



# NetSpeak: The integration of network communication mediums and open source speech synthesis.

**Ciaran Reidy**

School of Computing,

Dublin Institute of Technology,

Kevin Street, Dublin 8, Ireland.

A Dissertation Submitted in Partial Fulfilment of the Requirements of Dublin Institute of Technology for the Degree of M.Sc. in Computing Science (Information Technology)

September 2008

# Table of Contents

<b>1. INTRODUCTION.....</b>	<b>11</b>
1.1 BACKGROUND.....	11
1.2 PROJECT DESCRIPTION.....	11
1.2.1 <i>Speech Synthesis</i> .....	12
1.2.2 <i>Speech Synthesis Characteristics</i> .....	13
1.2.3 <i>Previous Work</i> .....	14
1.3 PREVIOUS WORK .....	15
1.4 INTELLECTUAL CHALLENGE .....	15
1.4.1 <i>Speech Synthesis</i> .....	15
1.5 RESEARCH METHODOLOGY .....	16
1.5.1 <i>Speech Synthesis</i> .....	17
1.6 PESTLE ANALYSIS.....	18
1.7 THESIS ROADMAP .....	18
<b>2. LITERATURE REVIEW .....</b>	<b>21</b>
2.1 INTRODUCTION .....	21
2.2 ACCESSIBILITY IN THE SOCIAL CONTEXT.....	21
2.2.1 <i>Implications of Inaccessible Technology</i> .....	22
2.2.2 <i>Legislation &amp; Guidelines</i> .....	23
2.2.2.1 Irish Legislation.....	23
2.2.2.2 U.S. Legislation .....	24
2.2.2.3 Accessibility Guidelines .....	25
2.2.2.4 Case Law .....	26
2.2.3 <i>Assistive Technologies</i> .....	27
2.2.3.1 Screen Readers .....	27
2.2.3.2 Screen Magnifiers.....	28
2.3 INSTANT MESSAGING & CHAT APPLICATIONS.....	28
2.3.1 <i>History of Instant Messaging</i> .....	29
2.3.2 <i>Instant Messaging Usability</i> .....	29
2.3.3 <i>Instant Messaging Accessibility</i> .....	30
2.4 OPEN SOURCE SOFTWARE.....	31
2.4.1 <i>Introduction</i> .....	31
2.4.2 <i>History of OSS</i> .....	32
2.4.3 <i>Closed Source Software</i> .....	34
2.4.4 <i>Open Source Software Development Methodology</i> .....	34
2.4.5 <i>Open Source Software Quality &amp; Performance</i> .....	35
2.4.5.1 Full Disclosure V. Security through Obscurity.....	36
2.4.5.2 Drawbacks of Open Source Software.....	37
2.4.6 <i>Open Source Software Success</i> .....	37
2.5 CONCLUSIONS.....	38
<b>3. DESIGN OF NETSPEAK.....</b>	<b>40</b>
3.1 INTRODUCTION .....	40
3.2 REQUIREMENTS .....	40
3.3 ANALYSIS .....	41
3.3.1 <i>Unified Modeling Language</i> .....	42
3.3.2 <i>Requirements Analysis</i> .....	42
3.3.3 <i>Use Case Modeling</i> .....	42
3.3.3.1 Initial Use Cases .....	43
3.4 DOMAIN ANALYSIS.....	46
3.4.1 <i>Class Diagrams</i> .....	47

3.4.1.1 Initial Class Diagram .....	47
3.4.1.2 Detailed Class Diagram .....	48
3.4.2 State Diagrams .....	48
3.4.2.1 Initial State Diagram.....	49
3.4.2.2 Detailed State Diagram.....	49
3.4.3 Sequence Diagram.....	50
3.4.4 Activity Diagrams.....	52
3.5 DESIGN .....	53
3.5.1 Architectural Design.....	53
3.5.2 Detailed Design.....	54
3.5.3 User-Interface Design .....	57
3.5.4 Prototype Screenshots of System Interface.....	58
3.5.4.1 Login Screen.....	58
3.5.4.2 Main Chat Area Screen.....	59
3.5.4.3 System Administrator Screen .....	59
3.6 JUSTIFICATION FOR TOOLS AND TECHNOLOGIES .....	60
3.6.1 TCP / IP .....	60
3.6.2 Java.....	60
3.6.3 MySQL.....	60
3.7 CONCLUSIONS .....	61
<b>4. DEVELOPMENT.....</b>	<b>62</b>
4.1 INTRODUCTION .....	62
4.2 METHODOLOGY .....	63
4.3 DEVELOPMENT REQUIREMENTS.....	65
4.4 DEVELOPMENT PROCESS.....	66
4.4.1 n-tier Architecture .....	66
4.4.2 Speech Technology .....	67
4.4.3 Justification for Tools and Technology.....	69
4.4.4 Environment Configuration.....	70
4.4.4.1 Java.....	71
4.4.4.2 MySQL .....	71
4.4.4.3 FreeTTS.....	71
4.4.4.4 Environment Settings.....	72
4.5 PHASES IN DEVELOPMENT ENVIRONMENT.....	74
4.5.1 Classes and Methods .....	74
4.5.1.1 ChatServer .....	76
4.5.1.3 ChatHandler.....	79
4.5.1.4 ChatClient.....	80
4.5.1.5 LoginScreen.....	81
4.5.1.6 ChatScreen.....	84
4.5.1.7 UserListServer .....	87
4.5.1.8 UserListClient.....	88
4.5.1.9 SpeakSynthesizerAPI .....	89
4.6 INTERFACE SCREEN SHOTS .....	91
4.6.1 User Interface Guidelines.....	91
4.6.2 Evolution of Screenshots.....	95
4.7 ADDITIONAL FUNCTIONALITY .....	97
4.7.1 Reduce Cognitive Load.....	97
4.7.2 Additional Access Keys.....	100
4.7.2.1 Logout of Chat Application .....	101
4.7.2.2 Saving a Conversation .....	102
4.7.2.3 Synthesizing list of Logged in Users .....	102
4.7.3 Acronyms interpreted.....	103
4.8 ERROR DETECTION AND EXCEPTION HANDLING.....	105

4.8.1 Examples.....	106
4.8.1.1 ArrayIndexOutOfBoundsException .....	106
4.8.1.2 NumberFormatException .....	107
4.9 CONCLUSIONS .....	109
<b>5. TESTING AND EVALUATION.....</b>	<b>110</b>
5.1 INTRODUCTION .....	110
5.2 TESTING.....	110
5.2.1 Development of Test Plan.....	110
5.2.2 Test Plan Features.....	111
5.2.3 Schedules and Resources.....	111
5.2.4 Testing Approach.....	112
5.2.5 Features to be tested.....	112
5.2.6 Environmental Needs.....	113
5.2.7 Item Pass/Fail Criteria .....	113
5.2.8 Test Deliverables .....	114
5.3 TYPES OF TESTING .....	114
5.3.1 Unit Testing .....	114
5.3.2 Integration Testing .....	115
5.3.3 Stress Testing.....	116
5.3.4 System Testing .....	116
5.3.5 Acceptance Testing.....	116
5.4 SOFTWARE EVALUATION .....	117
5.4.1 Introduction .....	117
5.4.2 Software Evaluation Process.....	118
5.4.3 Rubric Provided.....	118
5.4.4 Evaluator Feedback.....	119
5.5 CONCLUSION.....	122
<b>6. CONCLUSIONS.....</b>	<b>123</b>
6.1 INTRODUCTION .....	123
6.2 KEY FINDINGS .....	123
6.2.1 Review of Literature .....	123
6.2.2 Review of design .....	124
6.2.3 Review of Development.....	124
6.2.4 Review of Testing and Evaluation .....	125
6.3 PROBLEMS ENCOUNTERED.....	125
6.4 CONCLUSIONS.....	126
6.5 FUTURE WORK .....	126
6.5.1 Speech Synthesis .....	126
6.5.2 Speech Recognition.....	126
6.5.2.1 Previous Work .....	127
6.5.2.2 Performance Issues .....	127
6.5.2.3 Trained Data Concerns .....	128
6.5.2.4 Potential Speech Recognizer .....	128
<b>APPENDIX A: .....</b>	<b>130</b>
<b>APPENDIX B: .....</b>	<b>133</b>
<b>APPENDIX C: .....</b>	<b>134</b>
<b>APPENDIX D: .....</b>	<b>136</b>
<b>BIBLIOGRAPHY .....</b>	<b>145</b>

## Table of Figures

FIGURE 1.1 CONCEPTUAL EXAMPLE OF SYSTEM.....	12
FIGURE 1.2 THESIS OUTLINE.....	19
FIGURE 2.1 JAWS BASIC DIALOGUE BOX .....	28
FIGURE 2.2 ZOOMTEXT .....	28
FIGURE 2.3 CLASSIC WATERFALL DEVELOPMENT MODEL TYPICALLY FOUND IN PROPRIETARY SYSTEMS (NEW ROWLEY TECHVIEW).....	35
FIGURE 2.4 MARKET SHARE FOR TOP SERVERS ACROSS ALL DOMAINS AUGUST 1995 - JUNE 2008 (NETCRAFT).....	38
FIGURE 3.1 AUTHENTICATED USER USE CASE DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	44
FIGURE 3.2 ANONYMOUS USER USE CASE DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	45
FIGURE 3.3 SYSTEM ADMINISTRATOR USE CASE DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM].....	46
FIGURE 3.4 AUTHENTICATED USER USE CASE DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	46
FIGURE 3.5 INITIAL CLASS DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	47
FIGURE 3.6 DETAILED CLASS DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	48
FIGURE 3.7 INITIAL STATE DIAGRAM FOR CLASS USER [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	49
FIGURE 3.8 DETAILED STATE DIAGRAM FOR CLASS USER [SOURCE: AUTHOR USING VISUAL PARADIGM].....	50
FIGURE 3.9 SEQUENCE DIAGRAM FOR USE CASE LOG IN [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	51
FIGURE 3.10 SEQUENCE DIAGRAM FOR USE CASE SEND MESSAGE [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	51
FIGURE 3.11 ACTIVITY DIAGRAM FOR USER [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	52
FIGURE 3.12 ACTIVITY DIAGRAM FOR A SYSTEM ADMINISTRATOR [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	52
FIGURE 3.13 ARCHITECTURAL CLASS DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	54
FIGURE 3.14 STATE DIAGRAM VECTOR USERLIST [SOURCE: AUTHOR USING VISUAL PARADIGM] .....	55
FIGURE 3.15 DETAILED SEQUENCE DIAGRAM FOR USE CASE SEND MESSAGE [SOURCE: AUTHOR USING VISUAL PARADIGM].....	56

<b>FIGURE 3.16 DETAILED COLLABORATION DIAGRAM FOR USE CASE SEND MESSAGE [SOURCE: AUTHOR USING VISUAL PARADIGM].</b>	<b>57</b>
<b>FIGURE 3.17 INITIAL LOGIN SCREEN</b>	<b>58</b>
<b>FIGURE 3.18 INITIAL MAIN CHAT SCREEN</b>	<b>59</b>
<b>FIGURE 3.19 INITIAL ADMIN SCREEN</b>	<b>59</b>
<b>FIGURE 3.20 INITIAL ADMIN SCREEN</b>	<b>60</b>
<b>FIGURE 4.1 COMPONENT DIAGRAM FOR CHAT SYSTEM [SOURCE: AUTHOR USING VISUAL PARADIGM]</b>	<b>63</b>
<b>FIGURE 4.2 CRYSTAL FAMILY METHODOLOGIES (A PRACTICAL GUIDE TO SEVEN AGILE METHODOLOGIES, PART 2, WWW.DEVX.COM)</b>	<b>65</b>
<b>FIGURE 4.3 N-TIER ARCHITECTURE [SOURCE: AUTHOR USING VISUAL PARADIGM]</b>	<b>67</b>
<b>FIGURE 4.4 SPEECH ENABLED TECHNOLOGY</b>	<b>68</b>
<b>FIGURE 4.5 SPEECH ENGINE [SOURCE: AUTHOR USING VISUAL PARADIGM]</b>	<b>68</b>
<b>FIGURE 4.6 JAVA SPEECH STACK [SOURCE: AUTHOR USING VISUAL PARADIGM]</b>	<b>69</b>
<b>FIGURE 4.7 INSTALLING MYSQL 5.0</b>	<b>71</b>
<b>FIGURE 4.8 DOWNLOADING FREETTS 1.2.1-BIN.ZIP</b>	<b>72</b>
<b>FIGURE 4.9 DOWNLOADING FREETTS 1.2.1-BIN.ZIP</b>	<b>72</b>
<b>FIGURE 4.10 CONFIGURING THE CLASSPATH</b>	<b>73</b>
<b>FIGURE 4.11 CLASS DIAGRAM FROM CHAPTER 3 [SOURCE: AUTHOR USING VISUAL PARADIGM]</b>	<b>74</b>
<b>FIGURE 4.12 DETAILED DEVELOPMENT CLASS DIAGRAM [SOURCE: AUTHOR USING VISUAL PARADIGM]</b>	<b>75</b>
<b>FIGURE 4.13 CHATSERVER ACCEPTING A CONNECTION</b>	<b>76</b>
<b>FIGURE 4.14 METHOD GETLOGINDETAILS</b>	<b>76</b>
<b>FIGURE 4.15 INVOKING DATABASEHELPER CLASS FROM CHATSERVER CLASS</b>	<b>77</b>
<b>FIGURE 4.16 SERVER'S SUBMITRESPONSE METHOD SUBMITS SUCCESS / FAILED RESPONSE TO CLIENT</b>	<b>77</b>
<b>FIGURE 4.17 CHATSERVER SENDS PERSONALIZED IMAGE TO CLIENT AND CREATES AN INSTANCE OF THE CHATHANDLER CLASS TO BE RUN</b>	<b>77</b>
<b>FIGURE 4.18 CONVERTING IMAGE TO SERIALIZABLE FORMAT FOR TCP TRANSMISSION</b>	<b>78</b>
<b>FIGURE 4.19 VALIDATING USER WITH THE DATABASE</b>	<b>78</b>
<b>FIGURE 4.20 SAVING A MESSAGE TO THE DATABASE</b>	<b>79</b>
<b>FIGURE 4.21 MULTI-THREADING ENVIRONMENT DEPICTING THE CHATHANDLER CLASS BROADCASTING A MESSAGE TO CLIENTS</b>	<b>80</b>
<b>FIGURE 4.22 CHATCLIENT THREAD READING INCOMING MESSAGE FROM THE CHATHANDLER CLASS</b>	<b>81</b>
<b>FIGURE 4.23 ACTIONLISTENER'S ACTIONPERFORMED METHOD IMPLEMENTATION IN THE LOGINSCREEN CLASS</b>	<b>82</b>

FIGURE 4.24 ACTIONLISTENER'S ACTIONPERFORMED EVENT REGISTERED WITH 'LOGIN' COMPONENT.....	82
FIGURE 4.25 KEYLISTENER CODE EXECUTED ONCE THE 'ENTER' KEY IS PRESSED .....	82
FIGURE 4.26 LOGINSCREEN SUBMITS USER LOGIN DETAILS TO SERVER ON PORT 9200.....	83
FIGURE 4.27 POP UP PROMPT PROMPTING THE USER TO TRY LOGGING IN AGAIN	83
FIGURE 4.28 USER SUCCESSFULLY LOGS IN AND IS PRESENTED WITH MAIN CHAT SCREEN.....	84
FIGURE 4.29 MESSAGE SUBMITTED TO THE LOBBY .....	85
FIGURE 4.30 CHARACTER / WORD TYPED SYNTHESIZED BEFORE BEING SUBMITTED TO THE LOBBY .....	85
FIGURE 4.31 CREATESLIDER METHOD CALLED TO INITIALIZE PITCHSLIDER.....	85
FIGURE 4.32 SLIDER CREATED TO CONTROL THE SYNTHESIZER SETTINGS .....	86
FIGURE 4.33 ACTIONLISTENER USED TO SET THE PITCH SLIDER HIGHER USING ACCESS KEYS ('ALT' AND 'P') .....	87
FIGURE 4.34 USERLISTSERVER THREAD CLASS CREATED AND STARTED .....	88
FIGURE 4.35 CODE USED TO SEND LIST OF USERS LOGGED IN TO THE CLIENTS ONCE EVERY SECOND.....	88
FIGURE 4.36 CODE USED TO RECEIVE LIST OF USERS LOGGED IN FROM THE USERLISTSERVER CLASS ONCE EVERY SECOND .....	89
FIGURE 4.37 SPEAKSYNTHESIZERAPI CONSTRUCTOR .....	90
FIGURE 4.38 ACCESS KEYS 'ALT' AND 'L' TO LOGOUT OF THE SYSTEM.....	92
FIGURE 4.39 ACCESS KEYS 'SHIFT', 'ALT' AND 'S' TO SAVE A CONVERSATION .....	93
FIGURE 4.40 USE CASE FROM SECTION 3.3.3.1 .....	95
FIGURE 4.41 PROTOTYPE SCREEN SHOTS FROM SECTION 3.5.4 .....	96
FIGURE 4.42 LOGIN SCREEN APPLICATION SCREENSHOT .....	96
FIGURE 4.43 MAIN CHAT LOBBY APPLICATION SCREENSHOT .....	97
FIGURE 4.44 RECALL FUNCTION.....	98
FIGURE 4.45 HASH TABLE KEY (USERNAME) - VALUE (MESSAGE) PAIRS. [SOURCE: AUTHOR USING MS POWERPOINT].....	98
FIGURE 4.46 HASH TABLE KEY (USERNAME) - VALUE (VECTOR OF MESSAGES) PAIRS. [SOURCE: AUTHOR USING MS POWERPOINT].....	99
FIGURE 4.47 CODE TO DEMONSTRATE THE HANDLING OF SYNTHESIZING PREVIOUS MESSAGES.....	99
FIGURE 4.48 CODE TO RETRIEVE THE PREVIOUS MESSAGE .....	100
FIGURE 4.49 LOGGING OUT OF SYSTEM VIA ACCESS KEYS.....	101
FIGURE 4.50 SAVING CONVERSATION VIA MOUSE.....	102
FIGURE 4.51 SAVING CONVERSATION VIA ACCESS KEYS.....	102
FIGURE 4.52 SYNTHESIZING LOGGED IN USERS VIA ACCESS KEYS.....	103

<b>FIGURE 4.53 ADD ACRONYM GUI.....</b>	<b>104</b>
<b>FIGURE 4.54 CODE TO ADD ACRONYM .....</b>	<b>104</b>
<b>FIGURE 4.55 SYNTHESIZES MEANING OF ACRONYM .....</b>	<b>105</b>
<b>FIGURE 4.56 EXAMPLE OF ERROR HANDLING .....</b>	<b>106</b>
<b>FIGURE 4.57 INTERFACE COMMAND TO GENERATE ARRAY INDEX OUT OF BOUNDS EXCEPTION .....</b>	<b>106</b>
<b>FIGURE 4.58 CONSOLE ERROR - ARRAY INDEX OUT OF BOUNDS EXCEPTION .....</b>	<b>107</b>
<b>FIGURE 4.59 EXCEPTION HANDLING CODE - ARRAY INDEX OUT OF BOUNDS EXCEPTION .....</b>	<b>107</b>
<b>FIGURE 4.60 USER FRIENDLY ERROR - ARRAY INDEX OUT OF BOUNDS EXCEPTION .....</b>	<b>107</b>
<b>FIGURE 4.61 INTERFACE COMMAND TO GENERATE NUMBER FORMAT EXCEPTION .....</b>	<b>108</b>
<b>FIGURE 4.62 CONSOLE ERROR - ARRAY NUMBER FORMAT EXCEPTION .....</b>	<b>108</b>
<b>FIGURE 4.63 EXCEPTION HANDLING CODE – NUMBER FORMAT EXCEPTION.....</b>	<b>109</b>
<b>FIGURE 4.64 USER FRIENDLY ERROR – NUMBER FORMAT EXCEPTION .....</b>	<b>109</b>
<b>FIGURE 5.1 INTEGRATION OF TESTING METHODS INTO DESIGN AND DEVELOPMENT OF SYSTEM [SOURCE: AUTHOR USING VISUAL PARADIGM] .....</b>	<b>114</b>
<b>FIGURE 5.2 CONNEXCEPTION (GENERATED BY DEVELOPER).....</b>	<b>115</b>
<b>FIGURE 5.3 SOCKETEXCEPTION (GENERATED BY DEVELOPER). ....</b>	<b>116</b>
<b>FIGURE 5.4 BINDEXCEPTION (GENERATED BY DEVELOPER).....</b>	<b>116</b>
<b>FIGURE 5.5 ISO 9126-1 QUALITY MODEL CHARACTERISTICS [SOURCE: AUTHOR USING VISUAL PARADIGM].....</b>	<b>117</b>
<b>FIGURE 5.6 INTERVIEWEES CODED.....</b>	<b>119</b>



## **Abstract**

This project combines multi-threading technology, speech synthesis, and open source software to create a user friendly real time communication tool for the visually impaired and the blind to use at ease. A network instant messenger was implemented in Java with speech synthesis. This allows visually impaired users to make use of this communication medium where other main-stream applications lack the functionality they require. Users communicate with the program using access keys and mouse alternatives as the primary input and all application processes are dictated to the user through synthesized speech, so that they can navigate the application with ease.

