the PC's boot devices from a BIOS setup screen. To perform a remote boot process select PXE as the first boot device in system BIOS once the system is power on.

Some BBS systems also support Network Service Boot (NSB). With NSB, user can set PXE lower in the boot order. A message, such as "Press F12 to boot from network" will appear while the PC boots. In this secure system is needed. The two main uses of network booting today are for installing an OS in a brand new client PC that has no operating system, (or re-installing in a client PC where the operating system has failed), and booting into a guaranteed "clean" system. Below is the sequence of process that takes place in remote boot process.

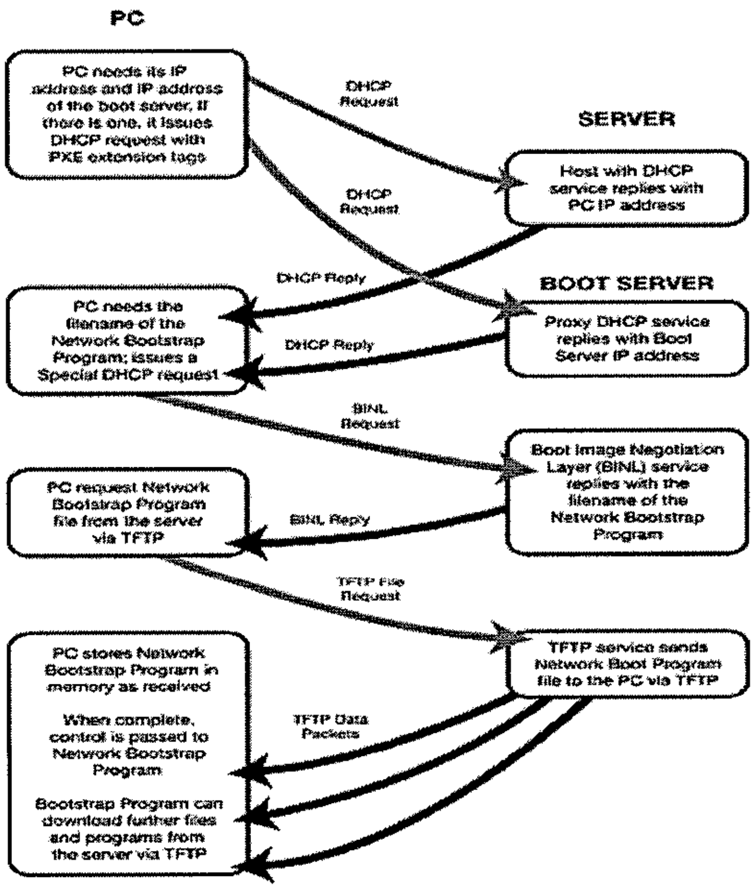

PC
PC needs its IP address and IP address of the boot server, if there is one, it issues DHCP request with PXE extension tags
DHCP Request
SERVER
Host with DHCP service replies with PC IP address
DHCP Request
DHCP Reply
BOOT SERVER
PC needs the filename of the Network Bootstrap Program; issues a Special DHCP request
DHCP Reply
Proxy DHCP service replies with Boot Server IP address
BINL Request
Boot Image Negotiation Layer (BINL) service replies with the filename of the Network Bootstrap Program
PC request Network Bootstrap Program file from the server via TFTP
BINL Reply
TFTP File Request
PC stores Network Bootstrap Program in memory as received
When complete, control is passed to Network Bootstrap Program
Bootstrap Program can download further files and programs from the server via TFTP
TFTP service sends Network Boot Program file to the PC via TFTP
TFTP Data Packets

Figure 1 Intel Remote Boot Access Process

2.1.2 Elements of the Intel Multiclient Validation System

2.1.2.1 Testing Platforms

In this system, testing platform with different Intel® architecture chipsets is used to check the compatibility of the designed graphic driver in different operating system environment. A typical block diagram of a testing platform motherboard based on Intel® 810 Chipset is shown below.

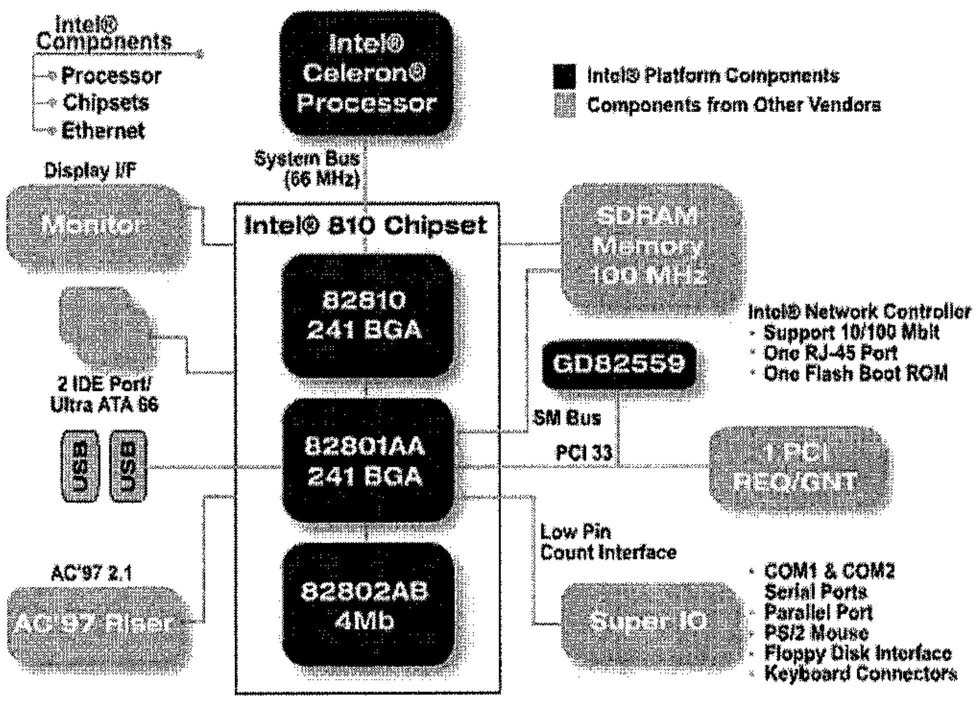


Figure 2 Intel® 810 Chipset Block Diagram

The 82559 fast Ethernet controller, with an integrated 10/100 Mbps physical layer device, is Intel's leading solution for PCI board LAN designs. This is the main controller that monitors and performs the remote boot process.

All the testing platforms are configured with 4 active partitions hard disk drives (HDD), where 4 different operating systems have been installed. As a result, they are robust, cost-effective, have fewer points of failure, and require less maintenance, yet do not sacrifice the performance and power necessary for validation engineers to achieve the benefits o f c omputer-assisted v alidation process in different c omputer architecture platforms. The configuration takes up less testing platforms in the laboratory and is not subject to insertion additional systems found in the typical graphic driver validation laboratory.

2.1.2.2 Centralized Server

The Multiclient Validation System incorporates the concepts of shared resources and centralized server management to offer affordable, low-maintenance solutions for validation process. This design allows the deployment of cost-effective testing platforms equipped with the processing power to handle sophisticated validation tasks, but relies on the server for all other resources such as applications software and storage.

In Multiclient Validation System Windows NT 4 operating system serves as the operating s ystem on server. The validation p rocess i s made p ossible by u sing t he PXE Product Development Kit (PXE PDK)

The PXE Product Development Kit is a solution that allows validation engineer to boot testing platforms remotely with/without a hard disk drive in the Preboot Execution Environment. A bootable floppy disk needs to be configured for remote boot process. The boot disk is then converted to a boot image with the PXE Product Development Kit. The PXE Product Development Kit provides the necessary services such as the boot server, TFTP protocols and APIs to download the boot image from the server to testing platforms upon power up. The testing platform then executes the boot image locally and boots up into the different operating system environment. The testing platforms are connected to the Windows NT4 Server through a network switch and the PXE server serves as main control station for the entire testing platforms. Figure 3 shows the physical layout of the Multiclient Validation System.

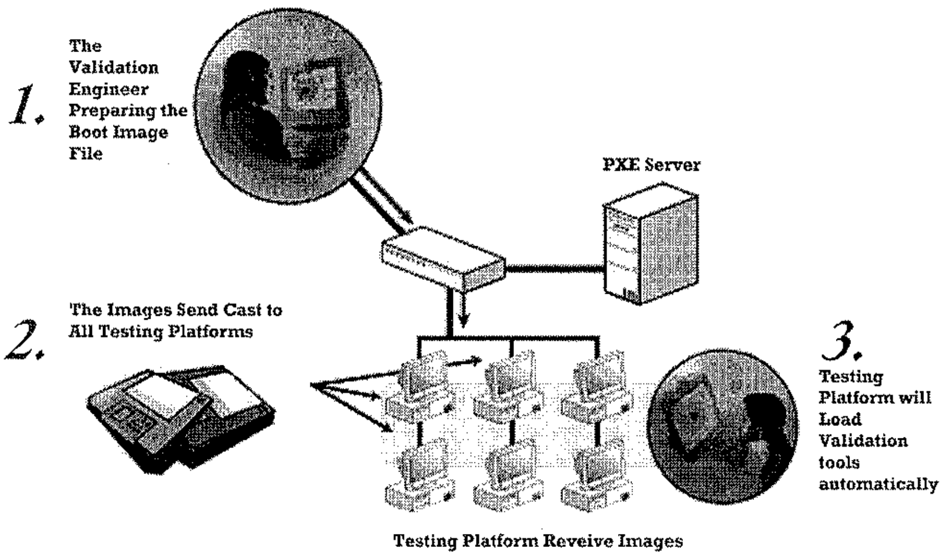

1.
The Validation Engineer Preparing the Boot Image File
PXE Server
2.
The Images Send Cast to All Testing Platforms
3.
Testing Platform will Load Validation tools automatically
Testing Platform Reveive Images

Figure 3 Multiclient Validation System Physical Layout

2.2 Technical Support for Multiclient Validation System

2.2.1 Dynamic Host Configuration Protocol (DHCP) Server

Dynamic Host Configuration Protocol (DHCP) is a TCP/IP standard for simplifying management of host IP configuration. The DHCP standard provides for the use of DHCP servers as a way to manage dynamic allocation of IP addresses and other related configuration details to DHCP-enabled clients on user network. Every computer on a TCP/IP network must have a unique computer name and IP address. The IP address (together with its related subnet mask) identifies both the host computer and the subnet to which it is attached. When user moves a computer to a different subnet, the IP address must be changed. DHCP will dynamically assign an IP address to a client from a DHCP server IP address database on local network:

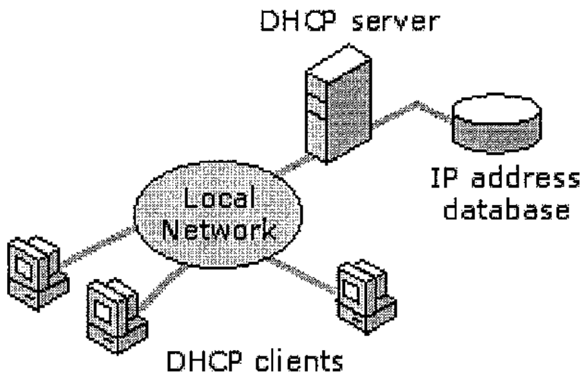


Figure 4 Server and Client Interaction in DHCP Protocol

2.3 Essential knowledge for Project Implementation

The knowledge that should pick in the course of project implementation are:

◆ Computer Networking	To understand the fundamental PXE service that provides communication link between server and testing platforms. There is also a need to understand the PXE protocol and the relationship b etween s tandard D HCP p rotocols before designing multiclient support sub-system.
◆ Visual C++ programming	To design a) Programs that able to change the hard disk's active partitions. This is crucial when the testing platform is changing from current operating system environment to the next active operating system environment. b) Program that able to update the server registry every time when the testing platform is restart. Once the registry has been updated, server will decided whether the testing platform should change to active partition or boot up with the current operating system.

	c) Edit the PXE service source code to enable multiclient support features.
♦ Java Script	This is the primary script that would be used to perform the validation tests on design graphic driver. Thus, it is vital to familiarize with the functionality of the java script.
♦ Batch Script	A batch script is being used to download and execute the partition changing application in remote boot environment.

Table 2.1: Knowledge base for the Multiclient Validation System project

CHAPTER 3

METHODOLOGY

3 METHODOLOGY & PROJECT WORK

The plan for this project is drawn across two semesters. The design and implementation phases of Multiclient Validation System are shown in Figure 5. There are basically 8 major steps, which reduces the scope for rework during system development.

3.1 Procedure Identification

3.1.1 Preliminary Investigation

The purpose of the preliminary investigation is twofold. First, it defines the perceived problems opportunities, and directives that triggered the project and access the risk of pursuing the project. Second, and assuming the project is worth looking at, the preliminary investigation phase must also establish the project charter, which defines the project scope, objective, and schedule after collecting factual information from the current users concerning the perceived problems, causes and effects. The preliminary investigation typically includes the following tasks:

a) Define problems statement.
b) Negotiate preliminary scope.
c) Plan the project.
d) Present the project and plan.

