Final Year Thesis
2001 /2002

WXES 3182

SmartBoard CHAT SYSTEM

BY

DAVID GAN CHYE ONG

WEK 990068

Under the supervision of :

Dr. Mazliza Othman

Moderator :

Pn Sameem Abd. Kareem

Faculty of Computer Science and Information Technology
University Malaya

ABSTRACT

"Collaboration is all about people working together. The end result is information sharing, problem solving, and support for ad hoc processes."
CSCW 2000 Conference

People need to connect with other people and business processes to get their work done. Much like the step between the telegraph to the telephone, the step to Internet technology connects people with a richer flow of information. Collaboration can bring people all over the world together with more efficient use of time and at lower cost, by eliminating the need to travel. The emergence and wide-spread adoption of the World Wide Web (WWW) offers a great deal of potential for the developers of collaborative technologies, both as an enabling infrastructure and a platform for integration with existing end-user environments.

This paper describes a system, which attempts to exploit that potential by extending the WWW to provide a set of basic facilities for collaborative information sharing. The *Smart***Board** system was conceived as an alternative tools which supports collaborative working for widely-dispersed working groups. This project focuses on building chat tool and whiteboard for collaborative computing.

We report on the design and implementation of our system, the requirements of the system and the benefits accruing from our choice of the development tools and using our experiences suggest how recent developments in the WWW point the way to more suitable and powerful mechanisms to support the development of collaborative systems.

AKNOWLEDGEMENT

First of all, I would like to greet all my readers a warm welcome. This report would never have been made possible without the assistance of all the people who gave their help in support.

First, I would like to express my deepest gratitude to Dr. Mazliza Othman for being an excellent supervisor for this project. She never fail to give me advice, correct my mistakes and allocate time for me to discuss about the project.

Special thanks also to my moderator, Pn. Sameem Abd Kareem for attending my viva presentation and also her encouragement that she gave. She has been a driving force and a helping hand in voicing out her ideas in improving my system through various ways.

Last but not least, I would also want express my utmost gratitude and appreciation to both of my project partners, Lee Tuck Khai and Yu Lian Kiong for the cooperation they have offered throughout the development of this project.

TABLE OF CONTENTS

LISTS OF FIGURES

LISTS OF TABLES

CHAPTER OVERVIEW

Chapter 1 presents an introduction to the *Smart***Board** project, where it describes briefly the project overview, motivation, modules, objectives, scope and also the project schedule.

After a brief introduction about the project, **Chapter 2** will elaborate on the studies and reviews conducted, to have a better understanding of the system. The client/ server concept is also elaborated on, covering the architecture and security issues related to a web based system. A short section of technology review is also covered in this chapter. The review will includes network protocol and Java programming language.

Next, **Chapter 3** will go into details of the system analysis. Based on the system methodologies, it is then applied to the system development together with the justification for using the methodology that was chosen. Requirements of the expected system are also elaborated here. This chapter also specifies the development tools, database server and platform that were chosen.

Subsequently, **Chapter 4** will cover the areas of system design of the proposed system, including network architecture design, functional design, system flow design and data flow design. This chapter will describe in detail of how the system will meet the requirements identified during system analysis.

When all the design are made, the development of the system will continue with the coding phase. This will be elaborated in **Chapter 5**. All the designs will be implemented into codes, classes, functions, modules and unit.

Following after that, **Chapter 6** will discuss on the testing phase of the system. There are four testing all together. They include unit testing, module testing, integration testing and module testing.

When the system is approve and tested as a whole, the system evaluation phase begins.
Chapter 7 will cover the areas of the system strength and benefits. The system limitation and future enhancement of the system is also elaborated here.

Finally, the **Conclusion** at the end of the report will conclude the report and what is expected in future of the newly completed product of *Smart***Board**.

CHAPTER 1 : INTRODUCTION

1.1 PROJECT OVERVIEW

*Smart***Board** is a web-based groupware or a collaborative tool, which enables users scattered at different locations on the Internet to communicate, interact and work upon a whiteboard. It is design to enable users to share material stored on the computer with other collaborators. Users could also share an electronic whiteboard among collaborators, so the team can diagram ideas as the meeting progresses. A chat room can complement this communication, where people can type messages to each other real-time or even have an audible conversation with each other.

1.2 PROJECT MOTIVATION

The following are the motivation aspect in developing *Smart***Board**.

Internet Advancement

Over the years, the Internet technology has evolved. The evolution has been from Web publishing to implementing Web-based applications, and continues to grow into collaboration projects. Besides the increment of information in the Internet, the number of Internet users is also growing in a very fast rate. According to the current statistics studied in year 2001, the number of Internet users has reach 500 million while the number of hosts on the Internet is 120 million [1]. For that reason, a collaborative groupware such as *Smart***Board** can be a tool that enables people all around the world to communicate and share resources virtually.

Increase of Technology

Nevertheless, the advancement of Internet depends on two main factors. In the last few years, the Internet network speed has increased tremendously. Most local networks have transitioned from twisted pair Ethernet to fast Ethernet and are now implementing or planning for even faster topologies such as Gigabit Ethernet. According to Michael Bundschuh, the Engineering Manager of Java Sun MicroSystem, by the end of 2003, Forrester Research predicts 4.5 million world subscribers will have broadband access [2].

Future computing environments will be designed to support interaction with people at least as much as with computers. The Computer Industry Almanac reports growth from 98 million PCs in 1990 to 222 million in 1995 and over 364 million in 1998 [1]. With the increase in the use of computers and Internet for human interaction, the drive is towards developing a collaborative and distributed computing environment.

Less delay, Less cost, Less effort

In today's technology, many are searching and developing a good and easy system that enables people from all over the world to communicate and work on something together in a wide network. In this fast pace growth of technology, we speak of efficiency, quick and real-time with less delay, less cost and less effort. A collaborative groupware is one important technology which brings us one step towards this goal.

People do not need to travel from afar just to meet together. They can just sit in front of their computers, at home or in the office, to conduct any meeting virtually. Since *Smart***Board** was proposed to be a free web-based application, users from different location can collaborate with each other on real-time basis without the need to pay for any access charge. Users even do not need to pay for expensive international calls to communicate with other users from different parts of the globe.

Multimedia Technology

The multimedia industry promises to be one of the fastest growing industry clusters in the coming years. For a collaborative tool such as *Smart***Board**, multimedia technology is important to combine text, audio and graphic together to provide quality communication services to users. Although some great new multimedia tools and toys will certainly be heading down the pike in the coming months and years, an impressive batch of programs that work multimedia wonders on the Internet is available today. Some of the tools and toys described include video conferencing, voice mail, voice chat, animation, voice over IP and whiteboard. This motivates the development of an application that presents a few of these multimedia tools to the Internet users. An example of such application is *Smart***Board,** which comprise of text chat, voice chat, message board and whiteboard technologies.

The Targeted Audience

The targeted users have also become the main motivation for developing *Smart***Board**. Three different types of users have been identified:

1. ***Professional Working Users***

*Smart***Board** can be used by the professional working people to conduct their meeting in a more efficient way. The delay and cost of physically meeting together can be reduced tremendously because meetings are done in a virtual and real-time environment. Users can share their ideas or proposed their design through this collaborative tool.

2. ***Educational Users***

*Smart***Board** can also function as a virtual class, where a teacher can teach art class or English class to students online.

3. ***Social Users***

*Smart***Board** can be a place of social interaction. Users can meet their friends or make new friends over the Internet. Kids can use it as a drawing playground or a friend may use it to show the map to their house to other friends.

1.3 PROJECT MODULES

The development of *Smart***Board** will be divided into three main modules. Even though, these modules are to be built separately by three different developers, the development of each module is done simultaneously. These three modules are as follows:

1. **Group Management**

Group management is a sub-system of *Smart***Board** that stores information about the groups and group members profile into a database. Group registration process and group administration systems are to be developed. Group security will be also be implemented in this module.

2. **Electronic whiteboard**

Electronic whiteboard is an application available in *Smart***Board**. Drawing tools and a drawing field will be developed here. Users can draw on the drawing field and be seen by other users. Users can also edit on the picture drawn.

3. **Real time chat**

Real-time chat is a communication tools for users in *Smart***Board**. Users will be able to communicate using text chat or voice chat.

This paper will only discuss on the real-time chat system development.

1.4 PROJECT OBJECTIVE STATEMENT

An objective statement of a project is the goal and vision that the project aims to achieve.

"The objective of this project is to design and develop a distributed text and audio chat system that allows multiple users to engage in real-time chat session so as to support the articulation and negotiation of ideas for working on a whiteboard and also to support communication in general"

*Smart**Board** Chat Developer*

1.5 PROJECT SCOPE

A project scope is the focus and limitation on what need to be developed in a system. For the real-time chat system, these are the three main scopes of the development.

1. **Text Chat Tools**

   Text chat tools are user-friendly tools that enable users to send text messages to other users in the same group. The tools may comprise of text fonts, text colors, animation gestures and so on. Group chat and private chat are also made available for users.

2. **Voice Chat Tools**

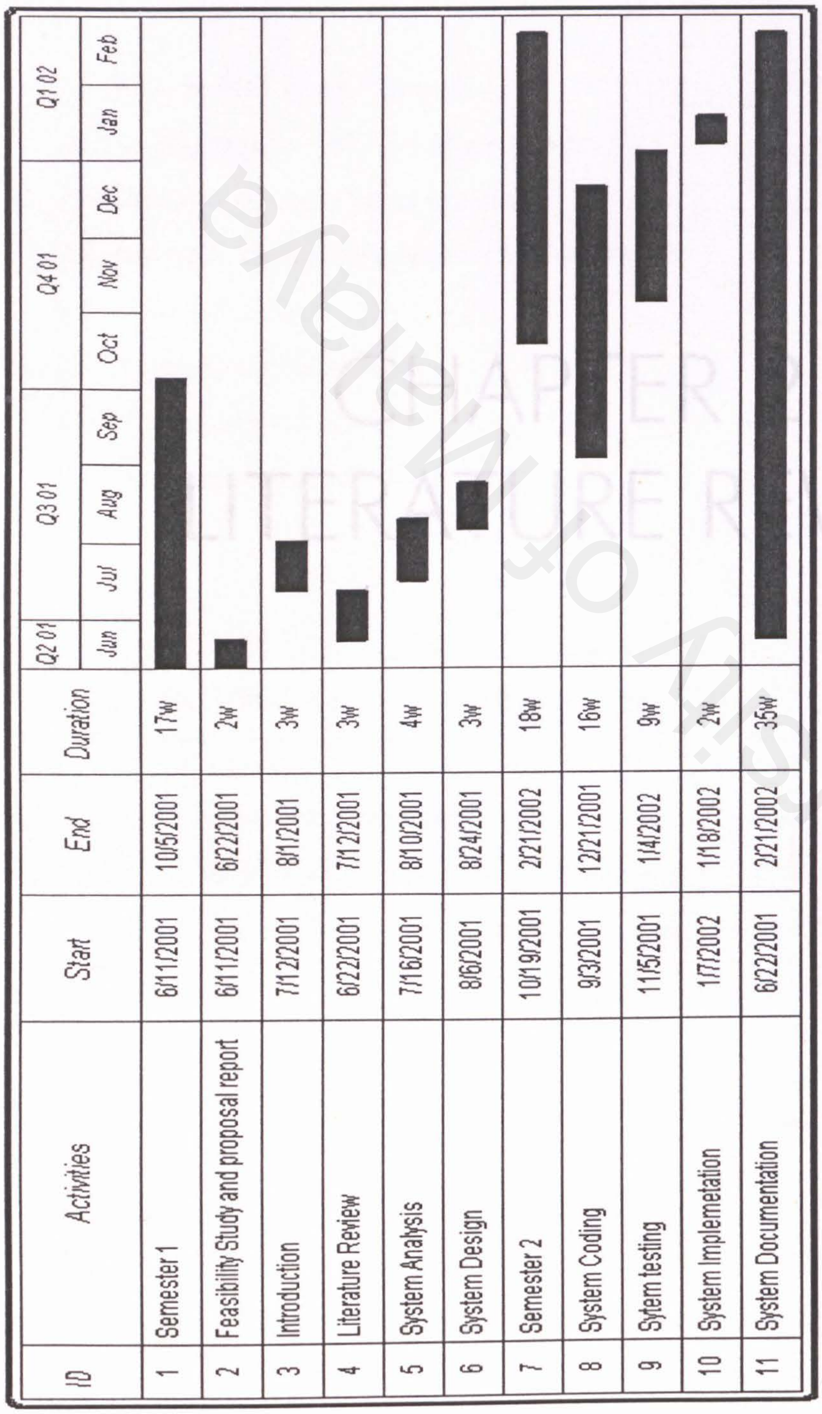
   Voice chat tools enable users to communicate by speaking through a microphone and hearing through a speaker or a headset. A voice analyzer will be build to enable users to adjust the sending and receiving voice sound. Besides that, a help feature will be included to guide the users in using the voice chat.

3. **Network for distributed computing over the Internet**

   The project will also focus on developing a proper network to support the distributed architecture of chat system. This development involves programming the chat server and chat client software.

1.6 PROJECT SCHEDULE

A time-line schedule development of the project is shown in the *Figure 1* below:



ID	Activities	Start	End	Duration
1	Semester 1	6/11/2001	10/5/2001	17w
2	Feasibility Study and proposal report	6/11/2001	6/22/2001	2w
3	Introduction	7/12/2001	8/1/2001	3w
4	Literature Review	6/22/2001	7/12/2001	3w
5	System Analysis	7/16/2001	8/10/2001	4w
6	System Design	8/6/2001	8/24/2001	3w
7	Semester 2	10/19/2001	2/21/2002	18w
8	System Coding	9/3/2001	12/21/2001	16w
9	Sytem testing	11/5/2001	1/4/2002	9w
10	System Implemetation	1/7/2002	1/18/2002	2w
11	System Documentation	6/22/2001	2/21/2002	35w

INTRODUCTION

Literature review is a study or research done to build a system. It is the relevant information, knowledge and facts that were gathered through many resources like the Internet, books, magazines and interviews. This chapter will elaborates on the following:

2.1 Introduction to real-time chat
2.2 Introduction to CSCW and groupware
2.3 Case studies on existing system
2.4 Network architecture review: Client / server
2.5 Stream Media
2.6 Network protocol review
2.7 Voice Bandwidth review
2.8 Programming tools review: Java

CHAPTER 2 : LITERATURE REVIEW

2.1 INTRODUCTION TO REAL-TIME CHAT

2.1.1 What is 'real-time'?

Real-time is a "live" session [illegible] that refers to process control or embedded system that must have immediate response. It can also be define as fast transaction processing as well as any electronic operation that [illegible] to keep up with its real-world counterpart (animating complex images [illegible]) [3]

2.1.2 What is 'chat'?

Chat or a chat session is a real-time synchronous conferencing system. It is a form of communication tool where people at different locations connected to a computer network meet [illegible] like a normal conversation, a *chat* can be a form of [illegible] project communication tools. [3]

INTRODUCTION

Literature review is a study or research done to build a system. It is the relevant information, knowledge and facts that were gathered through many resources like the Internet, books, magazines and interviews. This chapter will elaborates on the following:

2.1 Introduction to real-time chat
2.2 Introduction to CSCW and groupware
2.3 Case studies on existing system
2.4 Network architecture review: Client / server
2.5 Stream Media
2.6 Network protocol review
2.7 Voice Bandwidth review
2.8 Programming tools review – Java

2.1 INTRODUCTION TO REAL-TIME CHAT

2.1.1 What is 'real-time'?

Real-time is a "live" session environment that refers to process control or embedded system that must have immediate response. It can also be define as fast transaction processing as well as any electronic operation fast enough to keep up with its real-world counterpart (animating complex images, transmitting live video, etc.). [3]

2.1.2 What is 'chat' ?

Chat aka a chat system is a real-time synchronous conferencing system. It is a form of communication tool where people at different locations connected to a computer network meet talk together in a virtual environment. Just like a normal conversation, a *chat* can be a form of socialization tools, business communication tools or project communication tools. [3]

2.1.3 Types of chat

With the advancement of computer network, chat has become a very popular technology in the past few years. Typically, there are three types of real-time chat. They are text based chat, voice chat and video conferencing.

Text chat are communications between users, typically by typing text and sending text message to one another on real-time basis.

Voice chat or *audio conferencing* enables two or more people to use the computer as a telephone conferencing system. Users have their conversation using a microphone and a speaker or a headset.

Video Conferencing works like a portal mirror where not only can users hear the voice of the other person, but can also see the real time image of the person on the user's screen.

There are two forms of chat, a group chat and a private chat.

2.1.4 Group Chat

Group Chat consists of chat users that have the same interest or somewhat same identities and characteristic. A user can form or join a group of users in a chat channel. This may be popular if a group of users only want to communicate among themselves. Their conversations are only within the boundary of the group. For example, a group of recognized friends can form their own group chat. Users who have interest in sports or users who have interest in music can form their own group chat too. In *Smart***Board**, a group chat consists of group members working on a same whiteboard project.

2.1.5 Private Chat

Private Chat is a smaller scale of chat communication where it only involves two users across the network. This one-to-one conversation is personal and secure. The term *private* can be used as a means of chatting personally with another user. For example, user A would want to *private* with user B.

2.2 INTRODUCTION TO CSCW and GROUPWARE

Computer-supported cooperative work (CSCW) and Groupware has emerged as one of the new and important technology in the computer world recently. CSCW is a multidisplinary research field including computer science, economics, sociology, and psychology. CSCW research focuses on developing new theories and technologies for coordination of groups of people who work together [4]. Two notions and technologies that help to satisfy CSCW goals:

- **WYSIWIS**

  This stands for What You See Is What I See. It is analogous to two people each at their own homes watching the same television show at the same time. Computer technology extends this concept and allows people to interact and communicate in a WYSIWIS environment.

- **Collaboration**

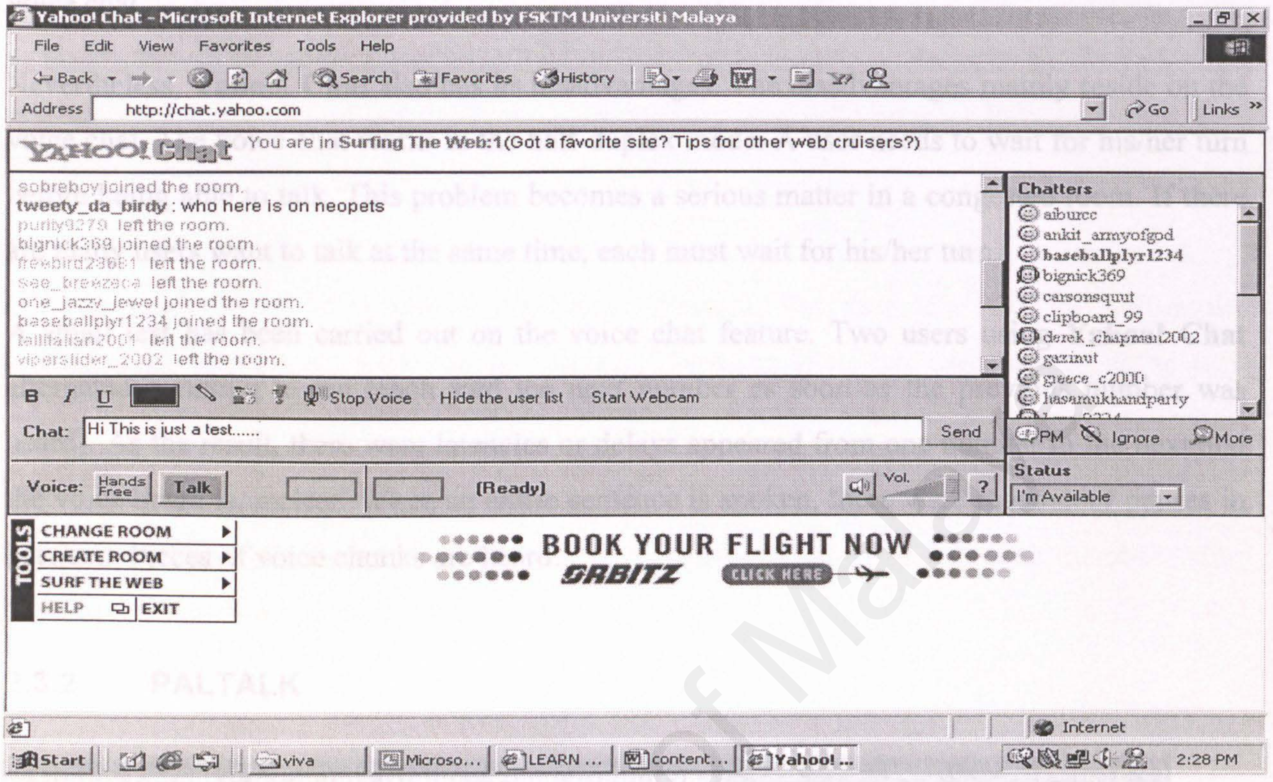
  Members of the group who are in physically different locations able to contribute to group activities such that the group performs as good or better compared to having all members in the same physical location.

Groupware is a computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment. So, while CSCW is the research, groupware is the product, such as *Smart***Board**.

2.3 CASE STUDIES ON EXISTING SYSTEM

Today, there exist several software solutions that have employ real-time communication technologies over the Internet. Few of the examples of such solutions are the Yahoo! Chat, HearMe, Paltalk, IRC and ICQ. This part of the chapter will review on two of these solutions and how *Smart***Board** can enhance the features available.

2.3.1 Yahoo! Chat


Yahoo! Chat - Microsoft Internet Explorer provided by FSKTM Universiti Malaya
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://chat.yahoo.com Go Links
You are in Surfing The Web:1(Got a favorite site? Tips for other web cruisers?)
B I U Stop Voice Hide the user list Start Webcam
Chat: Send
Voice: Hands Free Talk (Ready) Vol. ?
Chatters
PM Ignore More
Status I'm Available
TOOLS CHANGE ROOM CREATE ROOM SURF THE WEB HELP EXIT
BOOK YOUR FLIGHT NOW ORBITZ CLICK HERE
Internet
Start viva Microso... LEARN ... content... Yahoo! ... 2:28 PM

sobreboy joined the room.
tweety_da_birdy : who here is on neopets
purity9279 left the room.
bignick369 joined the room.
freebird23681 left the room.
see_breezeca left the room.
one_jazzy_jewel joined the room.
baseballplyr1234 joined the room.
tallitalian2001 left the room.
vipersllder_2002 left the room.

Chat: Hi This is just a test......

Chatters: aiburcc, ankit_armyofgod, **baseballplyr1234**, bignick369, carsonsquut, clipboard_99, derek_chapman2002, gazinut, geece_c2000, hasamukhandparty

Figure 2.1 : Yahoo! Chat Interface

Figure 2.1 displays a screen snapshot of **Yahoo! Chat** interface. Yahoo! Inc has long been a famous website for Internet users to find information and services. It started from being just a simple search engine to a rich feature website for Internet users. One of the new features is the **Yahoo! Chat**. **Yahoo! Chat** enables a new user to join an interest group from a selective list and to chat together via the voice chat and text chat. A user can create his/her own room or change from one room to another, according to the user's interest.

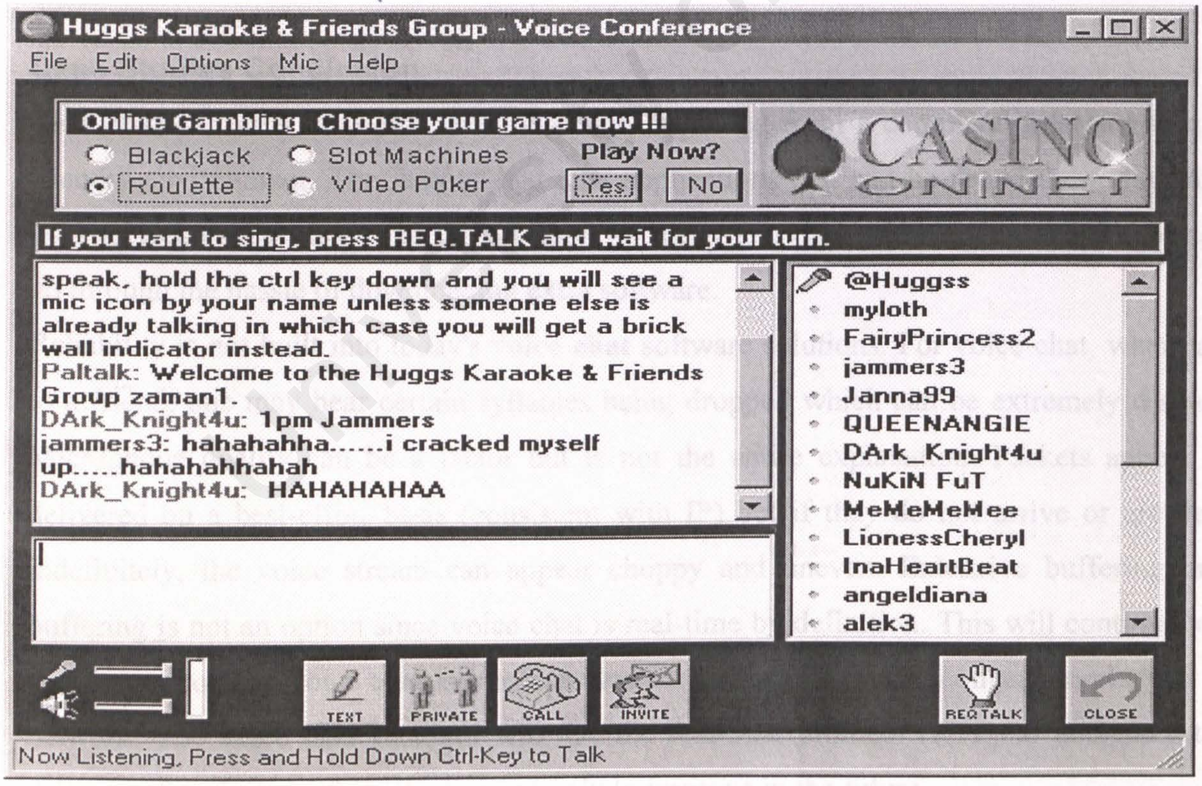
Yahoo! Chat is a very user-friendly tool. It consists of many types of chat rooms where users can select. For the text messaging, several functions available for users are the text font, text colors and gestures icon. Users can have private chat with other users, ignore messages from other users and hide the users list.

For the voice chat feature, users can press a Talk button to talk or use the "hands free" feature that will detect the voice of users. **Yahoo! Chat** also provides a wizard or a step-by-step guide for users to set their computer configuration so that it will be able to support the use of the voice chat.

Nevertheless, **Yahoo! Chat** also has its disadvantages. The disadvantages mainly reside on the voice chat. The voice chat works in the half-duplex mode. A user needs to wait for his/her turn before being able to talk. This problem becomes a serious matter in a congested room. If there are many users want to talk at the same time, each must wait for his/her turn.

A small test has been carried out on the voice chat feature. Two users using **Yahoo! Chat** alternated counting to ten (each said the next number as soon as the previous number was heard). As the result, there were latencies or delays appeared from one number to the next and the voice heard is unclear. When an entire sentence is spoken, there were mandatory pauses in between. Pieces of voice chunks are heard.

2.3.2 PALTALK


Huggs Karaoke & Friends Group - Voice Conference
File Edit Options Mic Help
Online Gambling Choose your game now !!!
Blackjack
Slot Machines
Roulette
Video Poker
Play Now?
Yes
No
CASINO ONINET
TEXT
PRIVATE
CALL
INVITE
REQ TALK
CLOSE

If you want to sing, press REQ.TALK and wait for your turn.

speak, hold the ctrl key down and you will see a mic icon by your name unless someone else is already talking in which case you will get a brick wall indicator instead.
Paltalk: Welcome to the Huggs Karaoke & Friends Group zaman1.
DArk_Knight4u: 1pm Jammers
jammers3: hahahahha......i cracked myself up.....hahahahhahah
DArk_Knight4u: HAHAHAHAA

@Huggss
myloth
FairyPrincess2
jammers3
Janna99
QUEENANGIE
DArk_Knight4u
NuKiN FuT
MeMeMeMee
LionessCheryl
InaHeartBeat
angeldiana
alek3

Now Listening, Press and Hold Down Ctrl-Key to Talk

Figure 2.2 : Paltalk Interface

Paltalk is another example of real-time chat. Just like Yahoo! Chat, **Paltalk** also offers a wide range of chat rooms for its users. Users can invite friends by sending an e-mail invitation to join in their room. A PC-to-phone or PC-to-PC call is made possible in **Paltalk**. So, **Paltalk** users will have the advantage of not needing to pay for telephone bills because **Paltalk** is free, even for international calls.

Unlike the Yahoo! Chat, **Paltalk** voice feature offers a better voice quality. It has less delays and the voice heard is clearer. Nevertheless, **Paltalk** also face the issue of half-duplex conversation. Users need to wait for turn to talk. There can be no two users talking at the same time.

Another disadvantage of **Paltalk** is also the need to download its software before being able to use it. The download takes time and users need to register themselves as Paltalk member before starting the download process.

Case Studies Conclusion

Text chat software solutions have long been improved from a command interface to a user-friendly environment. The famous text chat applications that can be found these days are the Internet Relay Chat (IRC) and ICQ. Today, for many text chat application, users do not need to go through the hassle of downloading extra software.

Reliability is not built into today's **voice chat** software solutions. For voice chat, when talking to a friend, one may hear certain syllables being dropped which can be extremely distracting. Microphone quality can be a factor but is not the entire explanation. Packets appear to be delivered on a best-effort basis (consistent with IP) but if they do not arrive or are delayed indefinitely, the voice stream can appear choppy and uneven. Extensive buffering or pre-buffering is not an option since voice chat is real-time by definition. This will continue to be a problem as long as voice chat is implemented across today's public, congestion-plagued inter-network. The voice chat standard will use the real-time protocol (RTP) to guarantee timely delivery of packets, so this situation is likely to improve in the future.

2.4 NETWORK ARCHITECTURE REVIEW – CLIENT / SERVER

Besides the software application, the network of the system is also an important criteria in CSCW and groupware such as *Smart***Board**. Here, the architecture of a client/server network is identified and reviewed.