## **Acknowledgments**

First and foremost, I would like to express my sincere thanks to my supervisor Mr. Damian Gordon for the guidance he provided through out the project. Mr. Damian Gordon not only served as my supervisor but he also provided assistance and encouragement throughout the research programme. Mr Damian Gordon also had some great suggestions on how this project could be made more user friendly for the visually impaired and the blind.

I am grateful for all the help provided by the staff in DIT Kevin Street.

Finally, I would like to thank my family and my friends for their continued support over the past two years in DIT. In addition I would also like to thank everybody who painstakingly provided assistance, suggestions, and feedback regarding the project. Thank you.

# **1. Introduction**

## ***1.1 Background***

When visually-impaired users attempt to access a range of online resources, they often face software access and communication problems, this is particularly true of messaging software. The available communication tools are not completely user-friendly for the visually-impaired and blind, and therefore intrinsically create accessibility barriers for them.

Software application accessibility has become more of an issue over the last few of years particularly because of legislation in both the U.S. and Europe. Computer applications' existing features to current applications are constantly being enhanced. However, the unfortunate reality is that not all of these high tech new features have been developed with the disabled in mind.

Having access to software controls through mouse alternatives is essential for people who cannot accurately control a mouse or for users with a visual disability who do not use a mouse. Additionally there are a wide range of open source communication mediums available (Redmond, 2002), such as MSN Messenger.

To facilitate new innovation in messaging software with the visually impaired and blind in mind, this research will seek to develop an open source speech instant messenger application for the visually-impaired and blind to use with relative ease. As it will be an open-source application, it will be available to a large variety of development platforms. This work will integrate some of the latest instant messaging and chat application technology with speech synthesis tools with the aim of helping the visually impaired.

## ***1.2 Project Description***

Instant messaging and real-time communication among personnel is growing in importance and it is no longer seen as just a way of communicating across networks and the broader internet for personal means. There is an urgent need for a real time communications tool with speech enabled technologies in order to allow everyone benefit from such a great piece of technology. Instant Messaging (IM) is one of the fastest growing methods of communicating over the web today. The ability to

communicate in real time via instant messaging is important to many as it is used for socializing, education, and for corporate related agendas.

IM has gained acceptance from all generations and it allows people communicate in virtual spaces where communicating in physical places may prove difficult at times for the visually impaired and the blind. However, the ironic truth is that such technology poses there own accessibility barriers for the visually impaired. There is a need to allow universal access to IM technology to allow all communicate in real time over the web. A conceptual example of real time communication in a virtual environment can be seen in figure 1.1.

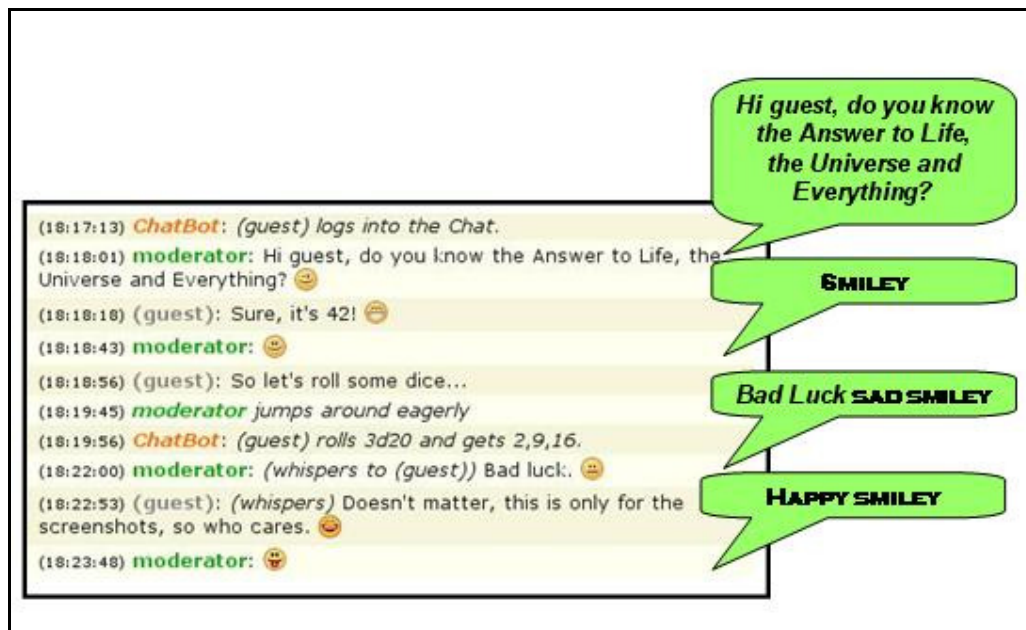


Figure 1.1 Conceptual Example of System

### 1.2.1 Speech Synthesis

Have you ever considered how an individual with no vision would launch a software application from the desktop? Similarly, how could one also close down applications or logout of a system? The introduction of Graphical User Interfaces' has made it far more difficult for visually-impaired and blind people to use computers. Many important operations now require the use of a mouse such as selecting and launching applications, copying and deleting files, and creating folders etc. However, successfully using a mouse requires continuous feedback about the position of the cursor, and this is difficult to achieve (Pitt & Edwards, 2003). It would be useful if there were a way for the computer to communicate the status of an application, or the actual keys that were typed through synthesized speech.

Until recently, synthetic speech was little more than a novelty to most people. It is only in the last few years, however, that synthetic speech has entered the mainstream, finding applications in a variety of areas from voice-mail systems to satellite navigation and traffic-avoidance systems (Pitt & Edwards, 2003). This move has occurred largely as a result of improvements in the technology, making synthetic speech more natural-sounding and hence more acceptable to a wide cross-section of potential users. (Karaali, 1996) states that speech synthesis equipment will only be accepted by the market if it has good speech quality, functionality, and low product costs.

Allen *et al.* (1987) presented the design of a Text-to-Speech synthesizer in some detail, and this has proven to be very influential. It ultimately led to the development of the *Dectalk synthesizer*, which in turn has been the basis of many subsequent synthesizers.

### **1.2.2 Speech Synthesis Characteristics**

Speech is known as a slow communication medium compared with visual displays. Barry Arons from the Speech Interface Group, pointed out, “*it is faster to speak than it is to write or type, however it is slower to listen than it is to read*” (Arons, 1993). And this quote by Arons is especially pertinent when one considers the impact this would have on the visually-impaired and blind. It will clearly be more complex and would therefore increase the cognitive load for the visually-impaired and blind user substantially but unfortunately, at times, there is just no other mechanism for communicating with the visually impaired and blind.

Thus the visually-impaired or blind user, who has no option but to access text through speech, is at a significant disadvantage compared with the sighted user who can rapidly skim through visually presented text. When developing any speech system, especially instant messaging systems where data is constantly flowing to and from users, we must accept the fact that the screen acts as a secondary memory for visual users. An experiment conducted by the US Air Force compared individual's ability to respond to visual alerts and synthetic speech. The results showed that the individual's responded more slowly to speech than those who received the visual warnings (Robinson and Eberts, 1987). Therefore one must look at ways to reiterate the synthesized speech if desired by the user and provide more feedback to the user of the system.

### 1.2.3 Previous Work

There exists a range of problems with current instant messaging applications such as Microsoft MSN Messenger. They are closed-source, meaning that the source code cannot be edited and reused to develop a new applications designed to alternative requirements. Similarly, there does exist text-to-speech (TTS) and speech recognition systems currently, but unfortunately they cannot be easily edited and integrated into existing software applications for end-user accessibility. Commercial technologies such as *NextUp Talker TTS* application is designed specifically for people who have temporarily or permanently lost their voice (NextUp, 2008).

These existing technologies are all excellent applications and certainly make life easier for the disabled. The problem is they are fixed and are standardized for all users. None of them can be tailored for particular user's desires. With open source and full disclosure of the software the application it is possible to implement a system that is portable to new hardware, the source code will be available and modifiable which will allow the unlimited tuning of the software product with the disabled in mind. There will be no dependence on software vendors and will be much easier to customize.

A chat application medium will be developed entirely in the Java programming language. This is the ideal technology to use as it is object-oriented; meaning the construction of interfaces permits the separation of the framework design from the actual implementation. Its modular nature means that it can accept add-ons at ease as the project matures. As it will be an open source application, it will be available to a large variety of development platforms. Java is also the ideal technology when one considers networking applications that involves multithreading. A separate thread will need to be created and run for every conversation.

A number of speech synthesizes applications have been developed for visually-impaired and blind people, for example, talking clocks and talking word processors. However, since they were designed from the beginning to use speech, the software tends to be specialized and therefore the market is relatively small, and in consequence such software is usually less well supported and less frequently updated than mainstream software (Pitt & Edwards, 2003). On the contrary, the speech synthesizer developed in this research will be of an open source nature. As the main instant messenger application will be developed in Java it is desirable that the speech technology used in this thesis has also been implemented in Java. This will ensure the smooth integration of the speech technology into the main application. Another benefit of developing an open source system is that it can be argued that in some cases *"the quality of software produced by the Open Source community sometimes exceeds that produced by purely commercial organizations"* (Peeling and Satchell, 2001).

### ***1.3 Previous Work***

The aim of this project is to research and implement a network instant voice messenger with speech enabled tools to allow a wider segment of users to access this system.

Objectives:-

- to research the ways of enabling existing instant messenger tools become more usable to visually impaired users.
- to gain a thorough understanding of the open source speech technologies on the market today and integrate these into an instant messenger system.
- to question the existing speech technologies performance and accuracy and raise appropriate questions to be answered by experts in the field. This will provide belief and drive direction in this research.
- contact experts in the field and engage at a Masters-level discourse with them.
- to gradually implement a new Speech Synthesis based network communications tool.

### ***1.4 Intellectual Challenge***

There will no doubt be numerous challenges in researching this topic up to and including the development of this project. My primary concern is with the speech tool in question as opposed to the networking involved in the project. This is mainly due to fact that speech technology is relatively new with regard to computing. In the next section I go into more detail about my concerns involving the synthesizer and where I will be in contact with the research groups directly involved with these technologies. The main challenge in this research project will be regarding speech software issues and technological constraints.

#### ***1.4.1 Speech Synthesis***

The effects of speech quality through synthesizers will be of a major concern. As the synthesizer TTS nature, it is inevitable that it will sound a little robotic, even though

major improvements have been recorded over the last few years. Nusbaum and Pisoni (1985) noted that listeners faced with poor-quality speech are forced to rely heavily on the semantic content of the speech in order to resolve ambiguities. They gathered a number of listeners to listen to grammatically correct English sentences and sentences that made no sense. They observed that when the sentences were presented using natural speech there was only a small difference between the recognition accuracy obtained on the meaningful sentences (99.2%) and on the nonsense ones (97.7%, a drop of 1.5%). However, when the sentences were presented using various speech synthesis systems the margin between grammatically correct English sentences and those that made no sense was much greater. It was down from 90.7% to 76.3%, a drop of 14.4%. They argue that synthetic speech requires further analysis by the end user. This will ensure that the system will need more synthetic speech enclosed around the conversations held on the instant messenger system, for example, "*Ciaran said: 'Hi, how are you?'*" etc. it has been found that audio imposes more cognitive load than an average human user can handle, a situation called cognitive overload (Sauer, Hochheiser, Feng, and Lazar, 2008).

### ***1.5 Research Methodology***

This research is personally significant to me due to my use of real time communications mediums on an almost daily basis. Using MSN messenger allows me to stay in contact with my family and friends when I'm traveling and on the road with work. Two of my aunts whom I converse with daily through messenger are blind since birth. I realized the difficulty for any blind person wishing to converse via the internet and then began to think of ways of improving this process for them. At present there exists voice-over-IP to help the visually impaired communicate, but there still poses the difficulty in the most basic functions, such as clicking the talk button in order to begin the communication ViaVoice. Similarly, there are dozens of other functions such as 'signing in', 'signing out', 'saving conversations', and the use of image icons which possess problems for the blind. These features are just to name a few.

The research methods that will be used in this research will be a combination of qualitative and quantitative methods, because as American philosopher Thomas Kuhn has pointed out that these two approaches are not entirely mutually exclusive in the research arena (Kuhn, 1961). He believed that qualitative work carried out in the area of physical sciences often led to more 'fruitful' quantitative results in the same area. Qualitative research is often used to gain an understanding of the theories and observations in the field which can later be tested by quantitative research.



### 1.5.1 Speech Synthesis

Speech synthesizers have limited capabilities which one needs to be aware of when attempting to integrate such a synthesis system into their mainstream application (Junqua, 2000). For example, even though a synthesizer can synthesize almost any word in the English language, at times the correct pronunciation of the word may not always be accurate. No matter how large the exceptions dictionary, TTS synthesizers will make different pronunciations. For example, "wind" can refer either to an air current or a circular action, such as the tightening of a screw. In order to decide which pronunciation to use, the synthesizer would have to determine which meaning is intended - and that could well be beyond the capabilities of most current synthesizers.

Apart from pronunciation issues which were addressed above, another issue that will need to be investigated is the use of pauses in character strings. Sometimes in our speech we consciously and unconsciously insert pauses into our speech: for breath, for humorous effect, to allow listeners better comprehend what we have just said etc. This inevitably raises the question of how this is reflected in speech synthesizers for the blind. Will the synthesizer you use cater for such pauses? Will a pause correlate to a comma character? etc. Juergen (YEAR) has investigated these issues in great detail concerning text analysis. Very often it is questions like these that are often overlooked as they appear to be trivial, when in reality these issues are very important for the blind, i.e. the potential user of a system.

Just how pauses aid speech comprehension, is still not fully understood, and a great deal of research is being carried out in this area. Nonetheless, most linguists now accept that pauses play a crucial role in speech comprehension. This belief is backed up by studies which show that inserting inappropriate pauses, or even just swapping around the position of pauses within a speech string, significantly impede speech comprehension.

Reich (1980) found that moving a short pause to an inappropriate place in a spoken sentence added several seconds to the average time taken by people to understand and respond. With this in mind, it raises the question of how FreeTTS would respond to various characters such as commas, question marks, etc, and how would this have an effect on the blind users of the system. This needs to be considered which pressed me into aiming to undertake primary research in the area with this specific synthesizer and thus consulting the developers behind it.

## ***1.6 PESTLE Analysis***

***Political:*** In The Disability Act (2005) it was put forth that any service offered by a public body must be accessible to persons with disabilities and the National Disability Authority created a code of practice that was published in 2006. In America Section 508 standards were created which involves not only web content being accessible to all but any ICT product, including documentation.

***Economic:*** Implementing a piece of software which is accessible to all will make the content available to a wider audience which is of benefit to any organization; be it public or private. Basing it on open source software means it will be made available at low cost as there are no licensing fees from vendors.

***Social:*** An accessible application will have the social benefit of having a wider scope and it will now be extended to include disabled people. Millions more people will be able to join the instant messaging and chat community. The use of open source software will allow a large number of developers to rapidly develop new features and fine tune an evolving application; thus making it a broader developer experience too.

***Technical:*** There are a number of real time communication tools available with all of these vendors shifting the productivity line outwards, essentially just raising the bar for all involved. With open source speech technology we can improve customized speech applications and any security vulnerabilities will be disseminated to the public where security researchers can work with developers on patching the software.

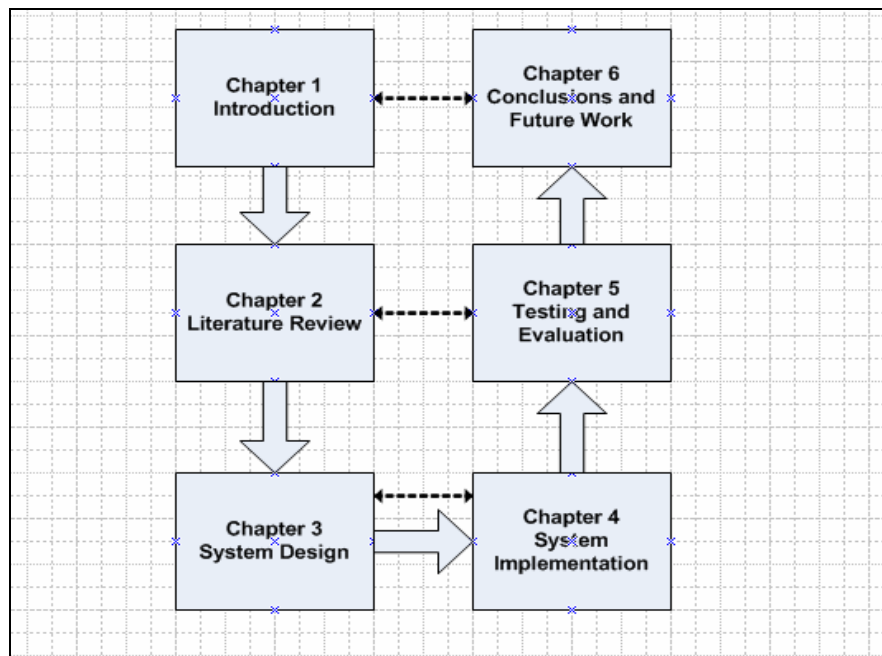
***Legal:*** It is within the rights of an individual to sue for data not being made available to them. Take for example the case of ‘Bruce Lindsay Maguire v Sydney Organizing Committee for the Olympic Games’; it was found that the respondent was guilty of discrimination against the complainant who was blind, as their website was not fully accessible and therefore the complainant could not access the information therein (Human Rights and Equal opportunity Commission, 2000).

***Environmental:*** Accessible sites will make for a better e-environment among numerous different cultures no matter the ability of the user of the system.

## ***1.7 Thesis Roadmap***

As seen from Figure 1.2, this thesis is structured in this way to give the reader a clear concise overview and see how the chapters are interlinked in this research. In this research the chapters have been structured so that a mirroring approach is seen in the dissertation. Chapters 1 and 6 mirror each other structurally since all of the main issues raised in the Introduction need to be concluded about in the Conclusion.

Similarly the main themes identified in the literature review will be used to test and evaluate the current research, and results discovered in this implementation will be compared with results of existing studies from the literature. Finally the main sections in the Design chapter clearly need to mirror the sections in the Implementation chapter, as all themes identified in the design should be reflected in some way in the implementation process.



**Figure 1.2 Thesis Outline**

Chapter 2 reviews existing research in the area of software accessibility, speech enabled technologies, open source software, and instant messaging. This chapter provides a framework for this research study. A solid documentation of previous work is produced to carry forward into the design stage

Chapter 3 is the system design for this thesis. It includes a complete sequence of UML diagrams from the requirements analysis, to the domain analysis, to the design architecture in order to safeguard from potential pitfalls during the development stage.

Chapter 4 outlines all the implementation of this speech enabled chat application in great detail. The various classes and methods are discussed and how these components are interlinked to produce a functionally working speech enabled chat application. Ways of improving this application for the visually impaired are

addressed by adding additional functionality to the application through an iterative development process.