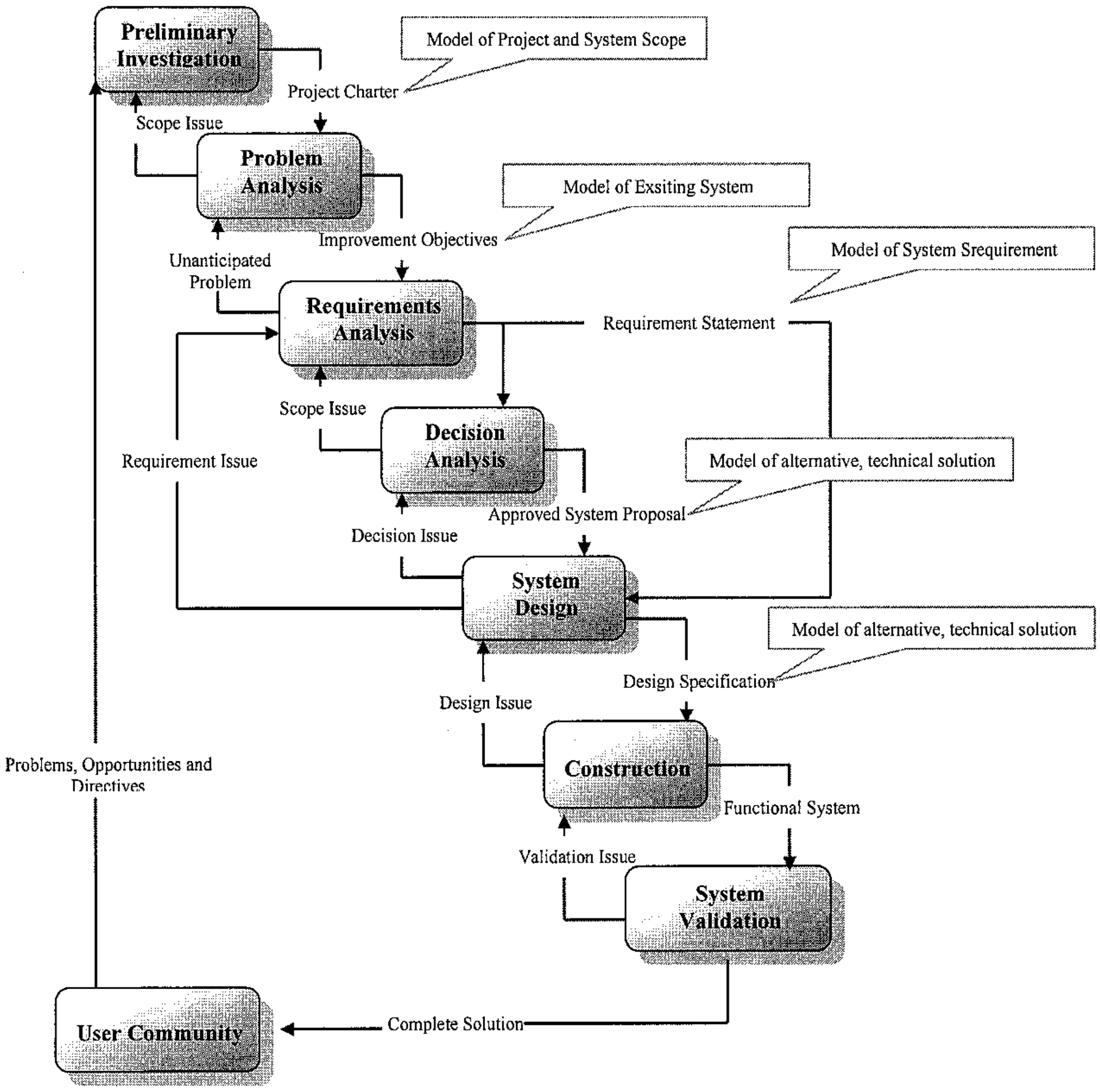


Figure 5: Multiclient Validation System Design Life Cycle

3.1.2 Problem Analysis

The problem analysis provides a more through understanding of the problems, opportunities, and directives that triggered the project. It always starts with study and analysis the existing system. The problem analysis provides a more through understanding of the problems that triggered the project. As shown in Figure 5, the key input is the project charter from preliminary investigation phase. The problem analysis phase typically includes the following tasks.

a) Study the problem domain
b) Analyze problems and opportunities.
c) Establish improvement objectives.
d) Update project plan.

The primary deliverable of the problem analysis phase is system improvement objectives. These objectives define the operational criteria on which any new system will be evaluated. For instance, one of the defined objective states the new system must perform the validation performance automatically, as mentioned in section **1.2.2 Significant of the Project.**

3.1.3 Requirement Analysis

The next phase of the methodology is to define and prioritize requirements. This is called the requirement analysis. The requirement analysis is intended to discover the requirements for the system as a whole. This is perhaps the most important phase of the methodology. The requirement analysis phase typically includes the following tasks.

a) Define requirement.
b) Analyze abstract functional requirements.
c) Trace and complete requirements.
d) Prioritize requirements.
e) Update project plan.

The deliverable for the requirement analysis phase is requirement statement. This statement is usually concentrates on deriving on these two types of requirements.

a) Abstract function requirements

The basic functions that the multiclient validation system must provide are:

i. Carry out selected validation process in different operating systems.
ii. Changing the operating system environment from one active partition to the other when finished validation process in one particular operating system environment.
iii. Support multiple testing platforms at the same time.

b) System Properties

There are non-functional emergent system properties, which are not directly concerned with the abstract functional requirement:

i. Results and error will be saved in the system log file in server to prove the validity of the designed software.
ii. The system reducing the validation cost and time consume of setting up multiple operating system platforms.

3.1.4 Decision Analysis

Given the requirement statement, there are usually numerous alternative ways to design a new validation system to fulfill those requirements. The purposes of decision analysis is to identify solutions, analyze the feasibility of the solutions from technical, operational, economics, schedule and risk aspect, and recommend the best solution to be designed.

The decision analysis phases typically include the following tasks:

a) Identify solutions.
b) Analyze solutions.
c) Compare solutions.
d) Update project plan.
e) Recommend a solution.

After completing the tasks that has been mentioned above, the decision analysis phase concludes with approved system proposal, which recommends a system solution. In this solution, there are 6 sub-systems being recommended, which govern all the functional requirements and essential specification.

a) Network Interface
b) Validation Application
c) Remote Boot Integration
d) Partition Changing Sequence
e) Booting Sequence
f) Multiclient System Support

3.1.5 System Design

The purpose of the design phase is to transform the requirement statement from the requirement analysis into design specification for construction. The main activity involved in this process is specifying design specifications. Different specifications that individually or collectively meet the requirements are identified as below:

◆ Networking Interface	A stand alone server will be setup with the following protocols to serve as client-server interaction support: a) ***TCP/IP*** - provides interaction between server and testing platform. It also assigns a static IP address to the server. b) ***DHCP*** - have capability of automatic allocation of reusable network addresses and additional configuration option. c) ***TFTP*** - service used to transfer the boot image file from the server to the client system.
◆ Validation Application	The primary script that would be used to link all the validation tools into a single application. Once this application has been, a text file will be created to inform the server. The validation application will be written in JScript.

♦ Partition Changing Application	Partition changing application will access to the Master Boot Record (MBR) directly and change the active partition into next operating system. This application will be written in C programming.
♦ Remote Boot Integration	A batch script is being used to call perform the partition changing application in remote boot system environment. After the active partition has been changed, the script will load the latest active operating system into client system.
♦ Booting Sequence	PXE Server Registry needed to be edited to specify the booting sequence for client system.
♦ Multiclient Support	Support multiple testing platforms, which connecting to the server at the same time. Make sure only the user-defined boot image is being loaded to the specific testing platform.

Table 3.1: Identified Sub-System Specification

3.1.6 Construction

Given some level of design specifications, sub systems for the design are constructed and tested. The purpose of the construction phase is twofold:

i) To build and test a system that fulfills design requirement and specifications.
ii) To implement the interfaces between new system and existing systems.

During construction phase, the identified specification has been realized by building some application programs, and user and system interfaces. Some of these components may already exist. As components are constructed, they are typically validated before demonstrate to user to solicit feedback. In this project, every specification has been developed into a single sub-system for a complete system. Figure 6 shows the steps to realize all the specifications that have been mentioned before.

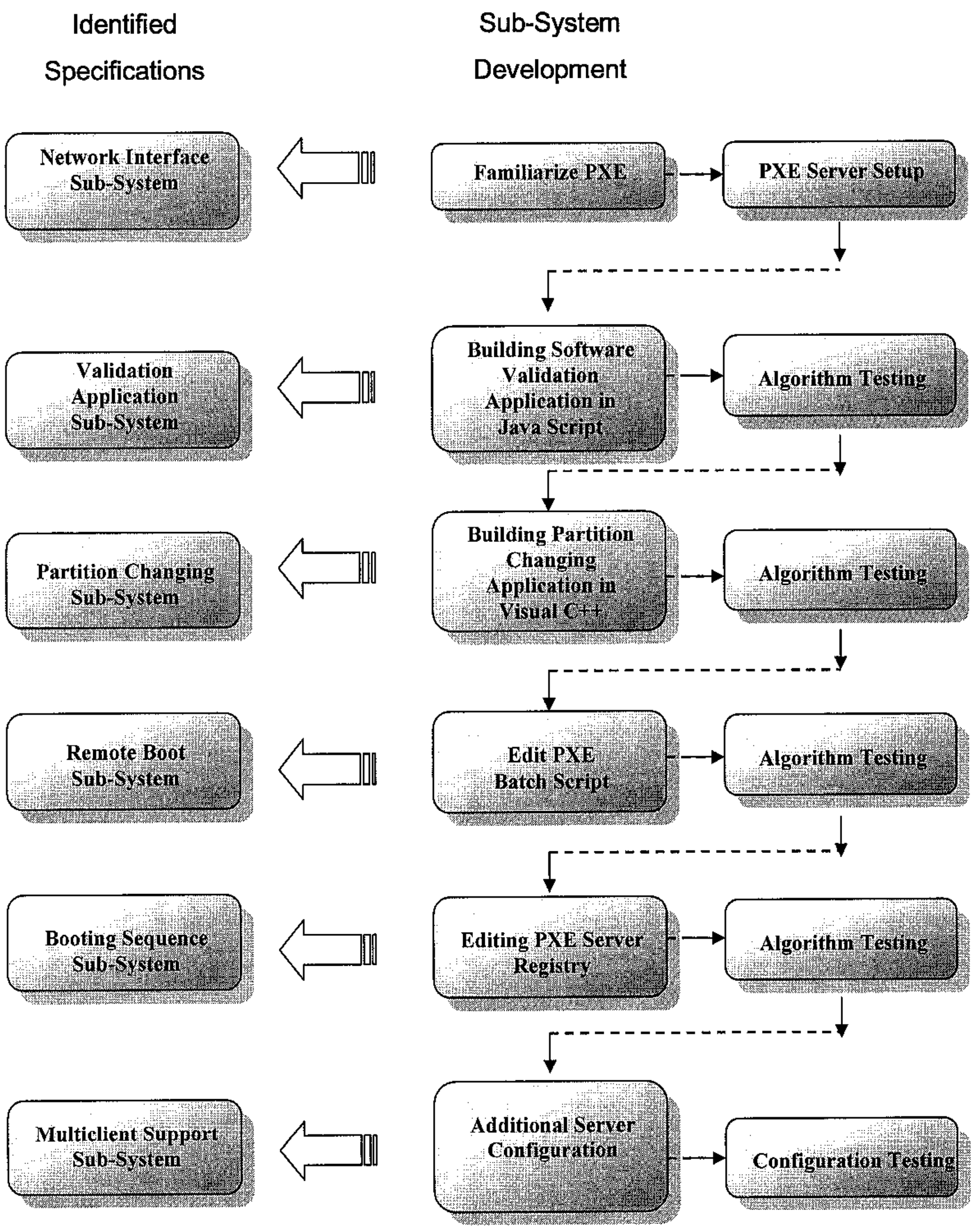

Identified Specifications
Sub-System Development
Network Interface Sub-System
Familiarize PXE
PXE Server Setup
Validation Application Sub-System
Building Software Validation Application in Java Script
Algorithm Testing
Partition Changing Sub-System
Building Partition Changing Application in Visual C++
Algorithm Testing
Remote Boot Sub-System
Edit PXE Batch Script
Algorithm Testing
Booting Sequence Sub-System
Editing PXE Server Registry
Algorithm Testing
Multiclient Support Sub-System
Additional Server Configuration
Configuration Testing

Figure 6: Construction Process

One important aspect of construction is conducting integration process, which taking independently developed sub-systems and putting them to make up a complete system. The integration process will basically do in ***incremental integration*** where all the sub-system is integrated one at a time. This incremental process is the most appropriate approach for two reasons:

i) It is usually impossible to schedule all the different sub-system development so that all development is completed at the same time.

ii) Incremental integration reduces the cost of error location. If many sub-systems are simultaneously integrated, an error that arises during testing may be located in any of these sub-systems.

While this may appear to be a simple process, but many problem arise in this stage:

i) The integration environment is not the same as the environment assumed by the developers of the system.

ii) When a single sub-system is integrated with an already working system, error might be occur and are probably in the newly integrated sub-systems or in the interactions between the existing sub-systems and the new sub-system.

3.1.7 System Validation

System validation is tended to show that this system conforms to its specification and meets the expectations that have been identified earlier. The sub-systems are integrated to make up the system. This process is concerned with finding errors that result from unanticipated interaction between sub-systems and sub-system interface problems. It is also concerned with validating that the system meets it functional and non-functional requirements and testing the emergent system properties.

3.2 Tools Required

3.3.1 Software

- Intel PXE Product Development Kits
- Intel PXE Software Development Kits
- Microsoft 32-bit C/C++* v5.00

- Microsoft 16-bit C/C++ v1.52c
- Microsoft 16-bit Assembler* v6.13
- MKS* Toolkit for Windows NT v6.1, or later

3.3.2 Hardware

- Intel InBusiness 8-Port 10/100 Fast Hub
- Linksys Preconnect 4 station KVM monitor switch.
- Intel EtherExpress PRO PCI Network Interface Cards.
- Intel 82810 Chipset Systems.