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request.

The client processes the user interface (Windows, Mac, etc.) and can perform some or all of the application processing. Servers range in capacity from high-end PCs to mainframes. A database server maintains the databases and processes requests from the client to extract data from or to update the database. An application server provides additional business processing for the clients.

For the usual client/server model, one server, sometimes called a daemon, is activated and awaits client requests. Typically, multiple client programs share the services of a common server program. Both client programs and server programs are often part of a larger program or application.

In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Relative to the Internet, a web browser is a client program that is responsible in sending the request of the client in HTML form to the web-server. Two types of web browsers that are commonly used are the Internet Explorer and Netscape Navigator. Web-server responsibility is to deliver HTML to the client browser which request for the information.

Similarly, a computer with TCP/IP installed allows users to make client requests for files from File Transfer Protocol (File Transfer Protocol) servers in other computers on the Internet.

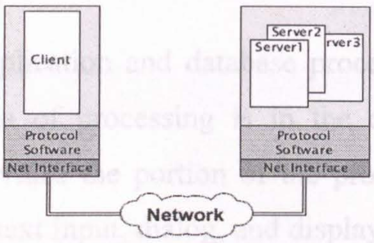


Figure 2.3 : Client / server architecture

For example, in the *Smart***Board** application, client may want to use the chat feature. A request will be made to the server. The server will, then, establish a connection with the client and also with other user whom the client wants to chat with.

In the following, a comparison of a single-tier host system example, a two-tier client server and a three-tier client server are discussed.

2.4.1 Single-Tier Host System

In the scenario in *Figure 2.3*, although there are clients and servers this is not "true" client/server, because the server is nothing more than a remote disk drive, and the client does all the processing. Lengthy searches can bog down the network, because each client has to read the entire database. At 1,000 bytes per record, a database with 100,000 records sends 100MB over the LAN.

From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.

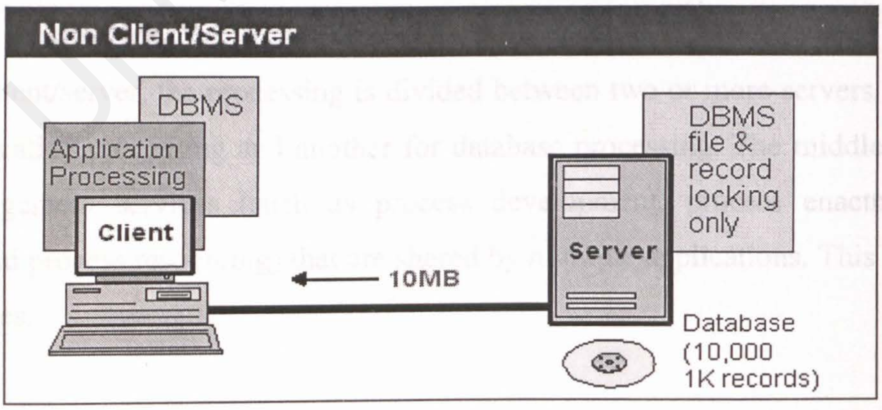


Figure 2.4: Single-Tier Host System

2.4.2 Two-tier client/server

In two-tier client/server, the application and database processing are done in the file server. Most of the application portion of processing is in the client environment. The database management server usually provides the portion of the processing related to accessing data. User interface (such as session, text input, dialog, and display management services) remains at the client side.

A SQL request is generated in the client and transmitted to the server. The DBMS searches for records in the server and returns only matching records to the client. If 50 records met the criteria in our 100,000-record example, only 50K would be transmitted over the LAN.

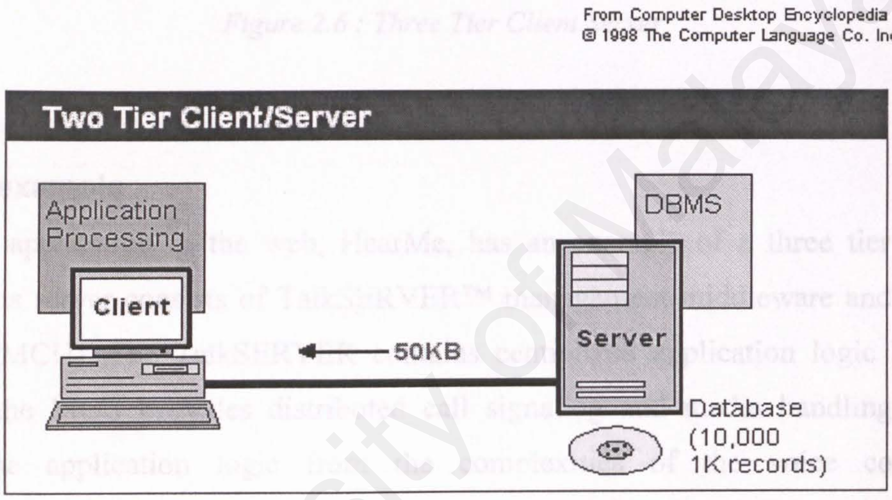

From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.
Two Tier Client/Server
Application Processing
Client
50KB
DBMS
Server
Database (10,000 1K records)

Figure 2.5 : Two Tier Client/Server

2.4.3 Three-tier client/server

In three-tier client/server, the processing is divided between two or more servers, one typically used for application processing and another for database processing. The middle tier provides process management services (such as process development, process enactment, process monitoring, and process resourcing) that are shared by multiple applications. This is common in large enterprises.

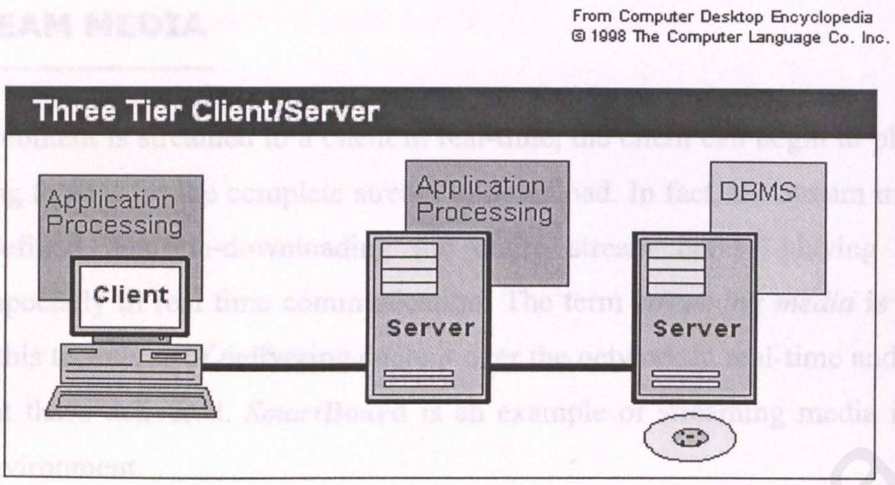


Figure 2.6 : Three Tier Client Server

"HearMe" example

A voice chat application in the web, HearMe, has an example of a three tier client/server architecture. Its server consists of TalkSERVER™ management middleware and a multipoint control unit (MCU). The TalkSERVER contains centralized application logic and platform APIs, while the MCU provides distributed call signaling and media handling capabilities. Separating the application logic from the complexities of the voice communication infrastructure allows application to supports point-to-point calling and multipoint conferencing.

The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve *usability*, *flexibility*, *interoperability*, and *scalability* as compared to centralized, mainframe, time sharing computing.

2.5 STREAM MEDIA

When media content is streamed to a client in real-time, the client can begin to play the stream without having to wait for the complete stream to download. In fact, the stream might not even have a predefined duration-downloading the entire stream before playing it would be impossible especially in real time communication. The term *streaming media* is often used to refer to both this technique of delivering content over the network in real-time and the real-time media content that's delivered. *Smart***Board** is an example of streaming media in a real-time groupware environment.

2.6 NETWORK PROTOCOL REVIEW

A literature review about network protocol is done, also, during this phase. A network protocol can be defined as a communication protocol used by a network. Three types of network protocol that will be reviewed here which is the TCP/IP, UDP and the RTP.
Transmitting media data across the net in real-time requires high network throughput. It is easier to compensate for lost data than to compensate for large delays in receiving the data. This is very different from accessing static data such as a file, where the most important thing is that all of the data arrive at its destination. Consequently, the protocols used for static data do not work well for streaming media.

2.6.1 Transmission Control Protocol (TCP)

The HTTP and FTP protocols are based on the Transmission Control Protocol (TCP). TCP is a transport-layer protocol designed for reliable data communications on low-bandwidth, high-error-rate networks. TCP ensures that a message is sent accurately and in its entirety. When a packet is lost or corrupted, it is retransmitted. The overhead of guaranteeing reliable data transfer slows the overall transmission rate. For this reason, underlying protocols other than TCP are typically used for streaming media.

2.6.2 User Datagram Protocol (UDP)

One that is commonly used is the User Datagram Protocol (UDP). UDP is an unreliable protocol; it does not guarantee that each packet will reach its destination. There is also no guarantee that the packets will arrive in the order that they were sent. The receiver has to be able to compensate for lost data, duplicate packets, and packets that arrive out of order. Like TCP, UDP is a general transport-layer protocol--a lower-level networking protocol on top of which more application-specific protocols are built.

2.6.3 Real Time Protocol (RTP)

RTP is defined in IETF RFC 1889, a product of the AVT working group of the Internet Engineering Task Force (IETF). RTP provides end-to-end network delivery services for the transmission of real-time data. RTP is network and transport-protocol independent, though it is often used over UDP. *Figure 2.7* depicts the RTP header architecture.

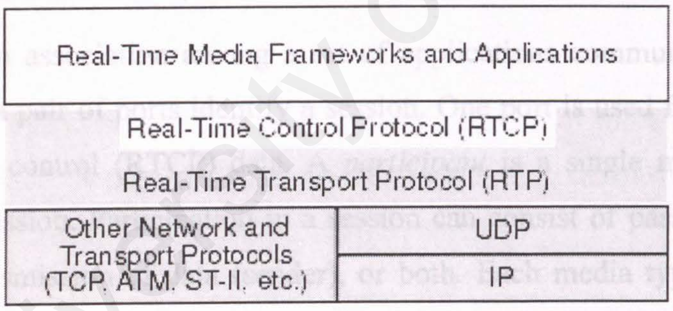

Real-Time Media Frameworks and Applications
Real-Time Control Protocol (RTCP)
Real-Time Transport Protocol (RTP)
Other Network and Transport Protocols (TCP, ATM, ST-II, etc.)
UDP
IP

Figure 2.7: RTP header architecture.

RTP can be used over both unicast and multicast network services. Over a *unicast* network service, separate copies of the data are sent from the source to each destination. Over a *multicast* network service, the data is sent from the source only once and the network is responsible for transmitting the data to multiple locations. Multicasting is more efficient for many multimedia applications, such as *Smart***Board** chat system. The standard Internet Protocol (IP) supports multicasting.

2.6.3.1 RTP Services

RTP enables user to identify the type of data being transmitted, determine what order the packets of data should be presented in, and synchronize media streams from different sources. RTP data packets are not guaranteed to arrive in the order that they were sent--in fact, they are not guaranteed to arrive at all. It is up to the receiver to reconstruct the sender's packet sequence and detect lost packets using the information provided in the packet header.

While RTP does not provide any mechanism to ensure timely delivery or provide other quality of service guarantees, it is augmented by a control protocol (RTCP) that enables developers to monitor the quality of the data distribution. RTCP also provides control and identification mechanisms for RTP transmissions. If quality of service is essential for a particular application, RTP can be used over a resource reservation protocol that provides connection-oriented services.

2.6.3.2 RTP Architecture

An RTP *session* is an association among a set of applications communicating with RTP. A network address and a pair of ports identify a session. One port is used for the media data and the other is used for control (RTCP) data. A *participant* is a single machine, host, or user participating in the session. Participation in a session can consist of passive reception of data (receiver), active transmission of data (sender), or both. Each media type is transmitted in a different session.

For example, if both audio and video are used in a conference, one session is used to transmit the audio data and a separate session is used to transmit the video data. This enables participants to choose which media types they want to receive--for example, someone who has a low-bandwidth network connection might only want to receive the audio portion of a conference.

2.6.3.3 Data Packets

The media data for a session is transmitted as a series of packets. A series of data packets that originate from a particular source is referred to as an *RTP stream*. Each RTP data packet in a stream contains two parts, a structured header and the actual data (the packet's *payload*).

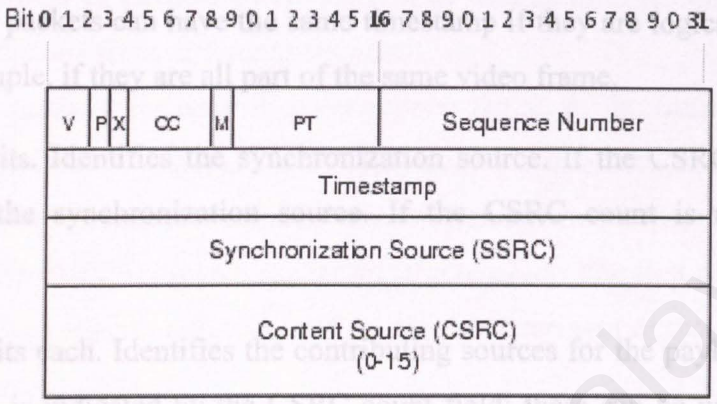

Bit 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 16 7 8 9 0 1 2 3 4 5 6 7 8 9 0 31
V
P
X
CC
M
PT
Sequence Number
Timestamp
Synchronization Source (SSRC)
Content Source (CSRC)
(0-15)

Figure 2.8: RTP data-packet header format.

The header of an RTP data packet contains:

- **The RTP version number (V)**: 2 bits. The version defined by the current specification is two.
- **Padding** (P): 1 bit. If the padding bit is set, there are one or more bytes at the end of the packet that are not part of the payload. The very last byte in the packet indicates the number of bytes of padding. Some encryption algorithms use the padding.
- **Extension** (X): 1 bit. If the extension bit is set, the fixed header is followed by one header extension. This extension mechanism enables implementations to add information to the RTP Header.
- **CSRC Count** (CC): 4 bits. The number of CSRC identifiers that follow the fixed header. If the CSRC count is zero, the synchronization source is the source of the payload.
- **Marker (M)**: 1 bit. A marker bit defined by the particular media profile.
- **Payload Type** (PT): 7 bits. An index into a media profile table that describes the payload format. The payload mappings for audio and video are specified in RFC 1890.

- **Sequence Number**: 16 bits. A unique packet number that identifies this packet's position in the sequence of packets. The packet number is incremented by one for each packet sent.

- **Timestamp**: 32 bits. Reflects the sampling instant of the first byte in the payload. Several consecutive packets can have the same timestamp if they are logically generated at the same time--for example, if they are all part of the same video frame.

- **SSRC**: 32 bits. Identifies the synchronization source. If the CSRC count is zero, the payload source is the synchronization source. If the CSRC count is nonzero, the SSRC identifies the mixer.

- **CSRC**: 32 bits each. Identifies the contributing sources for the payload. The number of contributing sources is indicated by the CSRC count field; there can be up to 16 contributing sources. If there are multiple contributing sources, the payload is the mixed data from those sources

2.6.3.4 Control Packets

In addition to the media data for a session, control data (RTCP) packets are sent periodically to all of the participants in the session. RTCP packets can contain information about the quality of service for the session participants, information about the source of the media being transmitted on the data port, and statistics pertaining to the data that has been transmitted so far.

There are several types of RTCP packets:

- Sender Report
- Receiver Report
- Source Description
- Bye
- Application-specific

RTCP packets are "stackable" and are sent as a compound packet that contains at least two packets, a report packet and a source description packet. All participants in a session send RTCP packets.

A participant that has recently sent data packets issues a *sender report*. The sender report (SR) contains the total number of packets and bytes sent as well as information that can be used to synchronize media streams from different sessions.

Session participants periodically issue *receiver reports* for all of the sources from which they are receiving data packets. A receiver report (RR) contains information about the number of packets lost, the highest sequence number received, and a timestamp that can be used to estimate the round-trip delay between a sender and the receiver. The first packet in a compound RTCP packet has to be a report packet, even if no data has been sent or received--in which case, an empty receiver report is sent.

All compound RTCP packets must include a source description (SDES) element that contains the canonical name (CNAME) that identifies the source. Additional information might be included in the source description, such as the source's name, email address, phone number, geographic location, application name, or a message describing the current state of the source.

When a source is no longer active, it sends an RTCP BYE packet. The BYE notice can include the reason that the source is leaving the session.

RTCP APP packets provide a mechanism for applications to define and send custom information via the RTP control port.

2.6.3.5 RTP Header Compressor

To lower the usage of bandwidth of voice transmission, the compression method is suggested. When text and financial data are compressed, they must be decompressed back to a perfect original, bit for bit. This is known as "lossless compression." However, audio and video can be compressed to as little as 5% of its original size using "lossy compression." Some of the data is actually lost, but the loss is not noticeable to the human ear and eye. *Figure 2.6.3.5* shows the compression of a RTP packet.

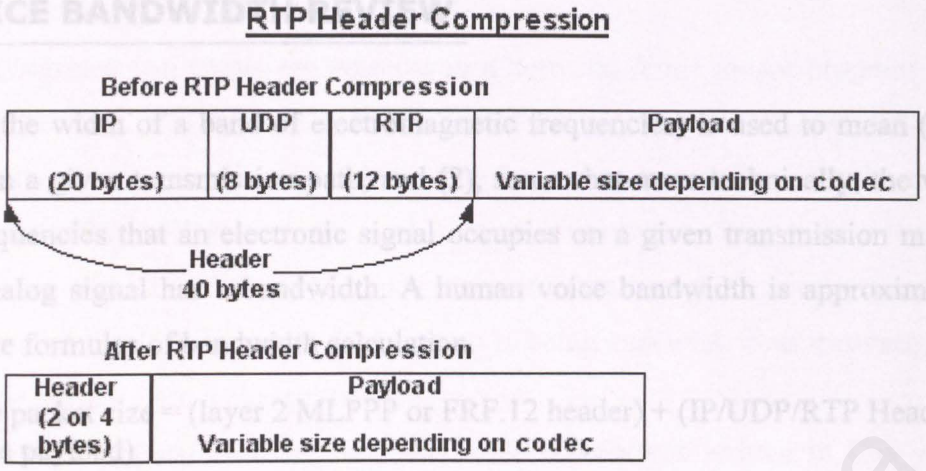

RTP Header Compression
Before RTP Header Compression
IP (20 bytes)
UDP (8 bytes)
RTP (12 bytes)
Payload Variable size depending on codec
Header 40 bytes
After RTP Header Compression
Header (2 or 4 bytes)
Payload Variable size depending on codec

Figure 2.9: RTP Packet Compression

2.6.3.6 RTP Applications

RTP applications are often divided into those that need to be able to receive data from the network (RTP Clients) and those that need to be able to transmit data across the network (RTP Servers). Some applications do both--for example, conferencing applications capture and transmit data at the same time that they are receiving data from the network.

2.6.3.7 Transmitting Media Streams Across the Network

RTP server applications transmit captured or stored media streams across the network. For example, in a voice chat application, a media stream might be captured from a microphone and sent out on one or more RTP sessions. The media streams might be encoded in multiple media formats and sent out on several RTP sessions for conferencing with heterogeneous receivers. Multiparty conferencing could be implemented without IP multicast by using multiple unicast RTP sessions.

2.7 VOICE BANDWIDTH REVIEW

Bandwidth (the width of a band of electromagnetic frequencies) is used to mean (1) how fast data flows on a given transmission path, and (2), somewhat more technically, the width of the range of frequencies that an electronic signal occupies on a given transmission medium. Any digital or analog signal has a bandwidth. A human voice bandwidth is approximately 3kHz. Below are the formulas of bandwidth calculation.

- voice packet size = (layer 2 MLPPP or FRF.12 header) + (IP/UDP/RTP Header) + (voice payload)
- voice packets per second (pps) = codec bit rate / voice payload size
- bandwidth = voice packet size * pps

MLPPP - Multilink Point-to-Point protocol

FRF - Frame Relay Forum [5]

2.8 PROGRAMMING TOOLS REVIEW - Java

A study has been conducted on the existing programming tools that can be used to implement the desired system. Programming language such as C++, HTML, ASP, Visual Basic, Java, Lotus and much more are the most common ones available. Nevertheless, the system developer has identified one suitable programming tool that may meet the needs of the system. The following is the literature review on Java.

Java is a programming language expressly designed for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces an object-oriented programming model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

The major characteristics of Java are:

- The programs you create are portable in a network. Your source program is compiled into what Java calls byte code, which can be run anywhere in a network on a server or client that has a Java virtual machine. The Java virtual machine interprets the byte code into code that will run on the real computer hardware. This means that individual computer platform differences such as instruction lengths can be recognized and accommodated locally just as the program is being executed. Platform-specific versions of your program are no longer needed.
- The code is robust, here meaning that, unlike programs written in C++ and perhaps some other languages, the Java objects can contain no references to data external to themselves or other known objects. This ensures that an instruction can not contain the address of data storage in another application or in the operating system itself, either of which would cause the program and perhaps the operating system itself to terminate or "crash." The Java virtual machine makes a number of checks on each object to ensure integrity.
- Java is object-oriented, which means that, among other characteristics, an object can take advantage of being part of a class of objects and inherit code that is common to the class. Objects are thought of as "nouns" that a user might relate to rather than the traditional procedural "verbs." A method can be thought of as one of the object's capabilities or behaviors.
- In addition to being executed at the client rather than the server, a Java applet has other characteristics designed to make it run fast.
- Relative to C++, Java is easier to learn.

2.8.1 Java Networking

One of Java's major strong suits as a programming language is of its wide range of network support. Java has this advantage because it was develop with the Internet in mind. There are three types of java networking that supports the technology distributed computing. They are RMI, CORBA and Socket.

2.8.1.1 RMI

Remote Method Invocation (RMI) defines a model of distributed computing. Its purpose is to allow local invocation of operations on objects located anywhere on a network. New code can be sent across a network and dynamically loaded at run-time by foreign virtual machines. Java developers have a greater freedom when designing distributed systems, and the ability to send and receive new classes is an incredible advantage. Developers do not have to work within a fixed code base - they can submit new classes to foreign virtual machines and have them perform different tasks. When working with remote services, RMI clients can access new versions of Java services as they are made available – there is no need to distribute code to all the clients that might wish to connect. While code can be accessed from a local or remote file-system, it can also be accessed via a web server, making distribution easier. RMI also supports a registry, which allows clients to perform lookups for a particular service. The following diagram shows the interaction between different components of an RMI system. Clients that know about a service can look up its location from a registry and access the service. If a new class is required, it can be downloaded from a web server [7].

2.8.1.2 CORBA

CORBA (Common Object Request Broker Architecture) is the standard distributed object architecture developed by the Object Management Group (OMG) consortium. It is a competing distributed systems technology that offers greater portability than remote method invocation. Unlike RMI, CORBA is not tied to one language, and as such, can integrate with legacy systems of the past written in older languages, as well as future languages that include support for CORBA. CORBA is not tied to a single platform (a property shared by RMI), and shows great potential for use in the future. That said, for Java developers, CORBA offers less flexibility, because it doesn't allow executable code to be sent to remote systems. Under communication between CORBA clients and CORBA services, method calls are passed to Object Request Brokers (ORBs). These ORBs communicate via the Internet Inter-ORB Protocol (IIOP). IIOP transactions can take place over TCP streams, or via other protocols (such as HTTP), in the event that a client or server is behind a firewall [8]. The following diagram in *Figure 2.7.1* shows a client and a servant communicating.

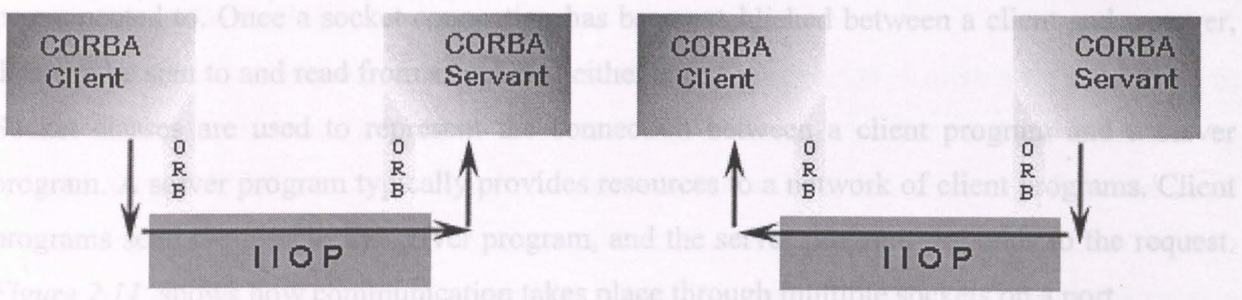


Figure 2.10 : Corba Communication

RMI vs CORBA

Remote method invocation has significant features that CORBA does not possess - most notably the ability to send new objects (code and data) across a network, and for foreign virtual machines to seamlessly handle the new objects. Remote method invocation has been available since JDK 1.02, and so many developers are familiar with the way this technology works, and organizations may already have systems using RMI. Its chief limitation, however, is that it is limited to Java Virtual Machines, and cannot interface with other languages.

CORBA is gaining strong support from developers, because of its ease of use, functionality, and portability across language and platform. CORBA is particularly important in large organizations, where many systems must interact with each other, and legacy systems cannot yet be retired. CORBA provides the connection between one language and platform and another - its only limitation is that a language must have a CORBA implementation written for it. CORBA also appears to have a performance increase over RMI, which makes it an attractive option for systems that are accessed by users who require real-time interaction.

2.8.1.3 Java Socket

The RMI allows multiple client programs to communicate with the same server program without any explicit code to do this because the RMI API is built on sockets and threads.
A socket is one end-point of a two-way communication link between two programs running on the network. Generally, a socket is a communication channel enabling data to be transferred through a certain port. Network sockets have a port number and an IP address so that they can

be connected to. Once a socket connection has been established between a client and a server, data can be sent to and read from a socket at either end.

Socket classes are used to represent the connection between a client program and a server program. A server program typically provides resources to a network of client programs. Client programs send requests to the server program, and the server program responds to the request. *Figure 2.11* shows how communication takes place through multiple sockets on a port.

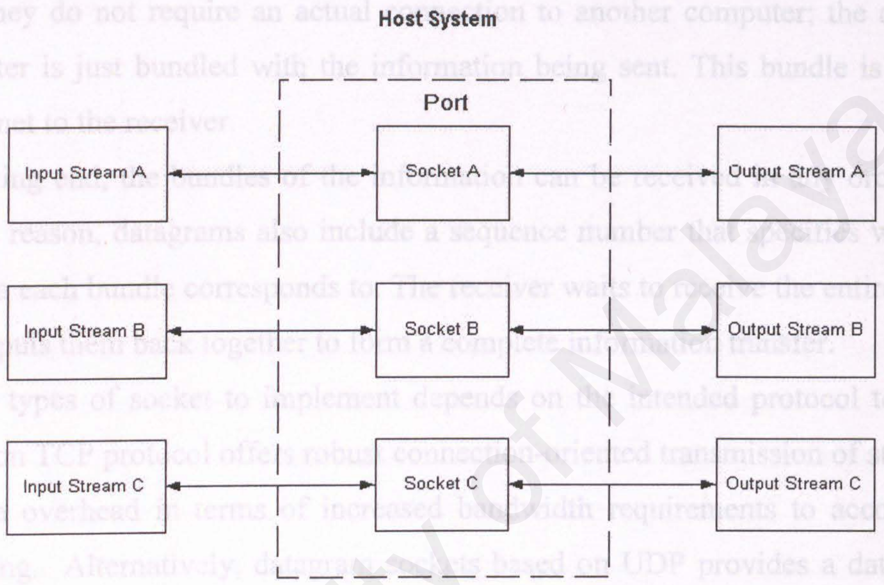


Figure 2.11 : Socket Communication on a Port.

Two main types of socket exist: a stream socket uses TCP, and a datagram uses UDP, both over Internet Protocol IP. A discussion about multithread is also included. Multithread has significant features when being used together with the Java socket.

- ***Stream Sockets***

A stream socket or connected socket; is a socket over which data can be transmitted continuously. Logically, the data might not being sent all the time, but that socket itself is active and ready for communication all the time. The benefit of using a stream socket is that information can be sent with less worry about when it arrives as its destination. The communication link is always available or live. Thus, data is transmitted immediately after it is sent.

- ***Datagram Sockets***

The other type of socket supported by Java is the datagram socket. Unlike stream sockets, in which the communication is akin to live network, a datagram socket tends to be a dial-up network in which the communication link is not continuously active. Because of the nature of communication medium involved, datagram sockets are not guaranteed to transmit information at a particular time, or even in any particular order. The reason datagram sockets perform this way is that they do not require an actual connection to another computer; the address of the target computer is just bundled with the information being sent. This bundle is then sent out over the Internet to the receiver.
On the receiving end, the bundles of the information can be received in any order and at any time. For this reason, datagrams also include a sequence number that specifies which piece of the whole data each bundle corresponds to. The receiver waits to receive the entire sequence, in which case it puts them back together to form a complete information transfer.
Choosing the types of socket to implement depends on the intended protocol to use. Stream socket based on TCP protocol offers robust connection-oriented transmission of streams of data – but with an overhead in terms of increased bandwidth requirements to accommodate the packet checking. Alternatively, datagram sockets based on UDP provides a datagram packet service but there is a possibility for packets to be lost, arrive out of order, or be duplicated. The advantage of UDP over TCP is its lower bandwidth requirement for the same data, as there is no checking that packets have arrived, resending if they have not, and are in the right order.
For the *Smar*t**Board** application, both types of sockets are important. Stream sockets can be used to transmit text messages among clients and servers. For voice communication, datagram socket was considered appropriate. Datagram socket can be used over the Real Time Protocol(RTP)

- ***Multithread***

One way to handle requests from more than one client is to make the server program multi-threaded. A multi-threaded server creates a thread for each communication it accepts from a client. A thread is a sequence of instructions that run independently of the program and of any other threads. Using threads, a multi-threaded server program can accept a connection from a client, start a thread for that communication, and continue listening for requests from other clients.