Chapter 5 tests and evaluates the software that was implemented in chapter 4 in order to gain feedback from the users of the system. The Data Analysis discusses the results from the testing, questionnaires, and interviews.

Chapter 6 includes what conclusions can be drawn from this thesis and comments on any future work that can be done in area of research of open source speech enabled chat applications.

## **2. Literature Review**

### ***2.1 Introduction***

In this chapter relevant research literature relating to this thesis is gathered and reviewed. Such literature includes accessibility constraints in the Information Technology (IT) arena, Instant Messaging (IM) technology, and Open Source Software (OSS).

The affects accessibility constraints in IT have on the visually impaired is reviewed and if legislation and guidelines has any affect on this. Also reviewed in this chapter are the current Instant Messaging (IM) tools on the market today and if such tools were designed of a universal standard. The idea of Open Source Software (OSS) is introduced, the history of the OSS movement, the methodology used in OSS, and the quality and performance of Open Source Software.

Reviewing such literature will pave an easier route for this thesis, which looks at ways of integrating network communication mediums with Open Source Speech Synthesis technology to allow the visually impaired use Instant Messaging real time communication mediums.

### ***2.2 Accessibility in the Social Context***

In order to achieve effective participation in today's economy it is essential to have computer literacy in all walks of life. With better broad-band connection speeds, enhancements in computing technology and better education systems, more and more people are using computers and are connecting to the Internet. In parallel with this trend is the evolution in the accessibility and affordability of both general use and assistive computer technologies for the disabled. As long as both of these trends are kept working in parallel, then it will always be possible to grant the disabled access to the same opportunities, skills, resources, entertainment, etc.

Software accessibility has improved significantly over the past couple of decades. Consider the changes that have occurred from when Microsoft were first developing the initial MS-DOS operating systems, to today's Microsoft Vista. Computers are becoming more graphically-orientated as opposed to being text-based. Many graphics and images are not properly supported by descriptive text, therefore software of this kind is only useful to people who can point, click, and view. It has been identified that 98% of all websites have some access barrier to people of a disabled nature (McGrane, 2000). Participating in the cyber society of modern age has prevented or proved difficult for roughly 10 million blind or visually impaired users in the United States alone (AFB - American Foundation for the blind).

### **2.2.1 Implications of Inaccessible Technology**

Through advancements in IT and networking, coupled with cheaper and more easily available technology, computers are becoming increasingly common in the home. This is a significant step in helping people, disabled or not, in gaining access to education, shopping, socialising etc. Accessible software can provide these avocational interests through online learning, online shopping, and socialising with friends through instant messaging software. With this in mind, it could be viewed that computing technology should be mainly designed, or geared towards the disabled portion of the population. This is reflected when Steven R. Reininger said in the Access Now V. Southwest case when he was quoted as saying:

“The Internet has become a huge shopping mall and is very important to blind people who sometimes have trouble getting around. It is critical that there be access”

Unfortunately however, a recent report that is a matter of grave concern (A nation online REFERENCE) demonstrated that people with disabilities are in fact less likely to use computers than the able-bodied. The statistics generated showed that those with disabilities were less likely to have a computer in the home than those without a disability, and those who did have a computer in the home were less likely to use it than those who were not disabled.

There were a couple of reasons for this which has a cascading effect on computer usage by the disabled. Firstly, it is found that people with a disability have less years of education. The report also found that the average percentage of Americans who do not have a diploma is 14% whereas the average percentage of disabled Americans who do not have a diploma is 22%. A similar pattern was found regarding undergraduate degrees and Masters Degrees. As a consequence, the disabled are not in a learning environment where they are exposed to a range of computing technology.

The above statistics inevitably have a negative ripple effect with regard to the percentage of disabled people who possess a computer. With less education, there will be less job opportunities and thus less income to be able to afford a computer. The report also looked at employment figures, which showed that the disabled were less likely to be employed. Of those who were employed, the report showed that about 29% of disabled people were on an income of less than €20k, while of the non-disabled this figure was only 12%. Cost is an important factor when one considers internet usage and the average person (Russell, and Stafford, 2002). Assistive

technology is not always readily available. This is due to inadequate funding for adaptive computer technologies (Fichten *et al.*, 2001).

Such technologies which are designed with the disabled in mind are more specific and are tailored to an individual's needs. As a result they tend to have a smaller market and therefore are more expensive. On an economic level, this paves the way for the current research and implementation being accepted into the market as it is entirely open source, meaning it will alleviate the costs typically involved in many closed source applications. Open source technologies will be discussed in more detail in *Section 2.4*.

An even more disturbing survey (Knight *et al.*, 2002) found that 54% of disabled people felt that internet access is essential to their daily life compared to only 6% of people with no disability. This is because the internet and software allows the disabled to more easily interact with the outside world when at times can be physically impossible. Having said that, the unfortunate reality is, people with disabilities have a more negative opinion on technology (Pew, 2003) than able-bodied people.

In the British Government Online Annual Report (2002) it acknowledges that there is a digital divide with regard to the disadvantaged communities – namely the poor, elderly and disabled. On many fronts they have delivered positive results, for example, they were successful in setting up 'online centers' around the UK in some of the less well-off areas, and they were successful in bringing the Internet to thousands of users, 61% of whom were indeed disadvantaged. However the report also found no evidence of assistive technology when using the computers.

## **2.2.2 Legislation & Guidelines**

Computer Software, like virtually anything in this world, can be designed to meet the needs of all people, including those with disabilities. Making software accessible is the process of making an application readily available to everyone no matter what their ability – or lack thereof. Regrettably, it is still the sad truth that a large portion of software is still not accessible to people with disabilities. The introduction of legal documentation (W3C, 1994-2006) and laws (ADA, DCA 1995) has raised the level of software and computer accessibility made available to the disabled. Modern legal systems began to allow the courts to review acts, in a process called judicial review, as people began to accept that access to information is a civil liberty.

### **2.2.2.1 Irish Legislation**

The Disability Act 2005 is part of a framework of Government legislative measures which promotes equal inclusion for all and considers the importance of services available from public bodies. The term "public body" includes Government

Departments, local authorities, and semi-state bodies, as well as most other state organizations. It includes the 'provision of information' even if this information's source is a website because a website comes under the remit of a service.

Section 28 (1) communications by a public body to a person with a hearing or visual impairment must, as far as practicable, be provided in an accessible format, following a request. Information provided electronically must, as far as practicable, be compatible with adaptive technology.

This indicates that it is unacceptable for a website to be made inaccessible to one or any. Failure to comply with section 28 of the Disability Act (2005) would be a breach of this act on the grounds of discrimination.

The Equal Status Act, 2000-2004 gives protection against discrimination in areas including provision of goods and services whereby a 'service' means a service or facility of any nature which is available to the public without prejudice. Services include access to public places, banking and insurance services, entertainment, facilities for refreshment and transport. In this day in age, many of these services are provided electronically to the public via the internet. Section 4 (1) states that discrimination

includes a refusal or failure by the provider of a service to do all that is reasonable to accommodate the needs of a person with a disability by providing special treatment or facilities, if without such special treatment or facilities it would be impossible or unduly difficult for the person to avail himself or herself of the service.

The next section of this act continues to state that

failure to provide the special treatment or facilities to which subsection (1) refers shall not be deemed reasonable unless such provision would give rise to a cost, other than a nominal cost, to the provider of the service in question

The term 'nominal cost' is not clearly defined in this piece of legislation which appears to be a safeguard for those who are involved in discrimination against the disabled.

#### **2.2.2.2 U.S. Legislation**

In the US, the Americans with Disabilities Act (ADA, 1990) and related legislation have had a major impact on all aspects of living for people with disabilities. These statutes declare that 'effective communication' and the provision of 'auxiliary aids and services' now apply to the internet and computer environment. In response to an ADA investigation, a set of guidelines were outlined to ensure access to distance education for all with disabilities (The High Tech Center Training Unit, 1999).



In 1998, section 508 of the Rehabilitation Act of 1973 was amended by Congress in the US with the hope of eliminating barriers in gaining e-information through IT and to encourage the development of user disability friendly technologies so that the disabled could perform their daily activities like anybody else. Section 508 '*will open many more doors to information for people with disabilities.*' (NDA Accessibility, 1998)

The ADA was signed into law in 1990 when the internet was in its infancy and therefore there was no mention of the Internet in this act. Unfortunately the section 508 amendment was subsequent to this, making it subject to judicial review, as to whether or not the Act should apply to the Internet. There are 5 Titles governing the ADA. The most appropriate with regard to accessibility and the internet is Title III. Title III of the ADA says:

“No individual shall be discriminated against on the basis of disability in the full and equal enjoyment of the goods, services, facilities, privileges, advantages, or accommodations of any place of public accommodation by any person who owns, leases (or leases to), or operates a place of public accommodation.”

What is under judicial review now is whether this act can be extended to cover cyberspace or not. Can the internet be classed as a 'place of public accommodation' even though the internet is never once mentioned in the legislation?

In cases which reached the courts the plaintiffs argued that the act is so broad it must cover the Internet, however the defendants believe that since the internet is not made of 'brick and stones' the Internet cannot be classed as a public accommodation.

### **2.2.2.3 Accessibility Guidelines**

The Web Accessibility Initiative (WAI) is a set of guidelines that were introduced with the hope of promoting the idea of accessibility for all. These guidelines are divided into

- Web Content Accessibility
- Authoring Tool Accessibility
- User Agent Accessibility

The Web Content Accessibility Guidelines (WCAG) was developed in 1999 by the WAI and is today the standard guideline used for accessibility. WCAG has 14 guidelines for achieving accessibility, each of which has one or more checkpoint. These checkpoints explain how the guideline applies in typical content development scenarios.

The Authoring Tool Accessibility Guidelines was developed to aid developers in creating authoring tools that produce web sites that are of an accessible nature and to assist developers in producing an equally accessible user interface.

Authoring tools aim to help developers create accessible Web content through prompts, alerts, repair functions, and help files (Authoring Tool Accessibility Guidelines 1.0).

The User Agent Accessibility Guidelines help to lower access barriers to Web content for the disabled. User agents include HTML browsers and any other form of software that retrieves data from a remote web server and displays this information to the end user (User Agent Accessibility Guidelines 1.0). A user agent that conforms to these guidelines promotes accessibility through its own interface and through ways of communicating with other speech enabled technologies.

Having a clear and consistent set of guidelines in place not only benefits people with cognitive disability or blindness but benefits all users.

#### **2.2.2.4 Case Law**

##### **PGA Tour vs. Casey Martin case (1998)**

In The PGA Tour vs. Casey Martin case (1998), the plaintiff Martin sued the PGA Tour for the right to use a golf cart during competition under the ADA. Since a golf course was viewed under this act as a physical public accommodation Martin won the case. This case was easily resolved by the courts as a golf course was easily identified as a public accommodation. The debate begins when one argues over internet accessibility, whether cyberspace is treated as nothing more than bunch of electrons making up a virtual place or whether it is a public accommodation providing 'services' and 'privileges' where nobody should be discriminated against.

##### **Doe v. Mutual of Omaha Ins. Co.**

In some circumstances the limiting of a public accommodation to physical structures would almost make the existence of Title III of the ADA and section 508 useless. This was not the case in Doe v. Mutual of Omaha Ins. Co where the court ruled that a website could be regarded as a 'public accommodation'. John Doe purchased health insurance policies from Mutual of Omaha Insurance Company (Defendant). Doe claimed that Mutual sold insurance policies that contained two different terms and conditions. One insurance policy was for persons with AIDS and the other one was for everybody else. On the basis of disability, people with AIDS are being denied the opportunity to receive potential insurance benefits that other insured persons can get. The statute defines "public accommodation" to include an "insurance office" whose

operations affect commerce. Since Mutual is engaged in the business of "offering and selling its insurance directly to members of the public at various office locations" in this country, it is a "public accommodation" covered by Title III (Doe v. Mutual of Omaha Insurance Company, 1999).

The debate of whether a 'public accommodation' can be extended to cover the internet becomes even more intense when one considers the issue of public governmental websites and private websites. There have been a number of cases (Martin v. Metro Atlanta Rapid Transit Authority, 2001) where the plaintiffs have sued governmental websites, even though the internet is not mentioned in the ADA, the website is viewed as a public accommodation.

### **Access Now v. Southwest Airlines (2002)**

One such case, Access Now v. Southwest Airlines (2002) addresses the issue of whether the ADA can be applied to private websites or not. The plaintiff alleged that Southwest Airlines website was completely inaccessible to the visually impaired. The defendant however argued that their website did not constitute a place of accommodation and so Title III of the ADA act did not apply. In its findings, the court ruled that 'to fall within the scope of the ADA as presently drafted, a public accommodation must be a physical, concrete structure. To expand the ADA to cover "virtual" spaces would be to create new rights without well-defined standards.' (Access Now v. Southwest Airlines, 2002)

As with Title III of the ADA, section 508 also 'requires access to electronic and information technology provided by the Federal government', however it 'does not apply to web pages of private industry' (The Rehabilitation Act Amendments, Section 508).

## **2.2.3 Assistive Technologies**

Assistive technologies are products used by people with disabilities to help accomplish tasks that they cannot accomplish on their own or could not perform easily. This section will identify some of the assistive technologies used by disabled people which are currently on the market.

### **2.2.3.1 Screen Readers**

A screen reader is a software application used to synthesize web content or information on the computer for the visually-impaired user to understand. The most commonly

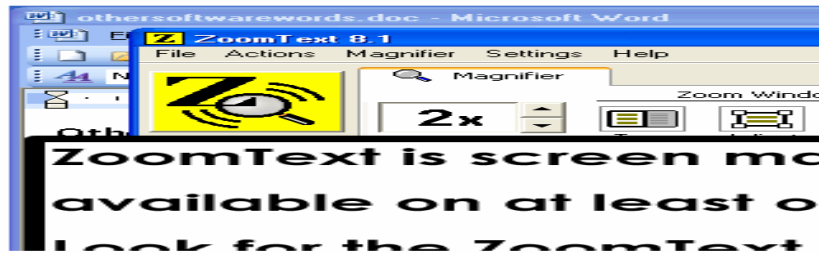
used screen reader is JAWS (Job Access with Speech). There are a wide range of other screen readers including Windows Eyes, SuperNova, Hal, and Outspoken. SuperNova is the only access product to combine magnification and screen reading in one easy to use product. The screen readers available for the Linux operating system include Gnopernicus and Speakup.



**Figure 2.1 JAWS Basic Dialogue Box**

### **2.2.3.2 Screen Magnifiers**

A screen magnifier is a software application for the visually impaired to magnify a portion of the screen to enable easier viewing of the content displayed. One of the most popular screen magnifiers is ZoomText, which ‘builds its unique view by enlarging the contents of a subsection of the display’ (Brunet, 2005).



**Figure 2.2 ZoomText**

From the control toolbar you can set ZoomText to suit your preferences, such as the screen area to be magnified, and the magnification factor to enlarge the selected area to.

## **2.3 Instant Messaging & Chat Applications**

From sending an instant message to a friend, to e-mailing co-workers, to placing phone calls, to conducting video conferences, the Internet offers a number of ways to

communicate. Instant Messaging (IM) is one of the fast growing methods of communicating over the web today. IM can be thought of as a two-way real time communication between two or more participants via text messaging in a chat room environment. The key part here is real time, which is exactly what distinguishes it from the traditional form of communicating via email. Chat systems, instant messaging and texting systems are synchronous, which means that correspondents must be co-present online. Typically, conversations are rapid and each individual comment is short (Preece *et al.*, 2003). Most IM servers allow the client to declare themselves to be in one of four states: online, busy, away, or offline (Resig *et al.*, 2004).

### **2.3.1 History of Instant Messaging**

Instant Messaging applications are around since the beginning of the Internet, where basic command prompts were used to chat via text. The UNIX talk command was popular in the 1980s and early 1990s. Internet Relay Chat (IRC) was developed in 1988 by Jarkko Okarinen (Preece, *et al.*, 2003). IRC is a console based form of real-time Internet chat or synchronous conferencing.

The GUI based IM clients as we know today weren't introduced to the chat community until around the mid 90s. ICQ (1996) being the first, followed by AOL Instant Messenger (AOL Instant Messenger, 1997). Meanwhile, other companies developed their own applications (MSN, Yahoo), each with its own proprietary protocol and client; users therefore had to run multiple client applications if they wished to use more than one of these networks.

Even though the technology that supports online communities has changed tremendously over the years, the biggest change lies not in technology but in who is using it (Preece, *et al.*, 2003). Therefore, we need to focus on the end user, however in certain circumstances; we need to address the technology being used first in order to focus on a particular set of end users.

### **2.3.2 Instant Messaging Usability**

Instant messaging is a popular and relatively new form of social interaction. It has gained acceptance by people from all generations. It is a major hit among the younger generation where (Oblinger, 2004) seventy percent use instant messaging to keep in touch and forty one percent indicated they use email and IM to contact teachers or schoolmates about class work, indicating that IM is also used as a means of education too.

Instant messaging and real-time communication among personnel is growing in importance and it is no longer seen as just a way of communicating across networks

and the broader internet for personal means. They also enable real-time communications between workers, and allow individuals to know instantly if coworkers are available for different business needs. Tang *et al.* (2002) discuss that real time communication is '*quickly gaining acceptance within the corporate environment*' too, so employers also seem eager at encouraging its use in daily work practices. Farmer (YEAR) noted that IBM reported that its 300,000 employees send over 3 million instant messages a day thus speeding up the decision making process from days or week to minutes. IBM provides help desk support to 1000 of its largest customers via instant messaging. Furthermore IBM reported that it has reduced telephone use by 4%, reducing the load on mail servers, increased responsiveness and collaboration, and improved employee productivity and teamwork.

There are numerous instant messenger technologies on today's market which bring millions of people together from all around the world. Communication via Instant Messenger over the Web has quickly become one of the Internet's "killer applications". This is confirmed by the popularity of AOL's Instant Messenger (AIM), MSN Messenger, Yahoo Chat, and ICQ among the most popular.

### **2.3.3 Instant Messaging Accessibility**

IM allows people to discuss real experiences in virtual spaces, where often physical accessibility barriers may prevent certain real life community situations occurring for the disabled, due to ramps, obstacles etc. The sheer existence of computer technology prevents people from having to move around, but unfortunately the IMs on the market today also possess an accessibility barrier and therefore a segment of the population cannot use this method of communication.

There are numerous new instant messaging tools (KDE & MSN) which are being developed which are more advanced with special added on features such as incorporating numerous other existing messaging tools into a single application (Trillian). These newly developed technologies include better customized user friendly interfaces. What is more urgently required is a communications medium designed to allow the visually-impaired users make use of such technology, where other main-stream applications lack the functionality they require.

AOL has developed an instant messenger tool aimed in aiding the deaf communicate effectively across the internet (AOL). However its software fell short in many respects when it came to creating and designing universal software usable by people from all walks of life.

On Nov 4<sup>th</sup> 1999, the national federation for the blind (NFB) sued the internet service provider AOL for failing to make its services available to the disabled. (Blanck & Sandler) have said that the NFB alleged that AOL's Internet browser and services were inaccessible to the blind and did not comply with the accessibility requirements

of Title III of the Americans with Disabilities Act (ADA). The software did not operate with the disabled in mind. On the contrary, their software had unlabeled graphics, mouse only activated commands, and not all functions were accessible from the keyboard. In fact it was almost impossible for a blind user to sign up for AOL because of the text used in registering is unlabeled graphics. The plaintiffs argued that its services such as signing up, welcome screens, and chat rooms were completely inaccessible because screen readers could not read text hidden within graphic displays. Such services like these, which fell short of being accepted as universal design, encouraged this particular dissertation. This particular case was settled out of court with AOL promising to make future versions of its software more accessible to the visually impaired.

As discussed in section 2.2.3, it is only in more recent years through legislation that software is becoming more usability friendly for the disabled. Bigham et al., (YEAR) noted that AOL was the second most accessible instant messenger after MSN Messenger. AOL being the same IM service provider who were sued by NFB in 1999. This is a clear example of how legislation and guidelines has helped make software more accessible for the disabled. Improvements have been made but there can still be a lot more done.