The detail layout of the system is shown in **APPENDIX F**.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 FINDINGS AND DISCUSSION

According to the design specification, the multiclient validation system is made up of 6 main sub-systems, where each sub-system is playing different role in the system. In order to verify that operational system comes out as intended in the design specification, each sub-system created is validate to make sure all the build applications are functioning well according to the project specification and can integrate with each another. The completion of each sub-system will display the overall process of the main system.

Figure 7 shows the overall system flow with PXE protocol. Initially a server with Dynamic Host Configuration Protocol (DHCP) service is network connected with testing platform through a network hub. After the testing platform is being powered on, it will automatically start the network booting process without performing the normal local boot from hard disk. This can be configuring by entering system BIOS and change the boot priority to LAN boot.

Once the testing platform is successful get into network booting process, the testing platform will request an IP address from the server. The server will assign a unique IP address to each of the requesting client. After receiving the IP address, PXE protocol will be loaded into testing platform from the server. Next, testing platform will read the boot server list and select the 1st booting sequence from the PXE boot menu. (In this case, the author select APITEST boot server as the only active boot server)

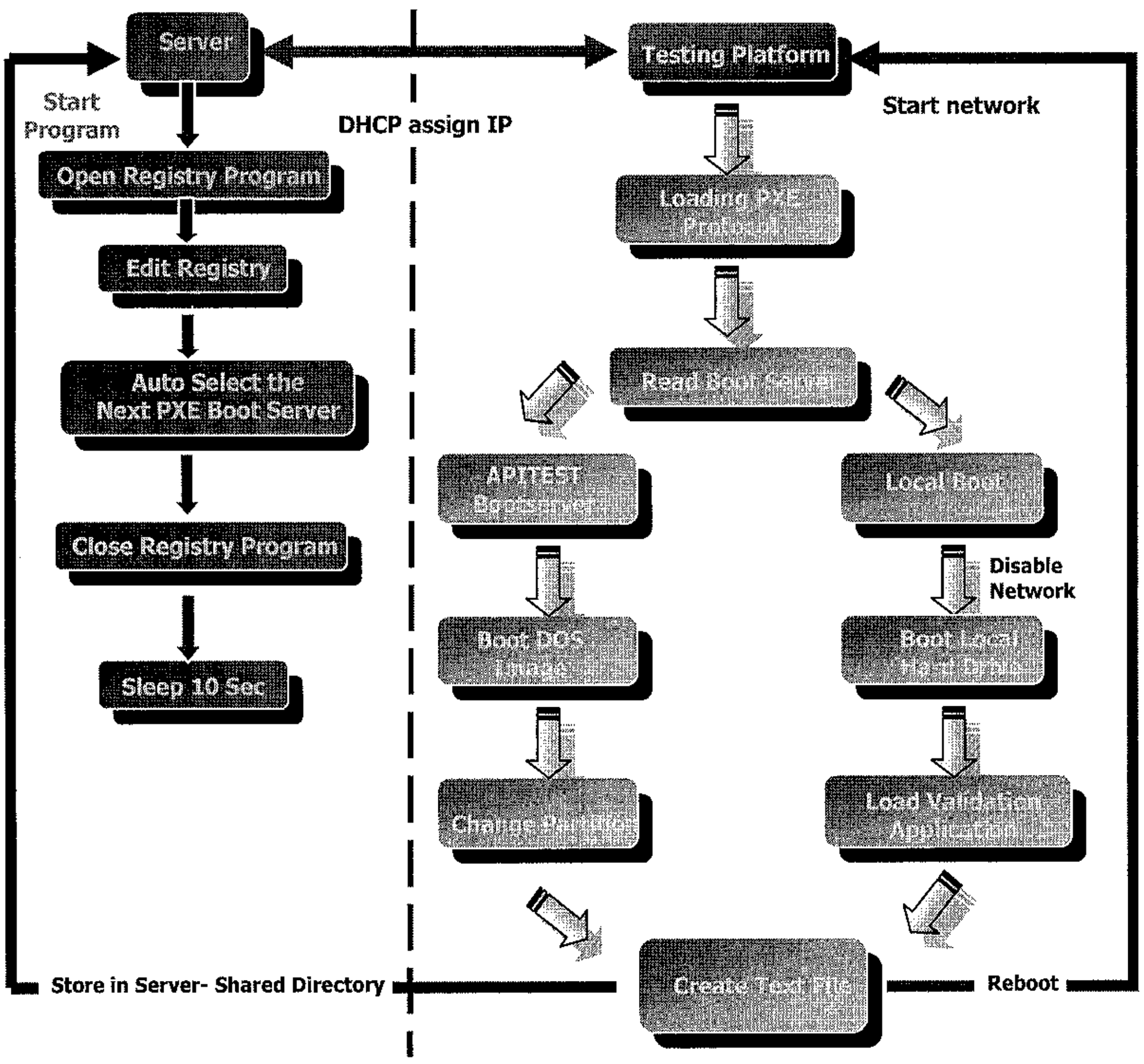


Figure 7: Remote Boot Validation Tests in PXE Environment Process Flowchart.

For first time boot, the testing platform will start with reading APITEST boot server. A boot server is channel in PXE server, where testing platform can received the name of an executable DOS image on the chosen boot server. Testing platform has been directed to download DOS image file from APITEST boot server into testing platform's memory. At this point, the testing platform state must meet certain requirement include the availability of certain areas of the testing platform's main memory, and the availability of basic network I/O services. After finished downloading, DOS image file will be executed from the memory and calling partition changing application to modify the Master Boot Record (MBR) in hardisk.

This time, the 1st active partition in the local disk is set to active status. A text file will be created and stored in the server server-shared directory. Testing platform will be rebooting and ready to load the first operating system.

After completing the partition changing process, testing platform will start booting with local boot, where operating system that stored in the 1st active partition will be loaded. "Local Boot" is a special case in remote boot process. Once "Local Boot" has been selected, testing platform will silently halt the network boot process and jump to the next boot device as listed in the system BIOS boot order, which is hard disk drive.

Once the Windows OS is successfully loaded, the validation tests will be executed automatically to check the compatibility of the graphic driver. This validation process is a continuous process, where it cannot be disturb by any external interrupt or unexpected errors. If one of the tests is failing to complete, it will recorded into a text file and stored in the server's network-share directory. Similarly, a text file will be created and stored in the server's network-share directory once the validation process is complete.

The PXE server plays an important role in this system. Once the testing platform is rebooting after performing a complete set of validation tests, a registry editor program will edit the PXE menu registry automatically once a text file have been created in server-shared directory. It will inform the server to set the next boot server for the testing platform. This time, APITEST will be selected so that testing platform can change into the next active partition, where different operating system will be loaded. The same process will be continuing until all the operating system has been tested.

4.1.1 Network Interface Sub-System Development

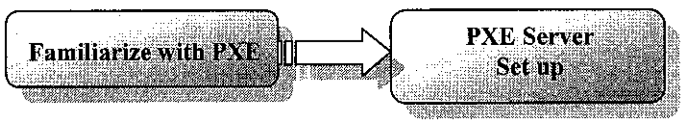


Figure 8: Network Interface Sub-System Development Process

This sub-system development is basically to set up the network interface between server and testing platform with PXE protocol. Basically there are 2 main steps involved in the sub-system development.

a) Familiarize with PXE

PXE offers standardization to process uses for network booting such as:

1. Booting diskless system such as thin client and dedicated system
2. Deploying software and OS for new systems
3. Automating system maintenance such as backups and validation.

PXE service is not operating system specific; it loads the image file of any OS to the client PC. It provides support for network booting, of embedded and other operating system. The PXE server can be on the same server as DHCP or on a different sever, so PXE can add into an existing network without affecting the existing DHCP server or configuration. The PXE server watches for special DHCP requests which include a tag identifying the client as a PXE client. If the discovery request includes the tag, the PXE server replies to the client with configuration information, including the name of a boot image file.

Network Bootstrap Program is used to invoke the initial bootstrap before loading any OS or disk boot manager. Bootstrap is a short program loaded by the BIOS upon system start up. BIOS have no information about the environment and cannot initialise the system beyond putting the hardware into a known state. It is necessary to load an appropriate operating environment by loading bootstrap from a known location and transfer control.

b) PXE Server Set Up

1. Operating System Installation.

To develop a remote boot system with PXE protocol, a server with Microsoft Windows NT 4.0 Server operating system with Dynamic Host Configuration

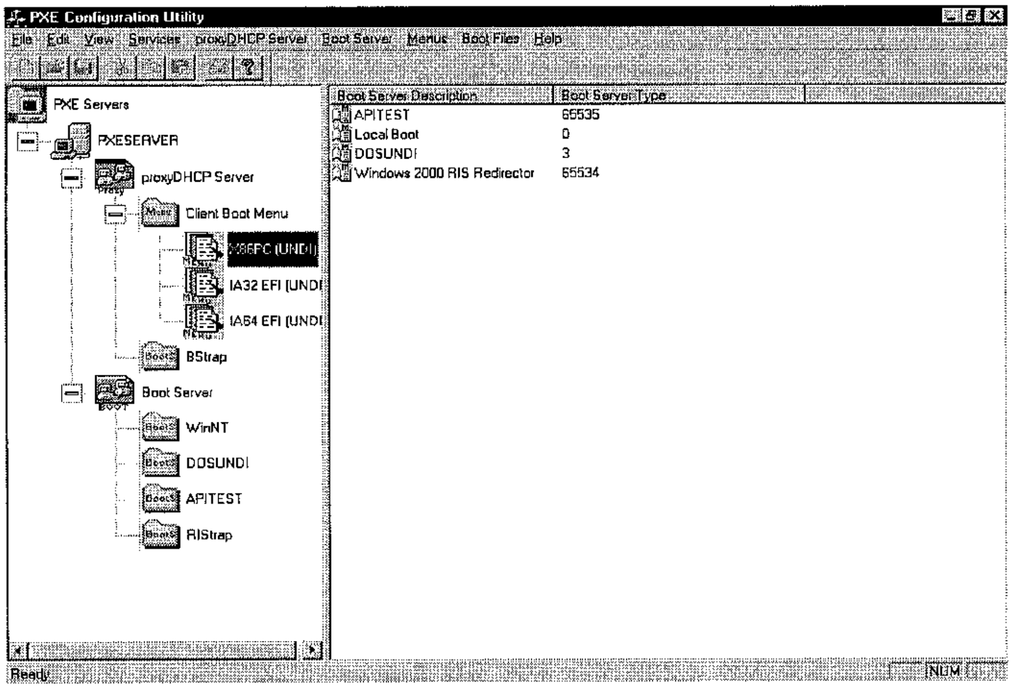
Protocol (DHCP) service has been installed. DHCP have capability of automatic allocation of reusable network addresses and additional configuration option.

2. Network Interface Set Up

During network interface installation, TCP/IP service needs to be installed and assign a static IP address to the server. Make sure that the network interface card (NIC) driver has been installed with the Windows NT 4 Server installation CD. This is because there are some of the files needed to copy from the CD to build the RAMDISK images. This followed by DHCP service. Make sure a valid scope is created on DHCP server.

3. PXE Product Development Kit Installation

After the server has been set up, PXE Product Development Kit (PDK) is installed. PXE PDK contains binaries and programming utilities to install the PXE ROM image into the FLASH memory on Intel EtherExpress PRO PCI Network Interface Card. Make sure PXE PDK is installed with the Windows NT 4 Server CD. This is because there are some of the files needed to copy from the CD to build the DOSUNDI and APITEST RAMdisk images. Figure 9 shows the PXE PDK configuration utility interface. Boot server of APITEST is selected as the foundation to build the boot image that performs the remote boot.



Boot Server Description	Boot Server Type
APITEST	65535
Local Boot	0
DOSUNDI	3
Windows 2000 RIS Redirector	65534

Figure 9: PXE Configuration Utility Interface

In this project, the PXE PDK is being installed on the same host server as the DHCP service. In PXE PDK, there is a service called proxyDHCP, which is configured to support boot clients of the Intel X86PC architecture that have a network interface. The DHCP Class Identifier (option 60) is needed to add to the DHCP server to inform the client that PXE proxyDHCP service is available on the same host as the DHCP service.

4. Testing Platform Network Boot Configuration

In network boot process, there must be always have 2 PCs connecting each another to enhance the ability to communicate with each another. One of the PC will be the server and the other PC is the client PC. After installing PXE PDK, make the Network Interface Card is being selected as the first boot device, which can configure in the BIOS Setup.

Most PCs support the BIOS Boot Specification (BBS), or other method that let the user to order the PC's boot devices from a BIOS setup screen. To perform a network boot each time the PC is powered on, select PXE as the first boot device. Even if the platform does not support BBS, client may still be able to reorder the Network Interface Card to be the first boot device in the BIOS setup as many manufacturers have implemented proprietary methods.

5. PXE API Test Environment Configuration.

Three different types of boot servers are provided in PXE PDK.

a) Packet analysis

b) UNDI Stress Test

c) PXE API Test.

Only PXE API test has been carried out for the purpose of understanding. API stands for Application Program Interface, which is the specific method prescribed by a computer operating system or by an application program by which a programmer writing an application program can make requests of the operation system or another application. An API can be contrasted with a graphical user interface to an operating system or a program.

The PXE specification contains a list of API calls that must be implemented by a PXE boot ROM. These API test program attempts to verify all API calls for their existence. It also verifies the contents of all the registers modified by the call. The test results are stored in a log file on the client and on the proxyDHCP server.