Sockets are significant to network programming by allowing developers to focus on input and output operations, independent of the intricacies and specifics involved with the network itself. This definitely matches the requirements that of handling great amount of input and output in the *Smart***Board** chat system.

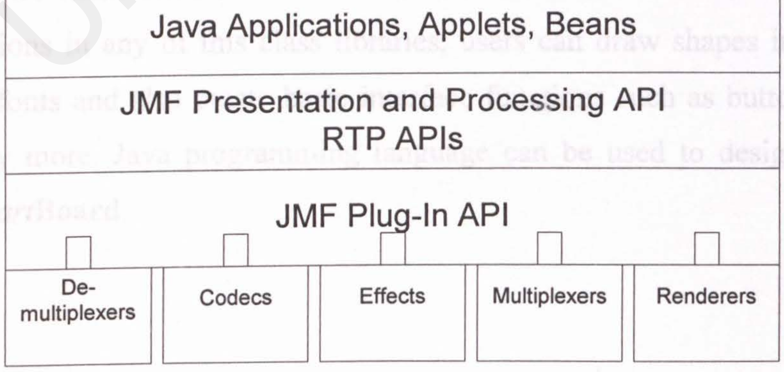
2.8.2 Java Application

2.8.2.1 Java™ Media Framework (JMF) API

The JMF is an application-programming interface (API) for incorporating time-based media into Java applications. It provides a unified architecture and messaging protocol for capturing, processing and delivery of time-based media data. By exploiting the advantages of the Java platform, JMF ensure the characteristics of "Write Once, Run Anywhere".

The high-level architecture of JMF is shown in *Figure 2.8.2.1a*. The application is built on top of the JFM Presentation and Processing API and RTP APIs (RTP architecture is explained later), which is in turn built on top of the JMF Plug-In API. The JMF Plug-In API manages different plug-ins such as Demultiplexers and Multiplexers, Codecs, Effect Filters, and Renders.

Figure 2.12 High-level JMF architecture



JMF (Java Media Framework) that runs RTP (Real Time Protocol) provides a unified architecture and messaging protocol for managing the acquisition, processing, and delivery of time-based media data.

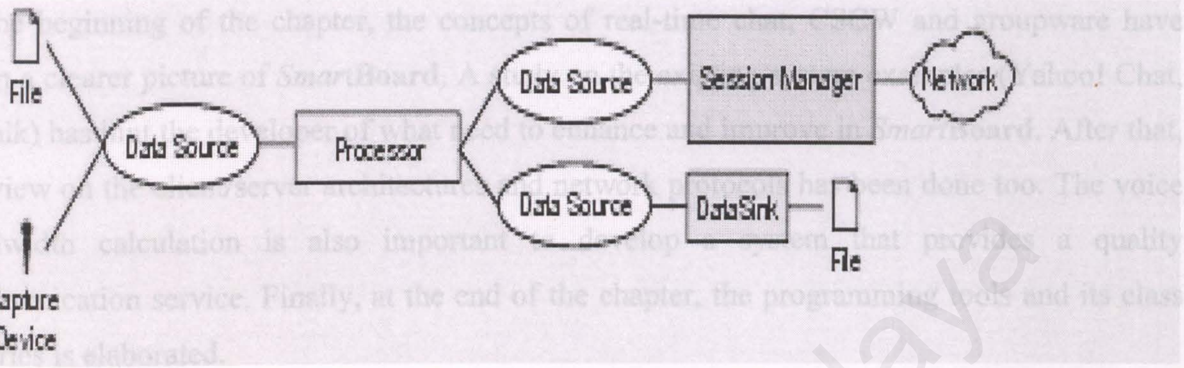


Figure 2.13 : JMF RTP design

Figure 2.13 display the JMF RTP APIs that are designed to work seamlessly with the capture, presentation, and processing capabilities of JMF. Players and processors are used to present and manipulate RTP media streams just like any other media content. Users can transmit media streams that have been captured from a local capture device using a capture **DataSource** or that have been stored to a file using a **DataSink.** Similarly, JMF can be extended to support additional RTP formats and payloads through the standard plug-in mechanism.

2.7.2.2 Java Graphic Interface

Java programming language has been popularly known over the years by its graphic interface programming. Two main class libraries in Java for this are the *Java AWT* and *Java Swing*. Using the functions in any of this class libraries, users can draw shapes in 2D or 3D, fill in colors, set text fonts and also create basic interface functions such as buttons, scrollbars, text frame and many more. Java programming language can be used to design the graphic user interface for *Smart***Board**.

CONCLUSION

The literature review is an important aspect as an initial stage to equip users with relevant knowledge and information to develop the system.

At the beginning of the chapter, the concepts of real-time chat, CSCW and groupware have given a clearer picture of *Smar*t**Board**. A study on the existing system examples (Yahoo! Chat, Paltalk) has hint the developer of what need to enhance and improve in *Smart***Board**. After that, a review on the client/server architectures and network protocols has been done too. The voice bandwidth calculation is also important to develop a system that provides a quality communication service. Finally, at the end of the chapter, the programming tools and its class libraries is elaborated.

The next chapter of this document will discuss about the methodology used to develop *Smart***Board**. The analysis of the system requirements will also be covered.

CHAPTER 3 : SYSTEM ANALYSIS

INTRODUCTION

In Chapter 3, the initial analyst phase of the *Smart***Board** system is being done. Firstly, we discuss about the methodology use to develop the system. Next, the requirement analysis is elaborated followed by the network architecture analysis, development tools analysis and last of all, the run-time analysis.

3.1 SYSTEM METHODOLOGY

System methodology describes the flow of the development stages or life cycle of the system. In the real-time chat module of *Smart***Board**, the development strategy used is the V Model.

3.1.1 Introduction to V Model

The V model is a variation of the waterfall model that demonstrates how the testing activities are related to analysis and design. As shown in figure below, coding forms the point of the V, with requirement analysis, system design and program design on the left, unit & integration testing, system testing, acceptance testing and operation & maintenance on the right.

Unit and integration testing will addresses the correctness of programs. The V model suggests that unit and integration testing will also be used to verify the program design. That is, during unit and integration testing, the developer should ensure that all aspects of the program design have been implemented correctly in the code. Similarly, system testing should verify the system design, making sure that all system design aspects are correctly implemented. Acceptance testing, which is conducted by the customer or user rather than the developer, validates the requirements by associating a testing step with each element of the specification; this type of testing checks to see that all requirements have been fully implemented before the system is accepted.

The model's linkage of the left side with the right side of the V implies that if problems are found during verification and validation, then the left side of the V can be re-executed to fix and improve the requirements, design, and code before the testing steps on the right side are reenacted. In other words, the V model makes more explicit some of the iteration and rework that are hidden in the waterfall depiction. Whereas the focus of the waterfall is often documents and artifact, the focus of the V model is activity and correctness [6].

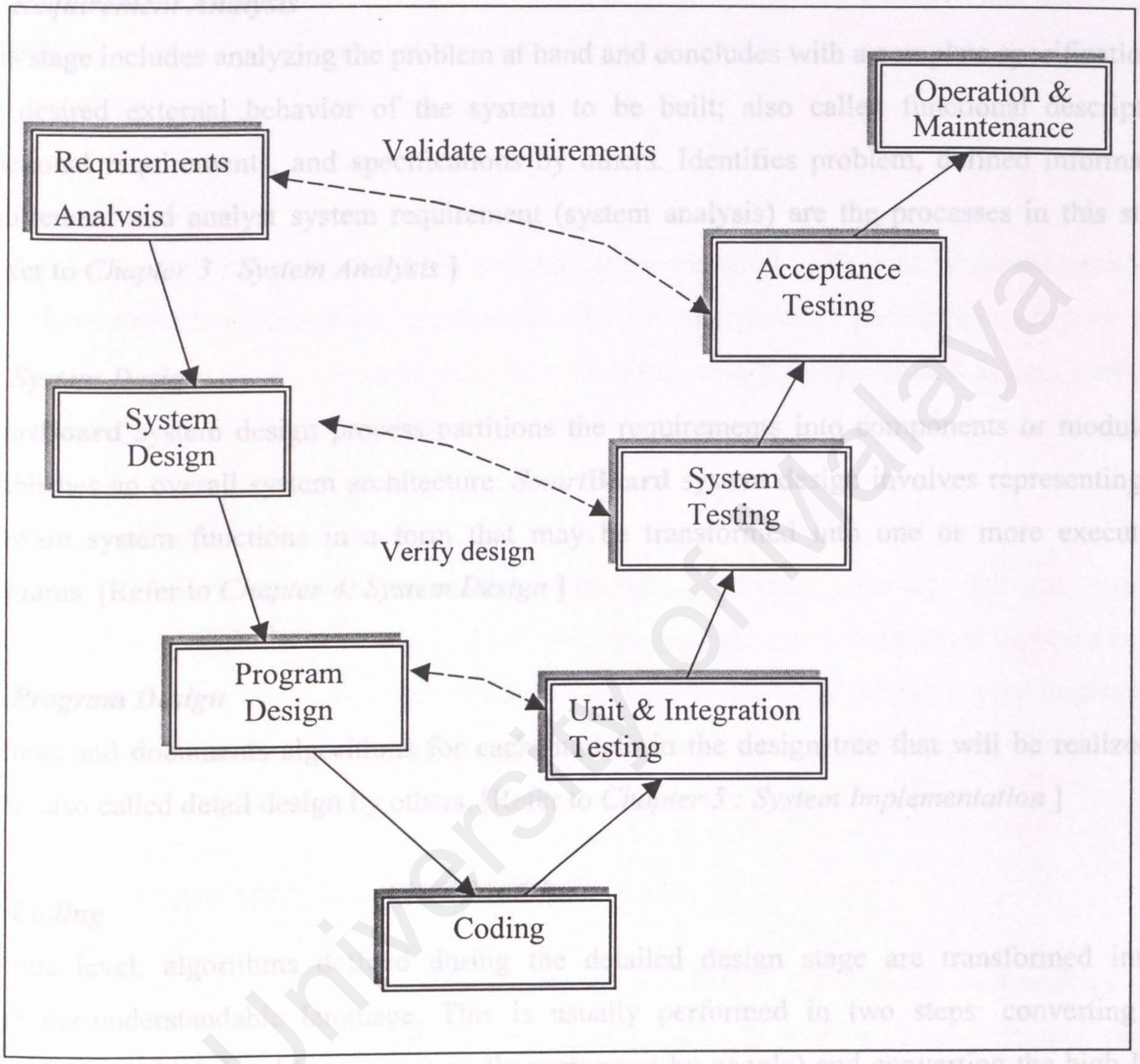


Figure 3.1 - The V model

3.1.2 V Model Stages Review

1. ***Requirement Analysis***

This stage includes analyzing the problem at hand and concludes with a complete specification of the desired external behavior of the system to be built; also called functional description, functional requirements, and specifications by others. Identifies problem, defined information requirement and analyst system requirement (system analysis) are the processes in this stage. [Refer to *Chapter 3 : System Analysis*]

2. ***System Design***

*Smart***Board** system design process partitions the requirements into components or module. It establishes an overall system architecture. *Smart***Board** system design involves representing the software system functions in a form that may be transformed into one or more executable programs. [Refer to *Chapter 4: System Design*]

3. ***Program Design***

Defines and documents algorithms for each module in the design tree that will be realized as code; also called detail design by others. [Refer to *Chapter 5 : System Implementation*]

4. ***Coding***

At this level, algorithms defined during the detailed design stage are transformed into a computer-understandable language. This is usually performed in two steps: converting the algorithm into high-level language (usually performed by people) and converting the high level language into a machine language (usually performed automatically by a compiler); also called programming. During this stage, *Smart***Board** system design is realized as a set of program unit. [Refer to *Chapter 5 : System Implementation*]

5. ***Unit & Integration Testing***

Checks each coded module for the presence of bugs. Unit & integration testing purpose is to ensure that each as-built module behaves according to its specification defined during detailed

design. This stage also will interconnect sets of previously tested modules to ensure that the sets behave as well as they did as independently tested modules. Ideally each set of modules should correspond to a component in the design tree defined during unit testing. Thus, this will ensure that each as-built component behaves according to its specification.
[Refer to *Chapter 6 : System Testing*]

6. System Testing

System testing will checks that the entire system embedded in its actual hardware environment behave according to system requirements. Every components, module and program unit are integrated and tested as a complete system to ensure that the *Smart***Board** system requirements are met. [Refer to *Chapter 6 : System Testing*]

7. Operation & Maintenance

After final system testing, the system and its surrounding hardware become operational. *Smart***Board** system will be installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of the system units and enhancing the system's services as new requirements are discovered.

3.2 REQUIREMENT ANALYSIS

Requirement analysis is the first step of a system life cycle in the V Model methodology.
During this first phase, a systematic approach to identify problems, opportunities, needs and requirements of the users are being carried out. The user's requirement with respect to the future system are carefully identified and documented. From this phase, the direction of *Smart***Board** development can be laid clearly so as to meet exactly the needs and requirements of the user. Questions of what are the specification requirements are answered here.
As the studies of the available chat system has been carried out in the Literature Review at Chapter 2, its advantages and disadvantages of the available system were also taken into account in the system requirement analysis. The studies of current sample system in Chapter 2 gave the idea of what need to enhance or improve for the chat application in *Smart***Board.**

The requirement analysis is divided into three parts

1. Requirements of the chat server
2. Requirements of the chat client
3. Overall requirement

3.2.1 Requirement of the chat server

- **Connect to multiple clients simultaneously**

In order to have a chat room, the server must be able to connect and interact with multiple clients. This can be accomplished by having the server class creates a thread object to deal with each connection. This will enable the group chat functionality

- **Search and connect to a particular client**

To enable a client to have a private communication (private chat) with another client, the server must be able to locate the IP address and establish a one-to-one connection with the other client. Message transmission must be between the two clients and not broadcast to the rest of the group members.

- **Receive and process commands from each client**

When a client issues a command to the server, the server must take the specified action to process the command and send a reply to the client. If a message is sent, the server should be able to broadcast this message (group chat) or only send to a particular connected client(private chat).

- **Maintain a list of current connections**

The chat server must be able to periodically update the list of connected clients. Any changes to the list (user leaves the room or new users log in) must be broadcast to all connected clients of the particular group.

- **Coordinate the voice chat**

For the voice chat function, the server must be able to coordinate between group members the turn to speak. In order to avoid collision of voice message across the network, there must be a set of rules to follow in order to ensure a smooth conversation between group members.

- **Scalability**

The chat server must be able to remain effective even when there is a significant increase in the number of resources and the number of users in one session.

3.2.2 Requirements of the chat client

- **Usability and User Friendly**

*Smart***Board** must be easy to use. It also must be able to accommodate any levels of user. The graphical user interface (GUI) plays a major part to develop a user-friendly system. The GUI must be clear without any confusion. It brings high degree of understanding to the user. It must also be simple yet attractive. The functions embedded in the interface must also be complete and effective.

- **Good Voice Quality**

The real-time voice chat must have minimal voice distortion, minimum voice latency, low packet lost rate and less jitter to ensure a high quality of audible communication.

- **Response Time**

The response time of the real-time chat system must be within reasonable interval time. The real-time chat must not have delays in between to ensure smooth conversation between users.

- **Loading time**

The chat application session must not have a long loading time. A client browser must be able to connect to the server and download the application in a minimal time.

3.2.3 Overall requirements

- **Reliability**

The real-time chat system, software and hardware shall be reliable and not cause any unnecessary downtime for the overall environment.

- **Maintainability and Modularity**

The system coding and design shall be implemented by using the modular approach so that it can be easily enhanced in the future. The procedure, subroutine and methods in the program are written in a modular form. With this it makes the program easier to understand in the later time. Its reusability of some common procedures or functions will save a lot of development time and prevent code redundancy. Later maintenance will be easier.

- **Robustness**

Real time chat system in *Smart***Board** consists of modules that will be completely tested to ensure that each module achieves its expectation. The module is integrated into system. Any errant that maybe discovered during system testing will be solved immediately. This is to make sure that the system is as robust as what had been expected before.

3.3 NETWORK ARCHITECTURE ANALYSIS

3.3.1 Three-Tier Client Server Architecture

After much consideration, *Smart***Board** will adopt the three-tier client/server architecture approach as its underlying network. A single tier host system is not feasible as this web-based system is in a client/server environment. A two-tier (client centric or server centric) needs the software to be installed on multiple clients. Two-tier architecture is less used in critical time processing system compared to three-tier. Three-tier is chosen because of the advantages below:

- Some upgrades can be done entirely at the server level

- An increasing number of homogeneous products are available off the shelf pre-made software is cheap.
- Since only the information to be displayed is sent on the network, there is little network bandwidth comparative to the client-centric model. The load may however be higher between the business logic and data services systems if they are on different servers.
- Allows for (actually encourages) component-based development, which can increase reusability.

3.3.2 Protocol

There are two proposed protocol to be implemented in transmitting the data over the client / server network. They are Real Time Protocol (RTP) and Transfer Control Protocol / Internet Protocol (TCP/IP)

- **Real Time Protocol**

RTP is the most suitable protocol to stream data such as voice packets in real-time development. RTP provides end-to-end network delivery services for the transmission of real-time data. It is network and transport-protocol independent, though it is often used over User Datagram Protocol (UDP). TCP is not a good choice for voice chat application considering the delay for retransmission of lost or corrupted packets.

- **TCP/IP**

Text chat application will be based on TCP/IP. TCP is a transport-layer protocol designed for reliable data communications on low-bandwidth and non real-time environment. Unlike the UDP, TCP guarantees that each packet will reach its destination. The stream socket will be used over the TCP to ensure transmission of text messages.

3.4 DEVELOPMENT TOOLS ANALYSIS

To carve a beautiful door a carpenter need to select his right and effective tools. Similarly, in order to develop a good chat application system for *Smart***Board**, the right and effecting tools must also be selected. In the context of software development, the "tools" here include the operating system platform, web server, web database and programming language. Besides considering the suitability of the tools to the system requirements, the tools used must also be easy to use, able to support each other and has scalability features.

3.4.1 Operating System Platform – Microsoft Windows 2000

Similar like the Microsoft Windows NT Server 4.0, Windows 2000 is said to be the most complete platform available for building and hosting web based application. Microsoft Windows 2000 was proposed as the operating system of choice due to several advantages that are distinct when compare to others operating system

- It gives high performance, reliability, security and easiness to manage server for sharing information and running applications in the most demanding business.
- It has a user friendly environment and easy usability compare to other operating system
- It can be integrated with the Microsoft Internet Information Server 5.0 considering that both products come from Microsoft. A customer service support is available for both products.
- It supports multitasking environment
- It is available at the development site which is in the computer lab of FSCIT
- It supports innovative web publishing features, customizable tools and its new a technologies

3.4.2 Web Server – Microsoft Internet Information Server 5.0 (IIS)

Microsoft Internet Information Server is the proposed web server for *Smart***Board**, due to several key factors:

- It is suitable for medium size to large size system doing high volume serving
- It is bundle with Windows 2000 operating system and this made it really easy to implement.
- IIS is the fastest web server for Windows 2000. It is completely integrated with Windows 2000 Directory Service. This combination of web and operating system services makes it possible to deploy scalable and reliable web-based application.
- IIS supports SSL 3.0, which provide a secure communications channel between a browser and server.
- IIS provides a comprehensive web services and it is designed to support multiple web server scenarios ranging from simple web sites on a corporate intranet to larger ISP web farms. This makes it the best suit web server for *smart***board**.

3.4.3 Web Database System – Microsoft SQL Server 7.0

For the *Smart***Board** database, Microsoft SQL Server 7.0 is proposed. Since the platform is a Microsoft product, it will work very well under the environment. Microsoft Access 2000 is not chosen because it is only capable to hold small database as it has an upper bound. Key features of SQL :

- SQL is a step further by providing dynamic sizing for databases and transaction logs. This means that database and transaction logs can grow and shrink without requiring any intervention by an administrator.
- It is easy to use. It has many wizards to assist the administrator in his/her work.
- The amount of RAM used by SQL Server can be automatically adjusted on an as-needed basis.
- Online backups can now take place with a 95% throughput rate. This is a big benefit for sites that must be available around the clock.
- SQL Server has a security model that is more tightly integrated with Windows 2000.

3.4.5 Programming Language - Java

Java programming language is proposed since it represents state-of-the-art in platform-independent application development. Java as a technology is poised to bring interactivity to the web in a general sense. Java provides a level of platform independent, security and network support that is still unattainable in any other languages. The web is a revolutionary medium for the distribution of information. Real-time chat that is created on the Java platform allows anyone with a web browser to access.

- ***Multiple Users Interaction***

The Java platform facilitates interaction by multiple users. Java has libraries that support Web and Internet protocols. Thus, applets running on different computer can communicate with each other. Users from around the web can communicate with each other via the chat.

- ***Object-oriented Programming***

The Java platform supports object-oriented programming. This makes Java a great language to develop the chat application since it helps to manage complexity and code-reuse, thereby, cutting down on development time. The object metaphor is ideal for creating graphics.

- ***Simplicity***

Java is viewed as a simple language in several ways. First, Java's syntax resembles that of C and C++. Thus, it is not too difficult to learn for C and C++ programmers. Second, Java eliminates features from C and C++, which are redundant or led to poorly written or insecure code. The result is a smaller, simpler language than either C or C++. Third, Java makes memory management simple with the elimination of pointers and he use of garbage collection to reclaim unused allocated memory. This is able to reduce common sources of bugs and frustration. The simplicity of the Java platform makes the chat program easier to understand and debug.

Proposed Java Libraries

Java Media Framework in conjunction with the **Java Sound API** will be used to develop the voice chat system. JMF is able to support the Real Time Protocol (RTP).

Java Swing will be used to design and encode the user interface

Java Multicast concept will be abducted to enabled broadcast message to users

Java Socket is a preferable choice to transmit text data compared to CORBA. Besides its complexity, CORBA is too slow to support real-time and time-based application such as the voice chat.

Below is the conclusion of the proposed development tools for the respective area.

Operating System

Windows 2000

Web Server

Microsoft Internet Information Server

Web Database System

Microsoft SQL Server

Programming tool

Java

3.5 RUN – TIME REQUIREMENT ANALYSIS

After a review and analysis done to proposed the underlying network architecture and the development tools, the run–time requirement needed for *Smart***Board** application to execute is studied and the results are proposed below. The run-time environment also consists of hardware and software configurations.

3.5.1 Server Hardware Requirements

The server requirements for *Smart***Board** system are

- A server with a least Intel Pentium 166MHz MMX CPU.
- At least 64MB RAM.
- Network Interface Card (NIC) and network connection with recommended bandwidth at 10Mbps.
- Keyboard and mouse as input devices.
- Other standard server peripherals. For example, 6.4GB hard disk.

3.5.2 Server Software Requirements

To host and run the system, the server needs to have various supporting software installed.

- Microsoft Windows NT Server 4.0 or higher
- Microsoft Internet Information Server 4.0 or higher
- Microsoft Access
- Any Web Server (For instance, JRun 3.1 Web Server).
- Java™ Development ToolKit Version 1.3.1
- Java™ 2 Runtime Environment, Standard Edition, v 1.3.1 (JRE).
- Java™ Media Framework Version 2.1.1

3.5.3 Client Hardware Requirements

The following hardware specifications for *Smart***Board** client run-time environment :

- Minimum Intel Pentium 166MHz MMX CPU.
- At least 32MB of RAM
- Network connection through existing network configuration or modem (recommended at least 28.8 kbps).
- A color monitor with a VGA card
- Full Duplex Soundcard
- Microphone (preferably 16 bits) and speakers / headset
- Keyboard and mouse as input devices.

3.5.4 Client Software Requirements

The following software specification for *Smart***Board** run time environment:

- Any web browsers (recommended Internet Explorer 5.5 or Netscape Navigator 4.0).
- Any operating system platform (recommended Windows 98 and above).

CONCLUSION

In this chapter, the developer has discussed about the methodology of the system development. The methodology gives a backbone to how *Smart***Board** will be developed. The expected requirements of the system have also been elaborated.

Following that, the type of network architecture and development tools has been chosen and its reasons are also included. Finally, after a thorough study, the developer has clarified the run-time requirements for the whole *Smart***Board** system at the end of the chapter.

After studying on the system requirements, the design phase begins. The following chapter discusses the design phase.

CHAPTER 4 : SYSTEM DESIGN

INTRODUCTION

Design is the creative process of transforming the problem into a solution and the description of the solution [Pfleeger, 1998]. System design is the essential nucleus of the software development process and is applied regardless of the development model or standard it is used. This chapter will meet the requirements identified during system requirement analysis. The system designer will address the question as 'how' to achieve the user's requirements in term of systems components and their interaction. In this chapter, the design of the system will include the module architecture reviews, description of module concept, module flow and other system components and their relationship.

4.1 DESIGN PROCESS

Software design is a process in which requirements are translated into a software representation. Subsequent refinement leads to a design representation that is agreeable with the source code. In order to achieve quality software, real-time chat *Smart***Board** has been designed to adhere to the following guidelines.

- Exhibits a hierarchical organization that makes intelligent use of a control among components of software.
- Modularity, that is, the software should be logically partitioned into components that perform specific functions and sub functions.
- Contain distinct representation of data and procedure.
- Lead to modules that exhibit independent characteristic.

4.2 DESIGN TECHNIQUE

The design technique used in *Smart***Board** chat application system is a combination of Event-Oriented decomposition and Object-Oriented decomposition technique. This design based on events that the system must handle and uses information on how the events (actions) have been generated by the users. The design also identifies classes of objects and their relationships. Each object class is described regarding its functions and how these objects are related to one another.

4.3 DESIGN CONSIDERATION

There are many issues or considerations involved in creating a design. Often, no one style or method is best for every situation. During the design of the chat system, the following considerations have been taken into account:

4.3.1 End-user consideration

The design of the graphical user interfaces should confirm to user acceptance and to anticipate future end users needs. On most things, the effect of attractiveness on users is weighted heavily.

4.3.2 Coding consideration

Basically, the system requirements are organized by objects as their abstract type. The design can also build its components around object classes. That is, each component is an instance of an abstract data type. Thus, the issue of encapsulation is given emphasis. The object must preserve the integrity of the data representation and the data representation must be hidden from other objects.

The primary advantage of using this design technique is data encapsulation making it easy to change the implementation without perturbing the rest of the system. Also, combining object access routines with the data manipulated by them encourages the designer to decompose the underlying problems into a collection of interacting agents. (in this case, classes).

However, one object must know the identity of the other objects in order for them to interact. This dependence means that changing the identity of an object requires all other components to be modified if they invoke the changed object class.

4.3.3 Synchronization consideration

One of the common problems faced in a network chat system design is maintaining synchronization. Synchronization refers to how multiple chat session instances running on different or remote computers can be maintained in the same state information. As noted, each user is running a separate instance of the chat system, but the overall goal is to make each of these different instances function logically as one instance. All the internal data structure modeling the state of the chat session should match exactly on each user's system. In the design, any changes in the chat session must be identified and communicated to all users.

4.3.4 Bandwidth consideration

Another concern to be dealt with in designing a real time chat application is bandwidth. The amount of data transferred between chat session instances can have dramatic effect on performance. The degree to which the bandwidth affects the chat session performance, however is determined by the particular communication strategy used. An example of this design method is to reach the point where a very small amount of information is sent and got by with lower communication speeds. In this case, the only information transferred between instances would be the interaction of users (text chat, voice chat, whiteboard)

4.4 NETWORK ARCHITECTURE DESIGN

Network architecture depicts the physical flow of data over the system network and it also shows the location of hardware peripherals in the system.

*Smart***Board** system is based on web application using the three-tier client/server approach, where a central server manages partial actions (events) generated from users to chat system and each user runs a client program that interacts with the others through a Graphical User Interface (GUI). Users may connect with the client program and they should be able to communicate with other users.

The diagram in *Figure 4.1* depicts the network architecture of the *Smart***Board**. The server will be located at the Faculty of Computer Science and Information Technology (FSCIT) in University of Malaya.