With the visually impaired in mind, and the regulations outlined in the Disability Discrimination Act 1995 (Disability Discrimination Act), it is clear that there is an urgent need for a real time communications tool with speech synthesis & voice recognition tools enabled. This is backed up in key findings found in the studies conducted by Fichten, et al. (2001), where a need for assistive technology for people who suffered with a number of disabilities was identified. Additionally the ‘importance of ensuring that different types of adaptive equipment worked together’ was highlighted; in particular the ‘compatibility between dictation software and voice technologies that read what is on screen should be taken into consideration’ was singled out.

## ***2.4 Open Source Software***

### **2.4.1 Introduction**

Software can be considered an ordered sequence of instructions for changing the state of the computer hardware in a particular sequence. Once these lines of human-readable source code are compiled, the result will be a machine-readable executable program which can then only be of use to a computer. Open Source Software (OSS) is a type of software where these ‘sequence of instructions’ or source code is ‘open’, meaning that it is available for the user of the software to modify at will, producing a more fine tuned and updated version of the original software.

The Open Source software market is gaining significant attention. There are hundreds of thousands of various types of OSS projects that exist such as web servers (Apache), Operating Systems (Linux), scripting languages (PHP, Perl), programming languages (Java), and Mail transfers agents such as Sendmail (Madey *et al.*, 2002).

## 2.4.2 History of OSS

The Open source software phenomenon has come a long way since its inception and it is one of the most influential aspects of information technology. It has produced some of the most stable and widely used software packages ever produced (Bretthauer, 2001). Since the early 90s it hasn't been a specific piece of software which caught the IT world's attention but more the way software has been developed and distributed. Needless to say it was the introduction of the OSS movement.

In the 1960s and 1970s mainframe computers stationed mainly in universities were used mainly as sources of research where source code was distributed freely (Weber, 2000). As computer technology spread, the level of incompatible and non-portable software across multiple platforms increased which led to the development of the Unix operating system and the C programming language in 1969 (Nuvolari, 2003). Furthermore in 1979 UNIX machines began connecting to a network which further encouraged the sharing of computer programs.

Following this in the 1980s the talented developers who once served the open source community in research labs moved to privately owned software company where they were now prohibited from disclosing the source code of the applications they developed (Weber, 2000). In retaliation to this copyright movement, Richard Stallman, an evangelist of the open source movement, created the open source GNU operating system in 1983 (GNU Operating System) and two years later founded the Free Software Foundation (FSF) which is a non-profit corporation set up to support the free software movement (Hars and Ou, 2001). The license schema used in the FSF was the General Public License (GPL) also referred to as 'Copyleft'. 'Copyleft' outlines that all users are not restricted from using, modifying, and distributing software (Kuan, 2002). In one of Stallman's motivational papers, leading up to the creation of the Free Software Foundation, when debating whether to join the proprietary software world:

“I could have made money this way, and perhaps amused myself writing code. But I knew that at the end of my career, I would look back on years of building walls to divide people, and feel I had spent my life making the world a worse place.” (Stallman, 1985)

Richard Stallman was one of the most vocal enthusiasts promoting the OSS movement who believed that software and its source code should flow as freely through networked communities of developers as *'data flows in bits through a microprocessor'* (Weber, 2000).



In 1987 the Perl scripting language was designed by Larry Wall, where it was originally used for text manipulation until the growth of the Internet where it was later used as a programming language for HTML form processing (Scotts, 2002). The GNU operating system founded by Stallman, lacked a Kernel which later came from Linus Benedict Torvalds in 1991 (West and Dedrick, 2001), a second year student at the University of Helsinki. Torvalds disclosed a short note on an Internet newsgroup which marked the birth of the Linux Operating system.

“I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix...I've currently ported bash(1.08) and gcc(1.40), and things seem to work. Any suggestions are welcome, but I won't promise to implement them :-)” (Dirk and Dusty, 2003)

OSS is a social movement with goals set by a volunteer network of developers (Healy, 2003) and since these developers can be dispersed across the world, the OSS movement relies heavily on communication mediums to spread their ideas and theories (Hann, Roberts, Slaughter, Fielding, 2002).

With this in mind, OSS activity received a burst of energy around the same time the Internet exploded in the early 90s (Weber, 2000). The fact that the Internet and the Open Source movement work in parallel coincides with the SourceForge website claims of having over 600,000 users with a further 700 members joining daily and over 60,000 projects being worked on by these users with a further 60 new projects added daily (Elliott & Scacchi, 2003).

The Apache HTTP web server emerged in 1995 and is primarily used to serve both static and dynamic Web pages on the World Wide Web (Apache YEAR). The open source database MySQL was introduced a year later in 1996, and subsequently a year after that again, PHP was born (PHP).

In 1998 the Open Source Mozilla project created its finest product, the web browser which is being worked on by one of the largest communities within the OSS (Reis and Fortes, 2002). The Mozilla web browser has been ‘designed for standards compliance, performance and portability’ (Mozilla). This open source product is also user-friendly for the visually impaired, as it has the capability of enabling the caret function which can then indicate to the screen reader the current content of interest (King, Evans, and Blenkhorn, 2004). Furthermore, in aiding the blind, the Mozilla web browser’s web authoring tool ‘requires that authors address the need for alternate text for images’ (Thompson, 2005).

In 1998 the term ‘free software’ was changed to ‘open source’ because it is more ambiguous and more acceptable by the corporate world (Times Daily). Also, Stallman

and other advocates didn't want people to view the software as just free with regard to cost, but that it was 'open' for all to use, modify and distribute (Elliott).

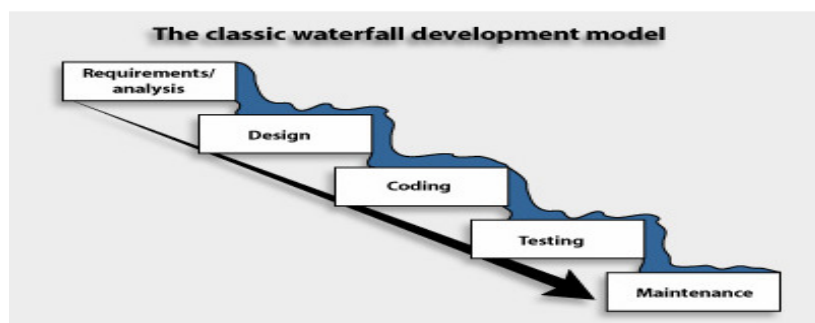
### 2.4.3 Closed Source Software

Software which is not open source has been labeled in numerous documents as 'proprietary', 'closed', or 'commercial' software. Software giants like Microsoft create proprietary software where the source code is never disclosed to the end user. Releasing the source code along with their applications would reduce their sales if the user modified the original source code for personal release. Such software companies treat their software, and the nature of it being closed source, as its secret ingredient which contributes to the company's success in the market. Such software can be viewed as a blueprint of its construction (Edwards, 2001).

### 2.4.4 Open Source Software Development Methodology

As a project grows, be it of an OSS or proprietary nature, the needs of the company or individual often changes. These changes subsequently means that the software needs to adapt to cater for those special purpose needs. Most, if not all software ever produced, undergoes continual redesign and upgrading, typically in the form of incremental releases which can either be in short intervals or long intervals (Jorgensen, 2001).

There are a number of software development models such as the Waterfall model, Spiral model, Evolutionary prototyping, Extreme Programming, Prototyping and many more. It is these models which allow software to evolve while at the same time ensuring that the new software produced remains reliable. Most traditional proprietary software projects use a variant of the waterfall model, where each release is broken into phases which must be signed off on completion (Bednar & Robertson, 2005).



**Figure 2.3 Classic waterfall development model typically found in proprietary systems (New Rowley TechView)**

The waterfall model derives its name due to the cascading effect from one phase to the other as illustrated in figure 2.3. This type of development model is carefully coordinated in order to ensure that all components and features as outlined in the functional specification at the requirements stage are incorporated into the final product.

In contrast the majority of open source documents refer to Raymond's paper 'The Cathedral and the Bazaar', which outlines two fundamentally different types of software development. This paper contrasts how proprietary software is built like 'cathedrals', 'carefully crafted by individual wizards' working in isolation and in a centralized environment. On the contrary to proprietary software development, he mentions the Linux software development methodology which he refers to as 'bazaar'.

"The Linux community seems to resemble a great babbling bazaar of differing agendas and approaches out ... which a coherent and stable system could seemingly emerge only by a succession of miracles."  
(Raymond, 2000).

The 'bazaar' mode of development initiates from a piece of code which evolves in the direction the developers choose. There is no formal centralized environment, but a group of uncoordinated programmers who modify the project on their own accord. Unlike traditional proprietary models there is no system-level design, little or no design documentation, and no system level testing (McConnell, 1999).

Raymond coined the phrase "release early, release often" to characterize the OSS development methodology (Raymond, 2000). In this paper the idea of mini-releases adds value on a motivational level too because the developers and users see the results of the work much sooner.

### **2.4.5 Open Source Software Quality & Performance**

Since programming began on the ENIAC computer in the 1940s there has been a large emphasis placed on the reliability of the algorithm or program which the software produced (ENIAC). The ongoing debate of whether open source systems or proprietary systems produces better software, better with regard performance, reliability, and security rages on. This is one debate which no article or study has ever been able to prove definitively as to which software produced is best.

From a motivational perspective, an Open Source project or application is different from a proprietary project. It is developed because it is wanted, because it is needed and not because of monetary reward. There is a desire by one to design an application

from scratch, or tweak an existing application to cater for the needs of the developers creating it. Since this in itself is a personal motivation for the developer, we would expect the end product to be better. Eric S. Raymond (1998b) reiterates this belief by noting that *'every good work of software starts by scratching a developer's personal itch'*. Raymond's theory could be the force behind claims that OSS produces more bug-free code than closed source software (Madey, Freeh, Tynan, 2002). Open source software is believed to be more stable and secure since developers can identify bugs and fix them on their own accord (Comino and Manenti, 2003).

As a result of a greater number of developers working on open source software, more bugs are exposed and corrected faster. As a result of this trend, that there exists far more developers working on OSS than proprietary software, open-source enthusiasts conclude that large numbers of reviewers lead to "efficient" development (McConnell, 1999).

In the seminal open source book "Open Sources: Voices from the Open Source Revolution", Paul Vixie states that open source software is better tested than closed source software which is due to the fact that users, often developers, are more helpful to one another when they are permitted to use, modify, and release the source code (O'Reilly, 1999).

#### **2.4.5.1 Full Disclosure V. Security through Obscurity**

Security in software, especially when discussing Open Source software, has two sides to it. The very topic *'full disclosure v. security through obscurity'* is almost a debate within a debate. 'Full Disclosure' can be viewed as the term which best describes open source as the source code is fully disclosed to the public, whereas 'Security through Obscurity' can be viewed as proprietary software. This debate is based around whether the public should have full access to security holes and vulnerability information within the software. Those who are against full disclosure believe that hacker's are aided by full disclosure in pursuing criminal and unlawful activity such as installing malicious malware on systems and theft (Ragragio, 2007). On the contrary, full disclosure would encourage developers to take their coding more serious as all flaws will be disseminated to the public which in turn would be bad publicity.

Having said this, the availability of source code does not increase the possibility of finding code defects within software. Miller *et al.* (1989) found code defects by submitting unusual data to the system and then analyzing how the program responded. Any unusual behavior may uncover a possible defect. Furthermore, it can be difficult to reverse-engineer compiled source code, but not impossible meaning that the original source code can be gotten anyway from proprietary software (Johnson, 2001).

### **2.4.5.2 Drawbacks of Open Source Software**

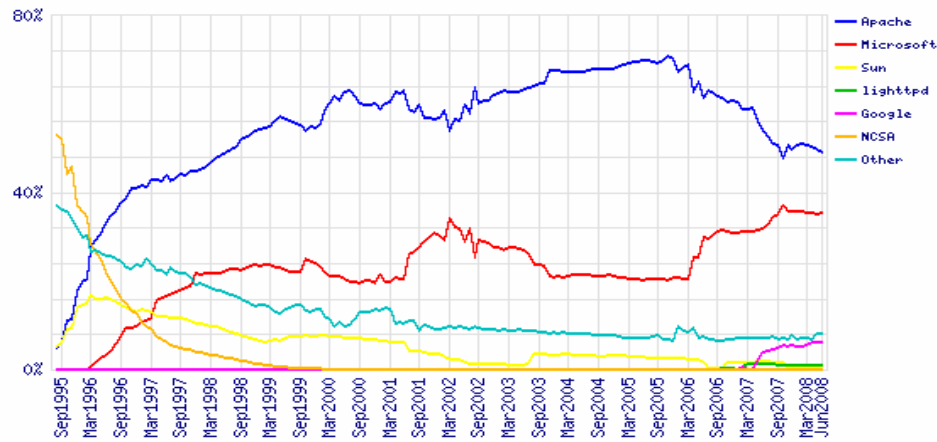
Despite the numerous advantages the open source movement has brought to the IT world over the past couple of decades there have been times when open source systems have underperformed and found to be insecure. Roughly 20 years ago, the mail transfer agent, sendmail, was found to be insecure. Robert Tappan Morris created a Worm whose primary aim, in the author's own words, was to find out the size of the internet (REFERENCE OTHER THAN WIKIPEDIA). However instead of actually gauging the actual size of the Internet he successfully managed to cripple a nationwide network and breached the Computer Fraud and Abuse Act (CFAA) by copying the worm onto each machine it found.

Similarly in 2002 the Slammer worm found vulnerabilities in the OpenSSL program which resulted in thousands of machines around the world being infected (Perriot and Szor, 2003). A patch for this bug was deployed within the same day it was found. All software, be it open source or proprietary, contains code defects and always will. The critical difference here is the low turnaround times that open source can boast about in comparison to the proprietary world. In June of 2002 Microsoft was notified of a critical flaw which was exploited via the clicking of a URL link which in turn caused files to be deleted on the user's system. Microsoft chose to play down this vulnerability until the release of Windows XP Service Pack 1 two months later in September 2002 (Gibson Research Corp).

### **2.4.6 Open Source Software Success**

It is hard to believe that the open source systems that were once limited to research groups back in the 60s are now being developed for mass usage. As the open source community continues to grow, it makes sense to place more emphasis on these systems being reliable and secure. There have been clear movements which would indicate the success of the open source movement which I would like to highlight below.

Large IT corporations such as IBM and Novell have been investing in Open Source software. For example the Open source Linux operating system was found to be the most popular on powerful networked systems and it is growing in use on desktop systems (Shankland, 2004).



**Figure 2.4 Market Share for Top Servers across All Domains August 1995 - June 2008 (Netcraft)**

From figure 2.4, we can see in 2000 the OSS Apache web server supported about 59% of all web pages which increased to 67% in 2004, while in 1998 the OSS Linux operating system was the operating system of choice for roughly 17% of users (Johnson, 2001). Although in more recent years the gap between open source and closed source web servers dominating the HTTP market has narrowed, we can still clearly see a significant margin that Apache has over Microsoft. In July 08, Apache still held almost 50% of the market share while Microsoft was in second with only 35% (Netcraft).

The billionaire software company Microsoft see open source systems as a significant threat and challenge to their own business model and the consequences this may have on the sales of their own products (Securities and Exchange Commission, 2004).

In 2000, Linux has developed into a robust and highly stable platform used by approximately 20 million people, with an annual growth rate of almost 200% (Weber, 2000). To remain competitive as a result of the open source success, some commercial companies, namely Sun, has switched most of its Solaris operating systems to an open source license to save costs (Hars and Ou, 2001).

## 2.5 Conclusions

In this chapter existing literature relating to this thesis was gathered and reviewed. Such literature included accessibility constraints in the Information Technology (IT) arena, Instant Messaging (IM) technology, and Open Source Software (OSS).

Accessibility from an IM perspective was also reviewed and how current IM software and vendors have fallen short of providing software of a universal nature. Ways to rectify this situation can now be envisaged after researching Open Source Software (OSS) and how OSS can be used to allow Instant Messaging tools be available for the visually impaired and the blind. Based on the literature reviewed in this chapter, in the next chapter we will see how to design such a system (NetSpeak) based on existing research and previous experiments.

## **3. Design of NetSpeak**

### ***3.1 Introduction***

Having done most research in the area of computer networking, instant messaging, software accessibility, and open source the design of a speech enabled chat application can now begin. The design stage is the prelude to the implementation stage which will be discussed in chapter 4.

If we are trying to design a computerized system it stands to reason that we need first to understand the business context within which the system will reside. The system will first be described using an analysis model with use cases and some domain analysis. The design model will aim to describe a technical solution, and finally it will be programmed in Java to create an application that will be ready to be deployed. This thesis will show how the usage of various UML diagrams created from analysis of requirements via a design model can be developed into the actual code and running application.

It is important to remember that the main purpose of performing systems analysis and design before constructing the system itself is to safeguard against delivering the wrong system to the users. This is particularly important for this project as it's designed with the visually-impaired in mind. As a result it will be even more crucial to ensure that all functionality as outlined in the design stage will be completed in the implementation stage. Use Case modeling is the technique UML recommends as a start-off technique in my quest for delivering the right system to the users.

In the first section a structured requirement analysis technique that will help ensure that the final system will fit within the mission and objectives of the organization will be introduced. This technique is called Use Case Modeling. This technique is UML's approach.

The UML approach will be used to find out what is going on in the environment of the system. Considerations of who is involved in each activity, when the activity is performed and why, are recorded and catered for, but the method assumes that there is no major conflict within the user community. The techniques that are introduced after Use Case modeling deal in depth with how the system works.

### ***3.2 Requirements***

The vast majority of software is developed for human beings to use. It is therefore essential to work with users when developing requirements.

Use-cases can be used to capture the requirements of the user. Through use-case modeling, the external actors that have interest in the system are modeled along with



the functionality they require from the system (the use cases). The actors and use cases are described in a UML use-case diagram. Each use-case is described in text, and that specifies the requirements of the user.

The following is an example of a specification written by a typical user of the system. The typical user of the system would be a visually impaired user where the speech chat application system would be tailored for the needs of this typical user.

- It is a chat application with speech functional tools for visually impaired users.
- The chat application allows visually impaired users to communicate.
- The chat application dictates conversations to the users and accepts voice commands as the primary input.
- The system administrator is the author of the system.
- The application allows users to chat to one or many different users in a user friendly environment.
- The user can navigate the system with ease through the use of instant synthesized feedback.
- The user can control the controls of the system using sticky keys.
- The application will only allowed authorized users to use the system.
- The user can login to the system and logout of the system at any time.
- The chat application can run on all popular technical environments (Unix, Windows etc.) and has a modern graphical user interface (GUI).
- On sign in to the system, the requesting user receives an acceptance or rejection notice immediately.
- A user must be accepted by the system before it may send chat messages.

### ***3.3 Analysis***

The analysis is intended to capture and describe all the requirements of the system, and to make a model that defines the key domain classes in the system (what entities in the system, which will hold valuable information). The purpose at this stage is to provide an understanding and to enable a communication about the system between

the developers and the users, therefore the analysis is typically conducted in cooperation with the end user.

At this early phase of the design, technical solutions or details will not restrict the analysis and will not be considered in terms of code or implementation. It is really just the first step towards really trying to understand the system requirements.

### **3.3.1 Unified Modeling Language**

The Unified Modeling Language is a universal standard for designing and documenting a system in an object-oriented manner. The diagrams produced in the design phase can easily be mapped to classes in the development stage. There are numerous diagrams which make up the suite of diagrams in the UML model. A set of stereotypes are defined using UML that help in the construction of intuitive analysis and design models in the development of Web applications (Koch, 2000). UML is used to express object-oriented analysis and design which is completely independent of the language used to implement the system. This is specifically important in the design stages of creating any system because at such an early stage it is often the case that the language has not been decided upon. Similarly the way architects use their drawings as blueprints for their constructions; business analysts use the notion of UML to write system blueprints outlining the functionality of their software system.