4.1.2 Validation Application Sub-System Development

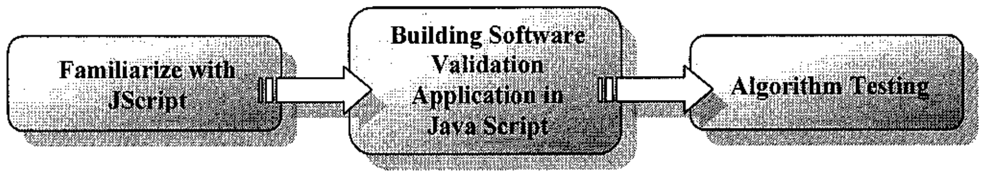


Figure 10: Validation Application Sub-System Development Process

In familiarizing the Java Script, research was done through Microsoft MSDN website which contains all fundamental and advanced Jscript programming code. Jscript is an interpreted, object-based scripting language. Although it has fewer capabilities than full-fledged object-oriented languages like C++, Jscript is more than sufficiently powerful for its intended purposes. In this sub-system, Windows Script Host has been used as a Host. The basic definition, function, assignation, classes, structure, etc of Jscript programming has been learned. Windows Script Host (WSH) and ActiveXObject form the basic structure of the program as these coding techniques are able to access and control the operating system.

Windows Script Host (WSH) is a Windows administration tool. WSH creates an environment for hosting scripts. That is, when a script arrives at user computer, WSH plays the part of the host — it makes objects and services available for the script and provides a set of guidelines within which the script is executed. Among other things, Windows Script Host manages security and invokes the appropriate script engine. It brings simple, powerful, and flexible scripting to the Windows platform, which allowing running scripts from both the Windows desktop and the command prompt. Windows Script Host is ideal for non- interactive scripting needs, such as logon scripting, administrative scripting, and machine or software automation. With combination of these scripting tools, the program was successfully created attachments with all the testing components. *Jscript is case sensitive to lower and upper class letter, in order to avoid error, made sure that the letters assigned properly.*

In this project, JScript is used to link up all the validation tools into a single application, which being used to validate the designed graphic driver. By just drag the script and move to Start | All Program | Start-up, the script will run automatically every time when the system is rebooting. Only one tool is being demonstrated because the ideal script needs about 3 hours to complete.

4.1.3 Partition Changing Sub-System Development

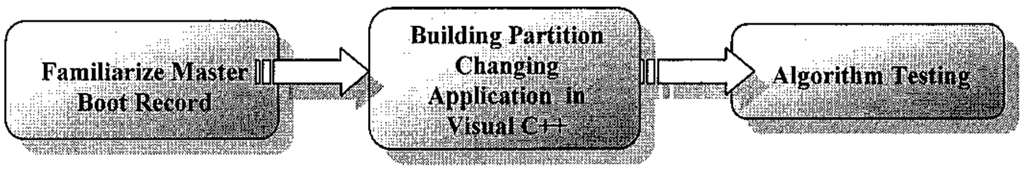

Familiarize Master Boot Record
Building Partition Changing Application in Visual C++
Algorithm Testing

Figure 11: Partition Changing Sub-System Development Process

In this development step, research on Master Boot Record (MBR) has been carrying out. Reading materials on disassembly of MBR and assembly code for searching, reading and setting active partition of the master boot record are needed to realize.

4.1.3.1 What is Master Boot Record?

Master Boot Record (MBR) is the information in the first sector of any hard disk or diskette that identifies how and where an operating system is located so that it can be boot (loaded) into the computer's main storage or random access memory. The Master Boot Record is also sometimes called the "partition sector" or the "master partition table" because it includes a table that locates each partition that the hard disk has been formatted into. (Refer to **Appendix C** for Master Boot Record Codes).

The MBR program code starts at offset 0000. The MBR messages start at offset 008b. The partition table starts at offset 01be and the signature is at offset 01fe. (Refer to **Appendix D** for Master Boot Record Information Table).

When the computer is powered up, the CPU will start executing the BIOS-ROM program. The BIOS runs Power On Self Test (POST) to test and initialise the system. Then it will execute the MBR. It then loads the MBR sector into memory at location 0000:7C00 and checks for a 0xAA55 signature at the end of the sector. The MBR codes will loads the kernel of the operating system in the active partition, while the kernel will loads the rest of the operating system. The kernel is what tells the big chip that controls user computer to do what user wants the program that users're using to do.

So, how is the system able to access to the Master Boot Record (MBR) and change the active partition and load different operating system? In order to modify the MBR, BIOS interrupts has bee used. There are a few numbers of interrupts available such as Interrupt 13h for reading and writing onto the disk and Interrupt 19h for rebooting. The most important interrupts will be Interrupt 13h as it provides functions for direct access of the hard disk. This is important especially for booting, since at this moment no operating system and no hard disk drivers are loaded. The INT 13h uses the CHS (cylinder, head and sector) notation with a width of 24 bits in order to address a boot sector in a disk.

The master boot record is always located at cylinder 0, head 0, and sector 1, the first sector on the disk. This is the consistent starting point that the disk will always use. When a computer starts and the BIOS boots the machine, it will always look at this first sector for instructions and information on how to proceed with the boot process and load the operating system. The master boot record (Figure 12) contains the following structures:

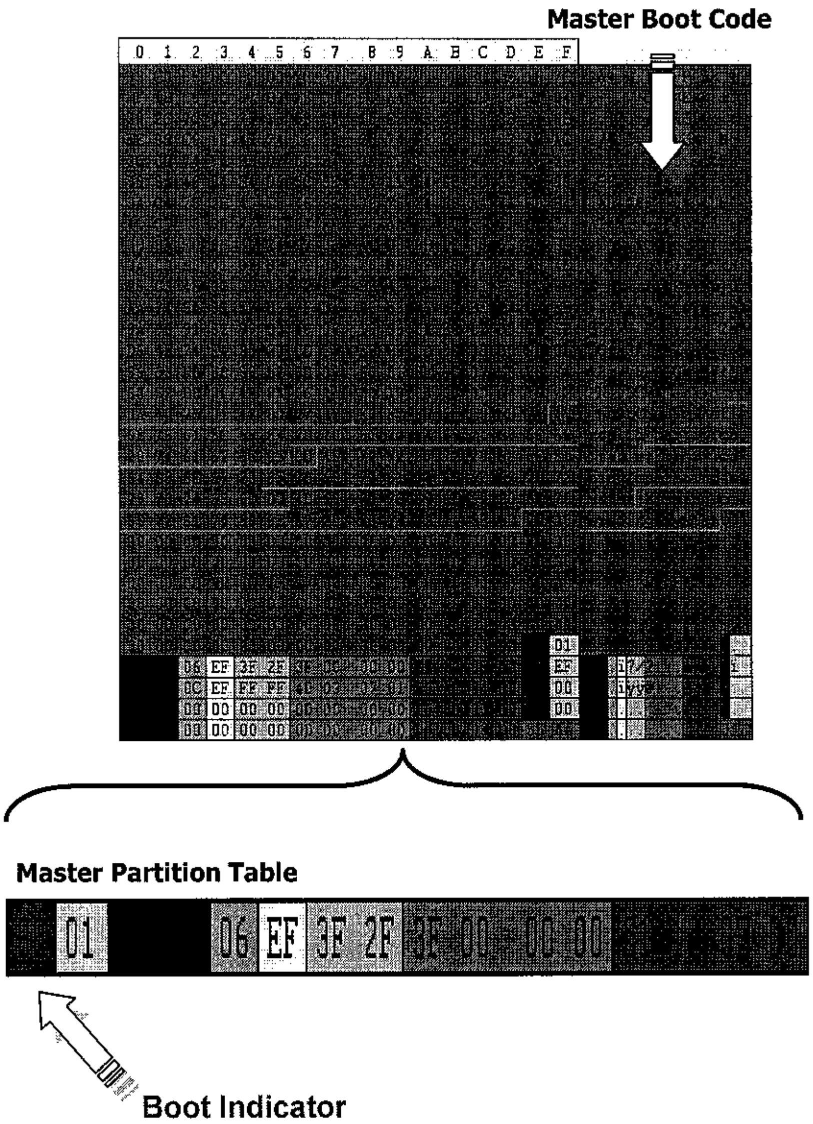
- **Master Partition Table**: This small bit of code that is referred to as a table contains a complete description of the partitions that are contained on the hard disk. When the developers designed the size of this master partition table, they left just enough room for the description of four partitions, hence the four partition (four physical partitions) limit. One of these partitions is marked as active, indicating that it is the one that the computer should used to continue the boot process.

- **Master Boot Code**: The master boot record is the small bit of computer code that the BIOS loads and executes to start the boot process. This code, when fully executed, transfers control to the boot program stored on the boot (active) partition to load the operating system.

Several problems were encountered in accomplishing this task especially in figure out the way to use the INT 13h to access to the MBR. The initial plan is to use a simple program called DEBUG to edit, display or manipulate the MBR. This has to be done in Diskette Operating System (DOS) mode because MBR is not accessible during n ormal w indow operating system. S ome b asics command t hat application uses are the (A) ssemble, (U) nassemble and the (W) rite commands. (Refer to **Appendix E** for Unassembling MBR using Debug Application).

By using DEBUG program, the system is able to manipulate the MBR. But Microsoft has changed its version of MBR from Windows 95 to Windows 95se. A far more complicated MBR was introduced for high version of Windows. Since this system require changing the active partition of the hard disk. So it is time consuming to rewrite the whole Master Boot Record to suit each of every operating system. The best way is to write a program similar to the operating system boot loader to change the active partition of the hard disk without override the MBR. Besides that writing directly to a hard disk using sector numbers will most likely result in loss of data or even corrupted the hard drive

The other alternative way would be using the FDISK program, which can found in Windows Start-up Disk to achieve the same objective. Manipulation of the source code has been carried out to generate the application by using Microsoft Visual C++ compiler. The FDISK utility provides capability in partitioning the hard drives. Partitioning a hard disk is a process of defining areas of the disk to be used by operating systems as volumes. FDISK allows the user to delete or create partitions and set active partition on the hard disk drive. Focus has been putting on reading and writing the Master Boot Record and the partition table. Additional BIOS functions



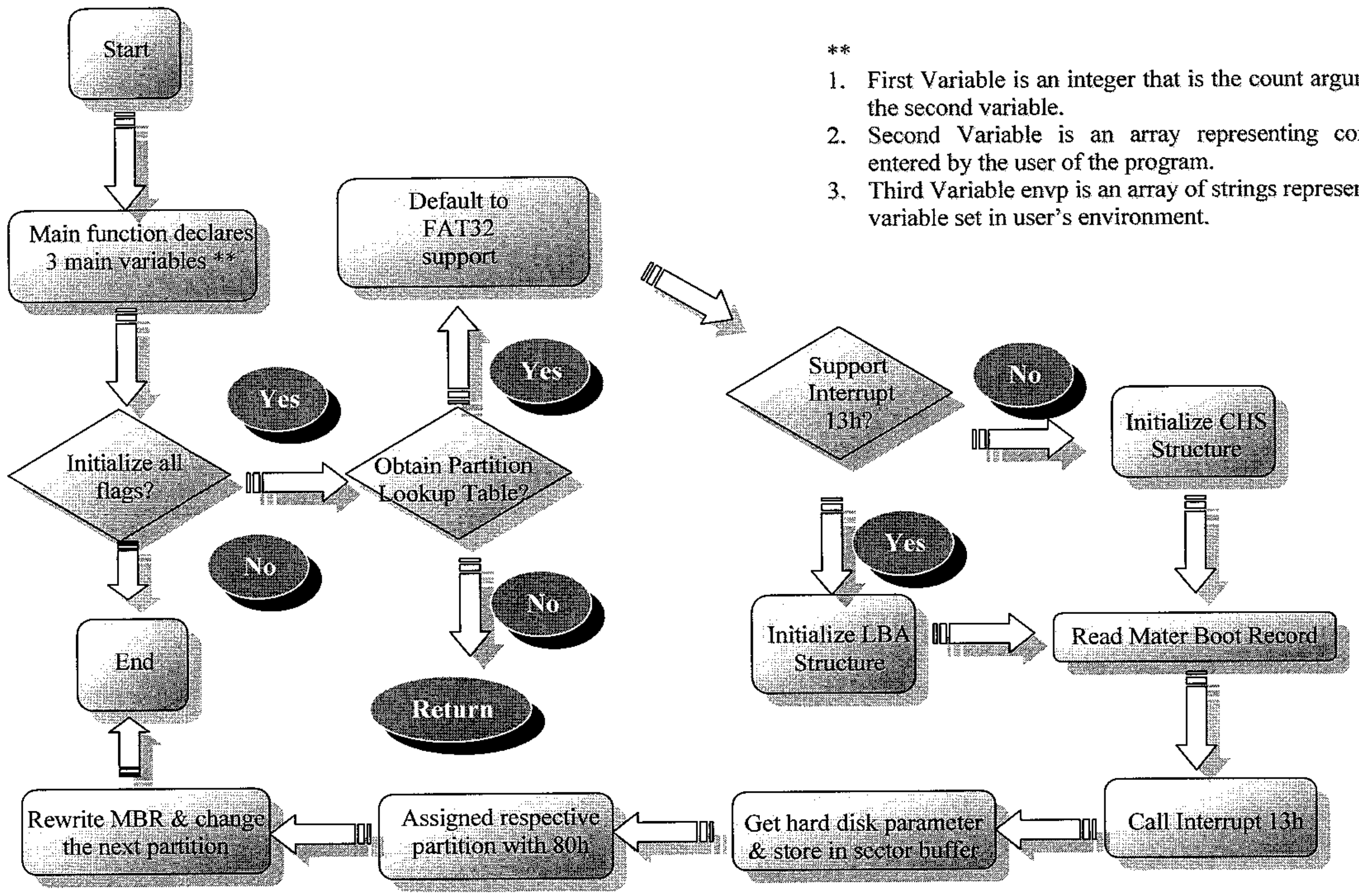
Boot Indicator

- **Determines which partition to boot from.**
- **00 = inactive and 80 = active.**
- **Only one partition marked as 80 at one time**

▨ -> Executable code	□ -> ending head
▨ -> boot indicator	▨ -> ending sector & cylinder
▨ -> starting head	▨ -> the starting sector
■ -> starting sector & cylinder	▨ -> no. of sectors in partition
▨ -> partiton type	▨ -> executable signature

Figure 12: Master Boot Record

library have been added during the compilation and building of the program. The general flow of the change partition program codes was listed in Figure 13.