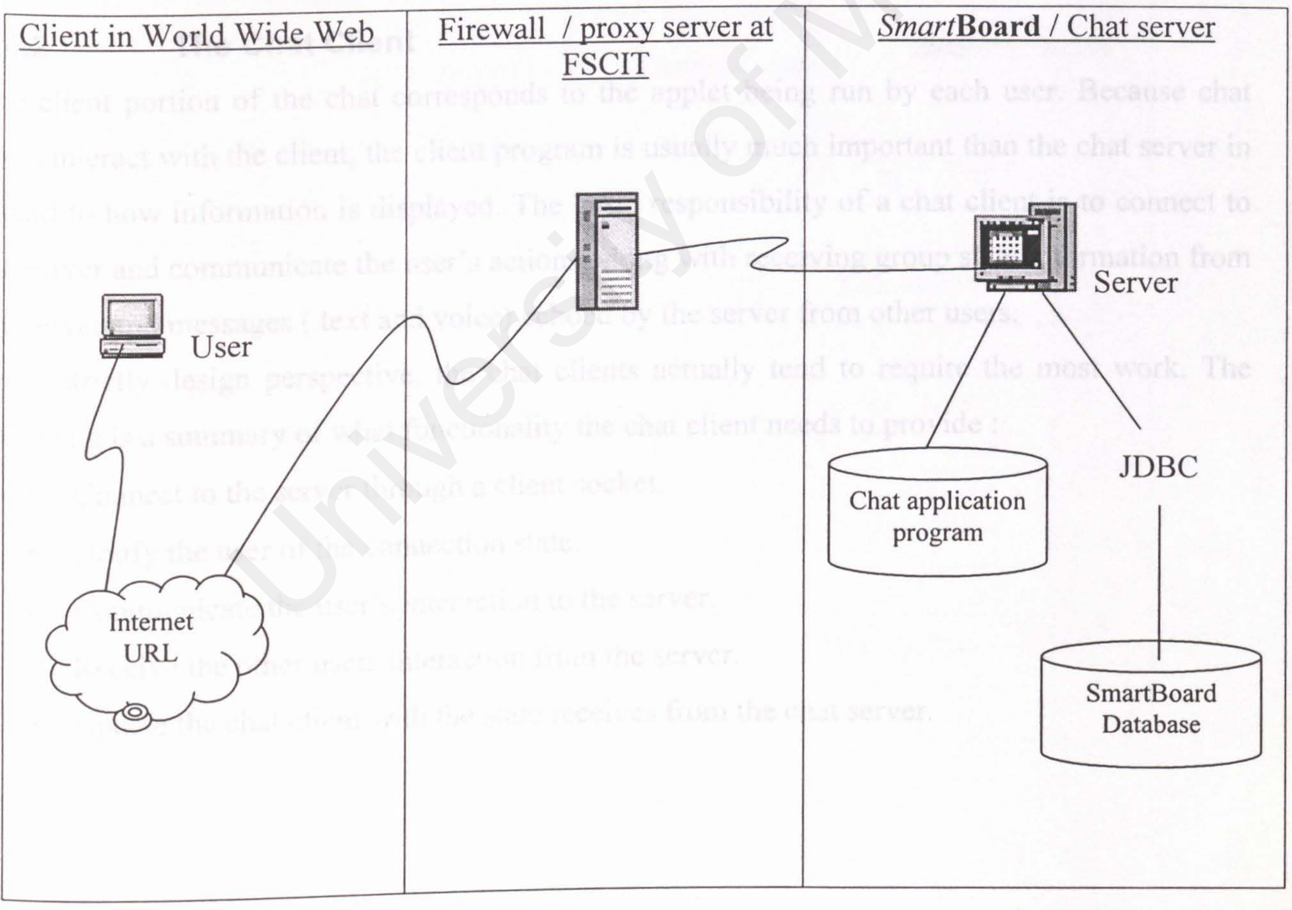


*Figure 4.1: Smart***Board** *Network Architecture*

ince *Smart***Board** is a web-based system, the system is assumes to be started once the users nters the URL of the web site. The users have to register (log in) before they can use any of the eatures of the *Smart***Board**

.4.1 The Chat Server

*mart***Board** chat system is designed to support multi-users. The server side of the system acts lmost like a middleman, managing the different users and helping them communicate ffectively. This system is designed to be completely client-driven; all the sever will do is echo ny message it receives. Most specifically, the server takes on the role of handling the network onnection through sockets for each user, along with querying for and responding to events for he users.

.4.2 The Chat Client

The client portion of the chat corresponds to the applet being run by each user. Because chat sers interact with the client, the client program is usually much important than the chat server in egard to how information is displayed. The basic responsibility of a chat client is to connect to he server and communicate the user's actions, along with receiving group state information from he server and messages (text and voice) echoed by the server from other users.

rom strictly design perspective, the chat clients actually tend to require the most work. The ollowing is a summary of what functionality the chat client needs to provide :

- Connect to the server through a client socket.
- Notify the user of the connection state.
- Communicate the user's interaction to the server.
- Receive the other users interaction from the server.
- Update the chat client with the state receives from the chat server.

.5 FUNCTIONAL DESIGN

rom the requirements discussed in Chapter 4, functional design describes what are the functions r services available for users in the chat application. The design is crucial to know the modules r program unit for coding in the later phase.

igure 4.2 is the functional diagram of the chat application in *Smart***Board**. In reference to the liagram, there are four main functions or modules to be created. They are the Text Chat Module, Jser List Module, Voice Chat Module and the Help Module

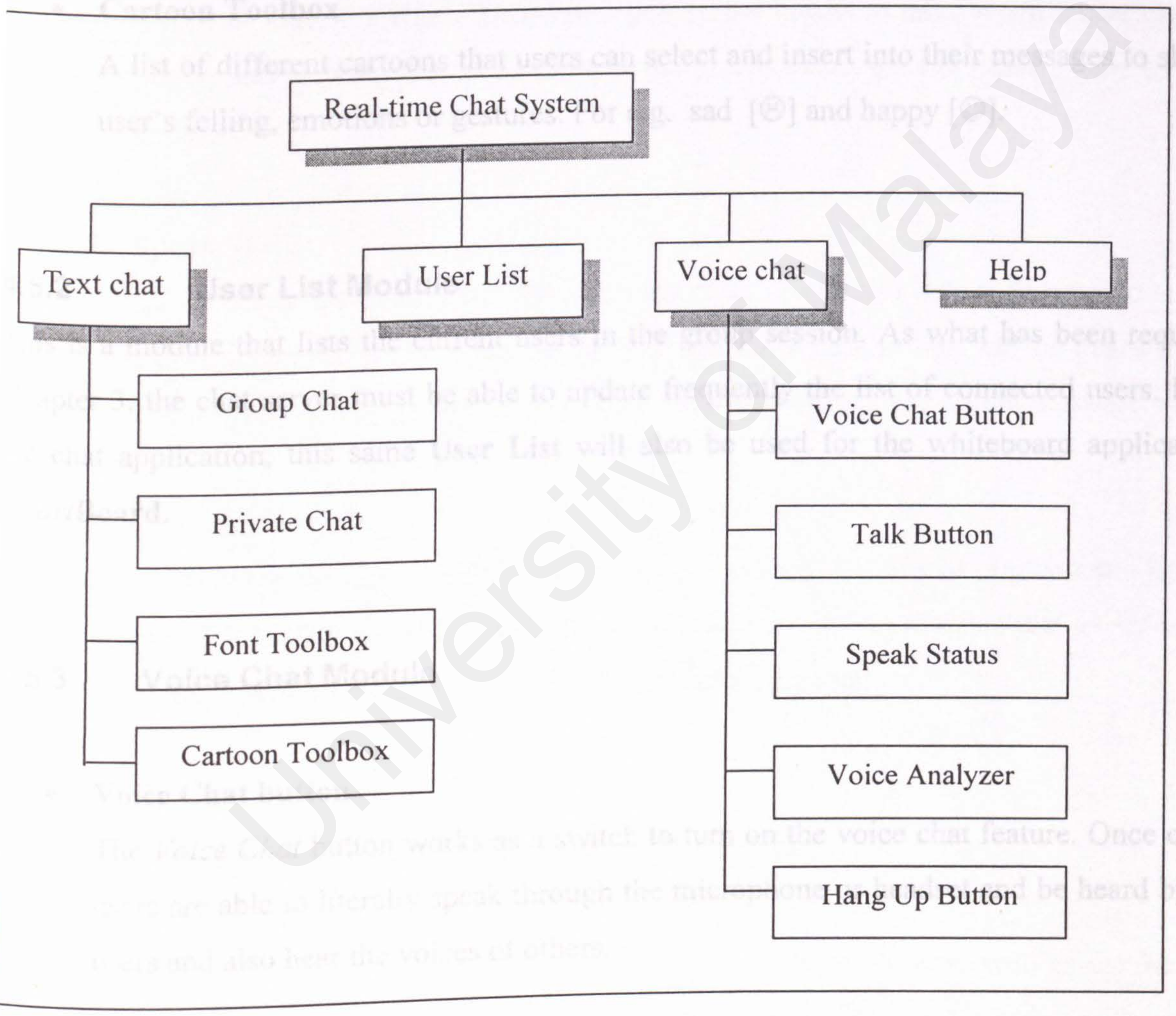


Figure 4.2 : Functional Diagram for SmartBoard Chat System

4.5.1 Text Chat Module

Below are the list and function of each sub-modules of Text Chat :

- **Group Chat**
  Message from a particular user will be broadcast to the group.
- **Private Chat**
  Message from a particular user will be send to another predefined user or destination.
- **Font Toolbox**
  This is a feature where users can select a particular font for their text messages.
- **Cartoon Toolbox**
  A list of different cartoons that users can select and insert into their messages to show the user's felling, emotions or gestures. For e.g. sad [☹] and happy [☺].

4.5.2 User List Module

This is a module that lists the current users in the group session. As what has been required in Chapter 3, the chat server must be able to update frequently the list of connected users. Besides the chat application, this same **User List** will also be used for the whiteboard application in *Smart***Board**.

4.5.3 Voice Chat Module

- **Voice Chat button**
  The *Voice Chat* button works as a switch to turn on the voice chat feature. Once clicked, users are able to literally speak through the microphone or headset and be heard by other users and also hear the voices of others.

- **Hang Up button**
  In prior to the *Voice Chat* button, the *Hang Up* button disable voice chat feature. A user may have low bandwidth transmission or problem of noise interference, the user may

wish to disable the voice chat feature and only use the text chat. In order to do this, user must click on the *Hang Up* button and the voice chat menu will disappear.

- **Voice Analyzer**
  Voice Analyzer is an essential tool for voice chat users to adjust the sound volume of receiving and sending voice messages.

- **Talk button**
  The *Talk* button is where users can click on the button to talk. When a user clicks on the button and is given the *Speak Status,* the user is given the floor and is able to speak and be heard.

- **Speak Status**
  Speak status is a display of who holds the floor or who is given the access to talk. When a user clicks on the *Talk* button, he /she has the access to talk and be heard by the rest. The rest has no access even if they click the *Talk* button also. A wait message will appear.

4.5.4 Help Module

In order to achieve the aim of being a user-friendly system, an idea to implement a help feature was proposed. This feature will include the solutions to common problems that a user might face. A step-by-step guide in using the system will be found here too.

4.6 CHAT APPLICATION SYSTEM FLOW

This section will gives an overview of the *Smart***Board** system flow as well as the *Smart***Board** chat application system flow. *Figure 4.3* shows the system flow of *Smart***Board** in detail. The following figure (*Figure 4.4*), will then shows the chat application system flow without taking the whole *Smart***Board** system flow into consideration. The starting point of the system flow can be either the "sending users" or the "receiving users", since both can start the flow.

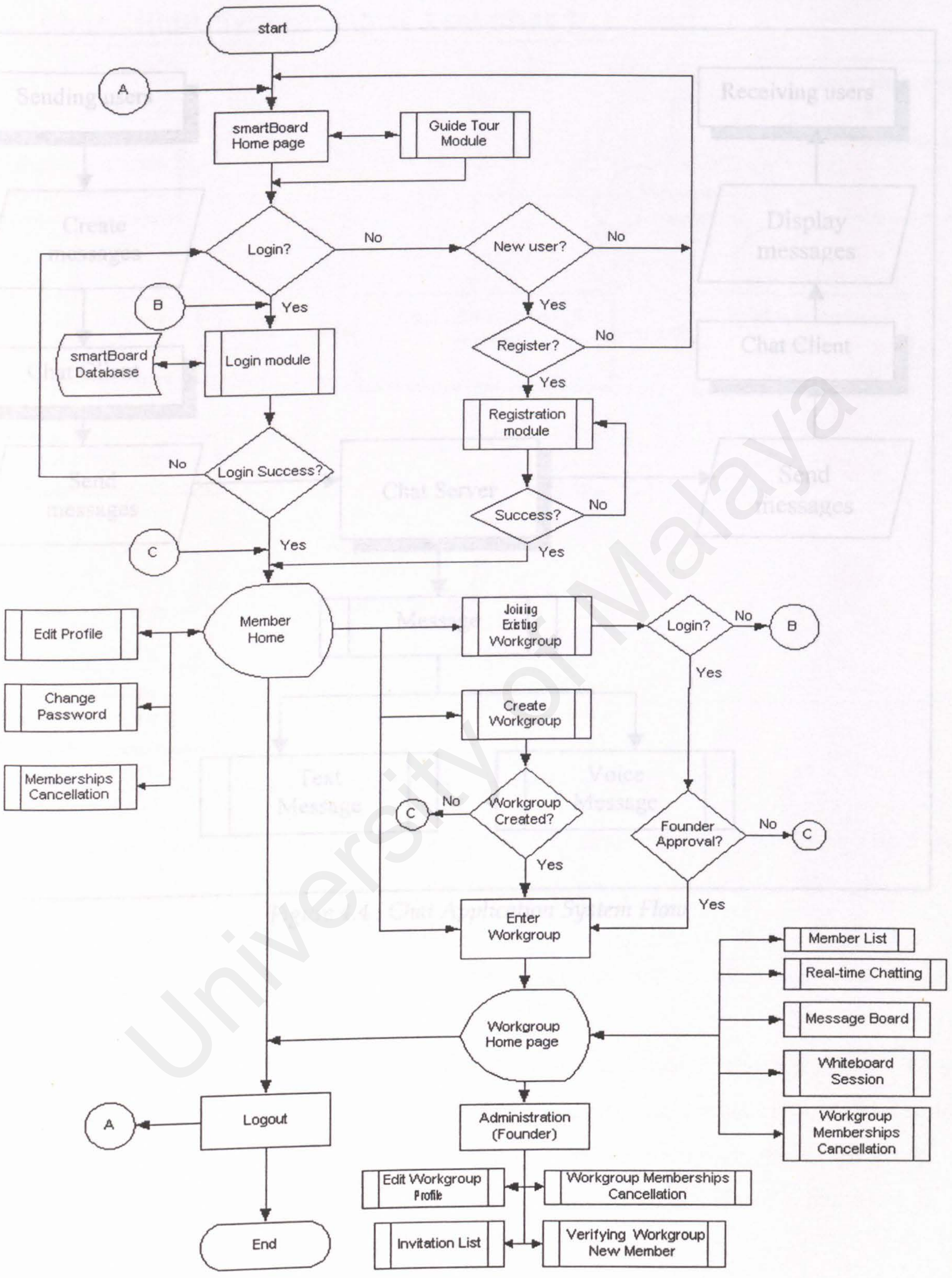


Figure 4.3 – SmartBoard System Flow

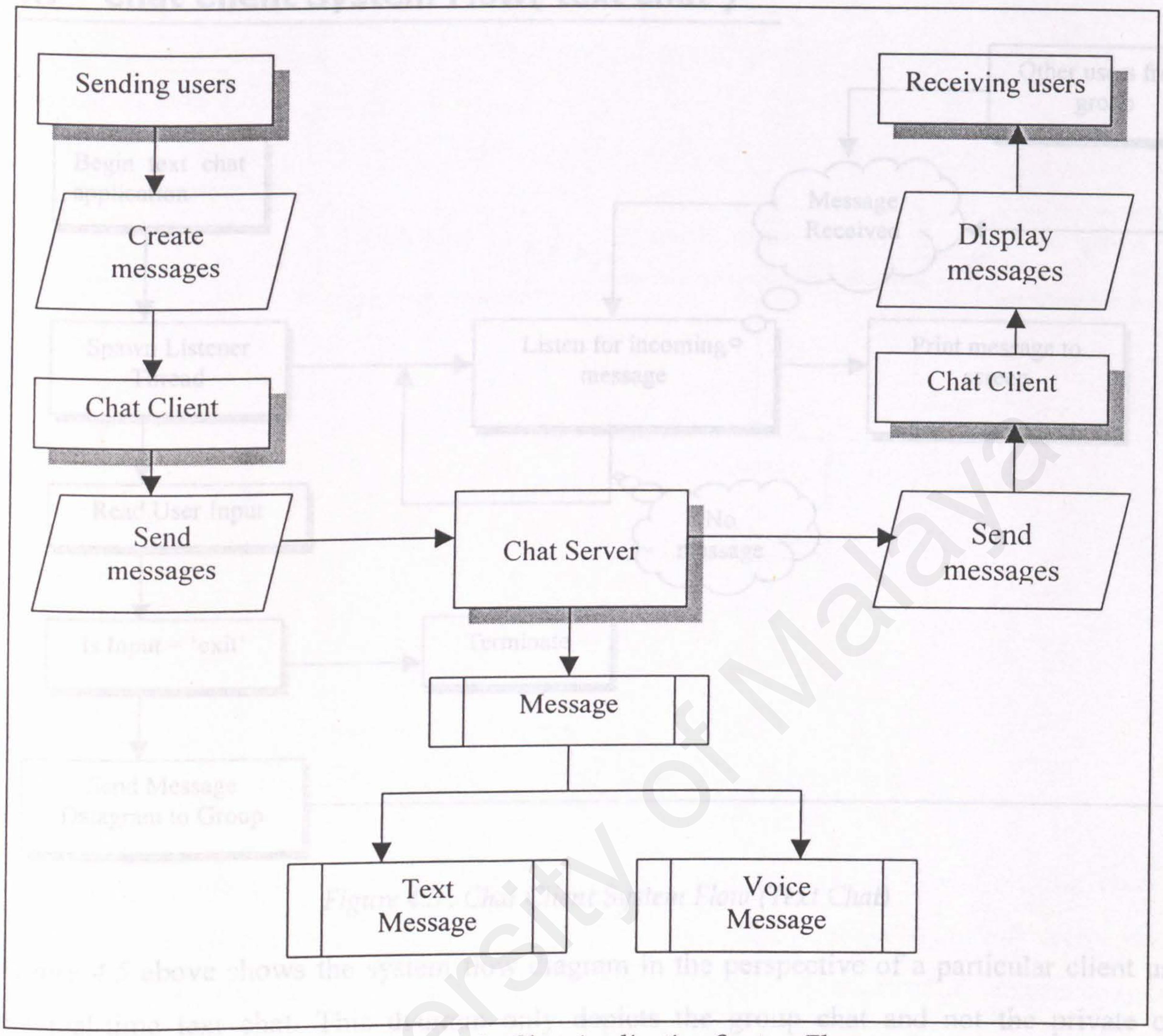


Figure 4.4 : Chat Application System Flow

4.7 Chat Client System Flow(text chat)

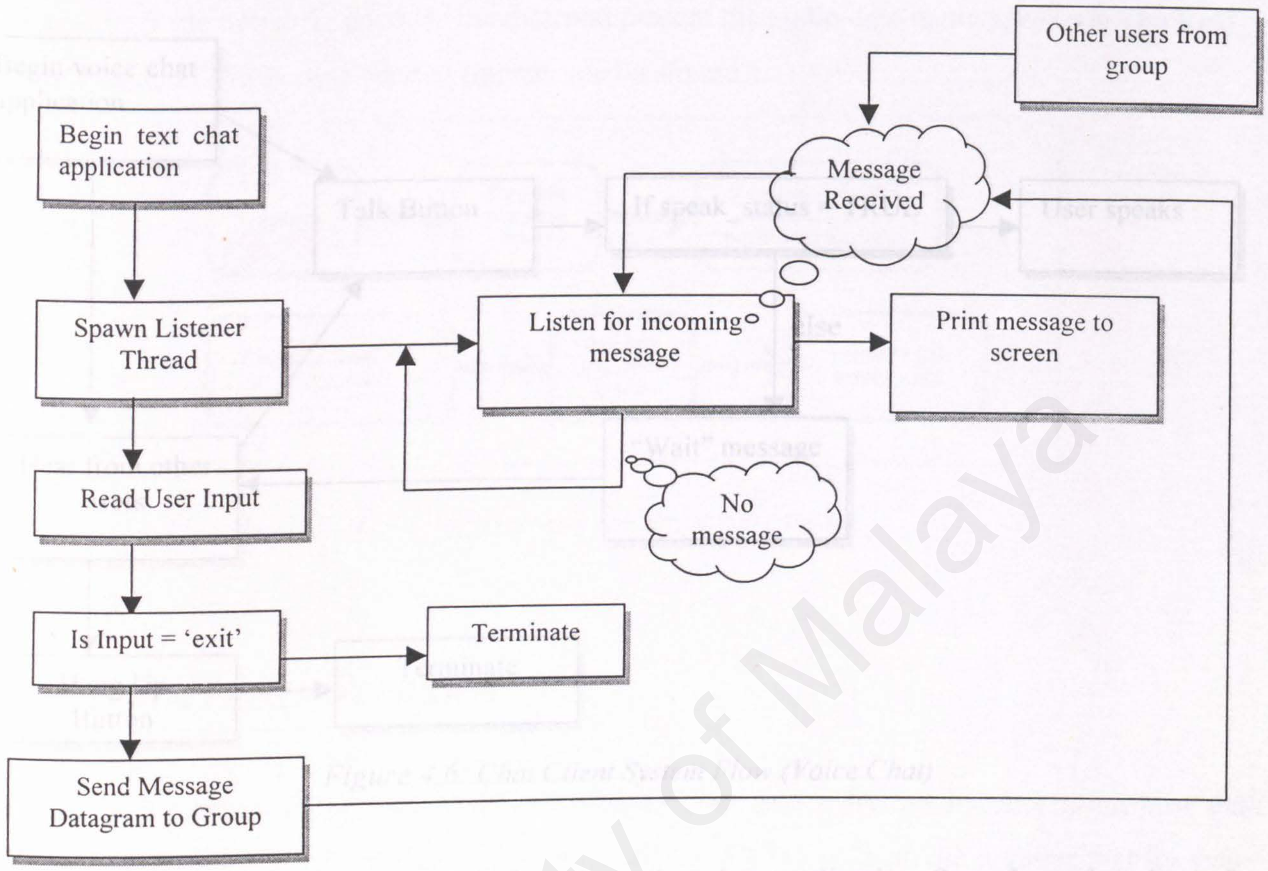


Figure 4.5 : Chat Client System Flow (Text Chat)

Figure 4.5 above shows the system flow diagram in the perspective of a particular client using the real-time text chat. This diagram only depicts the group chat and not the private chat. Nevertheless it is also applicable in the private chat.

There are two distinct system flows: one for the main user loop and the other listens for incoming messages.

The flow going downwards from the listener thread awaits the input of the users. Once the user exits the application, the program is terminated. The messages that the user type will be send to the group and are displayed on the user's screen to be viewed by other members of the group. Whereas, the flow going to the east, listens for incoming messages from other users in the particular group.

4.8 Chat Client System Flow (voice chat)

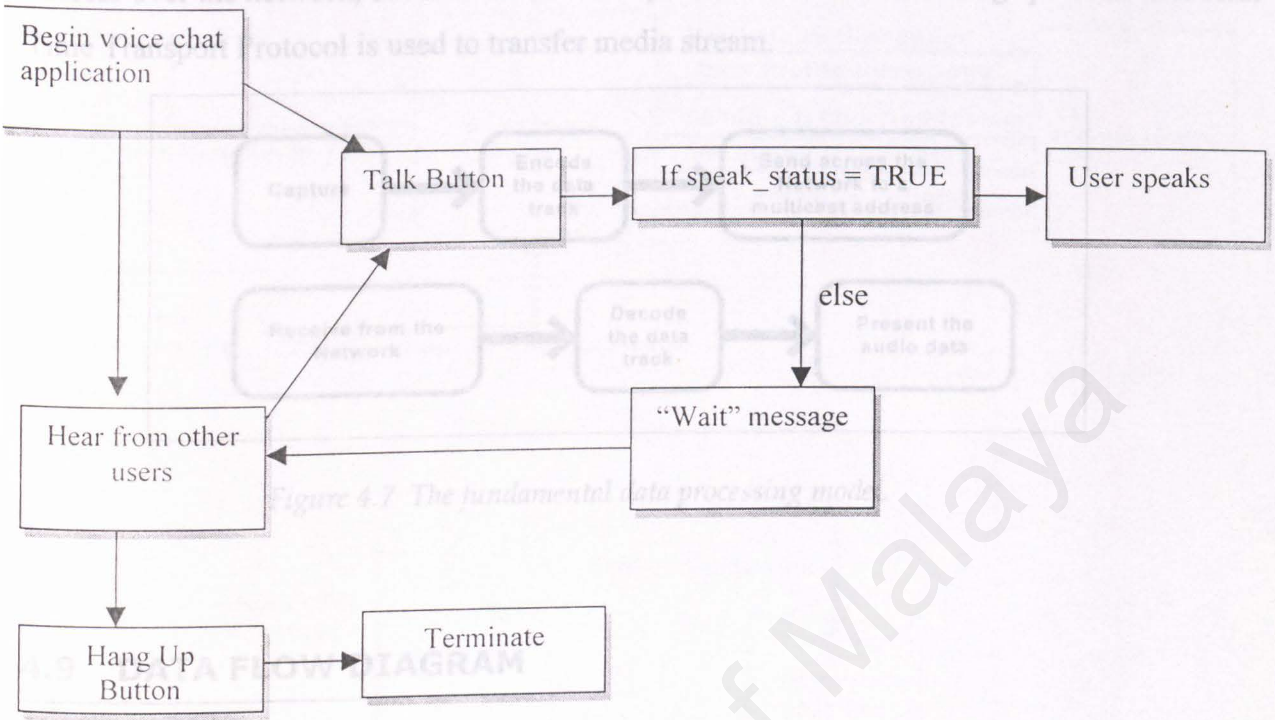


Figure 4.6: Chat Client System Flow (Voice Chat)

The diagram shown in *Figure 4.6* depicts the voice chat application flow from the view of a particular user. When a user clicks on the *Talk Button* to speak, the system will verify whether the user can be given the speak status (speak_status=TRUE) . If successful, the user may speak and be heard by the rest of the group members.Otherwise, a wait message will appear. When the user clicks on the *Hang Up* button the voice chat application will terminate.

Data Processing Model

The fundamental data processing model used in the voice chat application is shown in *Figure 4.7.* The Voice data is captured by microphone and is encoded by the JMF default Codec, then the media stream is sent to other registered clients (users) via a multicast address across the network (media resource locator).

The Voice Chat application can also receive media stream from other clients via a multicast address over the network, decodes the data and present the audio data using speakers. The Real Time Transport Protocol is used to transfer media stream.

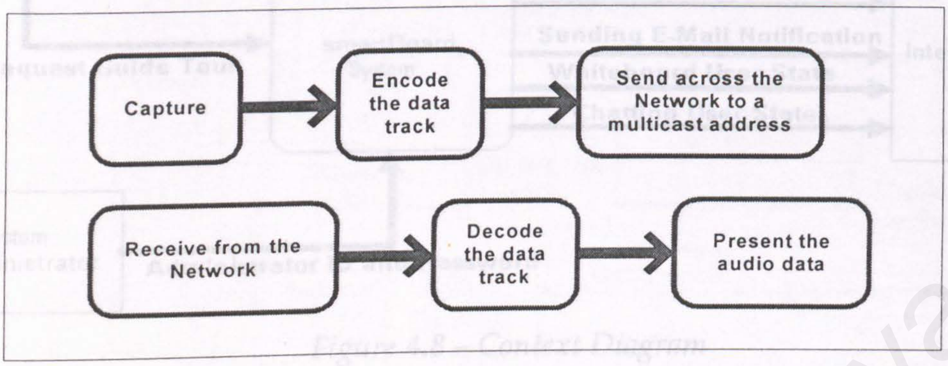
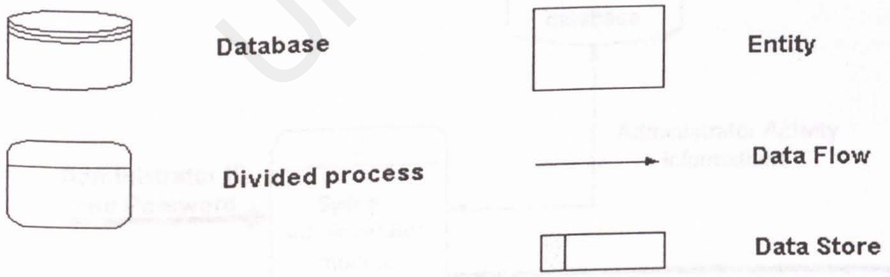


Figure 4.7 The fundamental data processing model.

4.9 DATA FLOW DIAGRAM

Data Flow Diagram (DFD) models objects, associations and activities by describing how data can or should flow between and around various objects. DFD works on the premise that for every activity, there is some communication, transference or flow that can be described as a data element. DFD describes what activities are or could be occurring to fulfill a business relationship or accomplish a business task; not how these activities are to be performed. That is, DFD shows the logical sequence of associations and activities are; not the physical process. Below shows the DFD symbols used.



The following figures (*figure 4.8 and 4.9*) show the context diagram and the diagram 0 for the SmartBoard system respectively. *Figure 4.10* shows the diagram for work group profile modules.