### **3.3.2 Requirements Analysis**

One of the first stages of the analysis phase is to define the use cases, which describes what the system provides in terms of functionality – i.e. the functional requirements of the system. Essentially it involves reading and analyzing the specifications, as well as discussing the system with potential users of the system. The model is eventually built through an iterative process during which discussions between the system developer (the author) and the customers lead to a requirement specification on which all can agree. It's extremely critical at this stage of the development process to continually communicate with the users as it will be them who will be using the system.

### **3.3.3 Use Case Modeling**

A use case model consists of a set of use cases, and an optional description or diagram indicating how they are related. A use case diagram consists of actors, depicted by stickmen, and use cases, depicted by bubbles. The actors or users require some functionality from the system and the use cases describe some functionality of the

system. Arrows represent relationships between the actors and use cases and also between different use cases.

The functionality is represented by a number of use cases, and each use case specifies a complete functionality. A use case must always deliver some value to an actor, the value being whatever the actor wants from the system.

### 3.3.3.1 Initial Use Cases

The actors in the speech activated chat application are identified as authenticated user, system administrator, chat room, and anonymous user.

The main use cases in the chat application system are:

#### Authenticated User

- Log on
- View user list
- Enter message
- Submit message
- View conversation
- Chat with users(s)
- Logout

#### Chat Room

- Message Storage
- Query Database
- Message Delivery

#### Anonymous User

- Try again
- Exit system

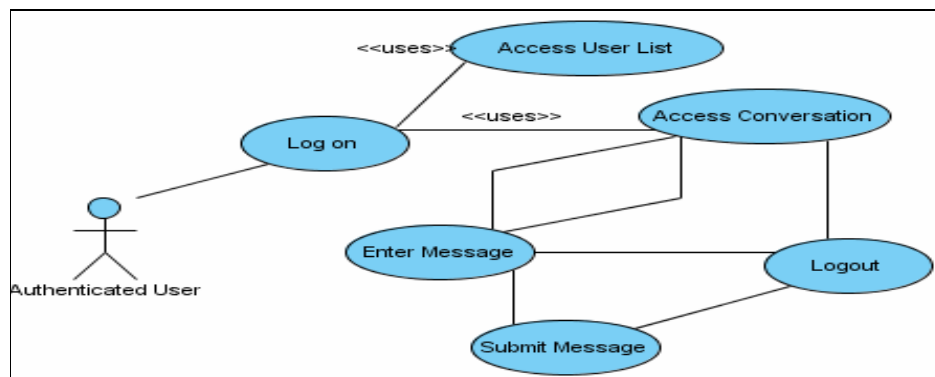
#### System Administrator

- Add user
- Remove user
- Edit user

Not included on the list is Maintenance and Control, which are more general use case that uses other use cases. It can be argued whether it is really a use case of its own, but I have decided it is to clearly separate the specific control tasks from the key functions of the system. The 'Maintenance' and 'Control settings' Use Cases will be introduced after the main use cases are discussed.

The authenticated user use case analysis for the system is shown below in a Use Case diagram. This Use Case describes the main areas of functionality in the chat system. The user of the chat system logs on to the system where they can then access messages typed by other members of the chat room as they appear on the screen. They then have the option to log out or to enter a message and post it to the room. This can be done many times before logging out of the room.

One Use Case may extend the behavior of another – typically when exceptional circumstances are encountered. For example, if a user signs into the system successfully, the user can automatically gain access to the user list and main chat area. The terms ‘access user list’ and ‘access conversation’ are used instead of ‘view user list’ and ‘view conversation’ because there will be some users who are unable to view these areas of the application, but all users will be able to access them through alternative methods of communication.



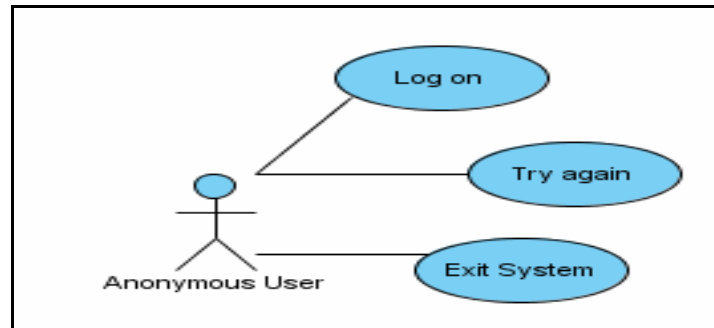
**Figure 3.1 Authenticated User Use Case Diagram [source: author using Visual Paradigm]**

A Use Case may ‘use’ another Use Cases’ functionality or ‘extend’ another Use Case with its own behavior. A Use Case may be included by one or more Use Cases, so it helps to reduce duplication of functionality by factoring out common behavior into Use Cases that are re-used many times. A good example of this can be seen in figure 3.1 where the logon Use Case automatically inherits the functionality of accessing the user list and conversations in the main chat arena. However we must be careful when it comes to use cases which ‘uses’ or ‘extends’ another use case, because some use cases can be abused in this manner. As Fowler (1998) points out in his article called ‘Abuse Cases’, the breaking down of use cases down to ‘some kind of elemental use case’ leads to functional decomposition which he describes as ‘the antithesis of object-oriented development’. This can be reflected directly into the code as ‘rich controller objects manipulate dumb data objects, which are little more than an encapsulated data structure’.

By design, this automatic step into the main chat arena would be best for the end user considering the strong possibility of the end user being blind. It is best to keep the logging in process, and the system as a whole as completely user friendly as possible.

The other user of the system, the anonymous user, although is functionally limited in using the system, must still be mentioned from the design stage so that when implementation begins the type of coding scenarios to cater for will be known. As Timothy C. Lethbridge points out in *Object-Oriented Software Engineering*, a ‘scenario is an instance of a use case

that expresses a specific occurrence of the use case with a specific actor operating at a specific time and using specific data' (Lethbridge, 2001).

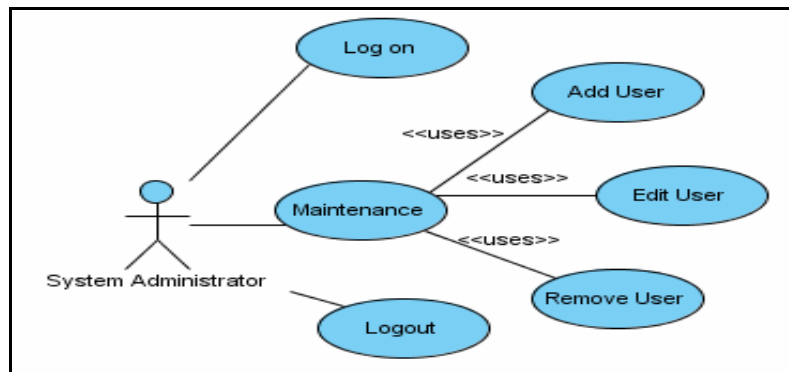


**Figure 3.2 Anonymous User Use Case Diagram [source: author using Visual Paradigm]**

Each Use Case is also documented with text, describing the use case and its interaction with the actor in more detail. For example, the use case 'log on' would be described as follows:

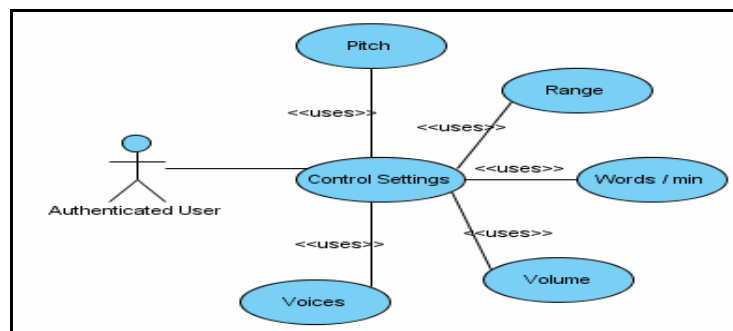
1. If the user has not yet logged into the system
  - User launches application
  - User's login credentials are entered
  - Boolean result returned on success / failure
  - The user is identified
  - User accesses main area of system
2. If the user has already logged into the system.
  - The user is identified
  - The available options are available to the user
  - The system responds to the users input
  - Once logged in, the user can choose to logout
  - Once logged out, the user is back to scenario one of this use case

The Use Cases are implemented throughout the development of the system to provide descriptions of the functional requirements of the system. They are also used in the analysis to check whether the appropriate domain classes have been identified. Below is another example of an actor (system administrator) which is involved with a number of different use cases.



**Figure 3.3 System Administrator Use Case Diagram [source: author using Visual Paradigm]**

In figure 3.4 below the second general use case as already mentioned above is used. This is another use case which was needed to be described in a separate context. It was decided to keep this Use Case separate from the other use cases to avoid confusion between control settings and the main functions of the system as already outlined above. With the visually impaired in mind, the control settings could well determine how easily navigable the system will be and determine how user friendly the system actually is. The user will be able to control these settings in a user friendly manner.



**Figure 3.4 Authenticated User Use Case Diagram [source: author using Visual Paradigm]**

### ***3.4 Domain Analysis***

An analysis also tried to identify the objects in the system. Looking at the use cases usually identifies the objects and deciding which concepts should be handled by the system. Essentially an object is an item we can talk about and manipulate. An object exists in the real world.

### 3.4.1 Class Diagrams

A class is a description of an object type and their main intention is to describe the data found in a software system. All objects are instances or examples of classes. For example 'user' would be considered a class and the actual user 'Joe Bloggs' would be an actual object of that class. The main symbols shown on class diagrams are:

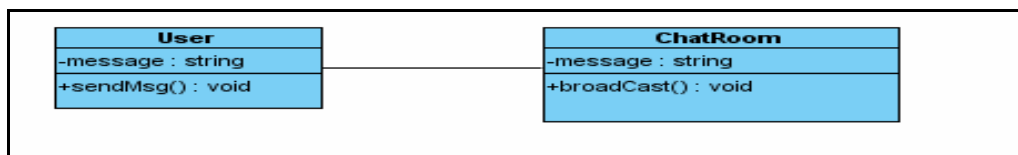
- Classes, which represent the types of data themselves
- Associations, which represent linkages between classes
- Attributes, which are simple data found in classes and their instances
- Generalizations, which group classes into inheritance hierarchies

A class is represented as a box with the name of the class inside. When we draw a class in a class diagram, we are saying that the system will contain a class by that name, and that when the system runs, instances of that class will be created. Optionally, the class diagram can be divided into three sections depending on how much detail you want your class diagrams to capture. The top section will include just the business object or class name, the second part includes the attributes and their types (optional), and the lower part will include operations or methods which are used to change the values of the attributes and thus the state of the object.

An association, depicted as a line, is used to show how two classes are related to each other. There is further multiplicity constraints associated with each association which indicates the how many instances of classes are linked with other classes.

#### 3.4.1.1 Initial Class Diagram

In line with the application needed to be developed, there are obviously two classes anyway which need to be modeled, i.e. the User and ChatRoom. At this early stage in the design process we explicitly model behavior (connection requests and chat messages) with a high level of abstraction. Later, this model of the user will be replaced by more specific interactions by users with the ChatRoom. At this stage in the design process, it is more of a guideline until more feedback from the users has been obtained.



**Figure 3.5 Initial Class Diagram [source: author using Visual Paradigm]**

The initial class diagram is relatively simple. There is a class User and a class ChatRoom. Each user can be associated with only one ChatRoom, whereas a ChatRoom can have one or many instances of a User associated with it. As the design evolves more classes and

associations linking these classes will be modeled. The classes will be more detailed including more attributes and operations.

### 3.4.1.2 Detailed Class Diagram

At this stage of the domain analysis a chat Manager or Handler class is added as it will be needed in the implementation stage. The Handler class acts as a mediator and relays all the communication between components. The ChatHandler class will alleviate much of the work from the Chatserver class. The ChatServer will accept incoming connections from the User which in turn will query the database with the DatabaseHelper class. On success an instance of the User class will be created. All messages the user sends will first be relayed to the ChatHandler class which in turn will broadcast the message to all clients connected to the Server at that time. As shown in figure 3.6 the User class will have an aggregation relationship with both LoginScreen class and the MainScreen class. In other words the User class is ‘composed of’ the parts LoginScreen and MainScreen. This is depicted on class diagrams as an empty diamond nearest the class which controls them.

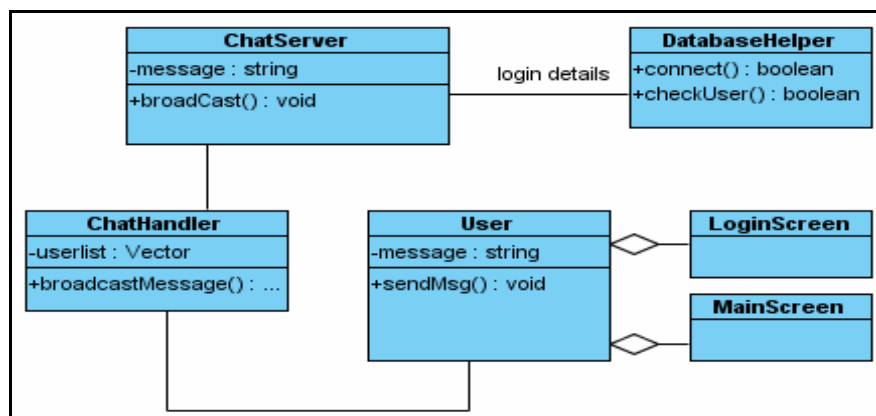


Figure 3.6 Detailed Class Diagram [source: author using Visual Paradigm]

### 3.4.2 State Diagrams

State diagrams are used to capture the life cycles of objects, subsystems, and systems. They identify all the possible states an object can be in at any given time and how different events such as received messages, time elapsed, errors, and conditions become true affect those different states over time. A state diagram should be attached to all cases that have clearly identifiable states and complex behavior, the diagram states the behavior and how it differs depending on the current behavior and how it differs depending on the current state. It will also illustrate which events will change the state of the objects of the class.

### States and transitions



All objects have a state, the state is a result of previous activities performed by the object, and is typically determined by the values of its attributes and links to other objects. A class can have a specific attribute that specifies the state, or the state can be determined by the values of the normal attributes in the object. An object changes state when something happens, which is called an event, for example a user logging into the system is an event which will change the state of the object.

### 3.4.2.1 Initial State Diagram

Some diagrams mix well with state diagrams but not all. For example below a state diagram will be illustrated for the User class.

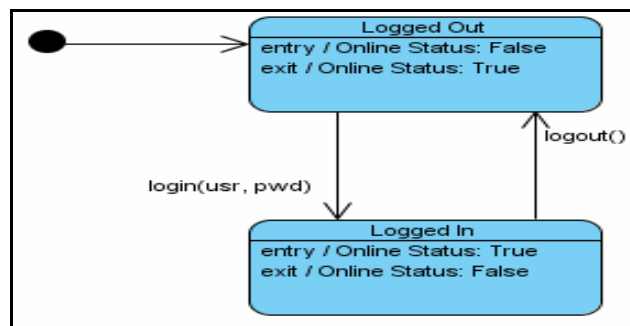
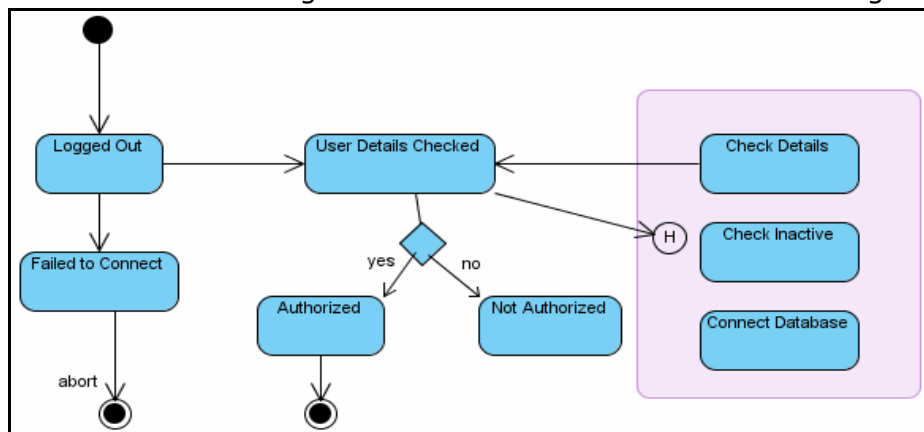


Figure 3.7 Initial State Diagram for class User [source: author using Visual Paradigm]

An obvious constraint on the above diagram would be to only allow a change of state from 'Logged In' to 'Logged Out' if the Online Status for that user has a boolean value of 'true'.

### 3.4.2.2 Detailed State Diagram

A more detailed state diagram for the User class can be seen in Figure 3.8.



**Figure 3.8 Detailed State Diagram for class User [source: author using Visual Paradigm]**

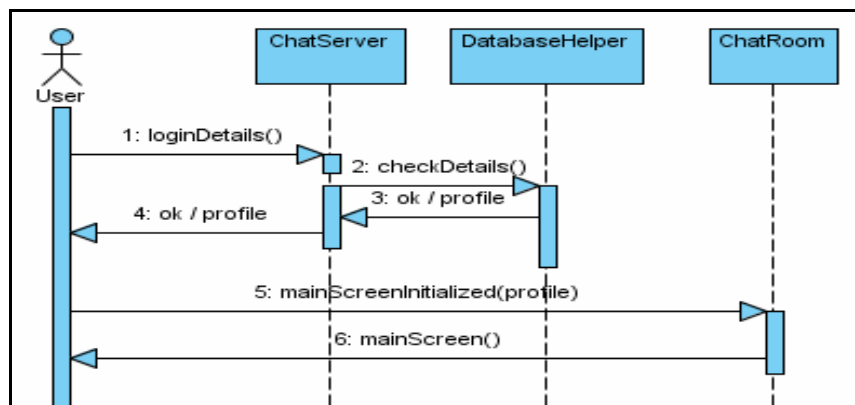
From above we can see all the possible states that a User can be in. If a user experiences an exception timeout or a connection error the user will not be able to log in and thus there will be no state change for the instance of the User class. If the user's authentication details are valid, then the user's state will change to 'Logged In' otherwise the state will remain as 'Logged Out'.

### 3.4.3 Sequence Diagram

Any of the dynamic (sequence, collaboration, or activity) UML diagrams can be used in order to describe the dynamic behavior of the domain class. The flow of logic is modeled using sequence diagrams. Sequence diagrams are used for both analysis and design purposes. The basis for the sequence diagrams are the use cases, where each use case has been described with its impact on the classes, to illustrate how the classes collaborate to perform the use case inside the system. When modeling the sequence diagrams new operations are usually found that can be added to the various classes already outlined at this stage. The goals of this analysis are to achieve a communication between the end user and create an understanding of the system being built; it is not a detailed design solution.

The sequence diagrams are not as abstract as the class diagrams. At this stage of the analysis phase we aim to capture more details of the system.

The rectangles across the top of the diagram represent classifiers or their instances i.e. typically actors, use cases, classes, user interfaces etc. The vertical dashed lines directly under the classifiers are called lifelines which represent the life span of the classifier during the use-case being described.

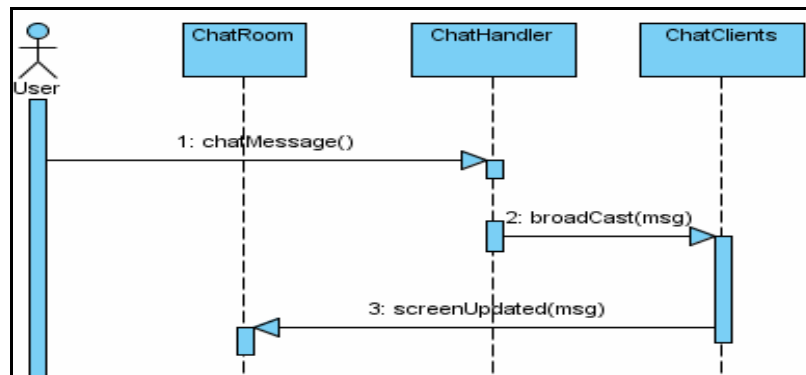


**Figure 3.9 Sequence Diagram for Use Case Log In [source: author using Visual Paradigm]**

We have now captured a much greater level of detail about the system being developed. Figure 3.9 details the full life cycle of a typical successful user log in to the system.