Start
Main function declares 3 main variables **
Initialize all flags?
Yes
No
End
Obtain Partition Lookup Table?
Yes
Default to FAT32 support
No
Return
Support Interrupt 13h?
No
Initialize CHS Structure
Yes
Initialize LBA Structure
Read Mater Boot Record
Call Interrupt 13h
Get hard disk parameter & store in sector buffer
Assigned respective partition with 80h
Rewrite MBR & change the next partition

**

1. First Variable is an integer that is the count arguments of the second variable.
2. Second Variable is an array representing commands entered by the user of the program.
3. Third Variable envp is an array of strings representing the variable set in user's environment.

Figure 13: General Flow of Partition Changing Application

4.1.4 Booting Sequence Sub-System Development

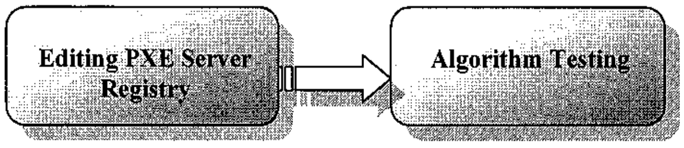


Figure 14: Booting Sequence Sub-System Development Process

The basic operation flow for booting sequence process is shown in Figure 15.

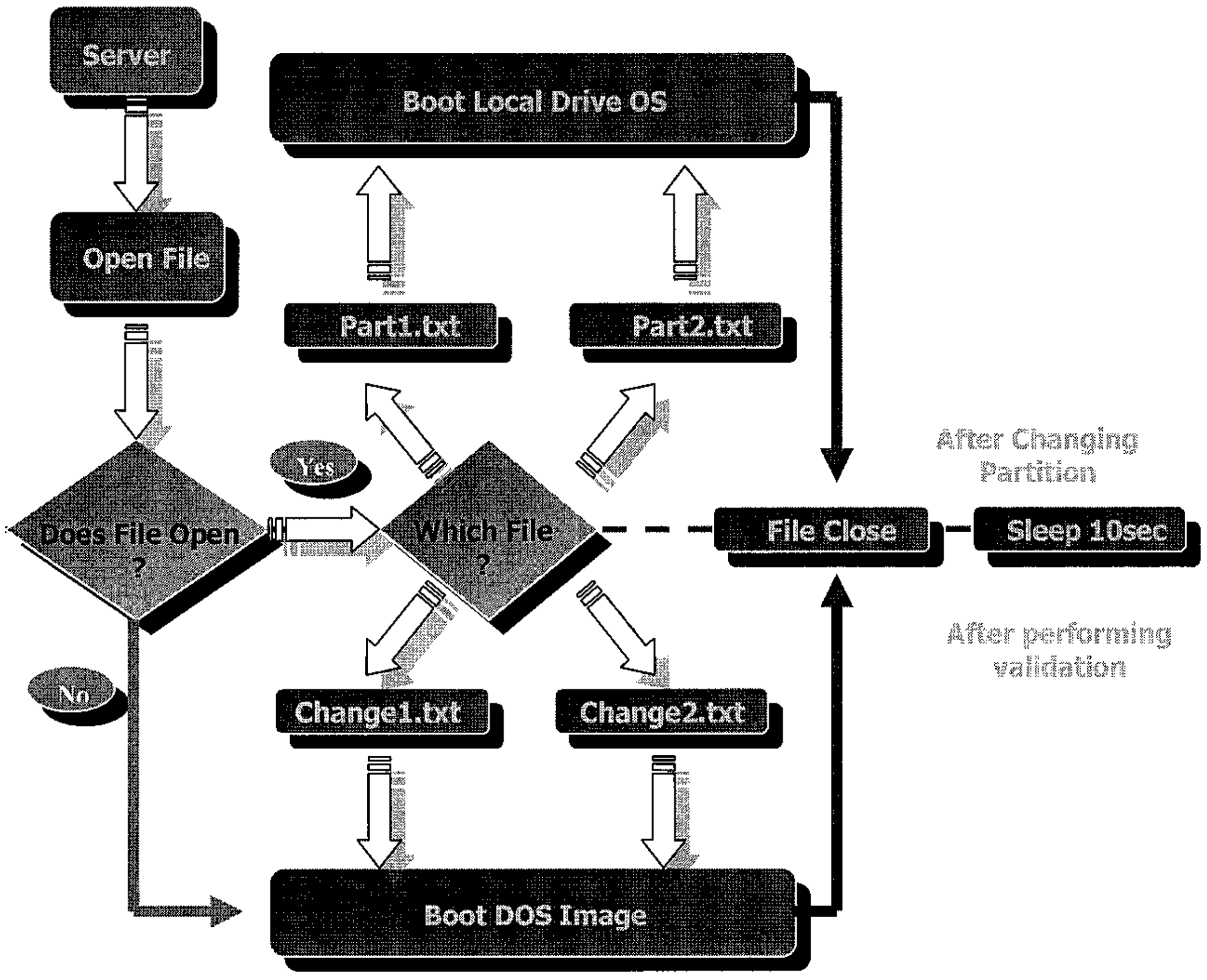


Figure 15: Booting Sequence Process Operation Flow

In order knowing the booting sequence, the testing platform has to perform the following steps:

1. The testing platform must generate a file to tell the server that it has performed the required operation and task. This can be done by writing a script or a batch file that will create a new text document, which can be identified by the server.

2. To generate a new text document, modified version of autoexec.bat and validation JScript has been made by adding a new command (echo> xxx.txt to batch script and fso.createtextfile ("xxx.txt") to validation JScript). Since the testing platform had mapped to the server-shared directory, the server will be able to read the newly created text document.

3. When the testing platform remote booting the DOS image file, the batch script in the image will set up the environment to changes the active partition from one to another. On the other hand, validation will be loaded after the testing platform has change to another active partition.

4. In responds of the server, a program has been developed to read the files generated by the testing platform. It will continuously read the files and edit the PXE menu registry. This program will determine whether the testing platform should remote boot the DOS image or start loading operating system from local boot.

5. Then the program will closes all open files and go into sleep mode for 10 seconds. This will provide sufficient resources for the Testing platformto remote boot the image file and increases the responds time in writing a new text file, else the server will be lagging and decreases the speed of image and files transferring

Although the result of the project is satisfying but there is one disadvantage. The boot sequence only changes after the time limit reaches and only one boot sequence that able to be booted. This will be a problem for too many clients booting at once from the server as the server will not able to identify which boot sequence to be boot.

4.1.5 Remote Boot Sub-System Development

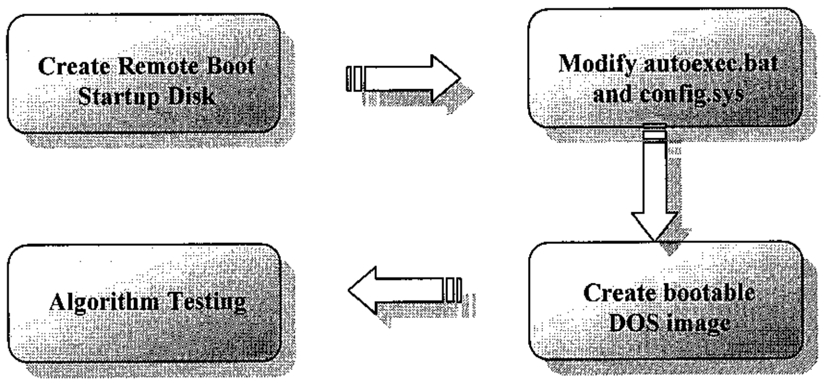

Create Remote Boot Startup Disk
Modify autoexec.bat and config.sys
Create bootable DOS image
Algorithm Testing

Figure 16: Remote Boot Sub-System Development Process

a) Create Remote Boot Startup Disk

The first activity to perform is to create a remote boot start up disk. MS-DOS 6.22 system boot disk image has been selected due to one of the software requirement by PXE PDK. This OS image (start up disk image) provides an environment for the testing platform to run the partition changing application later. There are 2 methods where a boot disk can be created. One is using format /s instruction in the DOS operating system environment. The second method is using boot disk utility, where can be downloaded from the Internet.

Just make sure these three files are inside the boot disk, **command.com, msdos.sys** and **io.sys**. The rest of the files can be deleted for space saving purposes. After this, make sure that these additional files are also being inserted into the disk:

a) HIMEM.SYS

b) RAMDRIVE.SYS

c) MORE.COM

d) FC.EXE

e) AUTOEXEC.BAT

f) CONFIG.SYS

HIMEM.SYS is a system driver, which manages extended memory and is generally located as a companion to **EMM386.SYS**. It coordinates the use of extended memory, including the high memory area **(HMA),** so that no application or devices driver attempt to use the same memory area at the same time. Generally it's located within the **CONFIG.SYS** file using **DEVICE** command. This allows the system to load the driver when it is booted or at starting up.

EMM386.EXE - expended memory system gives user computer access to the upper memory area, beyond the convention 640K. It's usually used in combination with **HIMEM.SYS** driver. **EMM 386.EXE** allows computer to use its hard disk drive as extended memory to simulate expanded memory. It allows user to load programs and other system drivers into upper memory blocks (UMB). UMB is the sections of memory that reside between 640K and 1MB. Sections of this memory are set aside for specific tasks but much is unused. Only microcomputer with an 80386 microprocessor, or better, can use this command. Below is the memory map for the DOS operation system.

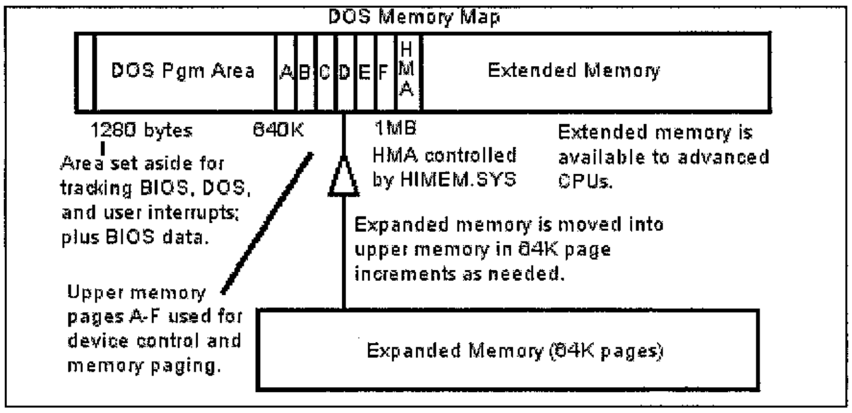

DOS Memory Map
DOS Pgm Area
A
B
C
D
E
F
HMA
Extended Memory
1280 bytes
640K
1MB
Area set aside for tracking BIOS, DOS, and user interrupts; plus BIOS data.
HMA controlled by HIMEM.SYS
Extended memory is available to advanced CPUs.
Expanded memory is moved into upper memory in 64K page increments as needed.
Upper memory pages A-F used for device control and memory paging.
Expanded Memory (64K pages)

Figure 17: DOS Memory Map

RAMDRIVE.SYS is a system driver that allows the computer to use part of its random access memory **(RAM)** to act in the place of a physical hard disk drive. The driver must be loaded in the **CONFIG.SYS** file so that it is mounted when the system starts or is booted. A **RAMDRIVE** can greatly increase the speed of some often repeated disk read/write operations and can be configured as the default for these operations.

The **MORE** command displays output from a file one screen ay a time with prompting for additional screens of information.

FC is used to compare the contents of two files for similarities and differences.

SMARTDRV is used to configure the **SMART Drive** software, which creates and manages the disk cache, which is used for extended memory. Although it can be invoked at the command prompt, the program is generally run within the **AUTOEXEC.BAT** file so that it initialises at every boot and start up.

CONFIG.SYS file loads device drivers and setup memory management. The **AUTOEXEC.BAT** file loaded after *config.sys* during system boot. This file launches programs that help set the operating environment and loading necessary terminate-and-stay resident (TSR) programs. TSR programs run in the background but provide system with added functionality.

b) Modify the autoexec.bat and the config.sys contents

APITEST Boot server is selected as the foundation to build the boot image that performs the remote boot. The related files of the APITEST such as the autoexec.bat and config.sys can be found in the following path: ***%ROOT_DIRECTORY%\ProgramFiles\Intel\PXE\PDK\System\Image\x86\UNDI \APITEST\mkimage.***

In order to produce the customized-boot image, modification of the autoexec.bat and the config.sys has been carried on.

c) Building DOS image.

Get into the Server's command prompt and run the command *mkimage.exe* in the following path:

%ROOT_DIRECTORY%\Program Files\Intel\PXE\PDK\System.

This mkimage.exe application files is used to create a DOS boot diskette image (such as APITEST.1). An image file named **test.bin** will be created with the size of 1.44 MB. Copy and paste the file to test folder. There are two image files in this directory, namely APITEST.0 and APITEST.1. APITEST.0 is a bootstrap environment image. This image is used to verify that the machine can run the OS image in the next layer. Download the next layer and set up the machine to run the next layer. After pasting the image, rename the file test.bin to APITEST.1. Before this, don't forget to delete the original APITEST.1 file inside this folder.