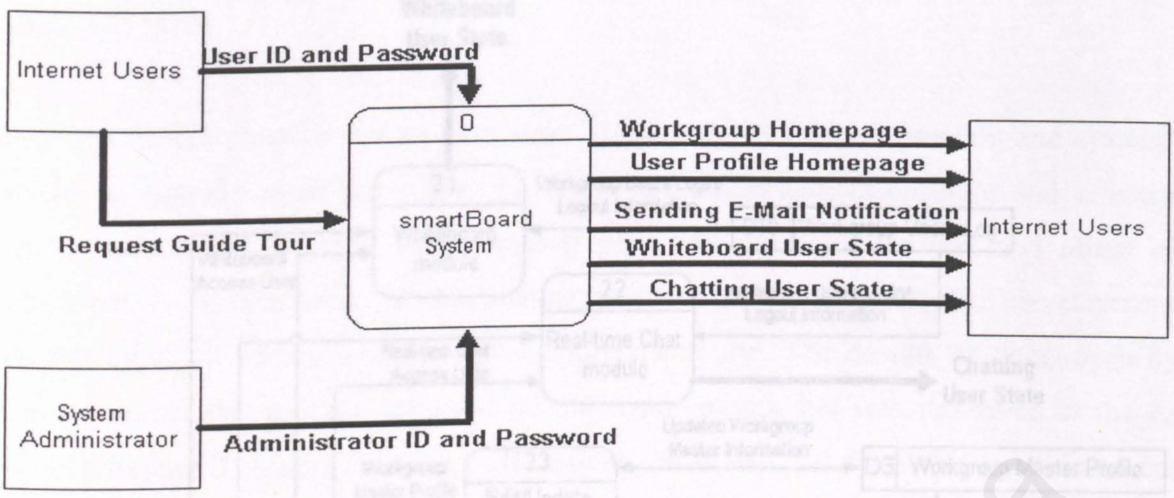


Figure 4.8 – Context Diagram

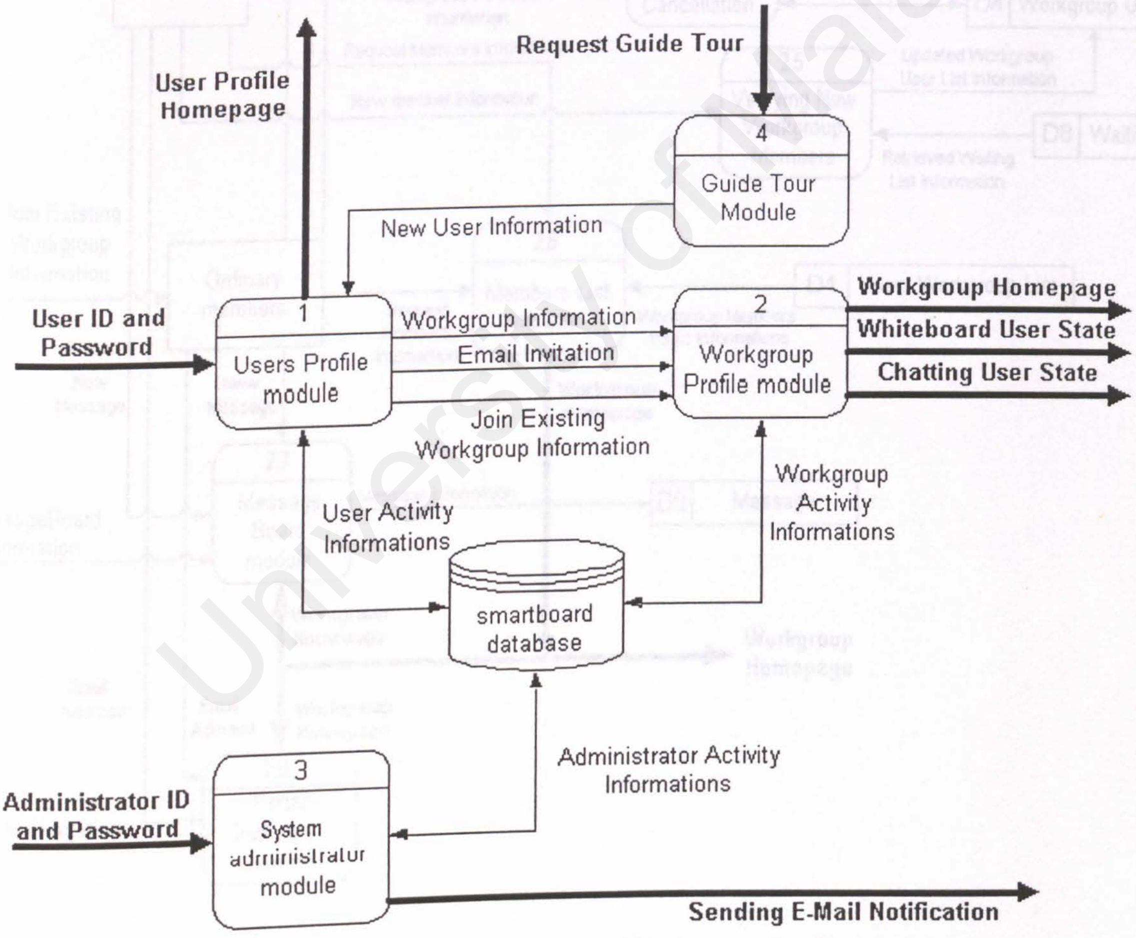


Figure 4.9 – Diagram 0 for smartboard

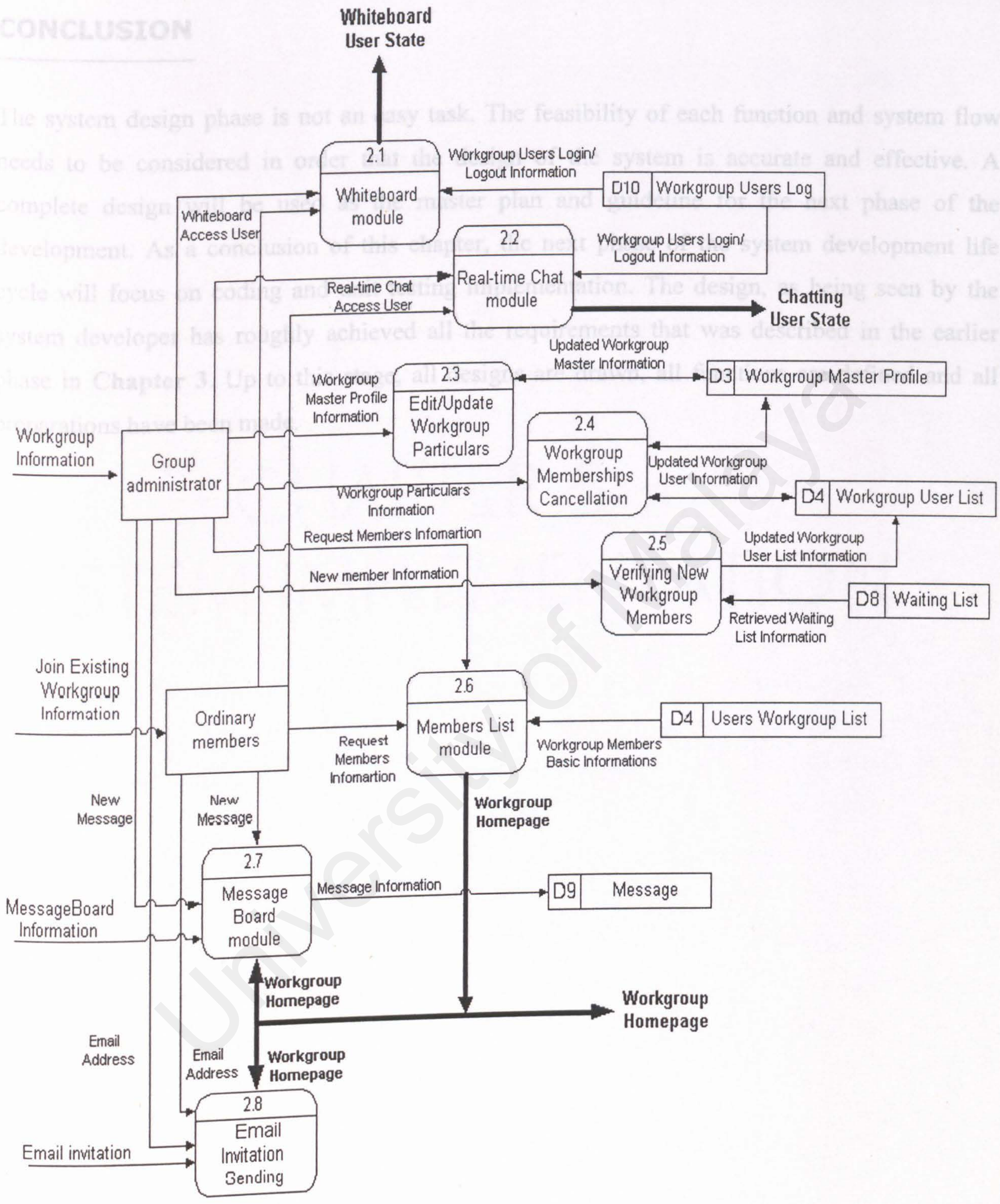

Whiteboard User State
2.1 Whiteboard module
Workgroup Users Login/ Logout Information
D10 Workgroup Users Log
Whiteboard Access User
2.2 Real-time Chat module
Workgroup Users Login/ Logout Information
Real-time Chat Access User
Chatting User State
Updated Workgroup Master Information
D3 Workgroup Master Profile
Workgroup Master Profile Information
2.3 Edit/Update Workgroup Particulars
2.4 Workgroup Memberships Cancellation
Updated Workgroup User Information
D4 Workgroup User List
Workgroup Information
Group administrator
Workgroup Particulars Information
Request Members Infomartion
New member Information
2.5 Verifying New Workgroup Members
Updated Workgroup User List Information
D8 Waiting List
Retrieved Waiting List Information
Join Existing Workgroup Information
Ordinary members
Request Members Infomartion
2.6 Members List module
D4 Users Workgroup List
Workgroup Members Basic Informations
Workgroup Homepage
New Message
New Message
2.7 Message Board module
Message Information
D9 Message
MessageBoard Information
Workgroup Homepage
Workgroup Homepage
Email Address
Email Address
Workgroup Homepage
2.8 Email Invitation Sending
Email invitation

Figure 4.10 – Child Diagram for Workgroup Profile Module

CONCLUSION

The system design phase is not an easy task. The feasibility of each function and system flow needs to be considered in order that the design of the system is accurate and effective. A complete design will be used as the master plan and guideline for the next phase of the development. As a conclusion of this chapter, the next phase of the system development life cycle will focus on coding and unit testing implementation. The design, as being seen by the system developer has roughly achieved all the requirements that was described in the earlier phase in **Chapter 3**. Up to this stage, all designs are drawn, all functions are defined and all preparations have been made.

CHAPTER 5 : SYSTEM IMPLEMENTATION

INTRODUCTION

This chapter describes in detail how the specified requirements and designs are implemented. The practical implementation done through system coding and debugging, focused on developing a series of classes that provides higher-level facilities for building a functional chat client-server system.

As an initial stage, the software development of the chat system begins by gathering requirements. A thorough study was done to evaluate the feasibility of each requirement and enhancement proposal before the overall objectives of the system were being clarified. The system design, then, focuses on a visible representation of those aspects of the system. After that was done, the system construction begins.

5.1 ENHANCING THE SYSTEM DESIGN

The design of the system discussed in the previous chapter acts as the master plan and framework during the coding phase. Nevertheless, due to some limitations and new opportunities discovered during this phase, the system needs to be redesign again in order to create a more enhance and newly improved system.

The voice-chat system will be implemented as a one-to-one conversation system. There are two main reasons why this is done.

Firstly, network bandwidth over the Internet is always inconsistent. A group conversation will require a broadcast server. The broadcast server must be able to receive and transmit multiple streams concurrently. This is not feasible for a large number of users considering that the bandwidth utilization at the server side will be extremely high. Worst, if the system is design to be a web-based system.

Secondly, a group conversation will eliminate the features of two-way live conversation. According to the previous design, users need to take turns to speak by pressing the Talk Button and the Hang Up Button. A one-to-one voice chat implementation scraps all these hassle. After activating the voice-chat, users are able to speak and hear each other without the need to perform

erform any steps. Thus, the Talk Button, Hang Up Button and Speak Status functions in the revious design are replaced by other functions such as Select User, Invite User and Voice Chat Jpdate functions. In accordance to this, it is no more necessary to include the private chat in the ext-chat system considering that it is already included in the voice-chat.

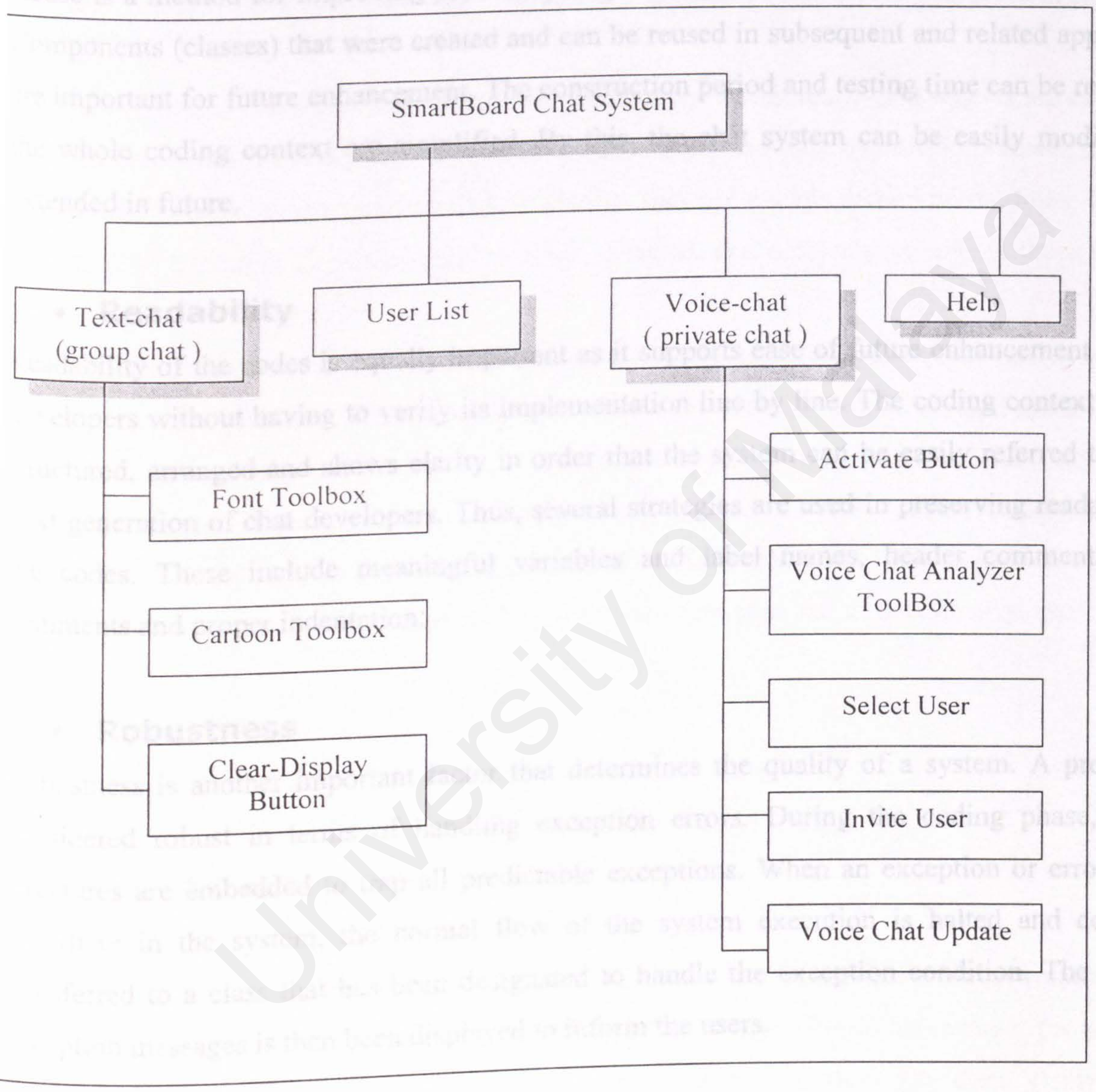


*Figure 5.1 : New Functional Diagram for Smart***Board** *Chat System*

Note : The role of these functions are described in more detail in the *User Manual Section* at *Appendix A.*

5.2 Coding Principles

The following are several criteria used as a guideline to convert the design into software coding.

- **Reuse**

Reuse is a method for improving product quality throughout the software development process. Components (classes) that were created and can be reused in subsequent and related applications are important for future enhancement. The construction period and testing time can be reduced as the whole coding context are simplified. By this, the chat system can be easily modified and extended in future.

- **Readability**

Readability of the codes is equally important as it supports ease of future enhancement by other developers without having to verify its implementation line by line. The coding context must be structured, arranged and shows clarity in order that the system can be easily referred to by the next generation of chat developers. Thus, several strategies are used in preserving readability of the codes. These include meaningful variables and label names, header comment blocks, comments and proper indentation.

- **Robustness**

Robustness is another important factor that determines the quality of a system. A program is considered robust in terms of handling exception errors. During the coding phase, control structures are embedded to trap all predictable exceptions. When an exception or error occurs anywhere in the system, the normal flow of the system execution is halted and control is transferred to a class that has been designated to handle the exception condition. The relevant exception messages is then been displayed to inform the users.

- **Maintainability & Ease of Testing**

An ideal system code should be structured systematically in such a way that program codes and functions of the same module are grouped together. In other words, high cohesion and loose coupling should be achieved. This is done through separating function classes between server side and client side, between modules and fields. Nevertheless, these functions must also be able to communicate with each other so that the whole system can be integrated.

5.3 CODING PARADIGM - OBJECT ORIENTED PROGRAMMING

Object Oriented Programming (OOP) is a paradigm in which real-world objects are each viewed as separate entities having their own state which is modified only by built in procedures, called methods. Because objects operate independently, they are encapsulated into modules, which contain both local environments and methods. *Smart***Board**'s objects are mainly the server and the clients.

In the chat coding context, objects are organized into classes, from which they inherit methods and equivalent variables. The object-oriented paradigm provides key benefits of reusable code and code extensibility and is ideal for the simulation-type problems commonly encountered in the chat system. There is no doubt that the principal features of OOP (namely encapsulation of related data and code into objects, and inheritance in a hierarchy of object classes) are of great benefit when it comes to constructing large software systems such as the chat system.

5.4 IMPLEMENTATION

The coding phase of the chat system reflects how the client-server architecture is being implemented. Due to the fact that the client and the server operates as separate programs, it is reasonable to develop the component as two different efforts. However, these efforts must be done concurrently in order that the two may communicate as expected. The *Figure 5.2* shows the client component and the server component for both the text-chat and the voice-chat application.

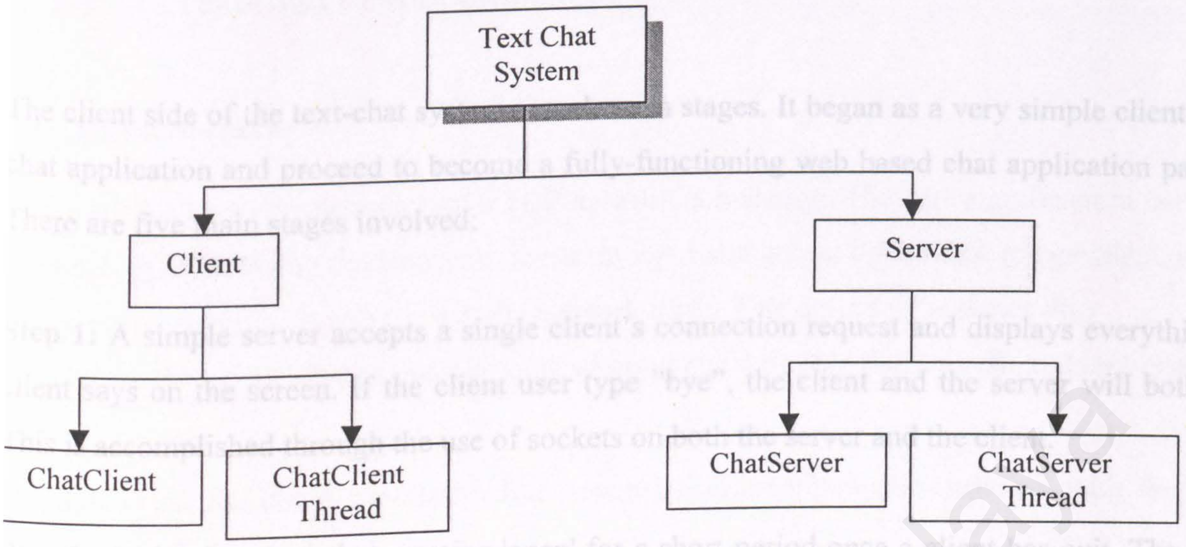


Figure 5.2a : Client and Server component for text chat application

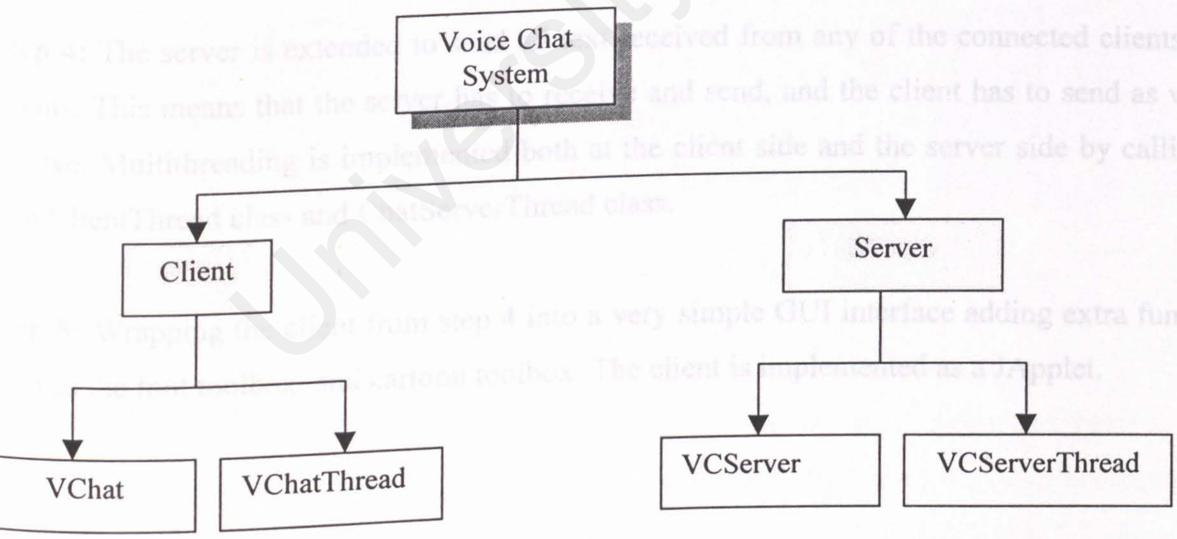


Figure 5.2b : Client and Server component for voice chat application

5.4.1 BUILDING THE TEXT-CHAT SYSTEM

The client side of the text-chat system was done in stages. It began as a very simple client-server chat application and proceed to become a fully-functioning web based chat application package. There are five main stages involved:

Step 1: A simple server accepts a single client's connection request and displays everything the client says on the screen. If the client user type "bye", the client and the server will both quit. This is accomplished through the use of sockets on both the server and the client.

Step 2: A simple server that remains 'open' for a short period once a client has quit. The server can handle at most one connection at a time. Here, single-threading is implemented on the server side.

Step 3: The server is extended to a multi-threading server that can handle multiple clients simultaneously. The outputs from all connected clients appear on the server's screen.

Step 4: The server is extended to send all text received from any of the connected clients to all clients. This means that the server has to receive and send, and the client has to send as well as receive. Multithreading is implemented both at the client side and the server side by calling the ChatClientThread class and ChatServerThread class.

Step 5: Wrapping the client from step 4 into a very simple GUI interface adding extra functions such as the font toolbox, and cartoon toolbox. The client is implemented as a JApplet.

5.4.1(a) Text-Chat Client Component

- **Client Sockets**

Sockets are the Java representation of a TCP network connection. They are significant to network programming by allowing developers to focus on input and output operations, independent of the intricacies and specifics involved with the network itself. This definitely matches the requirements that of handling a great amount of input and output in the *Smart***Board** text-chat system.

Using this class, the client can establish a stream-based communication channel with the text-chat server to send and receive messages.

To communicate with the server using TCP/IP, the client must first create a Socket to the server. This automatically establishes a TCP connection, throwing an exception if it fails. In addition to specifying a host name, it is necessary to specify a TCP port; For the *Smart***Board** system, all text-chat clients are connected to TCP port 12001 on the text-chat server

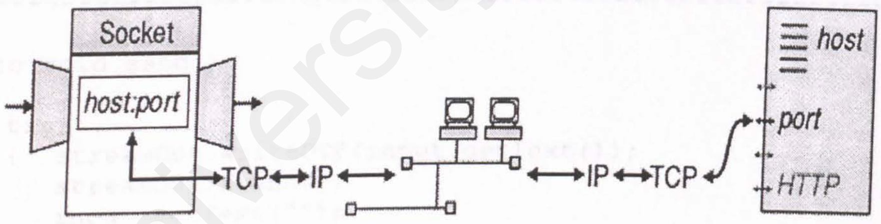


Figure 5.3 : Client Sockets

Once a Socket has been created, the Socket methods `getInputStream()` and `getOutputStream()` are used to obtain streams through which the client can communicate with the server. *Figure 5.4* shows how the socket method is implemented. *Figure 5.5* shows the class and function diagram for the text-chat client program.

Figure 5.4 : Client connects to the server and client sends message to the server

```
public void connect(String serverName, int serverPort)
{
    try
    {      socket = new Socket(serverName, serverPort);
           println("Let's Start Chatting! ");
           open();
     }

     catch(UnknownHostException uhe)
     {  println("Host unknown: " + uhe.getMessage()); }

     catch(IOException ioe)
     {  println("Unexpected exception: " + ioe.getMessage()); }
   }
//------------------------------------------------------------------------

  public void open()
  {
     try
     {  streamOut = new DataOutputStream(socket.getOutputStream());
        client = new ChatClientThread(this, socket);
     }

        catch(IOException ioe)
       {  println("Error opening output stream: " + ioe); }
  }

//------------------------------------------------------------------------

 private void send()
{
      try
      {  streamOut.writeUTF(input.getText());
         streamOut.flush();
         input.setText("");
       }

        catch(IOException ioe)
        {  println("Sending error: " + ioe.getMessage()); close(); }
 }
```

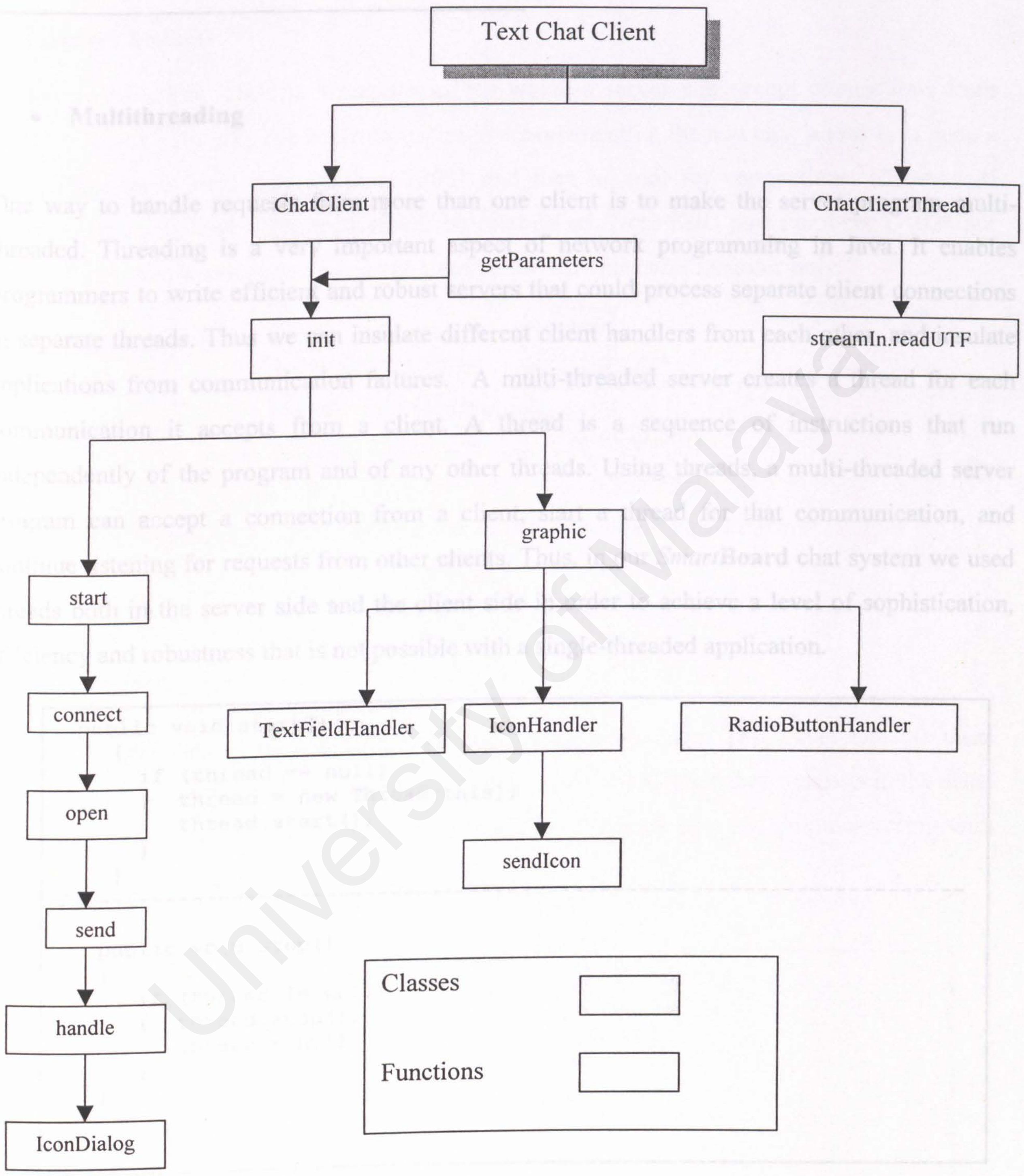


Figure 5.5 : Classes and function design for text-chat client program

5.4.1(b) Text-Chat Server Component

- **Multithreading**

One way to handle requests from more than one client is to make the server program multi-threaded. Threading is a very important aspect of network programming in Java. It enables programmers to write efficient and robust servers that could process separate client connections in separate threads. Thus we can insulate different client handlers from each other, and insulate applications from communication failures. A multi-threaded server creates a thread for each communication it accepts from a client. A thread is a sequence of instructions that run independently of the program and of any other threads. Using threads, a multi-threaded server program can accept a connection from a client, start a thread for that communication, and continue listening for requests from other clients. Thus, in our *Smart***Board** chat system we used threads both in the server side and the client side in order to achieve a level of sophistication, efficiency and robustness that is not possible with a single-threaded application.

```
  public void start()
     {
        if (thread == null)
        {  thread = new Thread(this);
           thread.start();
        }
     }
//--------------------------------------------------------------------

    public void stop()
    {
       if (thread != null)
       {  thread.stop();
          thread = null;
       }
    }
```

Figure 5.6 : Creating and stopping threads

- **Server Sockets**

The ServerSocket class is a mechanism by which a server can accept connections from clients across the network. The basic procedure for implementing the text chat server is to open a ServerSocket on a local port number, 12001 and then to wait for connections. Clients will connect to this port, and a connection will be established. Note that port numbers 1-1023 are reserved for system services and so can be used by the machine administrator only.