- The user first submits their login credentials to the chat server
- The chat server checks these details on the database
- A result is returned back to the chat server and the user. On Success this result will be a Boolean of true and a personalized profile
- The user is now presented with the main chat screen

When modeling the sequence diagram, it is important to show a user interface as part of the system. In the analysis, it is sufficient to be aware that interface windows are needed and to identify the basic interfaces. Windows that can be identified are for logging in, sending a message, a window for maintaining users is also necessary. The user interfaces are not specified at this point it's to show what they are and how they will be used in the context of the whole system.



**Figure 3.10 Sequence Diagram for Use Case Send Message [source: author using Visual Paradigm]**

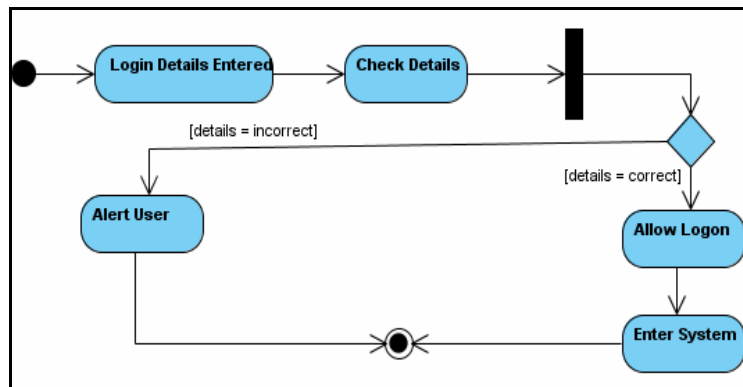
Figure 3.10 presents the interaction between the different system components for the send message use case.

- The user can post a message to the chat lobby
- This message is taken care of by the chat handler, which subsequently broadcasts this message to all clients who have a connection to the server
- Each clients screen is now updated and all screens are synchronized

### 3.4.4 Activity Diagrams

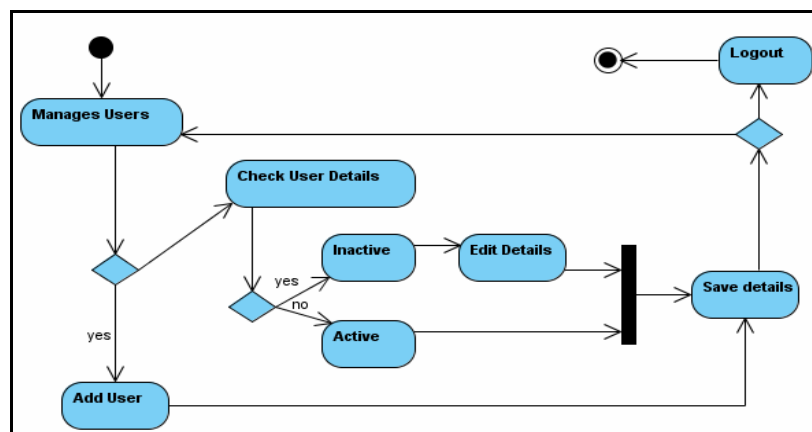
An activity diagram shows a sequential flow of activities, as shown in figure 3.11. The activity diagram is a cousin of a state diagram and has a slightly different purpose, which is to capture actions. The states in the activity diagram transition to the next stage directly when the action in the state is performed.

The rectangular boxes with round corners represent the activity performed in the system. The control flow lines show the order in which the activities are carried out. Synchronization bars, in an activity diagrams indicate that that all activities prior to the bar must complete before the flow of the activities continues. Diamonds represents decisions that the system may have to make e.g. are password valid, yes or no.



**Figure 3.11 Activity Diagram for User [source: author using Visual Paradigm]**

The above diagram shows the process involved in logging into the system. The user must enter their log in details, which are verified with the data stored for that user in the database. If they are invalid an appropriate message is displayed to the user notifying the user of the next steps to take. If the details are valid then the user is allowed to proceed in entering the main chat area.



**Figure 3.12 Activity Diagram for a System Administrator [source: author using Visual Paradigm]**

In the activity diagram in figure 3.12, the orders of activities are described for the system administrator for managing the users of the system. On log in the system administrator can add a new user or edit a current users details. After saving any changes the system administrator may choose to amend more details or log out.

### **3.5 Design**

The design phase and the resulting UML model expands and details the analysis model by taking into account all technical implications and restrictions. The purpose of the design is to specify a working solution that can be easily translated into programming code. The classes defined in the analysis are detailed, and new classes are added to handle technical areas such as a database, a user interface, a communication, devices, and more.

The design phase of the systems development life cycle involves using the findings uncovered during the analyses phase to create a strategy for developing the system. During the design phase, the outlines and illustrations of how to implement the project goals are documented.

The design can be divided into two segments:

- **Architectural Design:** This is the high-level design where the packages are defined, including the dependencies and primary communication mechanisms between the packages. Naturally, a clear and simple architecture is the goal, where the dependencies are few and bi-directional dependencies are avoided if at all possible.
- **Detailed Design:** This part details the contents in the packages, so that all classes are described in enough detail to give clear specifications to the programmer who will code the class. Dynamic models from the UML are used to demonstrate how objects of the classes behave in specific situations.

#### **3.5.1 Architectural Design**

A well defined architecture is the foundation for an extensible and changeable system. The packages can concern either handling of a specific functional area or a specific technical area. It is vital to separate logic (the domain classes discussed earlier in this chapter) from the technical logic so that changes in either of these segments can be done easily with out too much impact on the other part. Key issues to address when defining the architecture are to identify and set up rules for dependencies are created between packages, and so identify the need for standard libraries and to find libraries to use.

The packages, or subsystems, or layers in the system are:

**User interface package:** Classes for the entire user interface, to enable the user to view the data from the system and to enter new data. These classes are based on the Java AWT package and the Java Swing package, which are standard libraries in Java for writing user-interface applications. This package co-operates with the business objects package, which contains the classes where the data is actually stored. The UI package calls operations on the business objects to retrieve and insert data into them.

**Business objects package:** This includes the domain classes from the analysis model such as ChatServer, ChatHandler, User, DatabaseHelper classes etc. These classes are detailed in the design so that their operations are completely defined, and support for persistence is added. The business object package co-operates with the database package in all that business object classes must inherit from the persistent class in the database package.

**Database package:** The database package supplies services to other classes in the business object package so that they can be stored persistently.

**Utility package:** The utility package contains services that are used in other packages in the system. It is used to refer to persistent objects throughout the system and is used in the user-interface, business objects, and database packages. Below I present a class diagram showing an architectural overview with the application packages and their dependencies.

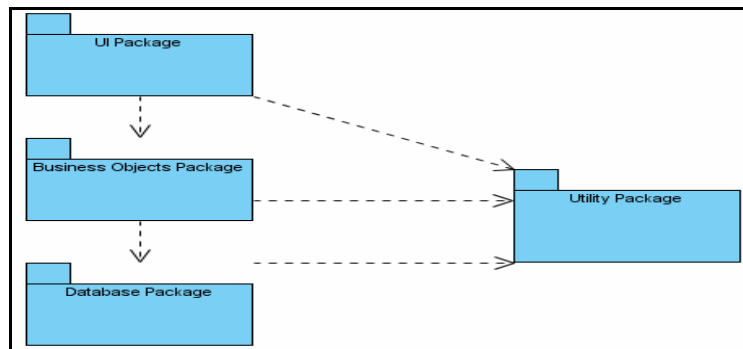


Figure 3.13 Architectural Class Diagram [source: author using Visual Paradigm]

### 3.5.2 Detailed Design

The purpose of the detailed design is to describe the new technical classes – classes in the user-interface and database packages – and to expand and detail the descriptions of the business object classes, which already have been sketched in the analysis stage. Creating new class diagrams, state diagrams, and dynamic diagrams (such as sequence, collaboration, and activity diagrams) do this. These are the same diagrams that are used in the analysis, but here,

they are defined on a more detailed and technical level. The user-case descriptions from the analysis are used to verify that the use-cases are handled in the design, and sequence diagrams are used to illustrate how each use case is technically realized in the system.

## Database Package

The application must have objects stored persistently; therefore a database layer must be added to provide this service. Details about the storage are hidden from the specification, which has to call only common operations such as connect(), store(), update(), delete() and so on for the objects. All the implementation of the persistent storage handling is done in a class called DatabaseHelper, which all classes that need to query the database must reference.

## Business Objects Package

The business objects package in the design is based on the corresponding package in the analysis, the domain classes. The classes, their relationships, and behavior are preserved, only the classes are described in more detail, including how their relationships and behavior are implemented. The operations from the analysis have been detailed, which means some of them have been translated into several operations in the design model and some have changed names. All operations in the design model must have well-defined signatures and return values.

The state diagrams from the analysis are also detailed in the design, showing how the states are represented and handled in the working system. The states are implemented in the design by using a vector called userlist, which contains a list of objects of type User. When the vector has a size of zero elements (i.e. it is empty), the User is in the state 'Logged Out', when the size is one or more, the state for the users logged in is 'Logged In'. When the userlist size is one or more it means that a user can log in and chat to another user. The userlist can increment and decrement via instances of the user class calling the operations signIn() and signOut().

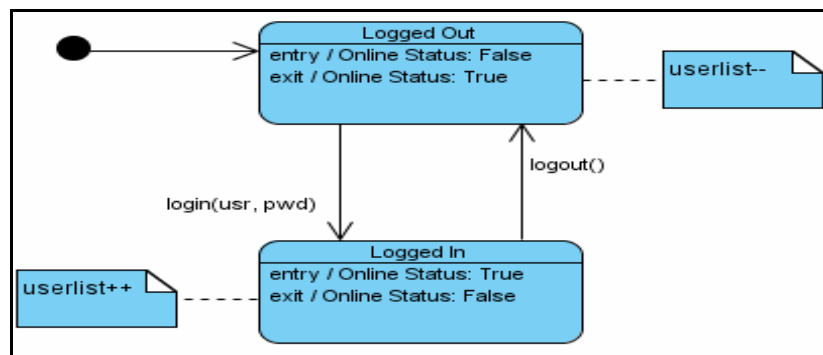


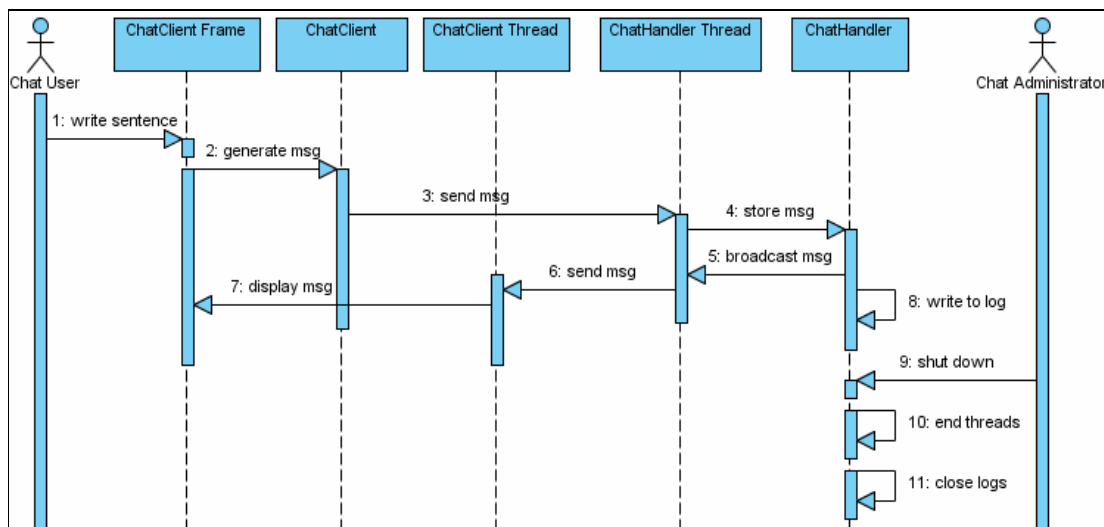
Figure 3.14 State Diagram vector userlist [source: author using Visual Paradigm]

## User interface Package

The user interface package is on top of other packages. It presents the services and information about them to a user. As stated already, this package is based on the standard Java SWT class library and Java Swing class library for writing user-interface applications in Java, and can be executed on all Java platforms.

The dynamic models in the design model have been allocated to the GUI package, since all interactions with the user are initiated through the user interface. Again, I have chosen sequence diagrams to show the dynamic models. The basis for the sequence diagram is like that in the analysis of the use case, except that the realizations of the use cases are shown in exact in the design model (including the actual operations on the classes, the analysis of the interaction was more of an overview). The sequence diagrams are not drawn on a once of basis, but rather they are drawn in iterations, whereby the final design is slowly generated. Other modifications in the sequence diagrams are generated by discoveries in the implementation phase. The operations and signatures appear exactly as they appear in the code.

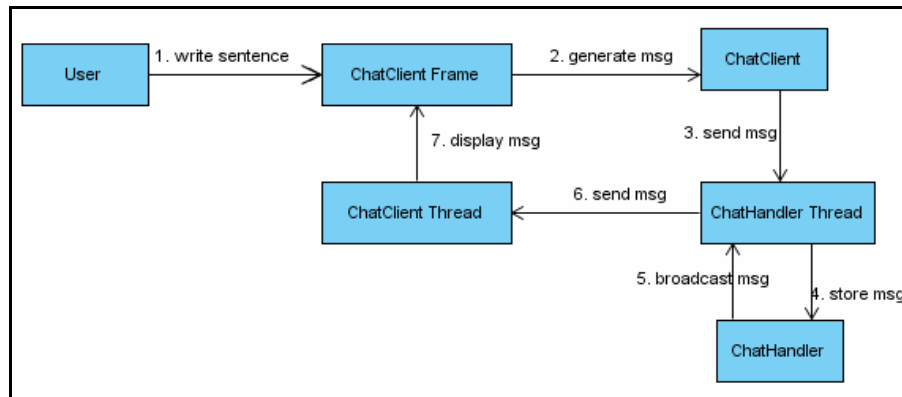
In figure 3.15 we can now see a more detailed sequence diagram for the user case 'send message'. Having gained enough information at this stage through the requirements, analysis, and design phases we could amend figure 3.10 accordingly to produce figure 3.15. It can now clearly be seen how the sequence of events must occur in order for one to send a message to the chat room.



**Figure 3.15 Detailed Sequence Diagram for Use Case Send Message [source: author using Visual Paradigm]**

Collaboration diagrams can be used as an alternative to sequence diagrams, but they still have the same idea, i.e. a sequence of steps which all must be completed in order for the use case to return a completed function to the actor. Collaboration diagrams are generally considered the easier alternative to sequence diagrams. Figure 3.16 essentially is just a different way of representing what is already modeled in figure 3.15.





**Figure 3.16 Detailed Collaboration Diagram for Use Case Send Message [source: author using Visual Paradigm].**

### 3.5.3 User-Interface Design

A special activity carried out during the design phase is the creation of the user interface, i.e. its look and feel. Initiated during the analysis phase, this is done separately, but in parallel to the other design work. The user interface in this application is based on the use cases, and has been divided into the following sections, each of which has been given a separate menu in the main window menu.

**Functions:** A window(s) for the primary functions of the application, i.e. logging in, typing a message, sending a message, controlling the synthesizer, logging out, synthesizing the buddy list, etc.

**Information:** A window for viewing the information in the system, buddy list, conversations, etc.

**Maintenance:** A window for maintaining the system that is, adding, updating and removing users from the system.

In the user interface environment, it is easy to add event handlers for user generated events such as mouse clicks and menu choices, by selecting specific component such as the `sendButton` and then selecting an event on that button that needs to be handled such as the `clicked` event. The resulting application user interface will be composed of a log in window, main chat window, and system administration window. The login window and the admin window will be of a simple nature. The login window will have a couple of text fields for the user to enter their username and password. There will be buttons called login, close and reset. When the user clicks the login button, their details will be submitted across the network to the server. This is also where the backend database will reside. The server in turn will query the database and then return a success or failed result to the user.

On success, an instance of the main chat class will be created. This will be the main chat area of the system where the user can:

- view a personalized profile for their account
- view the most up to date buddy list
- customize the speech synthesizer for their needs
- send messages
- view conversations etc

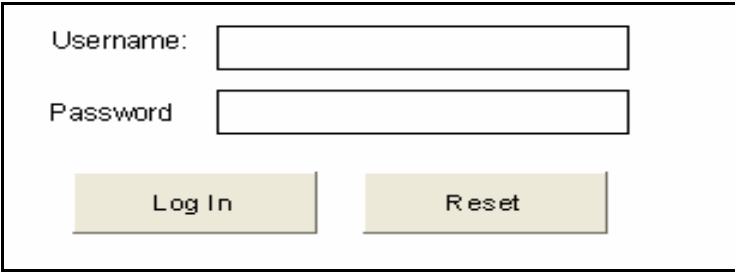
The admin window will be composed of options to add a user, edit a current user, and remove a user.

### 3.5.4 Prototype Screenshots of System Interface

The user interface is often the most complex and important part of any system. It can take over half of all development effort and is the part that is most likely to cause users to perceive a lack of quality (Lethbridge, 2001). It is therefore important to draft mock up GUIs from early on. Since the implementation stage has not yet begun, these screen shots are just a rough guide as to what the interfaces for the system will resemble. These screen shots were generated using Microsoft PowerPoint. As already mentioned in section 3.4.3, the windows that can be identified are for logging in, sending a message, and a window for maintaining users.

#### 3.5.4.1 Login Screen

The login screen in figure 3.17 as presented below is an initial impression of what the login screen for the system may look like. As coding begins at the implementation stage it may look different.



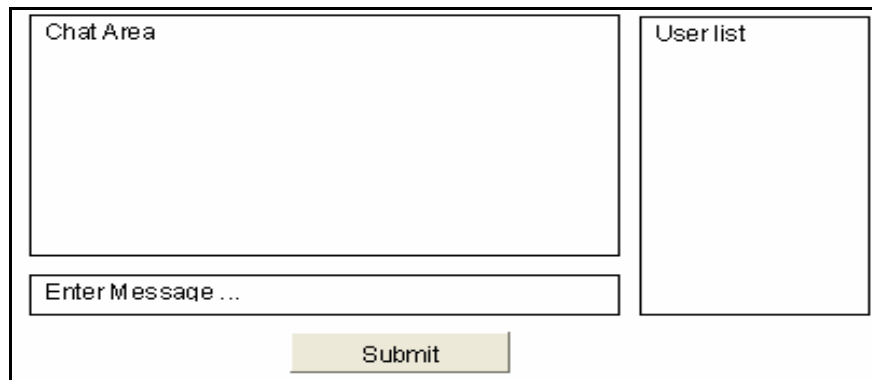
Username:

Password

**Figure 3.17 Initial Login Screen**

### 3.5.4.2 Main Chat Area Screen

This screen will no doubt be redesigned a few times in order to cater for all functionality in the system. When the implementation begins certain add-on features to be added to this screen as the system evolves. Also, feedback from the user will result in certain additional functionality becoming available. However, an initial screen show could look like the screen in figure 3.18.

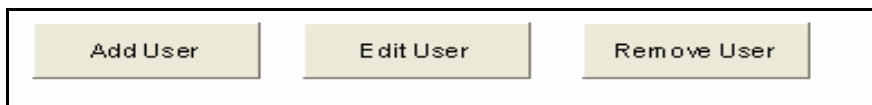


The diagram shows a rectangular window divided into two main sections. The left section is a large text area labeled 'Chat Area' at the top left. Below this text area is a text input field with the placeholder text 'Enter Message ...'. To the right of the 'Chat Area' is a vertical rectangular area labeled 'User list'. Below the 'Enter Message ...' field is a yellow button with the text 'Submit'.

**Figure 3.18 Initial Main Chat Screen**

### 3.5.4.3 System Administrator Screen

The admin screen would include buttons to add, remove and edit a current user.