4.1.6 Multiclient Support Sub-System Development

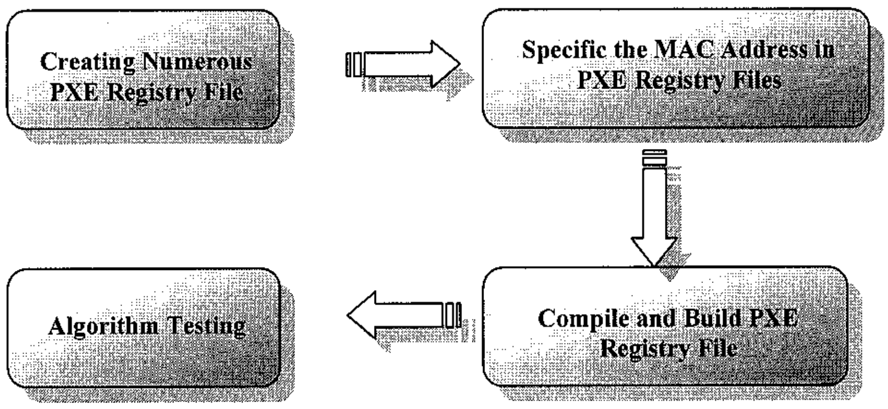

Creating Numerous PXE Registry File
Specific the MAC Address in PXE Registry Files
Compile and Build PXE Registry File
Algorithm Testing

Figure 18: Remote Boot Sub-System Development Process

Basically, the final task at hand is to complete the multiclient support sub-system and integrating this sub-system with the rest to form a complete graphic driver validation system. The approach to develop this sub-system is pretty straight forward. By using back the same PXE registry that has been created in booting sequence sub-system, specific the MAC Address for each of the testing platform in the file.

MAC Address, or also called as Absolute Address is an address that is permanently assigned to a specific storage location in memory. This means that, all testing platforms will have their own PXE registry file. Whenever a validation engineer needs to validation a testing platform, he just needs to make sure that he is executing the correct PXE registry file with the correct MAC Address. With all these, numerous validation processes can run parallel at the same time.

CHAPTER 5

CONCLUSSION

The project is basically in the working scope of utilizing the knowledge that has been acquired in 5 years of study. During the project design and implementation, more stones unturned in each step of progress that has been made. There are many other knowledge bases that need to pick up to complete this project. All these basically add more challenge to current task and would inevitably add more learning curve to the project.

5.1 Project Relevancy

After working on the project for so many months we have found that it is a relevant project to prove the two points. First, that part of the project can be successfully conducted and completed using free tools. We have used free software whether open source or other wise, at every step of process. Second, that an individual today can accomplish what took a team of engineers to accomplish several decades ago by building on existing work.

In addition, this project is very unique, as it benefits the past and present. This is because the system is a system that has been around. It benefits the past, as in recreating a compatible system from the past, we have managed to make several improvement to it to improve performance. It also benefits the present as the validation jobs are easier to perform compare to past.

At last, the project is a feasible one. With limited time and resources, we had sufficiently achieved the main objective of the project. The prototype has been implemented and although some bugs still exit in the design, most of them have been worked out and the processor is generally working. It is capable of executing in all operating system but they are probably still some bugs left to be ironed out.

5.2 Conclusion

In conclusion, the project looks promising with a lot of learning curves. A lot of skills and knowledge need to be picked up prior to the implementation of the core of the project. The undertaking of the project would provide a better solution for the software and driver validation process. The successful completion of the project could pave a way to the progress of bigger project that is currently being carried out by the external supervisor in Intel, Penang. Generally, the outline of the project had already been clearly justified and the preliminary stage of the project would focus primarily on gathering all the information pertaining to the project which is basically the PXE protocol and Visual C programming. The Gantt chart created in this report would provide a guideline on the progress of the project and would also serve to list the design steps necessary for the successful of the project.

5.3 Recommendations

5.3.1 Soft Copy Submissions

The author would like to suggest that the FYP submission of reports be done in PDF or DOC format instead of present print outs especially for weekly logbook reports. This is because the amount of paper generated is tremendous. If students could move towards a paperless method of submission, we would help save environment as well as reduce the hassle of submission. With the use of the UTP network, students could establish a local server for submission and distribution of FYP reports and updates. It could be supervised and controlled by one of the EE technicians doing the paperwork today.

5.3.2 Open Source Contributions

The author would like to suggest that U TP provide c ontinues contributions to the open source community. All FYP students' works should be made open source. There are many things that students can learn from open source and thus students should actively contribute to it. Open source allows students to learn things that are not taught in books and classrooms. It provides practical examples of knowledge application that is used in the industry as well as tips and tricks that are never taught except through real examples.

5.4 Suggested Future Work for Expansion and Continuation

Basically, the task at hand is to complete all sub-systems and integrating them together to form a complete Multiclient Validation System. The assembling of each sub-system requires student to understand all the programs that sending to the testing platform and their functionality.

The first sub-system would initially be difficult to design and comprehend, but with the successful implementation of subsequent sub-systems, the last sub-system, which is multiclient support, would inevitably be much easier to design since more experience, knowledge and exposure on the basic principle functionality have been acquired.

The suggested future work for expansion and continuation would be improving the multiclient support sub-system, which is not fully implemented in this project. It can be improve by editing the PXE protocol source code so that the server can support validation process for multiple testing platforms without creating specific MAC Address for every single testing platform in the PXE Registry File. Once the machine address has been detected, a table will be created to store all machine address variables. From the table, validation engineer will specify the boot image file and boot server for each testing platform to download. This will be eliminating conflicts on getting the same image boot file.

The PXE ROM Image with newly editing PXE source code can be installed into the FLASH memory on Intel EtherExpress PRO PCI Network Interface Card (NIC). With this additional feature, the validation work will become simpler and easier compare with the past.

Overall, the knowledge bases that I will need to pick up in my project are as follow:

1. Network Programming (Visual C++)
2. PXE Configuration Utility Interface (Visual C++)
3. Preboot Execution Environment (PXE) Protocol

Judging from the progress right now, it is a strong confidence that the project could be completed in time. However, a lot of effort needs to be put in not only to acquire all the knowledge base stated above but also to manipulate and apply them into the project.

REFERENCES

1. *Intel Wired For Management (WfM)*,2002 <http://www.intel.com/labs/manage/wfm/ >

2. *AppDeploy.com :FAQ :Preboot Execution Environment (PXE, PiXiE)*,2003 <http://appdeploy.com/faq/pxe.shtml>

3. *Preboot Execution Environment –Microsoft Internet Explorer*, 2003 <http://www.eu.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/distrib/dsed_dpl_UEPI.asp>

4. *Microsoft MSDN website*, 2003<http://msdn.microsoft.com/>

5. *Intel Corporation - Embedded Intel(R) Architecture in Communications*, 2004 <http://www.intel.com/platforms/applied/eiacomm/index.htm

6. Peter G. Brierley, 1997, "New Perspective on Microsoft Windows NT4 server 4.0", Collin Country Community College.

7. Ian Sommerville, 2001, "Software Engineering 6th Edition", Addison Wesley.

Overview of Remote Boot

The issue of the high total cost of ownership (TCO) of PCs in a corporate environment is well understood by the IT community today. The main question now is: What is the best solution for the TCO problem? There are many products available that help address this issue, ranging from simple, low-cost, single-task "tools" to costly, complex, multiple-task client m anagement products. One of the solutions i s using remote b oot (or network boot) w here a c omputer that does not relies on local resources to start, but uses centralized remote resources instead.

The common problem faced by IT managers in remote boot is to ensure that client systems in their enterprises can boot appropriate software images using appropriate configuration parameters. These selected boot images and configuration parameters must be acquired from selected servers in the enterprise as dictated by the needs of the particular environment, the capabilities or mission of the user, the resources available within the client, etc. Furthermore, these clients should boot consistently and in an interoperable manner regardless of the sources or vendors of the software and the hardware of both client and server machines.

There are several advantages and uses for network booting:

- Booting diskless systems such as thin clients and dedicated systems
- Deploying software and OS for new systems
- Automating system maintenance such as backups
- Automating system checking such as virus scanning
- Ensuring security where a guaranteed secure system is needed

The two main uses of remote boot today are installing an OS in a brand new client PC that has no operating system, (or re-installing in a client PC where the operating system has failed), and booting into a guaranteed "clean" system. By booting a brand new system or a defective-OS system from the network, user can install a new OS and/or applications without visiting each client PC with a stack of installation CDs.

APPENDIX A

Setting up a new client PC is as simple as connecting it to the network and powering it on. User can setup user servers to automatically detect the new clients and start installing the new software. This can dramatically reduce administration time. The administrator no longer has to physically visit to reinstall the software when a user's computer crashes. A remote boot does the re-install. It may now be more efficient to simply reinstall the entire user's software than to try to determine the problem with the existing installation. When u ser b oot from the n etwork, u ser g et a g uaranteed "clean" boot, with no boot-time viruses or user modified files. The system boot files are stored on the server where they are protected from infection.

Remote boot process also can be used to scan for viruses, ensuring that the local hard drive is clean before user boot from it. These tools include the Preboot Execution Environment (PXE) server side components and utilities to build boot image files. They enable administrators to set up a network booting environment to perform specific client management tasks using off-the-shelf utilities.

In summary, using Preboot Execution Environment, a newly installed networked client machine should be able to enter a heterogeneous network, acquire for itself a network address from a DHCP server, and then download a NBP to set itself up. This sets the stage to enable IT managers to customize the manner in which their network client machines go through a network-based booting process.

APPENDIX B

OVERVIEW OF PREBOOT

This section discusses Preboot, which can be another weapon in the IT manager's arsenal for tackling the TCO problems. Preboot is not a tool that performs specific tasks, but rather a framework that can be used to centrally deploy various client management tools.

As the name suggests, Preboot occurs before the operating system (OS) is loaded. The primary distinctive aspect of Preboot is the download of a boot image file from a boot server on the network, and the execution of this boot image on the client PC. To do this, the client PC must be equipped with remote boot firmware, as described later. Figure 19 shows a normal PC boot sequence without Preboot. Figure 20 shows the PC boot sequence with legacy Preboot capability. In Preboot environments, the PC normally reboots when the Preboot phase is completed

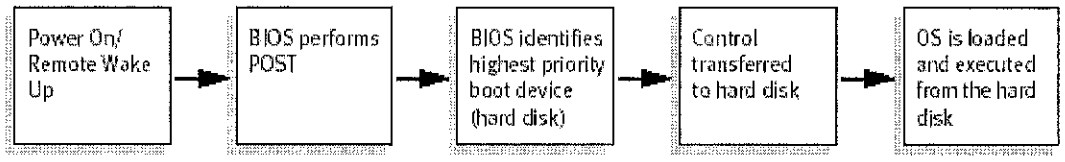

Power On/ Remote Wake Up
BIOS performs POST
BIOS identifies highest priority boot device (hard disk)
Control transferred to hard disk
OS is loaded and executed from the hard disk

Figure 19: Normal PC Boot Sequence

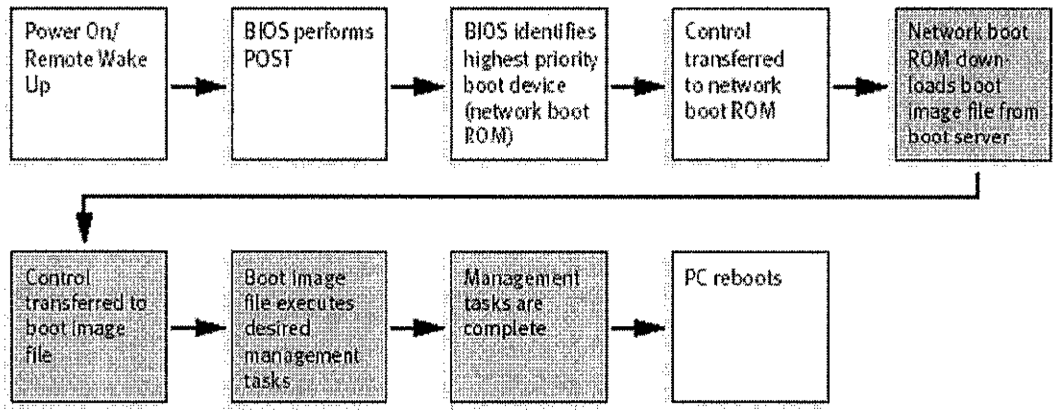

Power On/ Remote Wake Up
BIOS performs POST
BIOS identifies highest priority boot device (network boot ROM)
Control transferred to network boot ROM
Network boot ROM downloads boot image file from boot server
Control transferred to boot image file
Boot image file executes desired management tasks
Management tasks are complete
PC reboots

Figure 20: PC Boot Sequence with Preboot

The boot image file that is downloaded and executed on the client PC determines what actions are performed during the Preboot phase. A boot image file is the consolidation of the contents of a regular boot diskette (or diskettes) into one file. Typically boot image files contain two sets of files. The first set are files necessary to boot a PC, such as a small OS. To minimize the size of the boot image file (thus minimizing its download time) and to minimize the amount of memory needed to store the boot image file, the OS used during a Preboot session is typically a small, lightweight OS. DOS is the OS most commonly used during a Preboot phase.

The second set of files is those needed to perform the desired tasks. This is where the power of Preboot is unleashed. Programs, tools, utilities, and batch/script files can be included to perform a desired task on the client PC, as long as they are supported by the OS that is being used. The programs that are executed during the Preboot phase are often called "Preboot tasks." The types of Preboot tasks that are performed in a Preboot session are:

- Hard disk imaging. There are many imaging tools available today. They can be used to roll out OS installations and corporate desktops to client PCs.
- Virus scanning, especially of the boot sector.
- System updates, such as updating the client PC's BIOS to the latest version using a flash utility.
- Checking the existence and integrity of critical files.