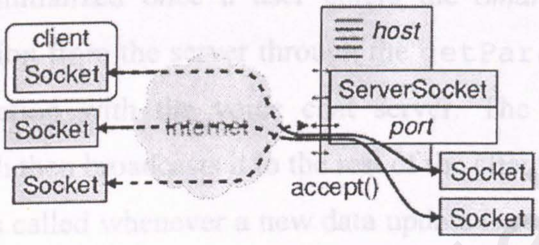


Figure 5.7 : Socket connection between the client and the server

- **ChatServer Class**

The ServerSocket class creates a Socket for each client connection. The server extracts these Sockets by calling the ServerSocket accept() method, and then handles these sockets in the usual manner, typically by extracting an InputStream and an OutputStream and communicating with the client through the standard streams interface.

```
public void run()
   {
      while (thread != null)
      {
       try
       {   System.out.println("Waiting for a client ...");
           addThread(server.accept());
        }
        catch(IOException ioe)
       {   System.out.println("Server accept error: " + ioe); stop(); }
      }
```

Figure 5.8 : ChatServer Class calls accept( ) method

5.4.2 BUILDING THE VOICE CHAT SYSTEM

Client Component

5.4.2(a) Initializing the voice chat system

The voice chat system is initialized once a user enters the *Smart***Board** workspace. After gathering sufficient information from the server through the `getParameter( )` function, the client establishes a connection with the voice chat server. The client sends its personal particulars to the server which then broadcasts it to the rest of the clients connected.
The `handle( )` function is called whenever a new data update is received from the voice chat server. The client then updates its own Vector list of client information whenever the new data is received.

5.4.2(b) Audio Transmit

The RTP data was transmitted via a session manager to the specified user. The following process is involved when implementing the AudioTransmit Class:

1. Create a Processor with a Data Source that represents the captured data.
2. Configure the Processor to output RTP-encoded data. This process needs to be blocked until the processor is configured. This phase is handled by the Blocker Class.
3. Get the output from the Processor as DataSource.
4. Create a SessionManager to the specified IP address.
5. Initialize the RTP session by calling RTPSessionMgr initSession.
6. Start the RTP Session by calling RTPSessionMgr startSession.

Figure 5.9 : Creating the processor for the microphone

```
AudioFormat format= new AudioFormat(AudioFormat.LINEAR, 8000, 16, 1);

 Vector devicesVector= CaptureDeviceManager.getDeviceList(format);

 CaptureDeviceInfo device = null;

 Enumeration devices = devicesVector.elements();
 boolean found = false;

 while (devices.hasMoreElements())
 {
   System.err.println("Capture device found!");
   device = (CaptureDeviceInfo)devices.nextElement();

 processor = Manager.createProcessor(device.getLocator());
 System.err.println("Processor started");
```

Figure 5.10 : Creating , initializing and starting the Session Manager

```
SessionManager sessionManager = new com.sun.media.rtp.RTPSessionMgr();
sessionManager.addReceiveStreamListener(new
sessionManagerListener(username));

sessionManager.addFormat(new dioFormat(AudioFormat.GSM_RTP,8000,8,1),18);

System.err.println("Create addresses");
SessionAddress localAddr = new SessionAddress();
InetAddress inetAddress = InetAddress.getByName(ipAddress);
SessionAddress destAddr = new SessionAddress(inetAddress, port,
inetAddress, port + 1);

dataOutput =  processor.getDataOutput();
System.err.println("Init Session manager!");

sessionManager.initSession(localAddr, srcDesList, 0.05, 0.25);
sessionManager.startSession(destAddr, 1, null);

System.err.println("Created RTP session: " + ipAddress + " " + port);
SendStream out = sessionManager.createSendStream(dataOutput, 0);
out.start();
processor.start();
System.out.println("Voice Chat On. Ready...");
```

5.4.2(c) SessionManagerListener

The session manager is also used to receive the RTP voice data from other users. The audio receive is handled by the SessionManagerListener class. These following procedures are involved in handling incoming voice:

1. Register the SessionManager in AudioTransmit class as a listener by calling RTPSessionMgr `addReceiveStreamListener( )`
2. In ReceiveStreamListener update method, watch for `NewReceiveStreamEvent`, which indicates that a new data stream has been detected.
3. When a `NewReceiveStreamEvent` is detected, retrieve the ReceiveStream from the `NewReceiveStreamEvent` by calling `getReceiveStream`.
4. Retrieve the `RTP DataSource` from the `ReceiveStream` by calling `getDataSource`. This is a PushBufferDataSource with an RTP-specific format. For example, the *Smart***Board** chat system used the GSM-RTP for audio encoding.
5. Pass the `DataSource` to `Manager.createPlayer` to construct a Player
6. Once the player has been constructed, the Player is sent to the `RTPPlayerWindow` where the VoiceAnalyzer Box is activated.

Figure 5.11 : Implementing the SessionManagerListener class

```
ublic class SessionManagerListener implements ReceiveStreamListener

public void update(ReceiveStreamEvent event)
{
  SessionManager source = (SessionManager)event.getSource();

  if (event instanceof NewReceiveStreamEvent)
  {
    ReceiveStream stream = null;
    stream =((NewReceiveStreamEvent)event).getReceiveStream();
    DataSource dsource = stream.getDataSource();
    newplayer = Manager.createPlayer(dsource);

    System.out.println("Voice Chat Running");
    showStatus("Voice Chat Running");
    }

  if (newplayer == null) return;

  playerlist.addElement(newplayer);
  PlayerControllerListener listener = new
  PlayerControllerListener(source, playerlist);
  newplayer.addControllerListener(listener);
```

5.4.2(d) RTPPlayerWindow

RTPPlayerWindow is a GUI component from the JMF package that triggers an interface to appear. This interface is the Voice Analyzer ToolBox that enables user to do the following

1. Mute the incoming voice
2. Lock and unlock the voice chat session
3. Increase the volume of the incoming voice
4. Check the voice packet flow through the whole voice chat system.
5. Halt the voice chat session by disposing the VoiceAnalyzer Box

A clock is also displayed to inform the user the duration of usage.

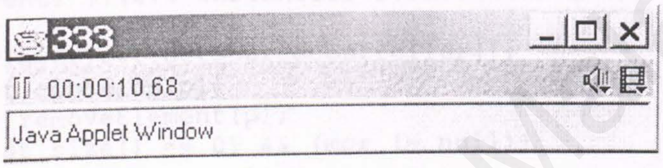


Figure 5.12 : Voice Analyzer ToolBox - RTPPlayerWindow

```
class RTPPlayerWindow extends PlayerWindow
  {

    public RTPPlayerWindow( Player player, String title)
    {
      super(player);
      setTitle(title);

    }

    public void Name(String title){ setTitle(title);}

  }
```

Figure5.13 : RTPPlayerWindow class

5.4.2 (e) PlayerControllerListener

The PlayerControllerListener class listens for update events generated by Controllers. Whenever it detects a ControllerClosedEvent, ControllerErrorEvent or DeallocateEvent, it triggers the voice chat session to stop. The particular player is then removed from the playerList Vector.

Figure 5.14 : Registering PlayerControllerListener as controllerUpdate

```
public void controllerUpdate( ControllerEvent evt)
{
  if ((evt instanceof ControllerClosedEvent) ||(evt instanceof
ControllerErrorEvent) ||(evt instanceof DeallocateEvent))
  {
    Player p = (Player)evt.getSourceController();
    if (playerlist.contains(p))
      playerlist.removeElement(p);
    if ((playerlist.size() == 0) && (mgr != null))
    {   mgr.closeSession();
        off();
     }
    }
```

5.4.2(f) Blocker

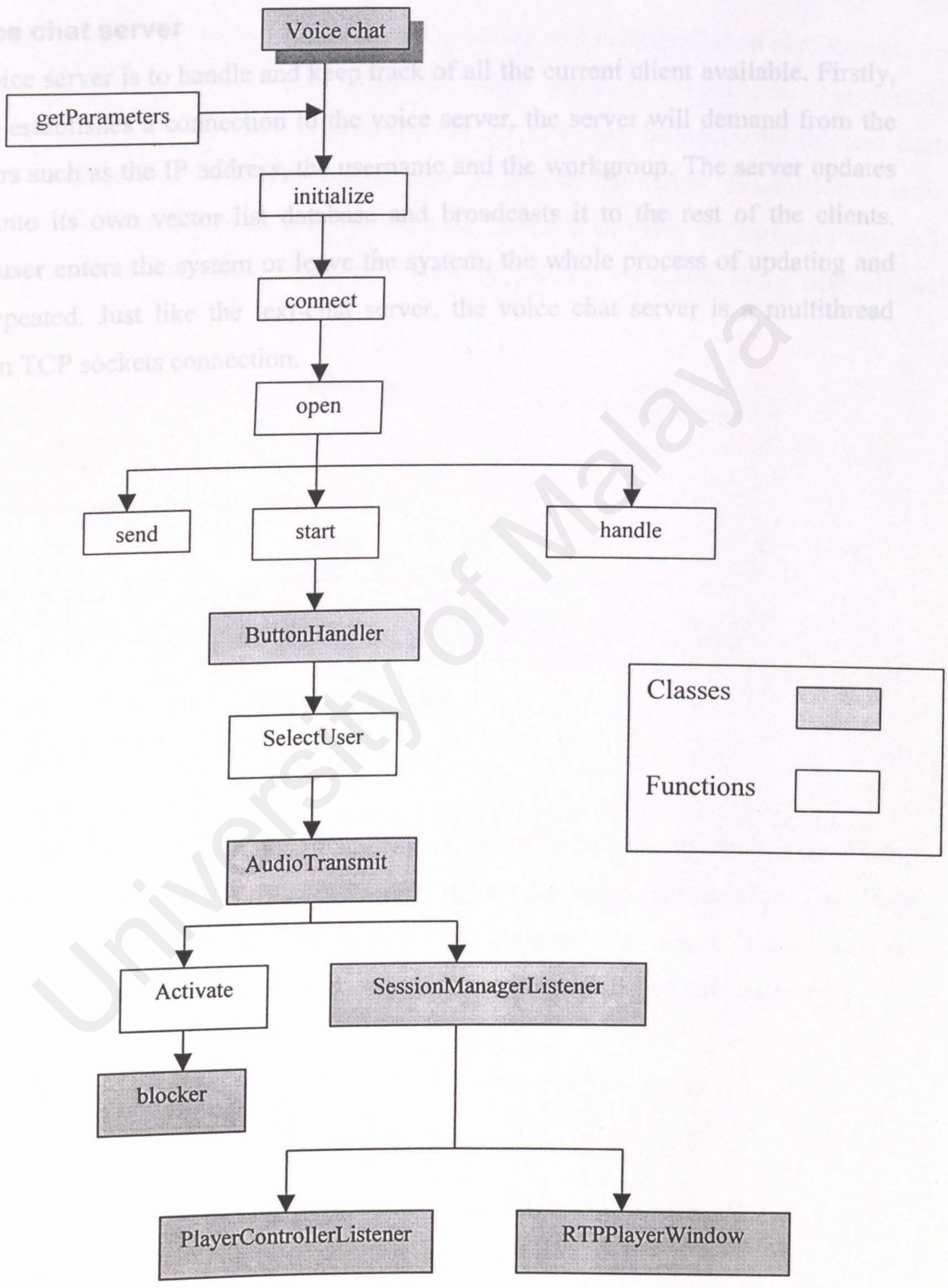
The objective of the Blocker class is to block the control program to continue until the processor for capturing new data is ready. Here, the processor is configured and moved into the realize state.

Figure 5.15 : Configuring the processor into different state

```
public boolean configure()
{
    ControllerListener listener = new ControllerListener()
      {

      public void controllerUpdate(ControllerEvent ce)
      {
      System.err.print("Event state = ");
      switch(processor.getState())
       {
      case Processor.Prefetched:  System.err.println("Prefetched");
                                  break;
      case Processor.Prefetching: System.err.println("Prefetching");
                                  break;
      case Processor.Realized:    System.err.println("Realized");
                                  break;
      case Processor.Realizing:   System.err.println("Realizing");
                                  break;
      case Processor.Started:     System.err.println("Started");
                                  break;
      case Processor.Unrealized:  System.err.println("Unrealized");
                                  break;
      case Processor.Configured:  System.err.println("Configured");
                                  break;
      case Processor.Configuring: System.err.println("Configuring");
                                  break;
      default: System.err.println("NOSTATE"); break;
      }
```

Figure 5.16 : Class and Function Diagram for voice-chat program



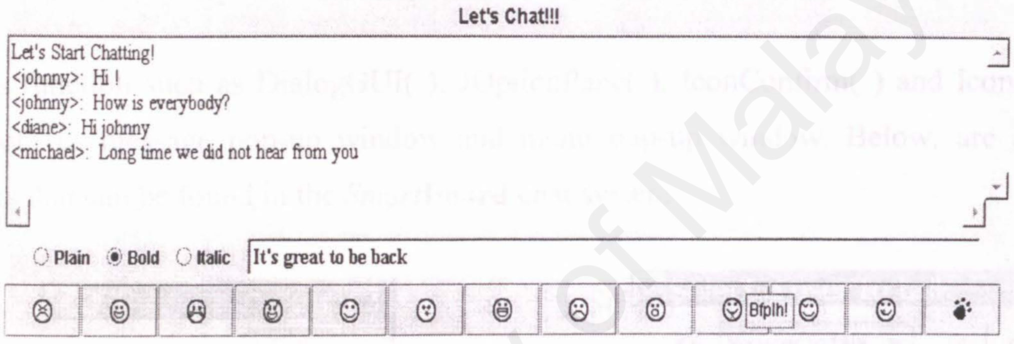
ERVER COMPONENT

Role of the voice chat server

The role of the voice server is to handle and keep track of all the current client available. Firstly, whenever a client establishes a connection to the voice server, the server will demand from the client its particulars such as the IP address, the username and the workgroup. The server updates this information into its own vector list database and broadcasts it to the rest of the clients. Whenever a new user enters the system or leave the system, the whole process of updating and broadcasting is repeated. Just like the text-chat server, the voice chat server is a multithread system that runs on TCP sockets connection.

5.4.3 GUI COMPONENT

The Java Swing package is deployed as the GUI programing tool in *Smart***Board** chat system. The TextChat Window includes the swing packages such as JTextArea, JTextField, JRadioButton and JButton. The text area is to display the messages broadcast by the text chat server while the text field is for input from the user to send to the server. JRadioButton is to handle the different fonts that is to be displayed and the JButtons contains the cartoon icons. All these were grouped into intermediate container JPanel. This JPanel are added to the contentPane of the JApplet.


Let's Chat!!!

Let's Start Chatting!
<johnny>: Hi !
<johnny>: How is everybody?
<diane>: Hi johnny
<michael>: Long time we did not hear from you

○ Plain ◉ Bold ○ Italic

It's great to be back

Figure 5.17: TextChat Window

The Voice Panel and the VoiceAnalyzer Box are part of the voice chat function. The Jbutton function allows the user to click on the activate button to start the voice chat function. The Voice Analyzer ToolBox automatically appears when the voice chat session is started. Voice Analyzer ToolBox is created using the RTPPlayerWindow class to represent the voice data received

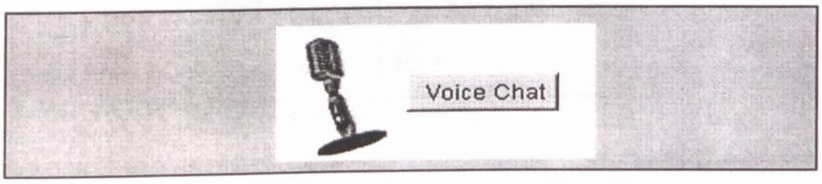


Figure 5.18 : Voice Panel

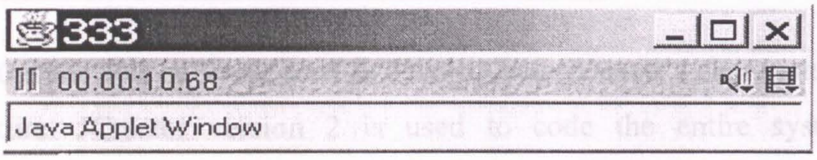


Figure 5.19 : Voice Analyzer ToolBox

Different Layout Manager are used for different purposes. For example, the iconPanel in the text chat Window are used GridLayout Manager to add buttons on the panel. The whole textChat Window uses BorderLayout to arrange its position to the Frame.

Dialogs function such as DialogGUI( ), JOptionPane( ), IconConfirm( ) and IconDialog( ) are used both as message pop-up window and menu pop-up window. Below, are a few of the diaologs that can be found in the *Smart***Board** chat system .

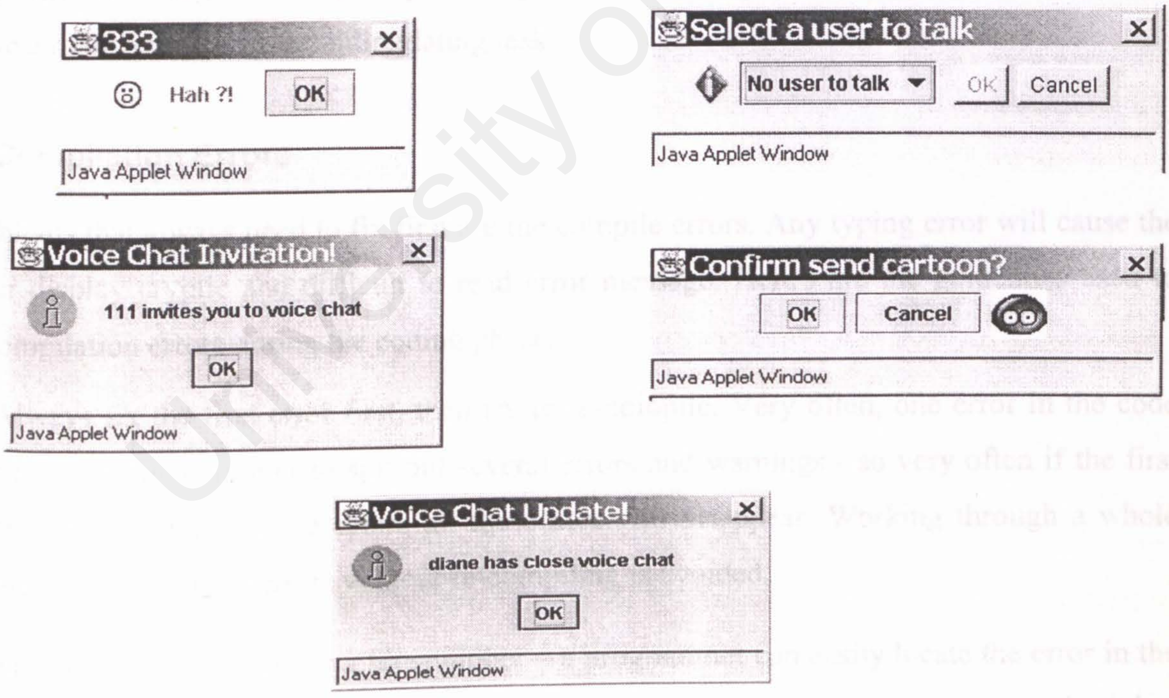


Figure 5.20 : Dialog () function creating pop-up window box

5.5 DEVELOPMENT TOOLS IMPLEMENTATION

JCreator version 2.0 is the sole software used to develop *Smart***Board**'s chat system. Based on its object-oriented nature. JCreator version 2 is used to code the entire system. This Java programming tool provides the basic development environment such as graphical editor and debugger. This had helped in reducing and streamlining a certain degree of organizational work involved in developing the class hierarchy for the applet.

5.6 DEBUGGING MECHANISM

Most programmers have errors in their code. For most programs, debugging and testing usually takes significantly longer than actually writing code. When a program is broken, fixing it can seem like a really confusing and intimidating task.

5.6.1 Compilation Errors

The problems that always need to fix first are the compile errors. Any typing error will cause the computer display cryptic and difficult to read error message. Here's are the guidelines used to debug compilation errors during the coding phase.

- Always fix the first error first, then try to re-compile. Very often, one error in the code can cause the compiler to spit out several errors and warnings - so very often if the first error is fix and compile again, all the others will disappear. Working through a whole long list of compile errors without re-compiling is avoided.
- The error always contains a line number – a programmer can easily locate the error in the code by referring to the line number. However, the line number isn't always exactly right.
- "Syntax error before or at" usually means that the programmer forgot to add important Java language symbols (semicolon, bracket, parenthesis).

- "undefined symbol : [symbolname]" means a word in the program that the compiler didn't recognize, usually a method or variable name. This may cause by typing error

- Other miscellaneous things that could cause errors such as unmatch types/parameters of the prototypes with those of the actual functions, clash variables and so on.

5.6.2 Runtime Errors -- my program compiles, but crashes or has bugs

This is where debugging becomes more of an art form. The program does something, but not what expected. This can be frustrating, because code is supposed to be predictable, and it's supposed to do what is expected. The following approaches is taken to see what iss *really* going on inside the code, which will help the fixing process.

- If the code seems to be misbehaving, executing one line at a time (this usually requires a pen and paper to keep track of variable values, etc.). This is a "sanity check", to make sure the code actually does what is originally intended it to do.

- A great way to make sure the code is getting executed is to put System.out.println() statements at strategic places in the code to find out what's getting executed and what's not. System.out.println is helpful as it can also display the corrupt data that causes a heap of errors.

- Comment out blocks of code that are suspected causing problems (using /* and */)... one may find that if comment out a single line, suddenly the program doesn't crash.

5.6.3 Debugger

The debugger is always said as the ultimate tool for debugging.A debugger is the only way a programmer can really "watch" the computer execute the code, step-by-step. It will display the values of variables in memory, it will display which lines make the programs crash, it will let the programmer stop the code in the middle of an infinite loop to find out what's going wrong. Obviously, this is a very powerful tool. The debugger tool is also included in the JCreator software, the tool that is used in the *Smart***Board** development.

5.6.4 Exception Handling

One of the useful debugging mechanisms used in this project is exception handling. This technique focuses on detecting and responding to unexpected events at runtime. To handle exceptions in the program code, some potentially troublesome codes are enclosed within a try clause. A try clause is a special Java construct that tells the runtime system that a section of code could cause trouble. Another piece of code (a handler) is needed in a corresponding catch clause that responds to errors caused by the code in the try clause. The error event itself is the exception and the code in the catch clause is known as an exception handler. The following is some exception handling code that embedded in the JApplet:

Figure 5.21 : Example of exception handling

```
try
          {   System.out.println("Waiting for a client ...");
              addThread(server.accept());
           }
         catch(IOException ioe)
          {   System.out.println("Server accept error: " + ioe); }
```

CONCLUSION

Implementing the design system into codes and program appears to be a complicated task. Not only it requires programming knowledge but one need to equip himself also with logic thinking, debugging skills and research ability. On top of that, the developer need to have a positive spirit and easily give up when difficult time comes. The next phase of the system development is to test and verify the efficiency and accuracy of each unit, modules, functions and classes. The next chapter will discussed on System Testing.

CHAPTER 6 : SYSTEM TESTING

INTRODUCTION

Whatever that is implemented in the system should be tested thoroughly to ensure that the system performs according to its specifications and in line with users' requirements and expectations. Generally, the main objective of carrying out the system testing is to search for faults or defects in the system, seek the root cause, strategize and finally fix it. This phase is done throughout the development of the chat system. When the coding phase was completed, the detailed and thorough test begins.

This chapter describes the steps and the techniques used to carry out the system testing

It begins with the unit testing which is the simplest test of all followed by the module testing. Then, all modules are integrated and integration testing is performed. Finally, system testing is done on the entire system to ensure it works as a whole. *Figure 6.1* depicts the life cycle of a testing process.

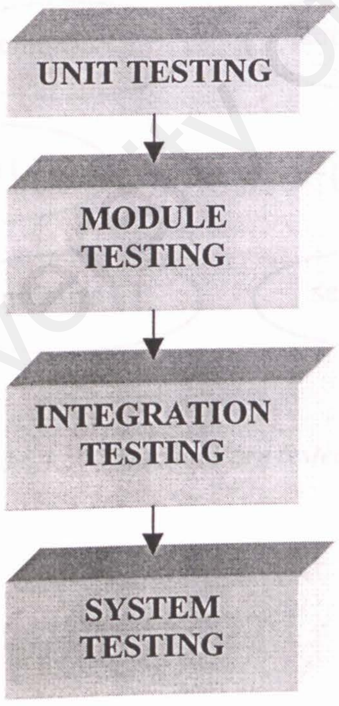


Figure 6.1 : Life cycle of a testing process

6.1 TESTING PROCESS

6.1.1 UNIT TESTING

Unit testing verifies the small unit of codes (subroutines or functions), as a stand-alone unit to locate errors. It is a way of making sure that the code works the way the developer thinks it should. It is the initial testing stage for the completion of each component class. This process enables the tester to detect errors in coding and logical mistakes that are contained within that component alone. In the testing of *Smart*__Board__'s chat system, functions in the program are usually tested as a unit. Below are the examples of different functions tested separately as an unit. Every line is tested to make sure that this unit is free from bugs and errors and it performs expectedly.

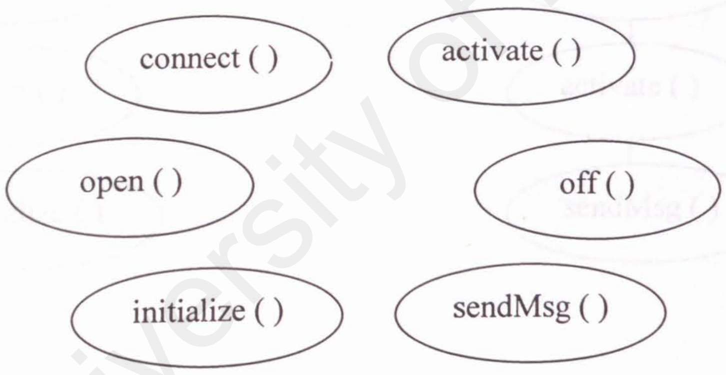


Figure 6.1 : Functions are tested as a unit

6.1.2 MODULE TESTING

The unit testing is followed by module testing. A module is a collection of dependent components. A module encapsulates related components. The module tests exercise a module independently of other modules. In the *Smart***Board** code development, related functions are grouped as a module called class. Here, these group of functions of the class is tested to ensure that it performs accordingly. *Figure 6.2* shows classes such as AudioTransmit class and SessionManager class

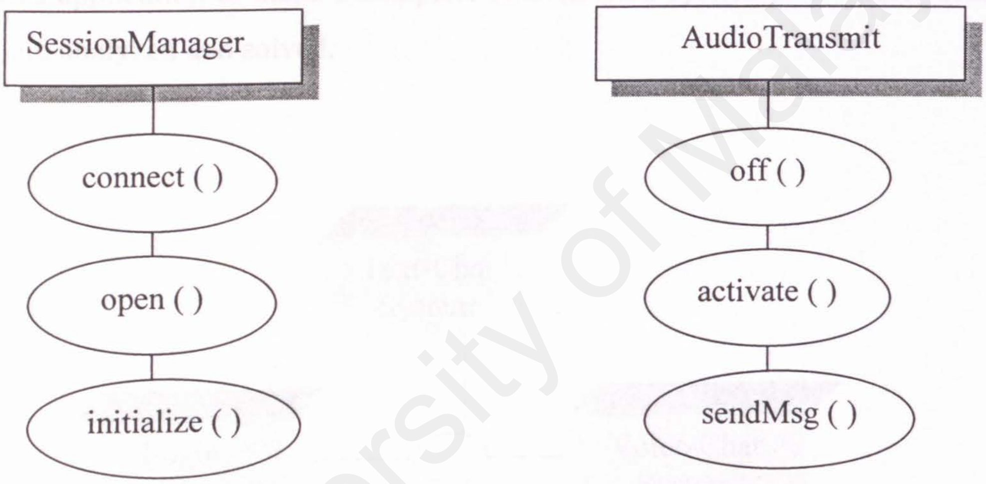


Figure 6.2 : Module Testing

6.1.3 INTEGRATION TESTING

After the module testing, the interaction and interfaces between the modules and subroutines are tested in order to ensure that the whole system are in accordance with the system specifications. In the *Smart***Board** development, the approach of merging components to test the system as a whole involves two stages. **First**, is the merging of all the classes to form the chat system. The relations between each class to another are tested and approve. Each parameter that is passed from one class to another is checked. The whole chat system is then, tested before the second stage of of integration testing begins. **Second** stage of the integration testing is the main effort when the chat system is implemented together with other systems such as the web page, the whiteboard application to make a complete *Smart***Board** system as a whole. Clashes with other systems are analyzed and solved.

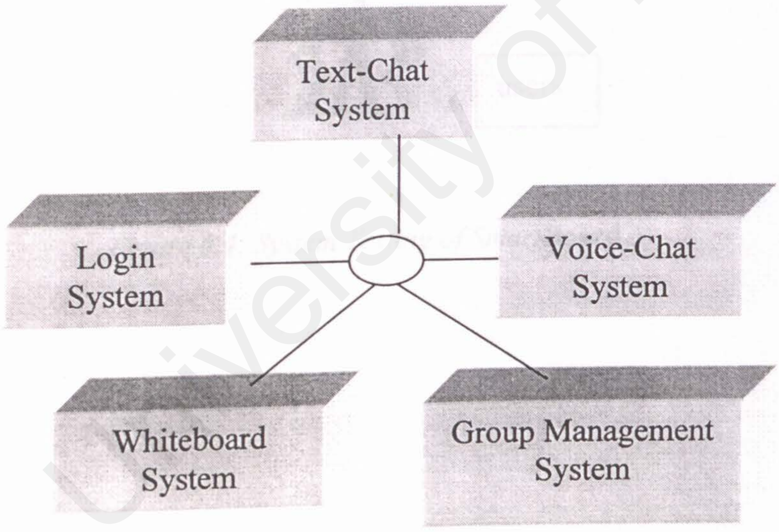


Figure 6.3 Integration Testing

6.1.3 SYSTEM TESTING

System testing is the final stage of testing prior to being placed in operational use. The chat system tested with procurer data rather than simulated data. Selected sample of users plays the main role to ensure that the system is effective, faultless and ready in use. Feedbacks and comments becomes the important resources for future enhancement. Diferent type of platform, or networks are used as a testing frame to ensure the robustness of the system

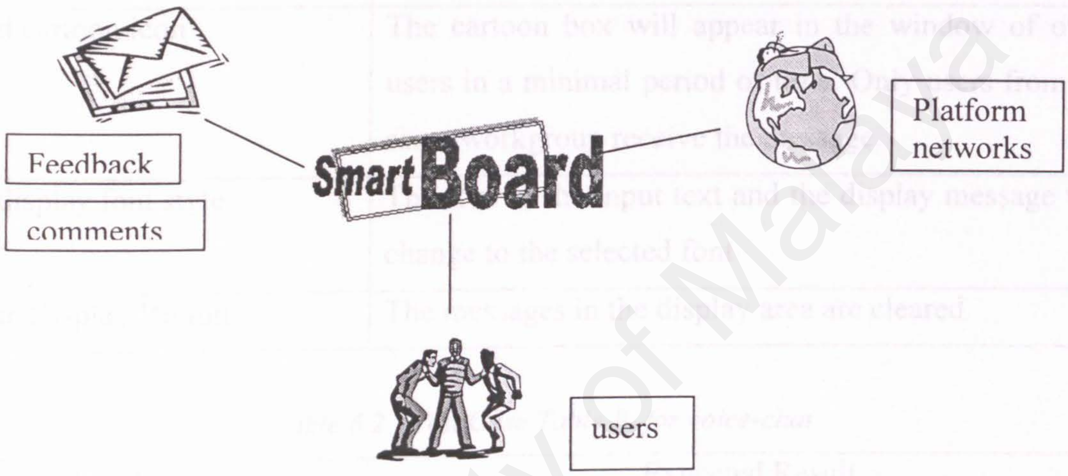


*Figure 6.4: System Testing of Smart***Board**

6.2 Test Case

A test case is a set of input data and expected results that exercises a system with the purpose of causing failures and detecting faults. *Table 6.1* shows the sample test case being used as a test bed in the development of the *Smart***Board** chat system.