The diagram shows a horizontal rectangular window containing three yellow buttons arranged side-by-side. The buttons are labeled 'Add User', 'Edit User', and 'Remove User' from left to right.

**Figure 3.19 Initial Admin Screen**

The clicking of any of these buttons would refresh this screen, hide these buttons and set the visibility of text fields figure 3.19 in order to insert, edit, or remove certain data from the database.

Forename	<input type="text" value="Ciaran"/>
Surname	<input type="text" value="Reidy"/>
Active	<input type="text" value="1"/>
<input type="button" value="Update"/>	

**Figure 3.20 Initial Admin Screen**

### ***3.6 Justification for tools and technologies***

#### **3.6.1 TCP / IP**

TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. This was the communications protocol used to handle the communication process in this system as it can be used in a private network (either an intranet or an extranet). Once a user has access to the Internet, their computer is provided with a copy of the TCP/IP program just as every other computer that you may send messages to or get information from also has a copy of TCP/IP. This meant no additional configuration would be needed in order to allow users to start communicating.

#### **3.6.2 Java**

Java was chosen as the language to implement the solution as the author has previous experience with Java, it maps well with domain classes, it is open source, and it should communicate with relative ease with the speech synthesizer (FreeTTS) as it was also coded using the Java programming language. Java makes mapping the logical classes to the code components easy, because there is a one-to-one mapping of a class to a Java code file (.java).

#### **3.6.3 MySQL**

The requirements specify that the system be able to run on a number of different machines, so Java was chosen as the Object-oriented programming language and the database chosen was MySQL; because both are open source. Also MySQL was designed to easily interact with Java.

### ***3.7 Conclusions***

In this chapter the system design was described in an analysis model with use cases in order to capture the requirements of the end user of the system. The next stage in the analysis and design stage was to identify the objects in the system which was obtained by reviewing the use cases and then deciding which concepts should be handled by the system. The domain classes modeled were more detailed than the use cases and thus were not as abstract as the use cases. This is the normal procedure in designing a system whereby the level of abstraction decreases as the design proceeds. Following the domain class analysis, state and activity diagrams were used to capture the life cycles of the objects identified in the class diagrams. Finally in the domain analysis, sequence diagrams and collaboration diagrams were used in order to describe the dynamic behavior of the domain class. The post analysis or design stage saw the already defined classes in the analysis model amended as more detail was added, and new classes added to handle technical areas such as a database, a user interface, and more. The end of the design stage specified a working solution that can now be easily translated into programming code which will be outlined in the next chapter.

## **4. Development**

The construction or implementation phase is when the classes are programmed. As already mentioned in section 3.6 Java will be the language of choice in order to implement this speech enabled chat application. This chapter will correlate with chapter 3 where the design will be transformed into a functionally working application.

### ***4.1 Introduction***

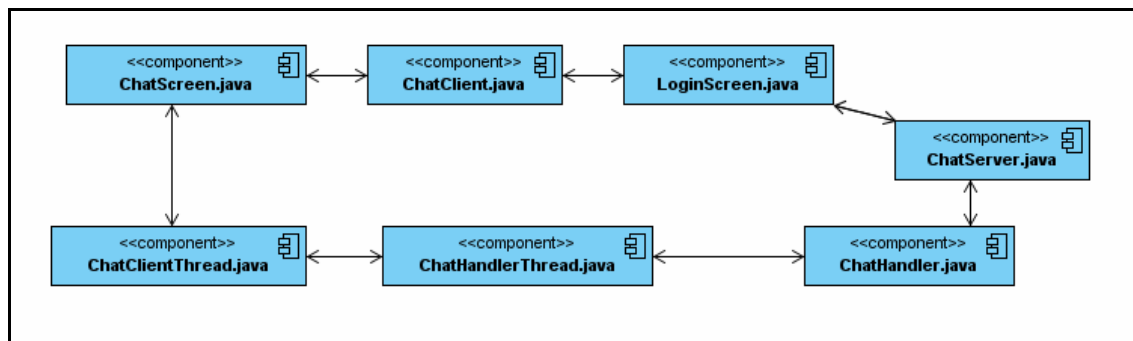
In the next section we will see how each part of the design phase was needed in order to create a smoother transition from the design to the implementation. Also in this section the idea of speech enabled technologies and how they will be integrated into this chat system will be introduced.

The main section of this chapter, section 4.5 will discuss the development process of how the design and analysis models map to the implementation classes. Such mappings are not always exact, and at times the design stage acts as more of an outline just to safe guard from designing an incorrect system. With this in mind, it is not uncommon to tweak certain classes in order to implement a required functionality.

In section 4.6 we will see how the initial use cases, as discussed in section 3.3.3, and the prototype screen shots, as discussed in section 3.5.4, have evolved into functionally working graphical user interfaces that allows the user communicate with the system.

In section 4.7, additional functionality outlines what added on features were added to the system after the development phase was completed. Such functionality was mainly geared towards the visually impaired. Finally in section 4.8, the idea of error detection and exception handling is introduced. This section uncovers the importance of allocating sufficient time so that all possible code defects are avoided and detected. The defects which can't be avoided will be handled accordingly so that system will continue to operate.

Below illustrates that the component diagrams in the design model contain a simple mapping of the classes in the logical to components in the component view. It identifies the components that implement the domain classes. The associations of these classes are bi-directional; therefore dependencies go in both directions between the classes involved in the association.



**Figure 4.1 Component Diagram for Chat System [source: author using Visual Paradigm]**

From above it can be seen that all the components have bi-directional dependencies. For example, when the user first logs in there is no thread involved. The user's login details go straight to the server. The response from the server goes straight back to the user. Assuming the login details are valid, the ChatServer then creates an instance of the ChatHandler class and passes details from the ChatUser class through the ChatHandler's constructor. The details passed through as parameters are the user's username and an instance of the socket created.

Once a successful result is passed back to the ChatUser class, an instance of the ChatClientFrame class is created. This is the main interface of the system. Once the main interface frame is created a separate client thread is also created which in turn communicates and has a link to the ChatHandler class via the ChatHandlerThread.

## 4.2 Methodology

According to an article submitted by Scott Berinato on CIO.com, almost three quarters of software projects have fallen ill to one of the following: total failure, cost overruns, time overruns, or an end product without all the requirements as outlined in the design spec (Berinato, 2001). This is mainly due to software projects not being managed adequately. Good software project management requires a particular methodology to be followed.

With regard to software development, there are many different definitions for the term methodology. Jayaswal in his book 'Software Development Methodology Today' defines a software development model as

“an organized strategy for carrying out the steps in the life cycle of a software application program or system in a predictable, efficient, and repeatable way”

Every software project needs a methodology which outlines the process of developing the software. A software process or methodology is needed in order to improve the quality of the software produced. If the correct methodical approach to software development is used then fewer defects, shorter delivery, and better value can be achieved (Hariska, 2007). The type of

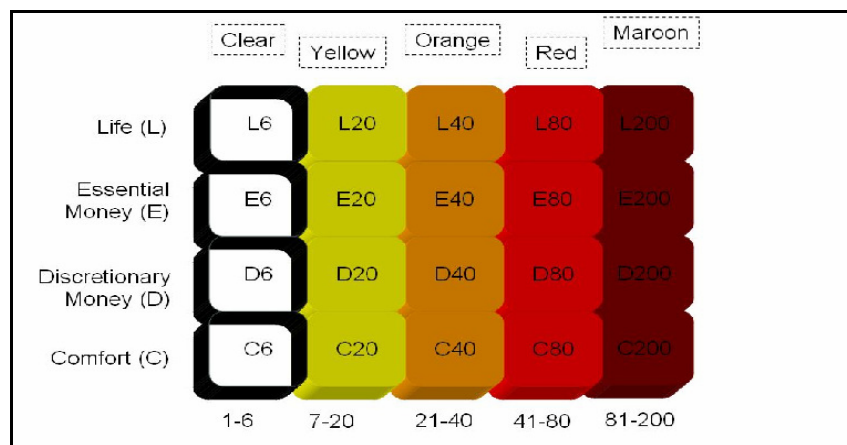
project, its criticality, and the size of the team involved in the project all determine the type of methodology used in developing the project.

There are numerous ways to develop software. Many approaches to software development include the classical traditional methods such as the Waterfall model, Rational Unified Process (RUP), Spiral, and Evolutionary model (Bednar & Robertson, 2005). Such approaches to software development involve requirements gathering, software design, software development, testing, and the delivery processes. Such approaches are also referred to as plan-driven methods that escalate from a sound foundation composted of documents outlining the requirements (Williams, 2007). From here the business analysts and software engineers have a strong plat-form to work from as they begin to design and implement the application. These traditional methodologies have been around since the 1970s (Nelson, 2007).

It's only in the last few years that Agile Methodologies have replaced some of the more classical approaches. There different from previous methodologies in that they put individuals and face to face interactions over processes and tool, and put the idea of changes over a set plan (Williams, 2007). Researchers in the new area of methodological work believe that the idea of heavy documentation and strict plans are no longer appropriate for many software applications (Lyons, 2004).

## Crystal Clear

The Crystal Methodology was created by Alistair Cockburn in the 1990s as an alternative approach to traditional methodologies. The Crystal methodology is one of the agile methodologies that favor flexibility over rigorous processes and face-to-face interaction over written documents. Cockburn created a family of Crystal methodologies because he believed that different projects with regard to team size and amount of risk involved required different methodologies (Wolak, 2001). The different Crystal methodologies are divided into color-coded brands of Clear, Yellow, Orange, Red, and Maroon as can be seen below in figure 4.2





**Figure 4.2 Crystal Family Methodologies (A Practical Guide to Seven Agile Methodologies, Part 2, [www.devx.com](http://www.devx.com))**

From the graph above, Cockburn used the X-axis to represent the number on the team and the Y-axis to represent the amount of risk involved in the project. As the number of team members working on a project increases, the need for documentation to co-ordinate the project also increases. The more documentation needed to co-ordinate a project means less face-to-face contact with the end user is needed. Each color brand has its own rules and basic elements. The lightest of the methodologies is crystal clear which is used for small projects of up to 4-6 people.

The methodology used in this thesis was Crystal Clear as the work was done solely by one person, i.e. the author, and the project was not critical. This is the most suited methodology for this thesis as it allowed maximum individual preference in the design and implementation of the application. With just one developer on the application (the author) and the lowest damage impact being loss of comfort this methodology is represented as C6 on the diagram in figure 4.2. In accordance with Crystal Clear, the author of this thesis has defined roles of sponsor, senior designer, programmer, user, and tester (Dubinsky & Hazzan, 2004).

### ***4.3 Development Requirements***

The design stage plays a significant part in the development process of any software application. In order to develop and code this application, the specifications were fetched from the following diagrams in the design model.

The class diagrams in the analysis and design stage show its static structure and its relationship to other classes in its environment. The specifications of each class show in detail the necessary attributes and operations that are required. Of course in the development phase these attributes and operations will change accordingly as the project evolves.

The state diagrams for the various classes showed the possible states that an object could be in at any one time and the transitions between these states that need to be handled.

The dynamic diagrams (sequence, collaboration, activity) in which objects of the class are involved show the implementation of a specific method within the class. They also show how other objects are using objects of the class. These diagrams are more closely related to code specifics as they are not as abstract as the class diagrams. From an implementation perspective these diagrams depict the parameters passed between the different objects which make up the system.

Use-Case Diagrams and specifications: Diagrams that show the result of the system are used when the developer needs more information regarding how the system will be used.

Naturally, deficiencies will be uncovered from the design model during the coding phase and the need for new or modified operations may be required, meaning that such changes will also need to be made to the design model. This happens in all projects and it is natural to expect these change requirements. It is important that the design model and the code are synchronized so that the model can be used as final documentation of the system.

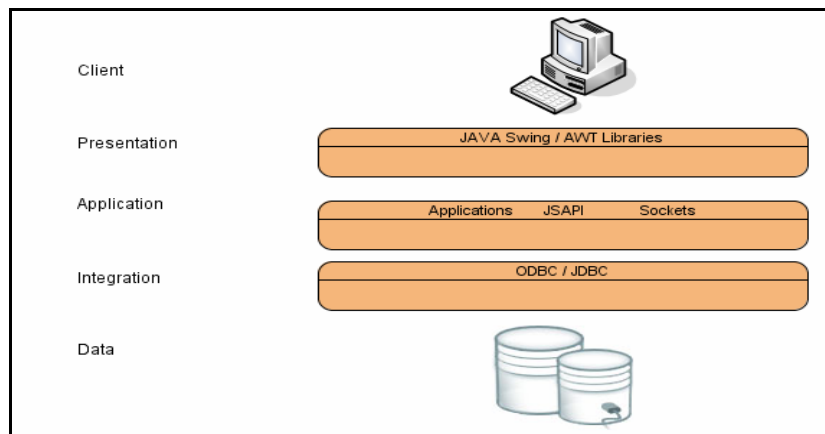
## ***4.4 Development Process***

In this section the system architecture and the tools used will be discussed. In section 4.4.1 the system architecture and the reasons why this type of architecture was used is outlined. In section 4.4.2 the idea of speech technology, speech API, and the background of speech synthesis is introduced. Finally in section 4.4.3 a brief note on each of the technologies used in developing this system is given and the reasons why using these technologies would be best in creating this system.

### **4.4.1 n-tier Architecture**

The architecture used in developing any system plays a crucial role in determining the success or failure of the application being developed. Robustness, scalability, performance, and future incremental releases are all very important elements which could determine the type of system architecture which is used.

In this section the concept of n-tier architecture will be discussed and how the type of architecture adopted could have a significant impact on the system's performance both now and for future releases. An n-tier architectural system typically has 3 or more layers or parts to the application. From the diagram below we can see how each layer rests on the layer just below it. For example, the user interface would have very little, if any, interaction with the database layer. This will be demonstrated later on in this chapter with specific code walk through examples. The physical actors in the system are the client / computer and the database.



**Figure 4.3 n-Tier architecture [source: author using Visual Paradigm]**

Between these two there are three different layers. The layer nearest the database is called the integration layer or data access layer (DAL). This layer deals with connections to the database, uses stored procedures to return results from the database to the layer above it, the application layer. The application layer is also sometimes referred to as the business logic (BL) layer. This layer handles all the business logic within the application, the multi-threading involved in a chat environment, the socket programming for connection clients / servers, and calls to the speech engines via the JSAPI. After the application layer handles the business logic, given the data from the integration layer, it then hands the results to the presentation layer which is then presented to the user via graphical user interfaces. The Java Swing / AWT libraries provide rich classes to do this. This architecture allows the decoupling of the interfaces from the implementation. This is the essence of object-oriented programming whereby a change in one section of the code will be isolated and have little or no affect across the application (Buhler, Starr, Schroder, Vidal, 2004).

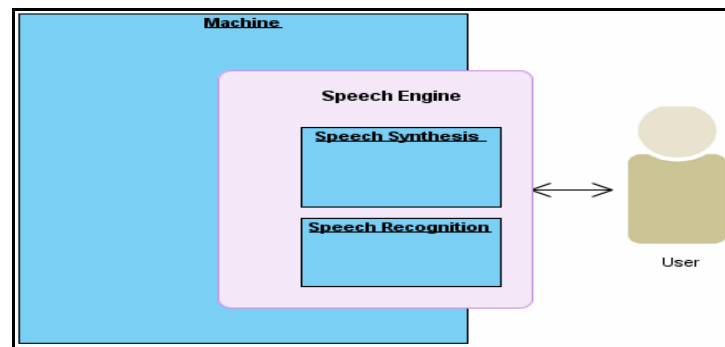
#### **4.4.2 Speech Technology**

In the design stage the chat area of the application was focused mainly upon whereas in this chapter the speech system and how it will link into the main chat system will be investigated. Speech synthesis is the process of generating speech from text. Speech applications have evolved and refined speech technology, making it now feasible to develop applications that use speech technology to enhance the user's experience.



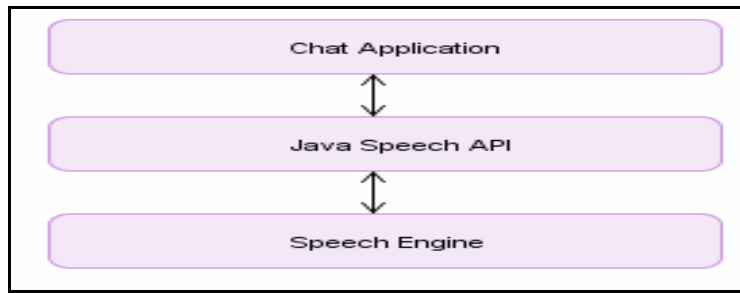
**Figure 4.4 Speech enabled technology**

A speech-enabled application does not directly interact with the audio hardware such as the microphone, speakers, and headphones. The Speech Engine, be it speech synthesis engine or speech recognition engine, provides speech capability and provides a smooth communication process between the speech enabled application and the audio hardware.



**Figure 4.5 Speech engine [source: author using Visual Paradigm]**

Similarly, Speech Application Programming Interfaces (APIs) allow applications communicate in a bi-directional process with the speech engine implementations. This allows the complicated speech code to be hidden from the developer so that the developer can spend their time focusing on implementing the project (Neumeyer, Franco, 1998). The Java Speech API (JSAPI) defines a standard cross-platform software interface to the latest speech technology. Two core speech technologies are supported through the Java Speech API: speech recognition and speech synthesis (Teppo and Vuorimaa, 2001).



**Figure 4.6 Java Speech Stack [source: author using Visual Paradigm]**

The Java Speech Application Programming Interface (JSAPI) is used as the gateway or interface to the world of speech enabled technologies. Like any speech synthesis application, it was important to use a JSAPI because it allowed the cross-platform development of such applications. It meant that it was not dependent on one platform or one specific implementation of a speech engine. Since the implementation was already developed by other vendors, the JSAPI allowed the use of speech technology in the application with as little as 10 lines of code.

Decoupling of the speech engine from the main chat application was important as this also meant that more than one voice could be used with this application. For this application one voice was enough to accommodate the functional requirements set from the offset, but for future releases of this application coupled with advancements in speech technology it was imperative to design a system which could cater for multiple voices. There are currently three different voices (Kevin, kevin16, Alan) that are packaged with FreeTTS , and it is possible to import other voices from Festfox and Mbrola (FreeTTS).

For this research the FreeTTS speech engine implementation was used. The reasons for this were that it is open source and written entirely in Java as is the rest of this chat application. This means that it is free and will work on Windows, Linux and Mac, as well as any other operating system that Java works on. The only real requirement is that you are using a least JDK 1.4.

The Java Speech API 1.0 was first released by Sun in 1998 (ITB Journal) and defines packages for both speech recognition and speech synthesis.

### **4.4.3 Justification for Tools and Technology**

#### **MySQL**

The database chosen to develop this system was MySQL. This was due to a number of different reasons, no least because it is open source and through the use of a Java DataBase Connectivity (JDBC) driver it maps well with Java. The version of MySQL used was 5.0.

JDBC is a Java API which is used to manipulate relational databases such as MySQL. The set of core classes that form the JDBC API can be found in the `java.sql` package. The JDBC driver used was MySQL Connector/J 3.0.17-ga.

## **Java**

Java was the programming language of choice as it is also open source which can work across multiple platforms. The version of Java used in the development of this application was Java JDK 6 Update 6 (Version: 1.6.0\_06-b02). It has also be shown (Julia,Neumeyer, YEAR) that multi-threading capabilities of Java are used to improve the speed and the efficiency in communication applications such as the one in this thesis.

## **Edit Plus**

The software tool used to write the code for this application was EditPlus v3.01. EditPlus is a programmer's editor for windows. It is more than a text editor, as it allows syntax highlighting for many different scripting, markup, and programming languages. It allows the rapid development of small applications as it provides full visibility to multiple classes concurrently. Unlike IDEs it's a lightweight program ideal for developing systems with a small number of modules. Furthermore, using a light weight editor as opposed to IDEs meant that very little hardware space and RAM was necessary to run the program.