There are two methods for executing Preboot tasks:

1 Include all the files necessary to perform the Preboot task in the boot image file. In this case, the files will be loaded into and executed from the client PC's memory.

2 Include only the files necessary to connect to and log into a file server where the Preboot task files are located. Execute the task from the server.

Three elements are needed to set up a Preboot environment:

1. **Client remote boot firmware.**

Remote boot firmware must be available on the client PC. When a PC boots, this firmware will communicate with a remote boot server to download the boot image file in the PC's memory and then execute the boot image. The firmware is usually located in an Option ROM on a network interface card (NIC) or integrated into the PC BIOS (Basic Input Output System). There are various remote boot protocols, but the specification that has become the industry standard is the Preboot Execution Environment (PXE) specification, which is part of the Wired for Management (WfM) specification. Most enterprise PCs purchased over the last two years typically have PXE firmware already included.

2. **Remote boot services.**

Remote boot services must be installed on a server that will act as the remote boot server. This can be an existing server or a new server that is added to an existing network. When using the PXE network booting method, four services are usually required: Dynamic Host Configuration Protocol (DHCP), PXE, Proxy DHCP (when the PXE service is installed on a boot server separate from the DHCP server), and Trivial File Transfer Protocol (TFTP). The DHCP service required is the usual one that issues IP addresses to the client PCs. Typically, most TCP/IP-based networks already have a DHCP service running. DHCP services are available with most network operating systems, including Windows NT and Windows 2000. The Proxy DHCP service is needed to supply the PXE client with the IP address of the PXE service. The service cannot run on the same server as the DHCP service. The PXE service supplies the PXE client with the filename of the boot image file to be downloaded. The TFTP service is a simple form of FTP. It is used to transfer the boot image file from the remote boot server to the PXE client. The Proxy DHCP, PXE, and TFTP services may be available with some network operating systems or they can be purchased from third-party vendors.

3. Administrator tools.

Some m iscellaneous tools are n eeded t o do v arious tasks such a s b oot image file creation. There are products available that c ontain this functionality. Preboot is a framework that can be used to centrally deploy various client management tools over a network. It allows DOS-based tools and utilities to be executed prior to the client's ultimate OS being loaded from the local hard disk. Setting up a Preboot environment and executing client management tools, either on a regular basis with every boot or only when needed, can help reduce system maintenance costs and increase maintenance efficiency.

MASTER BOOT RECORD SOURCE CODE

This sector is initially loaded into memory at 0000:7c00 but it immediately relocates itself to 0000:0600.

```
                BEGIN:                          NOW AT 0000:7C00,
RELOCATE

0000:7C00 FA            CLI                     Disable int's
0000:7C01 33C0          XOR     AX,AX           Set stack seg to 0000
0000:7C03 8ED0          MOV     SS,AX
0000:7C05 BC007C        MOV     SP,7C00         Set stack ptr to 7c00
0000:7C08 8BF4          MOV     SI,SP           SI now 7c00
0000:7C0A 50            PUSH    AX
0000:7C0B 07            POP     ES              ES now 0000:7c00
0000:7C0C 50            PUSH    AX
0000:7C0D 1F            POP     DS              DS now 0000:7c00
0000:7C0E FB            STI                     Allow int's
0000:7C0F FC            CLD                     Clear direction
0000:7C10 BF0006        MOV     DI,0600         DI now 0600
0000:7C13 B90001        MOV     CX,0100         Move 256 words (512
                                                bytes)
0000:7C16 F2            REPNZ                   Move MBR from
                                                0000:7c00
0000:7C17 A5            MOVSW                   to 0000:0600
0000:7C18 EA1D060000    JMP     0000:061D       Jmp to NEW_LOCATION


        NEW_LOCATION:                           NOW AT 0000:0600

0000:061D BEBE07       MOV     SI,07BE          point to first table
                                                entry
0000:0620 B304         MOV     BL,04            There are 4 table
                                                entries


        SEARCH_LOOP1:                          SEARCH FOR AN ACTIVE
ENTRY

0000:0622 803C80       CMP     BYTE PTR [SI],80  Is this the active
                                                 entry?
0000:0625 740E         JZ      FOUND_ACTIVE      Yes
0000:0627 803C00       CMP     BYTE PTR [SI],00  Is this an inactive
entry?
0000:062A 751C         JNZ     NOT_ACTIVE        No
0000:062C 83C610       ADD     SI,+10            incr table ptr by 16
0000:062F FECB         DEC     BL                decr count
0000:0631 75EF         JNZ     SEARCH_LOOP1      jmp if not end of
table
0000:0633 CD18         INT     18                GO TO ROM BASIC
```

```
        FOUND_ACTIVE:                               FOUND THE ACTIVE ENTRY

0000:0635 8B14          MOV     DX,[SI]             Set DH/DL for INT 13
                                                    call
0000:0637 8B4C02        MOV     CX,[SI+02]          Set CH/CL for INT 13
                                                    call
0000:063A 8BEE          MOV     BP,SI               Save table ptr


        SEARCH_LOOP2:                               MAKE SURE ONLY ONE
ACTIVE ENTRY

0000:063C 83C610        ADD     SI,+10                 incr table ptr by 16
0000:063F FECB          DEC     BL                     decr count
0000:0641 741A          JZ      READ_BOOT              jmp if end of table
0000:0643 803C00        CMP     BYTE PTR [SI],00    is this an inactive
                                                    entry?
0000:0646 74F4          JZ      SEARCH_LOOP2           yes


          NOT_ACTIVE:                               MORE THAN ONE ACTIVE
ENTRY FOUND

0000:0648 BE8B06        MOV     SI,068B             Display "Invld prttn
                                                    tbl"


         DISPLAY_MSG:                               DISPLAY MESSAGE LOOP

0000:064B AC            LODSB                          get char of message
0000:064C 3C00          CMP     AL,00                  end of message
0000:064E 740B          JZ      HANG                   yes
0000:0650 56            PUSH    SI                     save SI
0000:0651 BB0700        MOV     BX,0007                screen attributes
0000:0654 B40E          MOV     AH,0E                  output 1 char of
                                                    message
0000:0656 CD10          INT     10                        to the display
0000:0658 5E            POP     SI                     restore SI
0000:0659 EBF0          JMP     DISPLAY_MSG            do it again


                HANG:                               HANG THE SYSTEM LOOP

0000:065B EBFE          JMP     HANG                   sit and stay!


           READ_BOOT:                               READ ACTIVE PARITION
BOOT RECORD

0000:065D BF0500        MOV     DI,0005                INT 13 retry count


           INT13RTRY:                               INT 13 RETRY LOOP

0000:0660 BB007C        MOV     BX,7C00
0000:0663 B80102        MOV     AX,0201                read 1 sector
0000:0666 57            PUSH    DI                     save DI
0000:0667 CD13          INT     13                     read sector into
0000:7c00
0000:0669 5F            POP     DI                     restore DI
0000:066A 730C          JNB     INT13OK                jmp if no INT 13
```

```
0000:066C 33C0        XOR     AX,AX              call INT 13 and
0000:066E CD13        INT     13                    do disk reset
0000:0670 4F          DEC     DI                 decr DI
0000:0671 75ED        JNZ     INT13RTRY          if not zero, try
                                                 again
0000:0673 BEA306      MOV     SI,06A3            display "Errr ldng \
                                                 systm"
0000:0676 EBD3        JMP     DISPLAY_MSG        jmp to display loop


             INT13OK:                        INT 13 ERROR

0000:0678 BEC206      MOV     SI,06C2               "missing op sys"
0000:067B BFFE7D      MOV     DI,7DFE               point to
                                                    signature
0000:067E 813D55AA    CMP     WORD PTR [DI],AA55    is signature
                                                    correct?
0000:0682 75C7        JNZ     DISPLAY_MSG           no
0000:0684 8BF5        MOV     SI,BP                 set SI
0000:0686 EA007C0000  JMP     0000:7C00             JUMP TO THE BOOT
                                                    SECTOR
                                                       WITH SI
POINTING TO
                                                       PART TABLE
ENTRY
```

Messages here.

```
0000:0680 ........ ........ ......49 6e76616c *           Inval*
0000:0690 69642070 61727469 74696f6e 20746162 *id partition tab*
0000:06a0 6c650045 72726f72 206c6f61 64696e67 *le.Error loading*
0000:06b0 206f7065 72617469 6e672073 79737465 * operating syste*
0000:06c0 6d004d69 7373696e 67206f70 65726174 *m.Missing operat*
0000:06d0 696e6720 73797374 656d00.. ........ *ing system.     *
```

Data not used.

```
0000:06d0 ........ ........ ......00 00000000 *           .....*
0000:06e0 00000000 00000000 00000000 00000000 *................*
0000:06f0 00000000 00000000 00000000 00000000 *................*
0000:0700 00000000 00000000 00000000 00000000 *................*
0000:0710 00000000 00000000 00000000 00000000 *................*
0000:0720 00000000 00000000 00000000 00000000 *................*
0000:0730 00000000 00000000 00000000 00000000 *................*
0000:0740 00000000 00000000 00000000 00000000 *................*
0000:0750 00000000 00000000 00000000 00000000 *................*
0000:0760 00000000 00000000 00000000 00000000 *................*
0000:0770 00000000 00000000 00000000 00000000 *................*
0000:0780 00000000 00000000 00000000 00000000 *................*
0000:0790 00000000 00000000 00000000 00000000 *................*
0000:07a0 00000000 00000000 00000000 00000000 *................*
0000:07b0 00000000 00000000 00000000 0000.... *............    *
```

The partition table starts at 0000:07be. Each partition table entry is 16 bytes. This table defines a single primary partition which is also an active (bootable) partition.

```
0000:07b0 ........ ........ ........ ....8001 *            ....*
0000:07c0 0100060d fef83e00 00000678 0d000000 *...........x....*
0000:07d0 00000000 00000000 00000000 00000000 *................*
0000:07e0 00000000 00000000 00000000 00000000 *................*
```

```
0000:07f0 00000000 00000000 00000000 0000.... *............    *
```

The last two bytes contain a 55AAH signature.

```
0000:07f0 ........ ........ ........ ....55aa *..............U.*
```

MASTER BOOT RECORD INFORMATION TABLE

```
-------------------------------------
Partition sector (length 512 bytes)
-------------------------------------
Offset  Bytes  Meaning

 000h    446    boot loader code
 1BEh    16     1. partition entry
 1CEh    16     2. partition entry
 1DEh    16     3. partition entry
 1EEh    16     4. partition entry
 1FEh    2      signature (55h AAh)
-------------------------------------

-------------------------------------------------------
Partition entry (length 16 bytes)
-------------------------------------------------------
Off. Bytes Meaning

00h    1    80h = active partition / 00h = not active
01h    1    begin of partition (head number)
02h    1    begin of partition (sector number)   [*]
03h    1    begin of partition (cylinder number) [*]
04h    1    partition ID
05h    1    end of partition (head number)
06h    1    end of partition (sector number)     [*]
07h    1    end of partition (cylinder number)   [*]
08h    4    rel. sectors (# sec. to begin of partition)
0Ch    4    number of sectors in partition
-------------------------------------------------------
```

UNASSEMBLING MBR USING DEBUG APPLICATION

We were able to unassembled, display and save the MBR to a text file in MS-DOS using Microsoft DEBUG. The following shown the commands to obtained the MBR:

```
D:\>DEBUG                   (Start DEBUG program)
-A 100                      ;Assemble the following at offset 100
xxxx:xxxx MOV AX,201        ;Read 1 sector
xxxx:xxxx MOV BX,200        ;into memory at offset 200h
xxxx:xxxx MOV CX,1          ;from Cylinder 0, Sector 1
xxxx:xxxx MOV DX,80         ;Head 0, on Drive # 80h (81h for 2nd drive,
                             82h for 3rd, etc.)
xxxx:xxxx INT 13            ;Interrupt for Disk/Diskette functions
xxxx:xxxx INT 3             ;Breakpoint instruction
                            ;Press ENTER an extra time here to return
to dash prompt
-G=100                      ;Start program execution at memory offset
100h
-D 200                      ;This will display first 128 bytes of MBR -
Optional
-RBX                        ;Length of file - High 2 bytes in BX
register
:                           ;Type in 0 at : prompt
-RCX                        ;Length of file - Low 2 bytes in CX
register
:                           ;Type in 200 at : prompt (200h = 512d bytes)
-N MBR_BIN.TXT              ;Enter a name for the file
-W 200                      ;Write BX:CX bytes to filename starting at
                             memory offset 200h
-Q                          ;Quit Debug, return to DOS command prompt
```