Table 6.1 : Test CaseTable A for text-chat

Test Case	Expected Results
1. Connection to the system	The chat applet is downloaded in a minimal period of time. The client and the server are able to connect in a minimal period of time.
2. Send / receive text message	Message sent to the server is broadcasted in a minimal period of time. Only users from the same workgroup receive the message.
3. Send cartoon icon	The cartoon box will appear in the window of other users in a minimal period of time. Only users from the same workgroup receive the message.
4. Set display font style	The font of the input text and the display message will change to the selected font
5. Clear-Display Button	The messages in the display area are cleared

Table 6.2 : Test Case Table B for voice-chat

Test Case	Expected Result
1. Connection to the system	The voice-chat applet is downloaded in a minimal period of time. The client and the server are able to connect in a minimal period of time.
2. Clicking the activate button	A dialog box containing the correct current users appears. Users are able to select another user that he/ she wants to "voice" chat with.
3. Activate the voice chat	The system is able to detect the microphone attached to the computer. The processor is configured. The session manager is initialized and started.
4. Speaking through the mic	The voice of the sender is projected from the speakers at the recipient side in a minimal period of delay.
5. Receiving incoming voice	Vice-versa, voice from the sender is heard through the speakers. The voice is expected to be in good quality

	and delay must be minimal. Voice Analyzer ToolBox will appear.
6. VoiceAnalyzer ToolBox functions	Users are able to do the following a. Mute the incoming voice b. Lock and unlock the voice chat session c. Increase the volume of the incoming voice d. Check the voice packet flow through the whole voice chat system. e. Halt the voice chat session by disposing the VoiceAnalyzer Box.

6.3 TEST FRAME

Test Frame is the condition or the characteristic of the overall system when the program is tested. Several types of conditions and characteristics are deployed as different test frames to ensure the robustness of the chat program. Here, an example test frame of the test frame is being used together with the test case to produce the test results:

System : Windows 2000 system with 128MB RAM, Pentium III 500 MHZ processor.

Network : Internet with bandwidth of 10Mb

Speakers : 20 Watt

Microphone : 16 bit and 8 bit with 8000 Hz and 44.1 KHz

6.6 TEST RESULTS AND PERFORMANCE

The final product of the *Smart***Board** chat system are shown by some snap shots in the User Manual section at Appendix A. The following are the test results and the performance of the *Smart***Board** chat system based on the test case and the test frame discussed earlier.

Test Case Table A : Text-Chat

Test 1 : **Connection to the system**

Loading time varies depending on the overall system. The time of loading the TextChat window after the user logs into the Activities Park Window is about 2 seconds. The time to connect to the text-chat server is 3 seconds.

Test 2 and Test 3 : **Send - Receive text message / Send cartoon icon**

The delay between the time when client A sends a text message till the time when client B receives it is not noticeable. Only users from the same workgroup receive the message. The same applies when the user sends the cartoon icon and another user receives it. The cartoon box appeared in B's window instantly.

Test 4 : **Set display font style**

The font of the input text and the display message change to the selected font.

Test 5 : **Clear-Display button**

When clicked, the messages in the display area are cleared.

****Overall Testing** : It performs as expected without any exception or error occurred

Test Case Table B : Voice-Chat

Test 1 : **Connection to the system**

Loading time varies depending on the overall system. The time of loading the Voice-Chat Panel after the user logs into the Activities Park Window is about 4 seconds. The time to connect to the voice-chat server is 3 seconds.

Test 2 : **Clicking the activate button**

A dialog box containing the correct current users appears. Users are able to select another user that he/ she wants to "voice" chat with.

Test 3 : **Activate the voice chat**

The system is able to detect the microphone attached to the computer. The processor is configured. The session manager is initialized and started. Message on the status bar changes from *"Voice Chat Initializing"* till *"Voice Chat On. Voice Chat Ready"*. All this took place in a period of 6 seconds

Test 4 and Test 5 : **Speaking / Receiving**

The voice chat application was discovered to have latency, even though in a acceptable scale. The delay when client A speaks "Hi!" until the time when client B hears it is approximately 3 seconds. Jitters in the voice are not noticeable.

The voice quality that is heard in this application depends on the type of microphone used and the settings of the computer speakers. *Smart***Board** voice-chat application transmit voice packet data in this format:

Bit	:	16
Hertz	:	44.1 KHz
Format	:	Linear
Type	:	GSM

Microphones that meet exactly the criteria above work almost perfectly. Background noise behind is not noticeable and the voice received is clear, with minimum amount of distortion and jitters.

Microphones that capture data in 8 bits appear less favorable. This is because the noise and voice distortion is more noticeable. However, one can hear quite clearly what the other user is trying to say.

Test 6 : **Voice Analyzer ToolBox**

Users are able to do the following

f. Mute the incoming voice
g. Lock and unlock the voice chat session
h. Increase the volume of the incoming voice
i. Check the voice packet flow through the whole voice chat system.
j. Halt the voice chat session by disposing the VoiceAnalyzer Box.

****Overall Testing** : The program performs as expected.

CHAPTER 7 : SYSTEM EVALUATION

INTRODUCTION

The evaluation of the system is equally important as any other activities done in developing the *Smart***Board** chat system. It is important as it becomes the platform for future enhancement and expansion of the system process in order to become competitive when it is ready to penetrate into the collaborative market. Evaluation is done by a wide category of users to recognize the strength and weaknesses in the system.

7.1 STRENGTH AND BENEFIT

7.1.1 Cross Platform Support

The major benefit of the *Smart***Board** chat system is the usability in great variety of browsers and client platform. Any Java-capable web browser, such as Internet Explorer (version 5.5 and above) and Netscape Navigator (version 4.5 and above) can run the system as efficiently. As a result, there is always an open door for worldwide audience and users.

7.1.2 Multiple Connection

Through multithreading, the text-chat server and the voice-chat server are capable of handling more than one user simultaneously. *Smart***Board** chat users do not face any delay issue compared to many other Internet applications available.

7.1.3 Ease of Use Graphical User Interface

*Smart***Board**'s chat system provides an attractive and easy to use graphical interface for its audience. It is easy to understand and operate without involving any complicated steps.

7.1.4 Wide Accessibility

*Smart***Board** chat users are not required to pay anything to use the system. It is a free application for worldwide audience and it is also easily accessible through the World Wide Web.

7.1.5 High Quality System

Through the feedback and comments from different category of users, the *Smart***Board** chat system is proven to be a system that posses high quality. It is developed in such a manner that error-probability is at its minimum. Besides, it provides the sufficient functions and tools that a common user needs.

7.2 LIMITATIONS AND FUTURE ENHANCEMENT

"One can never find a perfect system but that doesn't mean that one can never find an almost perfect system". This is also proven for the *Smart***Board** system through the many evaluation received. Below are some of the limitation of *Smart***Board** chat system that can be enhance and improved in the future.

7.2.1 Loading time

The chapter before has described in detail the loading time report for the Voice Chat Panel and the Text-Chat Window. It depends mainly on the speed of connection that the user is using. Better compression methods are suggested to improve the overall loading period of the system.

7.2.2 Noise and Jitters

The issue of noise and jitters occurs because of incompatible types of microphone used. In future, *Smart***Board** chat system can be improved by allowing the user to select its own configuration that matches its microphone type. Better still, if the chat system itself can automatically trigger a Wizard application that can configure the system to match the users' microphone.

7.2.3 Voice delay and latency

Latency is still one of the main chat system limitations. Voice captured form one end of the network will only be heard at the other end after a small amount of delay. To reduce the impact of the problem, a developer needs to implement to better encoding and compression techniques. Voice data can be encoded and packetized into smaller units before being sent through the network.

7.2.4 Multi-User Broadcasting

Due to the constraints of the bandwidth, *Smart***Board** chat system is only limited to one-to-one voice chat. This, in future, can be enhance according to the progress of JMF technology, to support multi user broadcasting features.

7.2.5 Video Conferencing / Voice mail / File Sharing

Other communication tools that can included in the *Smart***Board** system are voice conferencing, voice mail, video mail and file sharing.

Through video conferencing, users are able to see each other live over the net. They could converse with each other, face-to-face as though they are at the same location.

Voice mail enables user to leave an audio message of the recipient is not online. Video mail function like wise but with the extra features of video images.

Through file sharing, users are able to send photos, documents, graphics, scanned images and multimedia file to each other.

CONCLUSION

People need to connect with other people and business processes to get their work done. Much like the step between the telegraph to the telephone, the step to Internet technology connects people with a richer flow of information. Collaboration can bring people all over the world together with more efficient use of time and at lower cost, by eliminating the need to travel. The emergence and wide-spread adoption of the World Wide Web (WWW) offers a great deal of potential for the developers of collaborative technologies, both as an enabling infrastructure and a platform for integration with existing end-user environments. The complete development of *Smart*Board system has contributed another success to the collaborative concept of online communication. With the features and functions available, it is hope to make an impact in the collaborative market which is ever expanding.

By then, the objectives are met and *Smart*Board will be a satisfactory product for users all around the globe.

CONCLUSION

People need to connect with other people and business processes to get their work done. Much like the step between the telegraph to the telephone, the step to Internet technology connects people with a richer flow of information. Collaboration can bring people all over the world together with more efficient use of time and at lower cost, by eliminating the need to travel. The emergence and wide-spread adoption of the World Wide Web (WWW) offers a great deal of potential for the developers of collaborative technologies, both as an enabling infrastructure and a platform for integration with existing end-user environments. The complete development of *Smart***Board** system has contributed another success to the collaborative concept of online communication. With the features and functions available, it is hope to make an impact in the collaborative market which is ever expanding.

By then, its objectives are met and *Smart***Board** will be a satisfactory product for users all around the globe.

APPENDIX A :
USER MANUAL

INTRODUCTION

*Smart***Board** chat system is a web-based application that enables users from different locations of the globe to communicate together in a real-time manner. It involves two types of online communication. They are text-message exchanging via text-chat and voice conversation via voice-chat.

*The Smart***Board** chat system has been developed to be a user-friendly and easy-to-use system that enables users of different categories to reap the benefit of the system. ***Appendix A*** describes and guides a common user on how to utilize the system functions and features. Here, an example of how a user, called "Johnny" manipulate the system to give a step-by-step guide for the reader.

The contents are as follows:

A.1 Logging into the Activities Park Window

The following are the step of how one user, called Johnny, could access to the *Smart***Board** chat system interface.

Step 1 : *Accessing the SmartBoard system*

Firstly, Johnny will have to open an Internet browser (IE version5.5 and above/Netscape Navigator version 4.5 and above). At the address text field, Johnny need to insert the *Smart***Board** URL below to access the main window:

http: *//202.185.109.102:88/smartboard/index.jsp*

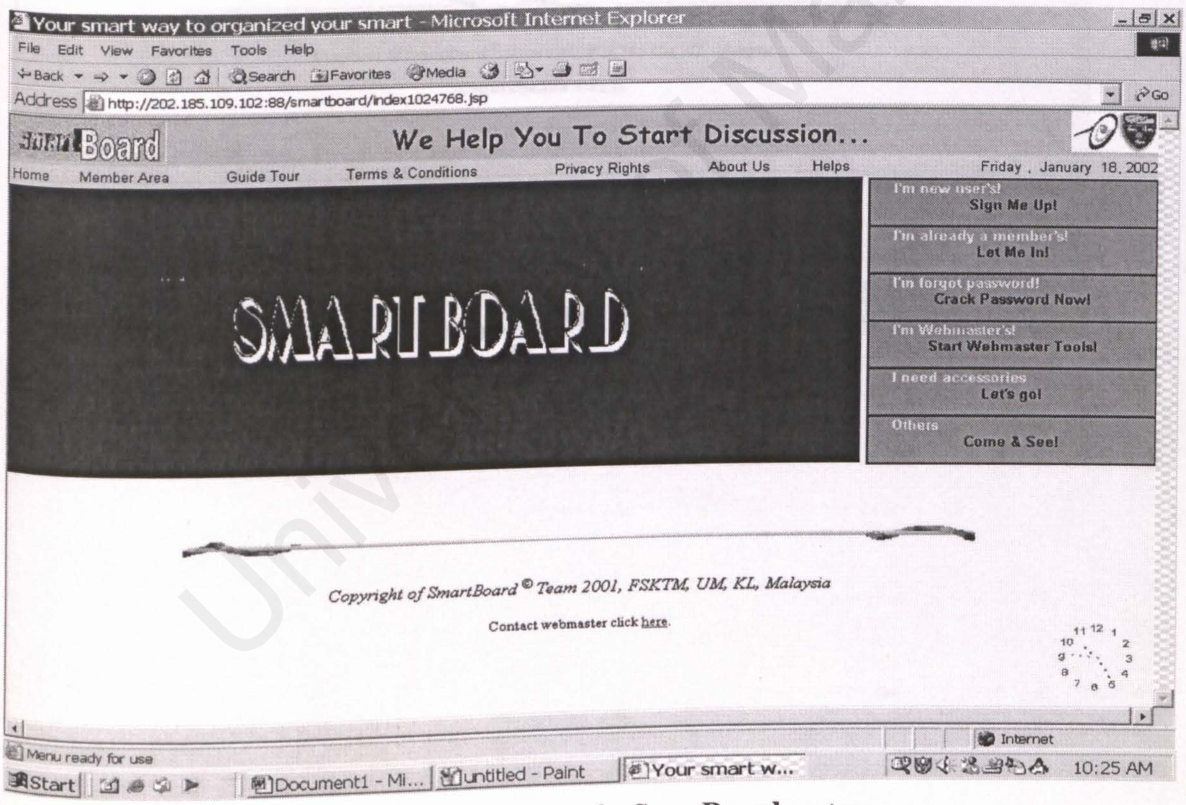
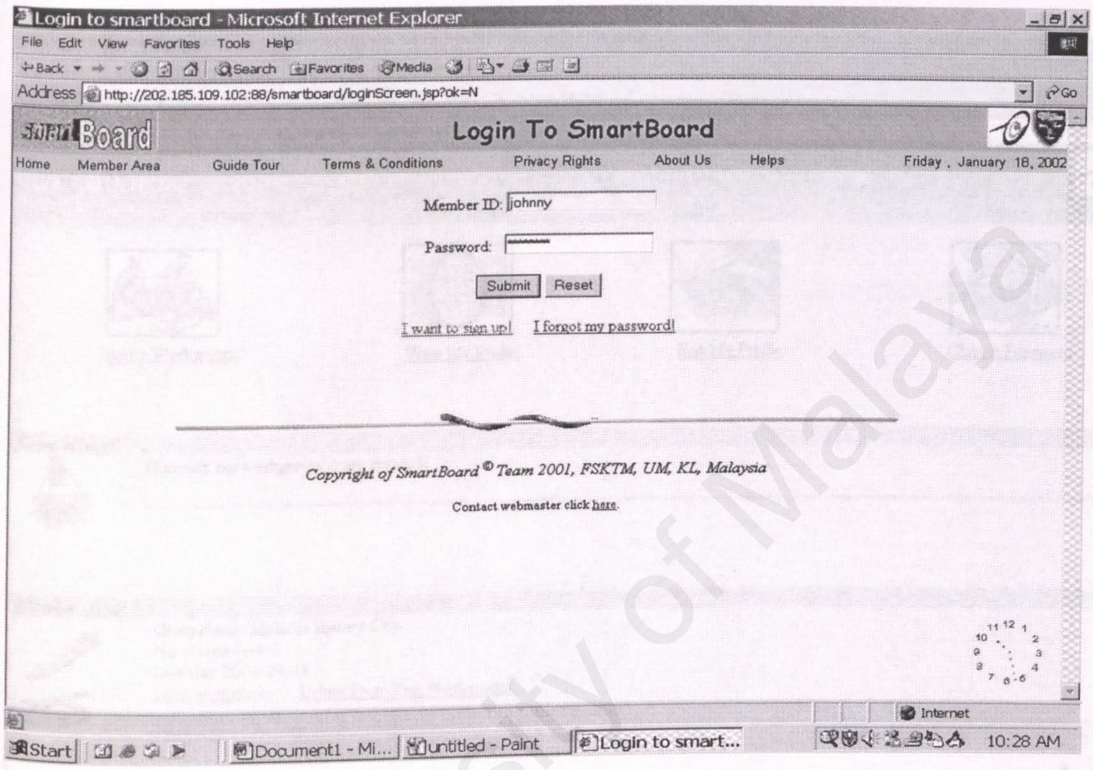

Your smart way to organized your smart - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://202.185.109.102:88/smartboard/index1024768.jsp
Go
We Help You To Start Discussion...
Home Member Area Guide Tour Terms & Conditions Privacy Rights About Us Helps
Friday , January 18, 2002
I'm new user's! Sign Me Up!
I'm already a member's! Let Me In!
I'm forgot password! Crack Password Now!
I'm Webmaster's! Start Webmaster Tools!
I need accessories Let's go!
Others Come & See!
SMARTBOARD
Copyright of SmartBoard © Team 2001, FSKTM, UM, KL, Malaysia
Contact webmaster click here.
Menu ready for use
Internet
Start Document1 - Mi... untitled - Paint Your smart w...
10:25 AM

*Figure a : Accessing the Smart***Board** *system*

Step 2 : *Login to the SmartBoard system*

To login to the system, Johnny inserts his username and correct password at the Login Page.


Login to smartboard - Microsoft Internet Explorer

Address http://202.185.109.102:88/smartboard/loginScreen.jsp?ok=N

Login To SmartBoard

Home | Member Area | Guide Tour | Terms & Conditions | Privacy Rights | About Us | Helps | Friday , January 18, 2002

Member ID: johnny

Password:

Submit | Reset

I want to sign up! | I forgot my password!

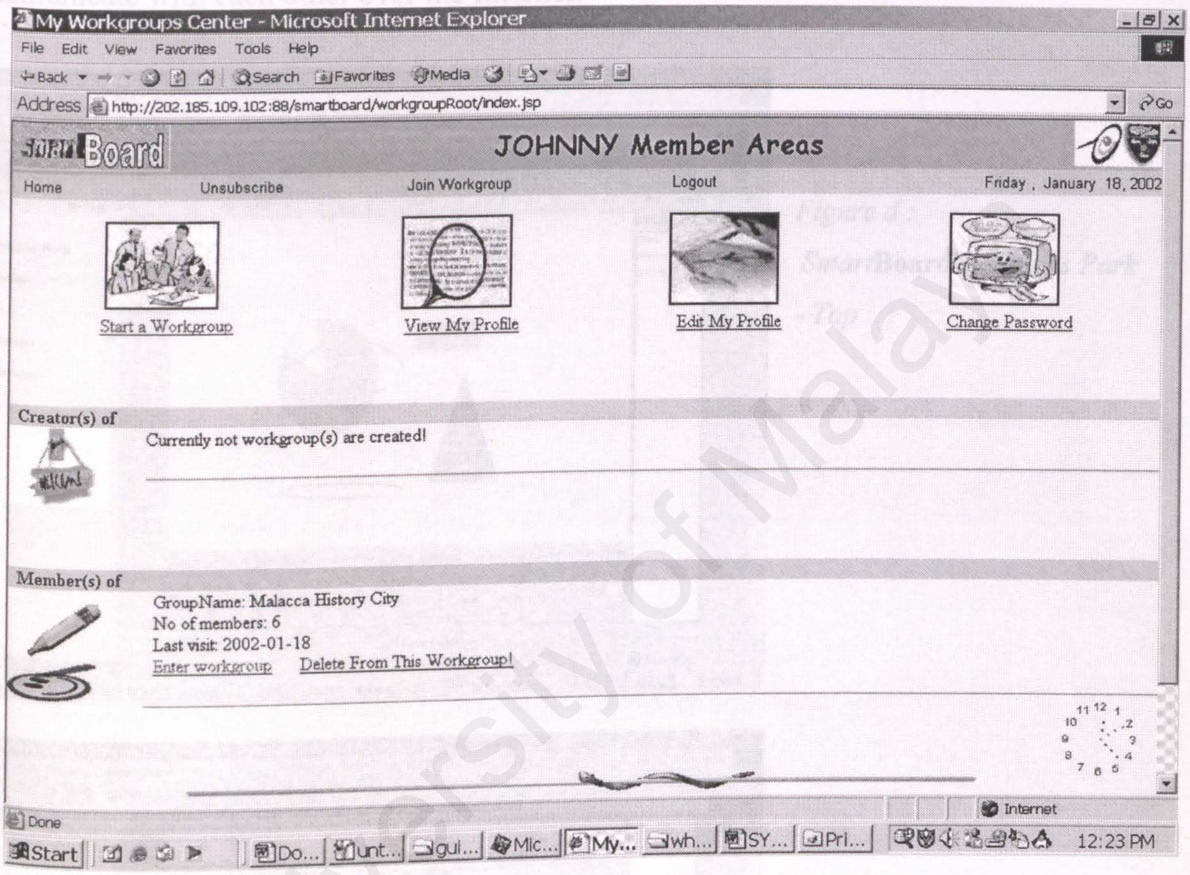
Copyright of SmartBoard © Team 2001, FSKTM, UM, KL, Malaysia

Contact webmaster click here.

*Figure b : Smart***Board** *Login Page*

Step 3: *Joining the workgroup*

After the correct login process, Johnny, successfully enters his member area. Here Johnny would like to begin his workgroup session. So, he will join his specific workgroup, called "Malacca History City".


My Workgroups Center - Microsoft Internet Explorer

Address: http://202.185.109.102:88/smartboard/workgroupRoot/index.jsp

JOHNNY Member Areas

Home | Unsubscribe | Join Workgroup | Logout | Friday , January 18, 2002

Start a Workgroup | View My Profile | Edit My Profile | Change Password

Creator(s) of

Currently not workgroup(s) are created!

Member(s) of

GroupName: Malacca History City
No of members: 6
Last visit: 2002-01-18
Enter workgroup   Delete From This Workgroup!

File Edit View Favorites Tools Help
Back Search Favorites Media
Go
Done
Internet
Start
12:23 PM

Figure c : Johnny's Member Area Window

Step 4 : *Welcome to Johnny's Activities Park*

Johnny enters his "Malacca Workgroup" Activities Park window, where he meet with his two other friends online, Diane and Michael. They could now draw on the whiteboard and communicate with each other over the Internet.

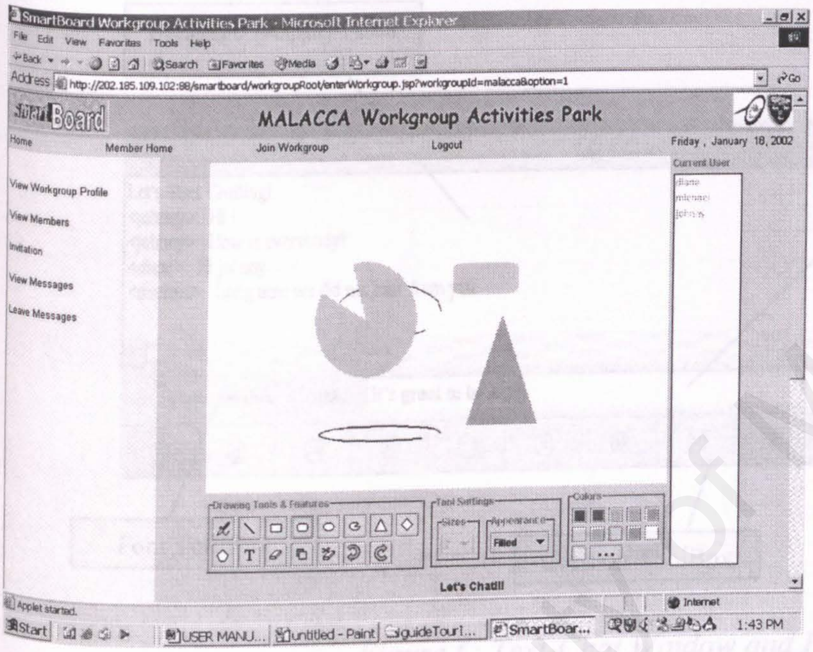
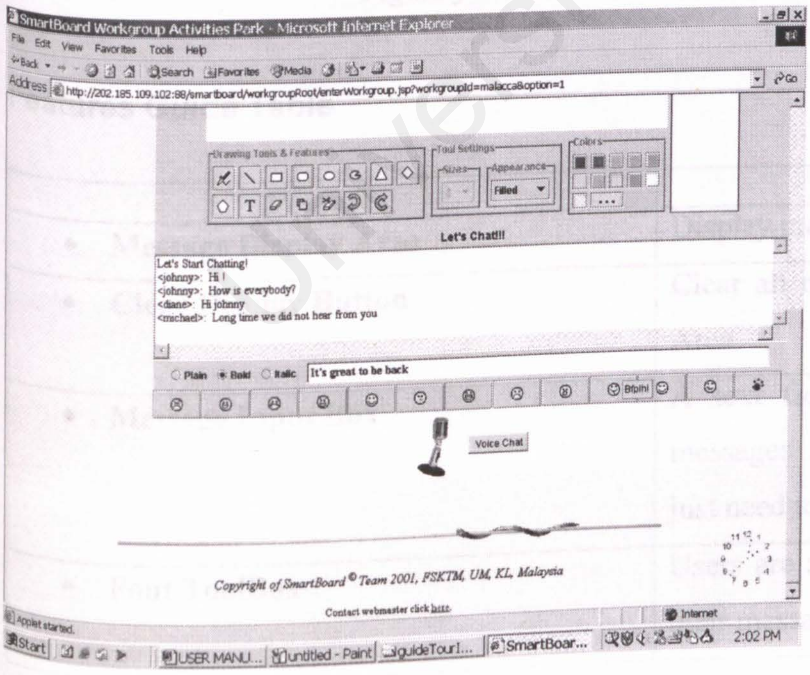


Figure d :

*Smart***Board** *Activities Park*

- Top


SmartBoard Workgroup Activities Park - Microsoft Internet Explorer
http://202.185.109.102:88/smartboard/workgroupRoot/enterWorkgroup.jsp?workgroupId=malacca&option=1
Drawing Tools & Features
Tool Settings
Appearance
Filled
Colors
Let's Chat!!
Plain
Bold
Italic
Voice Chat
Copyright of SmartBoard © Team 2001, FSKTM, UM, KL, Malaysia
Contact webmaster click here.
Applet started.
Internet
2:02 PM

Let's Start Chatting!
<johnny>: Hi !
<johnny>: How is everybody?
<diane>: Hi johnny
<michael>: Long time we did not hear from you

It's great to be back

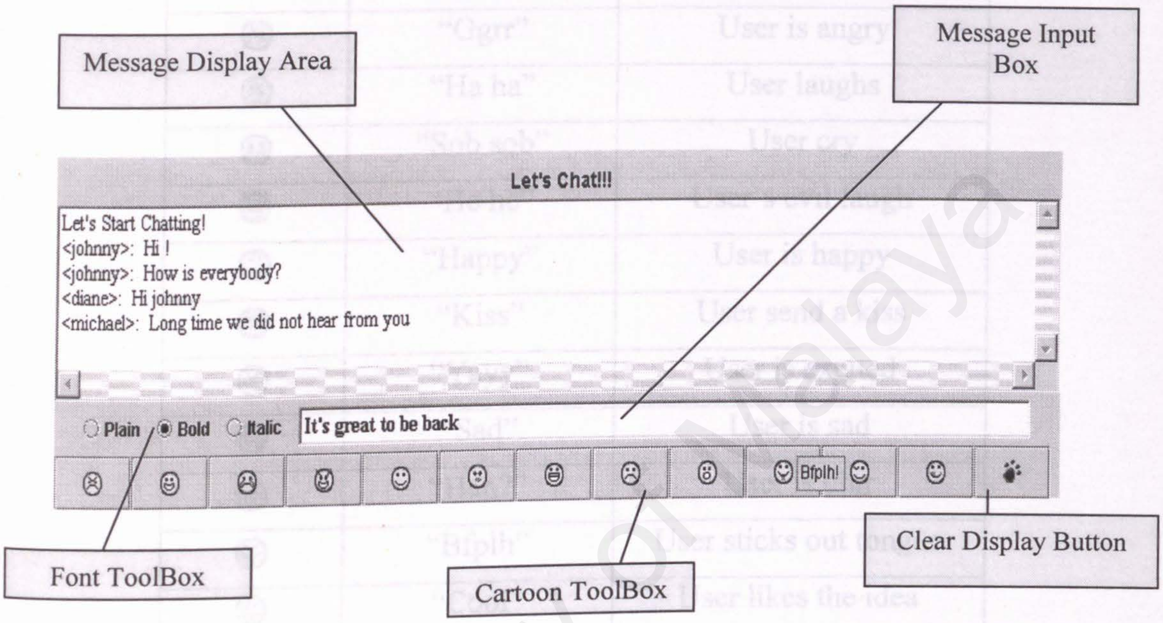
Figure e :

*Smart***Board** *Activities Park*

- Bottom

A.2 USING THE TEXT- CHAT

Figure f shows the text-chat window and its features. Here, we can see that Johnny, Michael and Diane are exchanging text messages with each other. To send a message Johnny must type his message at the Message Input Box and after that, press <enter> to send.



```
Let's Start Chatting!
<johnny>:  Hi !
<johnny>:  How is everybody?
<diane>:  Hi johnny
<michael>:  Long time we did not hear from you
```

Message input: It's great to be back

Figure f : Text-Chat Window and Function

Features Guide Table

Features	Functions
• **Message Display Area**	Display messages
• **Clear Display Button**	Clear all messages in the Message Display Area.
• **Message Input Box**	A text field for users to type and send messages. After typing their messages, users just need to press the <Enter> key to send.
• **Font ToolBox**	Users are able to select the type of font for their message characters.

- **Cartoon ToolBox**

The cartoon toolbox is a selection of cartoons that the user can select and choose to show their expression and feelings through cartoon animation. Below is the list of animation, expression and meaning for each of it.

Cartoon	Expression	Meaning
😠	"Ggrr"	User is angry
😆	"Ha ha"	User laughs
😢	"Sob sob"	User cry
😈	"He he"	User's evil laugh
🙂	"Happy"	User is happy
😗	"Kiss"	User send a kiss
😁	"Yayy"	User is excited
☹	"Sad"	User is sad
😲	"Hah?"	User is blur
😛	"Bfplh"	User sticks out tongue
😌	"Cool"	User likes the idea
😋	"Yum"	User is hungry

Table b : CartoonToolbox – Expression and Meaning

Now, Johnny is excited. He wants to express his excitement to Diane and Michael. So, he clicks 😁 button to say "Yayy!". When Johnny clicks it, a window box will appear asking Johnny to confirm whether to send or not. The cartoon will automatically be send once Johnny clicks "OK"

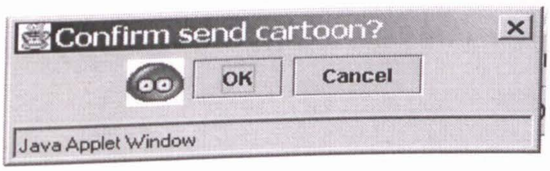
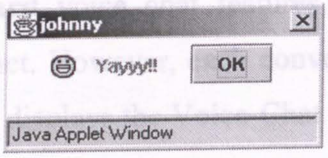


Figure g : Confirmation Box

Now, at the other side, Diane and Michael will receive a window box from Johnny, indicating Johnny's expression


johnny

Yayyy!!

OK

Java Applet Window

Figure h : Johnny's cartoon box

The following pages will cover on the usage of the voice-chat system

A.3 USING THE VOICE-CHAT

Through the *Smart***Board** voice chat features, Johnny, Diane and Michael are able to talk verbally over the Internet. However, each conversation can only involve not more and not less than two users. *Figure i* displays the Voice-Chat Panel.

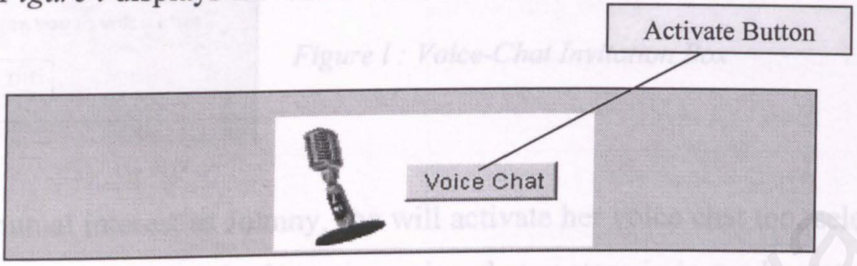


Figure i : Voice-Chat Panel

Selecting a user to talk

Now, Johnny is tired of typing messages in the text-chat. He thought of speaking privately to Diane. So, Johnny clicks the Activate Button. Now, for instance, if Diane and Michael or no one else is in the workgroup, it will be ridiculous if Johnny speaks to himself through the voice-chat. So, a "no user to talk" message will appear.

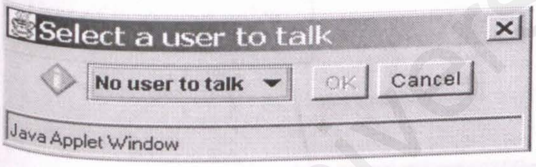


Figure j : "No user to talk" message

If Michael and Diane are in the *Smart***Board** Activities Park, Johnny is required to select one user to talk. Johnny selects Diane and clicks "OK". Johnny's voice chat system is configured and enters the ready state waiting for Diane's respond.

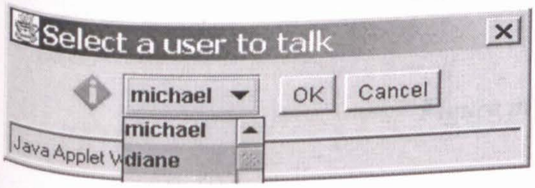
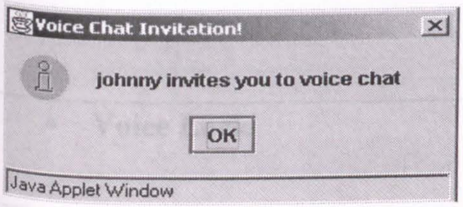


Figure k : Select-User Box

Invitation to voice chat

Automatically, an invitation to "voice-chat" from Johnny will be send to Diane in a window box below:


Voice Chat Invitation!

johnny invites you to voice chat

Figure l : Voice-Chat Invitation Box

If Diane has a mutual interest as Johnny, she will activate her voice chat too, selecting Johnny as her partner to voice-chat. Once both parties voice-chat system is in ready state, then the voice chat will be activated. Both can hear each other and speak at the same time, just like the operation of any common telephone.

Voice Analyzer ToolBox

The voice analyzer toolbox will then appear to give Johnny full control over the voice-chat session. Below is the diagram of the Voice Analyzer ToolBox

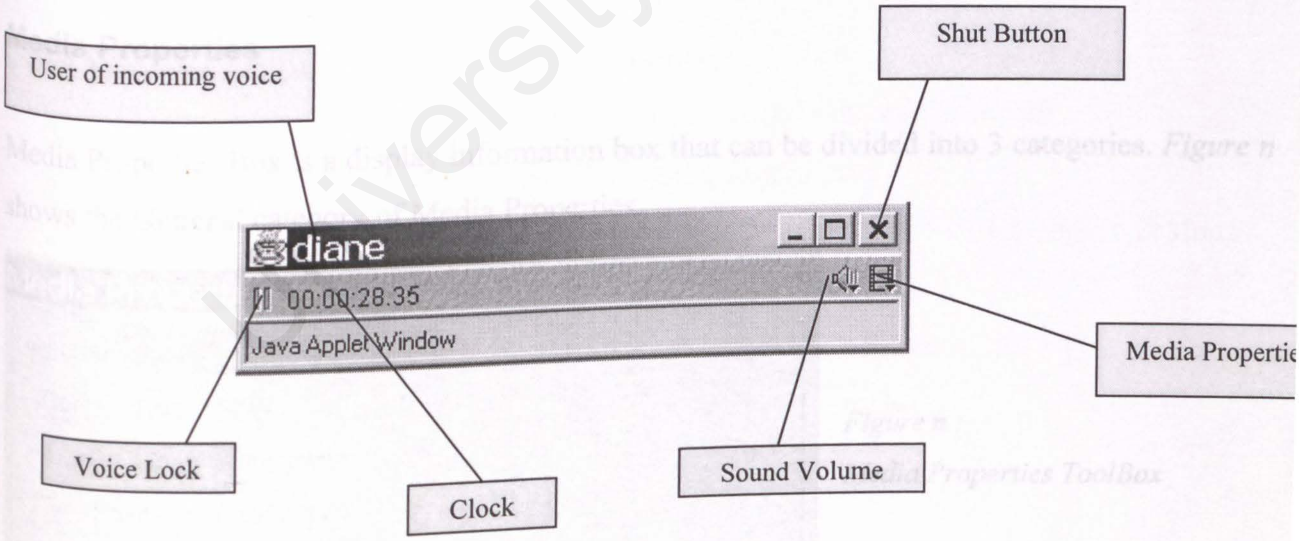


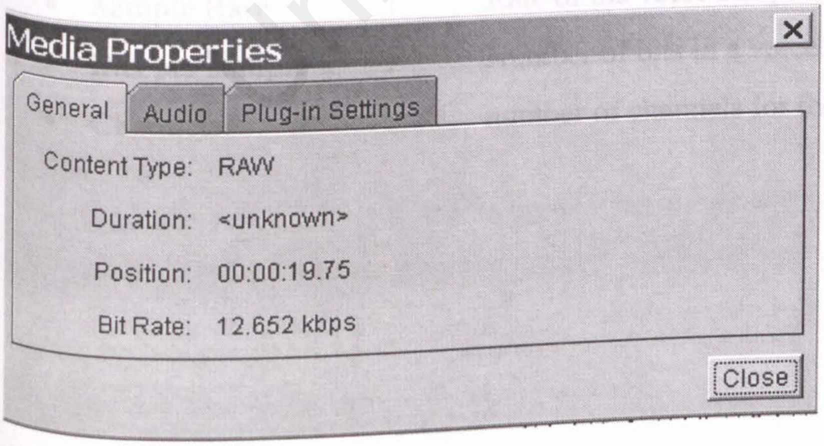
Figure m : Voice Analyzer ToolBox

Features Guide Table

Features	**Function**
• **User Name**	The title on the Voice Analyzer ToolBox indicates the user name of the incoming voice.
• **Voice Lock**	Locks or blocks the incoming voice. However, Johnny can still talk to Diane
• **Clock**	A digital clock is displayed to notify Johnny the duration of his conversation with Diane. It will pause when the Voice Lock is activated.
• **Shut Button**	Johnny clicks the Shut Button if he do not intend to continue the voice-chat session.
• **Sound Volume**	Johnny is able to control or adjust the volume of the incoming voice.

Media Properties

Media Properties Box is a display information box that can be divided into 3 categories. *Figure n* shows the **General** category of Media Properties.



Content Type: RAW

Duration: <unknown>

Position: 00:00:19.75

Bit Rate: 12.652 kbps

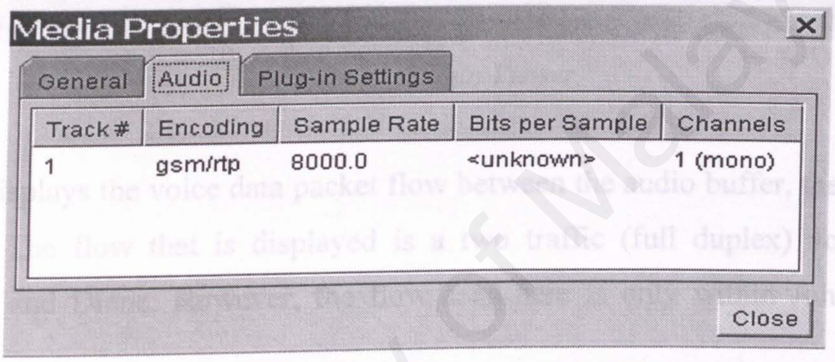
Figure n :

Media Properties ToolBox

Listed below are the items found in the **General** category:

• **Content Type**	:	The type of voice packet received
• **Duration**	:	Average duration of one packet transmission over the network
• **Position**	:	Current duration of the voice chat session
• **Bit Rate**	:	Transmission bit rate

Audio Category



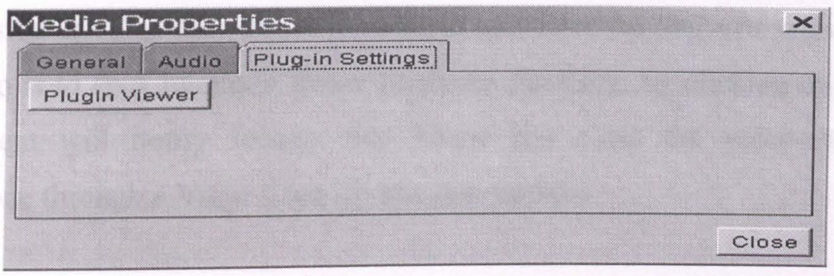
Track #	Encoding	Sample Rate	Bits per Sample	Channels
1	gsm/rtp	8000.0	<unknown>	1 (mono)

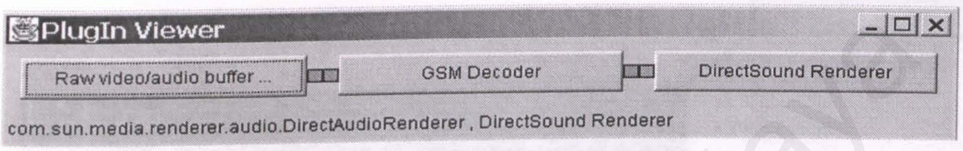
Figure o: Audio Category

Listed below are the items found in the Audio Category:

• **Tracks**	:	Number of sound track in the media stream
• **Encoding**	:	Type of encoding format used
• **Sample Rate**	:	Rate of the voice sample transmitted . Calculated in Hertz
• **Bits per sample**	:	Number of bits in a voice sample
• **Channels**	:	number of channels for the capture voice

Plug-in Settings Category





com.sun.media.renderer.audio.DirectAudioRenderer , DirectSound Renderer

Figure p : PlugIn Viewer

PlugIn Viewer displays the voice data packet flow between the audio buffer, the decoder and the sound renderer. The flow that is displayed is a two traffic (full duplex) voice transmission between Johnny and Diane. However, the flow seen here is only within Johnny's voice-chat system.

- **Audio Buffer** : Temporary memory storage to store the voice data.
- **Decoder** : A media processing unit that accepts voice data from the Buffer as its input, decodes the data into signals that can be transferred across the network, and then puts the result in an output Buffer object.
- **Renderer** : A media processing unit that renders input media to the system speakers. It has one input and no outputs--rendering is the final stage of the media-processing pipeline.

Closing the Voice Chat Session

After a long session of voice chat, Diane decided to quit from the *Smart***Board** application. What Diane need to do is to shut Johnny's Voice Analyzer ToolBox, by clicking on the Shut Button. The closing event will notify Johnny that Diane has close the voice-chat session. This notification is done through a Voice Chat Update pop-up box.

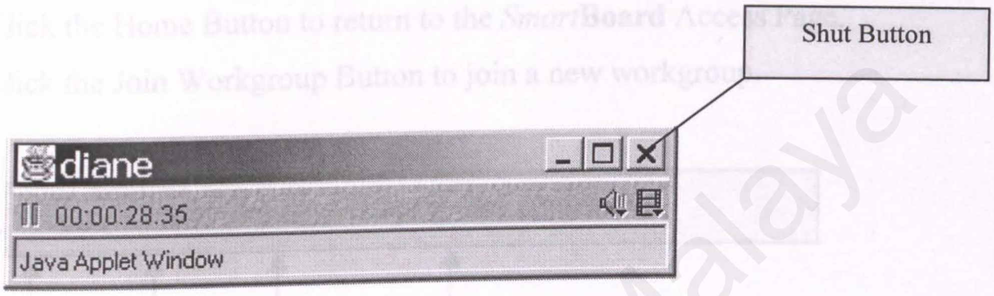
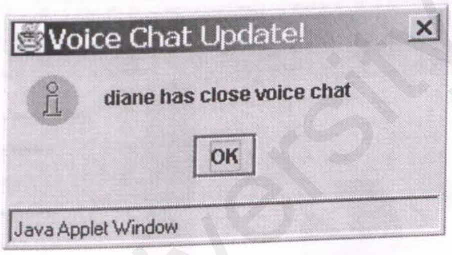


Figure q: Closing the voice chat session



diane has close voice chat

Figure r : Voice Chat Update pop-up box

A.4 Closing the SmartBoard Chat System

Soon, Johnny decided to quit the *Smart***Board** application. Johnny can do the following to log out from the Activities Park Window:

i. Click the Logout Button to logout as user "Johnny".

ii. Click the Back Button of the web browser.

iii. Click the Home Button to return to the *Smart***Board** Access Page.

iv. Click the Join Workgroup Button to join a new workgroup.

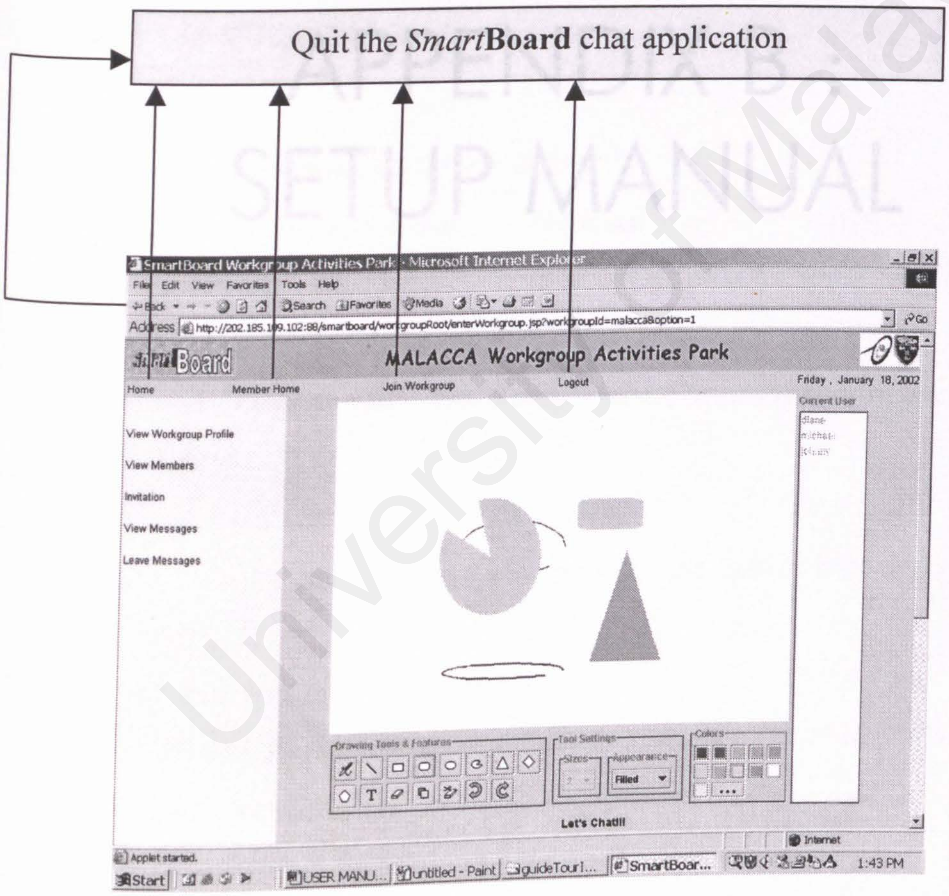

Quit the SmartBoard chat application
SmartBoard Workgroup Activities Park - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://202.185.109.102:88/smartboard/workgroupRoot/enterWorkgroup.jsp?workgroupId=malacca&option=1
MALACCA Workgroup Activities Park
Home
Member Home
Join Workgroup
Logout
Friday , January 18, 2002
Current User
View Workgroup Profile
View Members
Invitation
View Messages
Leave Messages
Drawing Tools & Features
Tool Settings
Sizes
Appearance
Filled
Colors
Let's Chat!!
Applet started.
Internet
Start
1:43 PM

Figure s : Closing the chat system

APPENDIX B : SETUP MANUAL

*Smart***Board's** Web-Based Application System

Copyright (C) 2001 *Smart***Board**, Inc.

All Rights Reserved.

INTRODUCTION

The setup manual is a guideline for setting up the *Smart***Board** system. The requirement of hardware and software may subsequently change based on the current technology and environment of where the setup takes place.

B.1 SYSTEM REQUIREMENTS

B.1.1 Server Hardware Requirements

The server requirements for *Smart***Board** system are

- A server with a least Intel Pentium 166MHz MMX CPU.
- At least 64MB RAM.
- Network Interface Card (NIC) and network connection with recommended bandwidth at 10Mbps.
- Keyboard and mouse as input devices.
- Other standard server peripherals. For example, 6.4GB hard disk.

B.1.2 Server Software Requirements

To host and run the system, the server needs to have various supporting software installed.

- Microsoft Windows NT Server 4.0 or higher
- Microsoft Internet Information Server 4.0 or higher
- Microsoft Access
- Any Web Server (For instance, JRun 3.1 Web Server).
- Java™ Development ToolKit Version 1.3.1
- Java™ 2 Runtime Environment, Standard Edition, v 1.3.1 (JRE).
- Java™ Media Framework Version 2.1.1

B.1.3 Client Hardware Requirements

The following hardware specifications for *Smart***Board** client run-time environment :

- IBM PC or any compatible personal computer.
- 64MB of free memory (128MB of free memory is recommended).
- 120MB free disk space.
- 56 Kbps Modem or 10/100 Mbps Network Card.
- A color monitor with a VGA card
- Soundcard (preferably supports full duplex)
- Microphone (preferably 16 bits) and speakers / Headset
- Keyboard and mouse as input devices.

B.1.4 Client Software Requirements

The following software specification for *Smart***Board** run time environment:

- Any web browsers

(recommended Internet Explorer 5.5 or Netscape Navigator 5.0).

- Any compatible operating system platform

(recommended Windows 98 and above).

B.2 DISTRIBUTION TERMS

*Smart***Board**'s system may be distributed freely in its original unmodified and unregistered form. The distribution has to include all files of its original distribution. The following is the view of overall *Smart***Board** system folder. From the CD distributed to you, copy the entire folder under *Smart***Board** folder into your Web server default directory.

```
SmartBoard (Main Folder)
  ↳allFlash (Flashing File)
  ↳allImages (Graphics File)
      ↳siteGraphics
      ↳guideTourImages
      ↳Sponsor
  ↳appletClasses (JavaScript and Applet File)
  ↳audioTextChat (Audio and Text Chat classes file)
      ↳icon
      ↳voiceChat
          ↳VoiceServer
          ↳graphic
  ↳java (SmartBoard Java Source Code)
  ↳tourGuide (Java Server Page file of guide tour)
  ↳helpDesk (Java Server Page file of help desk)
  ↳whiteboard (Whiteboard Classes File)
      ↳image
  ↳workgroupRoot (Java Server Pages Files For Workgroup Activities)
  ↳webmaster (Java Server Pages Files For Webmaster Activities)
  ↳messageboard (Java Server Pages Files For Posting Messages)
```

Under the classes\my\edu\um\fsktm\siswazah\smartboard\ and classes folder includes below files.

- smart.properties File
- Java Bean Classes Files

B.3 TOOLS INSTALLATION

B.3.1 INSTALLATION JAVA^{TM} 2 SDK, STANDARD EDITION, V 1.3.1

The Java™ 2 Platform, Standard Edition (J2SE™) has revolutionized computing with the introduction of a stable, secure and feature-complete development and deployment environment designed from the ground up for the Web. It provides cross-platform compatibility, safe network delivery, and smartcard to supercomputer scalability. It provides software developers with a platform for rapid application development, making it possible to deliver products to market in Internet time. It defies traditional software development and deployment models by delivering on the promise of cross-platform compatibility. It comes with three different versions to provide support to three different versions of operating systems:

- Solaris SPARC/x86
- Linux x86
- Microsoft Windows

Firstly, users need to download the *Java^{TM} 2 SDK, Standard Edition, v 1.3.1* installer from http://java.sun.com/j2se/1.3/. Once the installer is downloaded, double-click the installer to install the *Java^{TM} 2 SDK, Standard Edition, v 1.3.1.*

Users can also refer to the following URLs for further information regarding the installation processes.

- Solaris SPARC/x86
  - http://java.sun.com/j2se/1.3/install-solaris.html
- Linux x86
  - http://java.sun.com/j2se/1.3/install-linux-sdk.html
- Microsoft Windows
  - http://java.sun.com/j2se/1.3/download-windows.html

B.3.3 INSTALLATION JAVA^{TM} MEDIA FRAMEWORK STANDARD EDITION, V 2.1.1

JMF 2.1.1 is the latest release of the Java™ Media Framework API reference implementation. This new version is build using the same source code as in the SCSL release. Please use this latest version for the *Smart***Board** applications; It provides cross-platform compatibility and safe network delivery. There are three different types of download format that a user can choose depending on the users system.

They are the

i. Cross-platform Java

ii. Windows Performance Pack

iii. Solaris SPARC Performance Pack

The total size of the download file is only 2MB and for a common network, the downloading period takes not more than 5 minutes. Please refer to the download page of the java.sun website at this URL : http://java.sun.com/products/java-media/jmf/2.1.1/download.html

B.4 SETTING UP THE CONNECTION TO CHAT SERVER

Just as the above, implementation of the *Smart***Board** text-chat system and the voice-chat system involves the setup process of the server. The IP address and port number need to be redefined to match with the administrator's network. Changes in the client-side classes are unnecessary. An administrator just need to follow these steps :

1. Access to the applet code file

<default-app>/smartboard directory>/workgroupRoot/enterWorkgroup.jsp

2. Go to Line 147 to change the text-chat settings, go to Line 158 to change the voice-chat settings.

3. Change the port number and the IP address depending on the administrator's network

4. Restart the text-chat server and the voice-chat server

B.6 STARTING THE SERVER

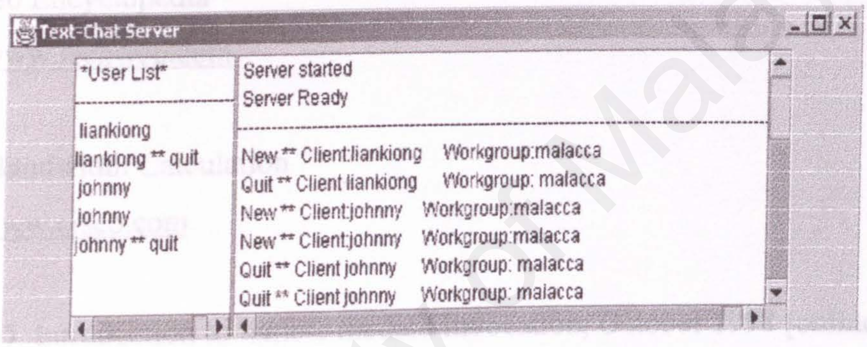
The text-chat server and the voice-chat server need to be started before any client makes any connection to use the system.

To start using **text chat** system, administrator need start the text chat server.

- To start the server goes to below folder.

  C:\Program Files\Allaire\JRun\servers\default\default-app\smartboard\audioTextChat
- Double click on the textChatServer.bat file.

Figure B.1 shows the graphical user interface of the text-chat server.



```
*User List*          Server started
                     Server Ready
liankiong            ----------------------------------------
liankiong ** quit    New ** Client:liankiong   Workgroup:malacca
johnny               Quit ** Client liankiong  Workgroup: malacca
johnny               New ** Client:johnny      Workgroup:malacca
johnny ** quit       New ** Client:johnny      Workgroup:malacca
                     Quit ** Client johnny     Workgroup: malacca
                     Quit ** Client johnny     Workgroup: malacca
```

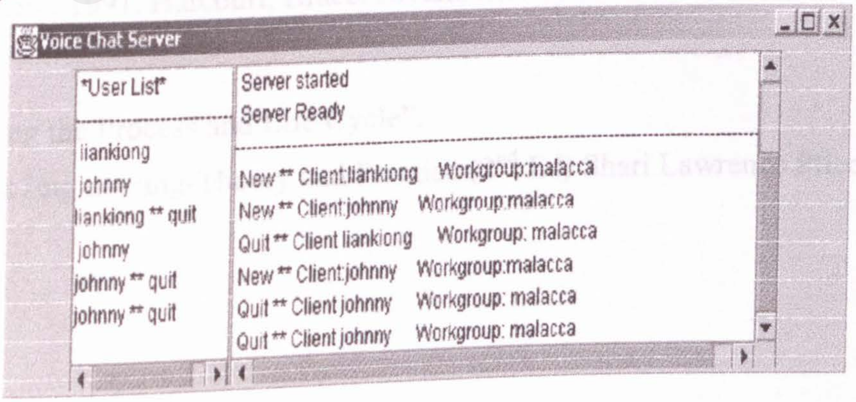
To start using **voice chat** system, administrator need to start the voice chat server.

- To start the server goes to below folder

C:\Program\Files\Allaire\JRun\servers\default\default-app \smartboard\audioTextChat\voiceChat\VoiceServer

- Double click on the voiceChatServer.bat file.

Figure B.2 shows the graphical user interface of the voice-chat server.



```
*User List*          Server started
                     Server Ready
liankiong            ----------------------------------------
johnny               New ** Client:liankiong   Workgroup:malacca
liankiong ** quit    New ** Client:johnny      Workgroup:malacca
johnny               Quit ** Client liankiong  Workgroup: malacca
johnny ** quit       New ** Client:johnny      Workgroup:malacca
johnny ** quit       Quit ** Client johnny     Workgroup: malacca
                     Quit ** Client johnny     Workgroup: malacca
```

REFERENCE

[1] Internet Growth Statistics
http://www.isoc.org/

[2] Java Sun Microsystem
http://www.java.sun.com

[3] Techweb Encyclopedia
http://www.techweb.com

[4] Voice Bandwidth Calculation
http://www.cisco.com

[5] Reilly, D. Introduction to remote method invocation, October 1998 [online] at
http://www.davidreilly.com/jcb/articles/javarmi/javarmi.html

[6] Morgan, B. CORBA meets Java, JavaWorld October 1998 [online] at
http://www.javaworld.com/jw-10-1997/jw-10-corbajava.html

[7] Greenberg, Saul (editor). Preface. *Computer Supported Cooperative Work and Groupware*. 1991. Harcourt, Brace, Jovanovich. ISBN 0-12-29920-2

[8] "Modeling the Process and Life Cycle",
Software Engineering-Theory and Practice (2nd Ed) Shari Lawrence Pfleeger, 2001