APPENDIX F

MUTLICLIENT VALIDATION SYSTEM PHYSICAL LAYOUT

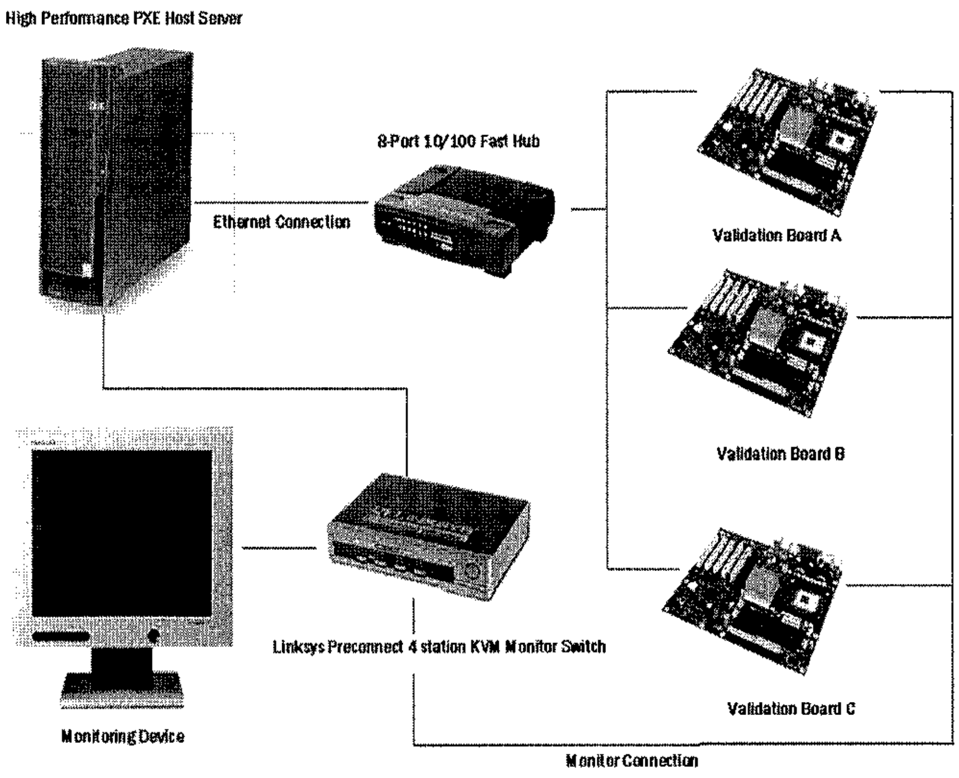
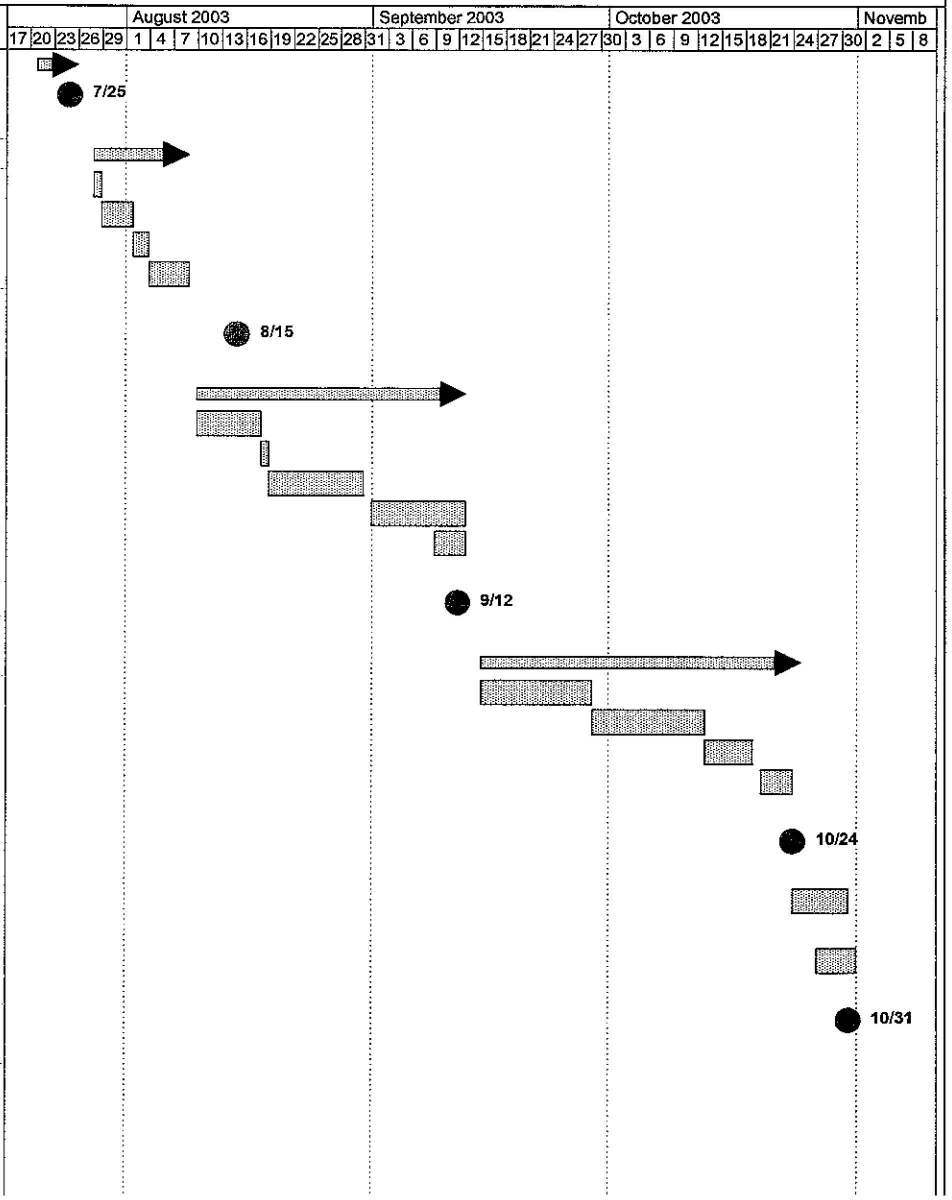


Figure 21 Multiclient Validation System Physical Layout

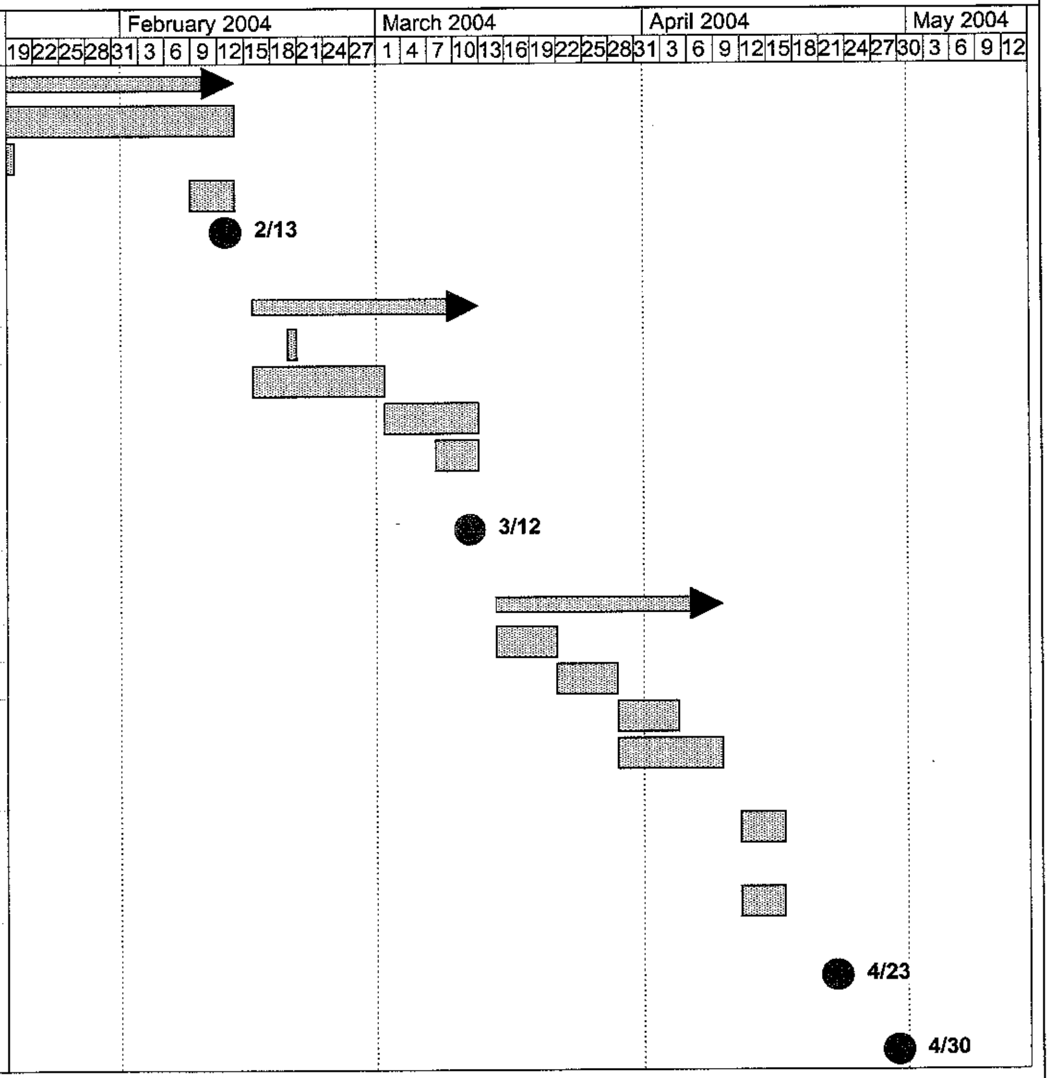
Multiclient Validation System with Multiboot Image Support on PXE
5th Year 1st Semester Gantt Chart

ID		Task Name	Duration	Start	Finish
1		***Selection of Project Topics***	5 days	Mon 7/21/03	Fri 7/25/03
2		Title: Multiclient Validation System with Multiboot Image Support on PXE	1 day	Fri 7/25/03	Fri 7/25/03
3					
4		***Preliminary Research Work***	2 wks	Mon 7/28/03	Fri 8/8/03
5		Understanding Problem Statement	1 day	Mon 7/28/03	Mon 7/28/03
6		Literacture Review on Proposed Project	4 days	Tue 7/29/03	Fri 8/1/03
7		Project Planning	1 day	Sat 8/2/03	Sun 8/3/03
8		Prepare Preliminary Report	4 days	Mon 8/4/03	Fri 8/8/03
9					
10		***Submission of Preliminary Report***	1 day	Fri 8/15/03	Fri 8/15/03
11					
12		***Project Work***	5.2 wks	Sun 8/10/03	Fri 9/12/03
13		Setup PXE Server in Windows NT4 Server environment	6 days	Sun 8/10/03	Sun 8/17/03
14		Remote Boot DOS Operating System	1 day	Mon 8/18/03	Mon 8/18/03
15		Pratices in Visual C++ and Java Programming	1.8 wks	Tue 8/19/03	Sat 8/30/03
16		Building the Software Validation Application in Java Script	2 wks	Mon 9/1/03	Fri 9/12/03
17		Prepare Progress Report	4 days	Tue 9/9/03	Fri 9/12/03
18					
19		***Submission of Progress Report***	1 day	Fri 9/12/03	Fri 9/12/03
20					
21		***Project Work Continue***	30 days?	Mon 9/15/03	Fri 10/24/03
22		Building the Partition Changing Application in Visual C++	2 wks	Mon 9/15/03	Sun 9/28/03
23		Editing the Batch Script in PXE Source Code	2 wks	Mon 9/29/03	Sun 10/12/03
24		Editing the server registry file	1 wk	Mon 10/13/03	Sat 10/18/03
25		Prepare interim Report Final Draft	0.8 wks	Mon 10/20/03	Thu 10/23/03
26					
27		***Submission of interim Report Final Draft***	1 day	Fri 10/24/03	Fri 10/24/03
28					
29		Prepare Oral Presentation	1 wk	Fri 10/24/03	Thu 10/30/03
30					
31		***Oral Presentation***	1 wk	Mon 10/27/03	Fri 10/31/03
32					
33		***Submission of Interim Report***	1 day	Fri 10/31/03	Fri 10/31/03
34					
35					
36					
37					
38					
39					



Multiclient Validation System with Multiboot Image Support on PXE
5th Year 2nd Semester Gantt Chart

ID	Task Name	Duration	Start	Finish
1	***Project Week***	20 days	Mon 1/19/04	Fri 2/13/04
2	Solving Last Semester Problem	20 days	Mon 1/19/04	Fri 2/13/04
3	Project Planning	1 day	Mon 1/19/04	Mon 1/19/04
4	Prepare Progress Report 1	5 days	Mon 2/9/04	Fri 2/13/04
5	***Submission of Progress Report 1***	1 day	Fri 2/13/04	Fri 2/13/04
6				
7	***Project Work***	4 wks	Mon 2/16/04	Fri 3/12/04
8	Confirm the Conference Details	1 day	Fri 2/20/04	Fri 2/20/04
9	Detecting Machine Address	11 days	Mon 2/16/04	Mon 3/1/04
10	Design Multiclient Support Sub-System	9 days	Tue 3/2/04	Fri 3/12/04
11	Prepare Progress Report	5 days	Mon 3/8/04	Fri 3/12/04
12				
13	***Submission of Progress Report 2***	1 day	Fri 3/12/04	Fri 3/12/04
14				
15	***Project Work Continue***	4 wks	Mon 3/15/04	Fri 4/9/04
16	System Validation	1 wk	Mon 3/15/04	Sun 3/21/04
17	Prepare Extended Abstract	1 wk	Mon 3/22/04	Sun 3/28/04
18	Prepare Exhibition	1 wk	Mon 3/29/04	Sun 4/4/04
19	Prepare Dissertation	2 wks	Mon 3/29/04	Fri 4/9/04
20				
21	Prepare Oral Presentation	1 wk	Mon 4/12/04	Fri 4/16/04
22				
23	***Oral Presentation***	1 wk	Mon 4/12/04	Fri 4/16/04
24				
25	***Submission of Dissertation***	1 day	Fri 4/23/04	Fri 4/23/04
26				
27	***Submission of Extended Abstract***	1 day	Fri 4/30/04	Fri 4/30/04



Task ▨ Milestone ◆ External Tasks ▭