## **Speech Technology**

There are a number of speech synthesis tools on the market today. However most of them are either of a commercial nature (TextAloud, SVTTS, JAWS) or specifically designed (WeatherAloud) and thus are not adaptable for my particular application. The Festival speech synthesis software is open source but since it is developed in C++ it wouldn't have been as easily integrated into my system (Festival). Also, the JSAPI is not part of the Java process itself, rather it talks to a Festival sever elsewhere (Festival - JSAPI). Having researched the possible speech technologies available, it was decided to use the FreeTTS speech synthesizer with this system. FreeTTS provides loosely coupled rich speech APIs, it is open-source, and it was written in the Java programming language, making it easily compatible with my application. The FreeTTS version used in the implementation of this application was freetts-1.2.1.

### **4.4.4 Environment Configuration**

In order for this application to run the following downloads, installations, and configurations need to be addressed.

#### 4.4.4.1 Java

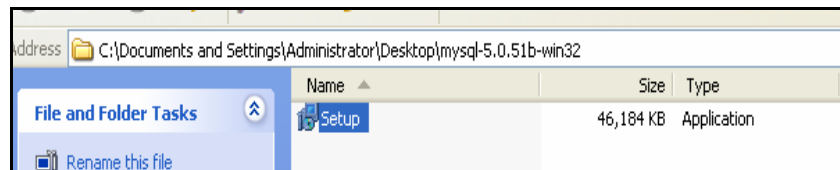
To compile and run a java program we need to install java platform. JDK (Java Development Kit). JDK is the basic set of tools required to compile and run java programs. This section enables the JDK to be downloaded and teaches you the steps to install it. The latest version of jdk is 6 (update 1) at the time of writing this thesis. The latest version of jdk can be downloaded from the following website;

<http://java.sun.com/javase/downloads/index.jsp>

Once the .exe file is downloaded it can be installed it by double clicking on the downloaded exe file (jdk-6u1-windows-i586-p.exe) and just following the screen shots as instructed.

#### 4.4.4.2 MySQL

MySQL is downloaded in a similar way to Java. The executable can be downloaded from <http://dev.mysql.com/downloads/mysql/5.0.html>. Once downloaded MySQL 5.0 can in installed by double clicking the executable as can be seen below.





**Figure 4.7 Installing MySQL 5.0**

Once Java and MySQL are running, we need a bridge to allow these two technologies talk to one another. This is achieved by downloading the JDBC driver which allows applications use MySQL. The JDBC driver used in this application was mysql-connector-java-3.0.17-ga and it can be downloaded from <http://downloads.mysql.com/archives.php?p=mysql-connector-java-3.0&o=other>

#### 4.4.4.3 FreeTTS

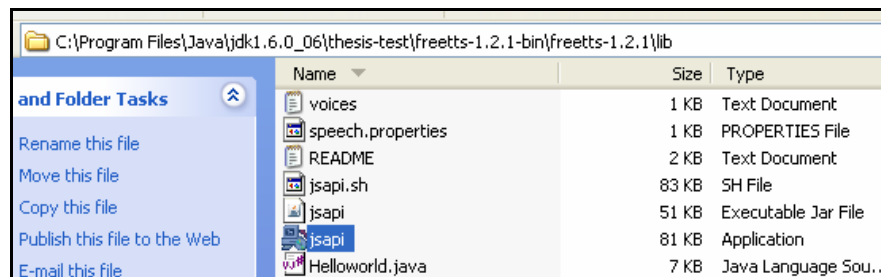
The FreeTTS speech synthesizer used in this application can be downloaded from this website

[http://sourceforge.net/project/showfiles.php?group\\_id=42080&package\\_id=34183&release\\_id=311626](http://sourceforge.net/project/showfiles.php?group_id=42080&package_id=34183&release_id=311626) . As can be seen in figure 4.8, there are three options to choose from. The option needed for this application was freetts-1.2.1-bin.zip. Since this application only wanted to use FreeTTS with this application the bin package was sufficient. If modifications need to be made with FreeTTS then the src package is required. As its open source this is permitted.

Latest	<a href="#">FreeTTS 1.2.1</a> 	(2005-03-10 00:24)	
	<a href="#">freetts-1.2.1-bin.zip</a> 	13136956	Platform-Independent
	<a href="#">freetts-1.2.1-src.zip</a> 	14100414	Platform-Independent
	<a href="#">freetts-1.2.1-tst.zip</a> 	3927166	Platform-Independent

**Figure 4.8 Downloading FreeTTS 1.2.1-bin.zip**

Once the FreeTTS has completed downloading, these file can be extracted anywhere. In order for FreeTTS to be used in any application the Java Speech API (JSAPI) environment needs to be set up. This can be achieved by double clicking on the jsapi icon in the lib directory of freetts-1.2.1 as seen in figure 4.9.



Name	Size	Type
voices	1 KB	Text Document
speech.properties	1 KB	PROPERTIES File
README	2 KB	Text Document
jsapi.sh	83 KB	SH File
jsapi	51 KB	Executable Jar File
jsapi	81 KB	Application
Helloworld.java	7 KB	Java Language Sou...

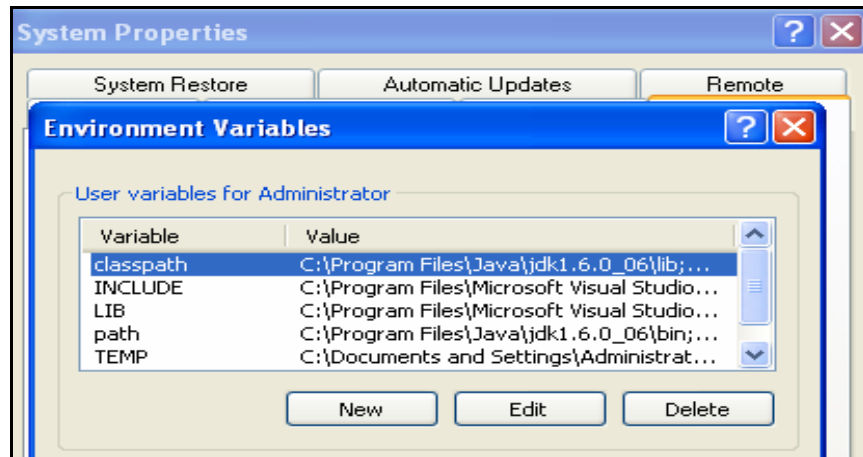
**Figure 4.9 Downloading FreeTTS 1.2.1-bin.zip**

Finally the speech.properties file needs to be copied to the home directory before running the demos that use JSAPI. The speech.properties file can also be gotten from freeTTS lib directory as seen in figure 4.9.

#### 4.4.4.4 Environment Settings

Once the relevant technologies and packages were downloaded and installed the relevant environment variables and class paths need to be set. First of all select the "My Computer" icon from the desktop and right click the mouse button. Now click on "system Properties" option. It provides the "System Properties" window, click the 'Advanced' tab. Then Click the "Environment Variables" button as seen in figure 4.10.





**Figure 4.10 Configuring the classpath**

The following paths need to be appended to the existing list:

- C:\Program Files\Java\jdk1.6.0\_06\lib;
- C:\Documents and Settings\Administrator\Desktop\mysql-connector-java-3.0.17-ga\mysql-connector-java-3.0.17-ga;
- C:\Program Files\Java\jdk1.6.0\_06\thesis-test\freetts-1.2.1-bin\freetts-1.2.1\lib\freetts.jar;
- C:\Program Files\Java\jdk1.6.0\_06\thesis-test\freetts-1.2.1-bin\freetts-1.2.1\lib\jsapi.jar;

## 4.5 Phases in Development Environment

In this section the various classes and methods that are implemented and invoked in order to produce a functionally working chat application are discussed. In section 4.5.2 the individual classes and the methods that make up these classes and how they are linked together will be addressed. The various interfaces that these classes implement and the classes they inherit will also be discussed.

### 4.5.1 Classes and Methods

Below we see the various classes used to implement this system. In the class diagram in section 3.4.1.2 six main classes were identified which were to be used in this application.

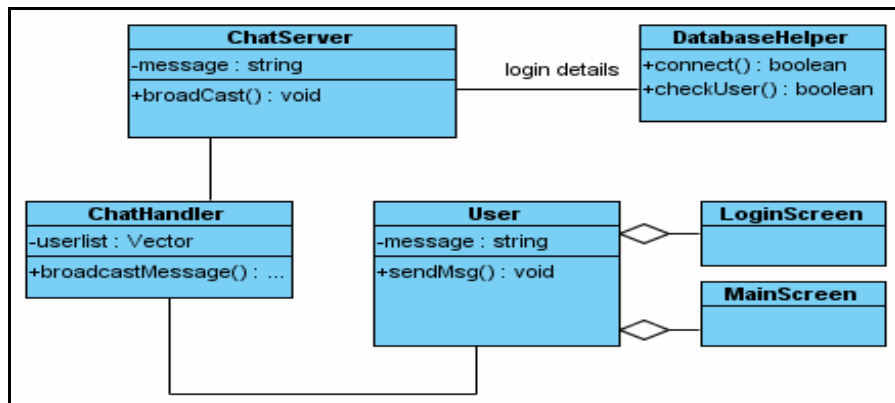
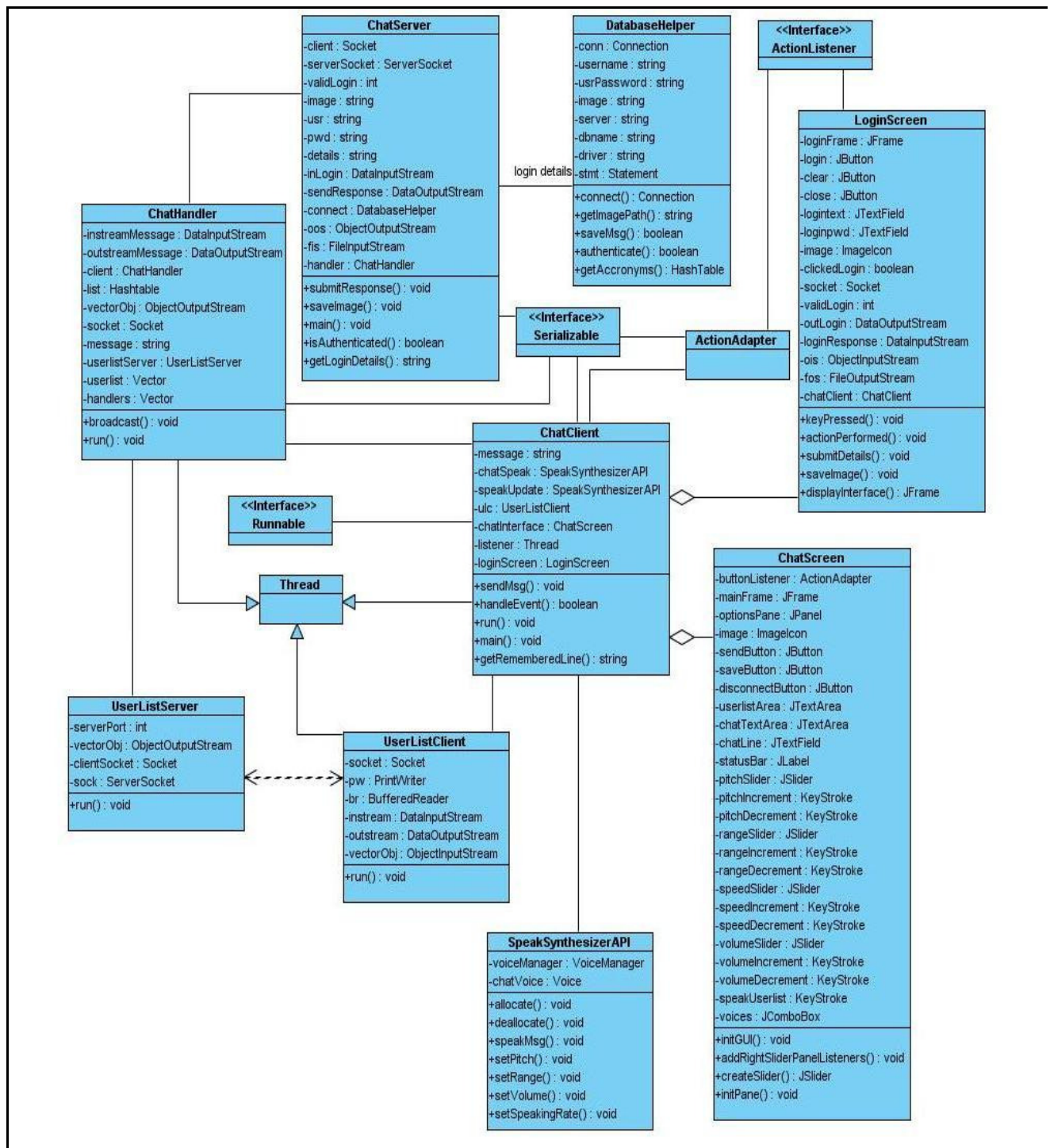


Figure 4.11 Class Diagram from Chapter 3 [source: author using Visual Paradigm]

However, since the speech technology was not integrated into the system at this early stage there was a need to alter this diagram to include the speech classes. Also, in the design stage the various methods and their parameters were not outlined. In the diagram below we can see the full application and how the various classes are interlinked. In this updated diagram, extra classes and interfaces are included which were needed in order for the application to function correctly. These classes were not identified in the earlier design stage. Other classes such as `UserListServer`, `UserListClient`, and `ChatClient` have also been integrated into the development stage. Also highlighted in figure 4.12 are the associations, generalizations / inheritance, and dependencies among the various classes and interfaces. The `ChatClient` class was added to separate the design code from the sockets and threading code so that the application could be maintained more easily.



**Figure 4.12 Detailed Development Class Diagram [source: author using Visual Paradigm]**

### 4.5.1.1 ChatServer

In this client / server architectural model a ChatServer class was developed, which, as the name suggests, acts as a server to the various clients / users who use the system. The ChatServer takes incoming connections from clients and determines whether the user is authorized to use the system or not. It creates a thread for each authorized user which the ChatHandler takes care of.

The ChatServer class uses an infinite while loop so that the server is constantly running. A ServerSocket is instantiated which constantly listens on port 9200 for TCP/IP connections from clients. Once the server accepts an incoming connection, a DataInputStream is opened to read in the client's login details, which are captured by the LoginScreen's interface. A database connection is then opened so that the client's details can be checked. A notification is sent back to the client indicating success or failure. On success, the main ChatScreen is presented to the client, and on failure the user is given the option to try again.

```
ServerSocket server = new ServerSocket (9200);
while(true)
{
    Socket client = server.accept ();
    System.out.println ("Accepted from " + client.getInetAddress ());
    String details = "";
    details = getLoginDetails(client);

    String[] splittDetails = null;
    if (details != null || !details.equalsIgnoreCase(""))
    {
        splittDetails = details.split("\\\\,");
    }
    usr = splittDetails[0].trim();
    pwd = splittDetails[1].trim();
}
```

**Figure 4.13 ChatServer accepting a connection**

Where getLoginDetails is a method created that opens a DataInputStream reader so that the users details can be read. This implementation for getLoginDetails can be seen in figure 4.14.

```
DataInputStream inLogin = new DataInputStream
    (new BufferedInputStream (client.getInputStream ()));
DataOutputStream sendResponse = new DataOutputStream
    (new BufferedOutputStream (client.getOutputStream()));
// accept both usr + pwd from client
String details = inLogin.readUTF();
```

**Figure 4.14 Method getLoginDetails**

Once the details have been read by the server, the string is split so that the user name and password can be assigned into two variables. This is required for the next stage which involves querying the database. The next screen shot outlines how the DatabaseHelper is invoked by the ChatServer class.

```

DatabaseHelper dbconnect = new DatabaseHelper(usr,pwd);
if(dbconnect.authenticate())
{
    System.out.println("This is a valid user");
    validLogin = 1;
    image = dbconnect.getImagePath().trim();
}

```

**Figure 4.15 Invoking DatabaseHelper class from ChatServer class**

Once the user's login details have been checked by the server, a response is sent to the user. The variable validLogin is set to true and the user has been successful. The LoginScreen then instantiates an instance of the ChatScreen. If variable validLogin has a value of 0 the user is given the option to try again.

```

private static void submitResponse(Socket client) throws IOException
{
    DataOutputStream sendResponse = new DataOutputStream
        (new BufferedOutputStream (client.getOutputStream()));
    sendResponse.writeInt(validLogin);
    sendResponse.flush();
}

```

**Figure 4.16 Server's submitResponse method submits success / failed response to client**

Finally, if the client has been successful, the image obtained from the database is sent back to the client. Part of the user's chat screen is populated with this image for a customizable effect. A new instance of ChatHandler is then created which takes care of all the clients. It will maintain a list of users in a vector. A new thread will also be created (c.start()) for each client, and when it received a new message from any one of the clients, it will broadcast this message to each client on the list.

```

if(validLogin == 1)
{
    sendImage(client);
    ChatHandler c = new ChatHandler (client, usr);
    c.start ();
}

```

**Figure 4.17 ChatServer sends personalized image to client and creates an instance of the ChatHandler class to be run**

Since the transmission of an image is not like transmitting strings, it proved difficult in trying to submit the image across the network initially. Trying to send images via DataOutputStream, and ObjectOutputStream has failed. With the aid from other developers on Suns forums, it was established that the image had to be first read into a FileInputStream as a series of bytes, which could then be sent as an object by an ObjectOutputStream. This is demonstrated below.

```

private static void sendImage(Socket client) throws IOException
{
    // write image path as object
    ObjectOutputStream oos = new ObjectOutputStream(client.getOutputStream());
    FileInputStream fis = new FileInputStream(image);
    byte[] buffer = new byte[fis.available()];
    fis.read(buffer);
    oos.writeObject(buffer);
}

```

**Figure 4.18** Converting image to serializable format for TCP transmission

### 4.5.1.2 DatabaseHelper

The DatabaseHelper class is used as a communication link between the application logic and the database. It separates a large amount of database connection code from the other classes, where the other classes' main functionality is not connecting to the database. It also encourages the idea of code reuse which is one of the main aspects of object-oriented programming (Garrigue, 2000). Almost every class needs to communicate with the database at some stage, whether it's the ChatServer checking user details, or the ChatClient saving conversations to the database. It is for these reasons that the database connection code should be implemented only once but invoked any number of times.

```

try
{
    Class.forName(driver).newInstance();
    Connection conn = DriverManager.getConnection(server+dbName,userName,password);
    System.out.println("Connected to the database\n");
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("select * from tblusers where username = '" + username + "'
    += and password = MD5('" + usrPassword + "')");
    while (rs.next())
    {
        image = rs.getString("image");
        return true;
    }
}

```

**Figure 4.19** Validating user with the database

The connection code from DatabaseHelper class can be seen above. This example piece of code opens a connection to the database and if the where clause of the SQL statement is met, a record from the table will be returned. This code is executed when the ChatServer class determines whether the user is authorized to use the system. Also, in this code snippet we can see where an image path is selected from the database. As already mentioned in the previous section, this is needed for customizing each user's interface with their personal picture.

The user can save a conversation in one of two different ways. The user can click the 'save' button, or hold down 'Alt', 'Shift' and 'S'. Each event will make a call to the database via the saveMsg method of the DatabaseHelper class seen below. The conversation along with the current timestamp will be inserted into the table tblmessage.

```

public boolean saveMsg(String msg)
{
    Connection conn = connect();
    int valid = 0;
    try
    {
        Statement stmt = conn.createStatement();
        valid = stmt.executeUpdate("insert into tblmessage (timestamp, message)"
            += "values (now(), '" + msg + "')");
    }
}

```

**Figure 4.20 Saving a message to the database**

Since a database connection is opened on more than one occasion, a connect method was created which will reuse the connection code each time a call to the database is issued.

### 4.5.1.3 ChatHandler

Although this class is less than 100 lines of code it handles most of the business logic that makes up this chat application. It maintains a list of socket connections, one for each client which has successfully logged into the system. It makes great use of multithreading by extending the Thread class. A thread is a separate line of control or functionality within a single process or program. Multi-threading can be implemented in Java in two ways – either by extending the Thread class or implementing the Runnable interface.

It was necessary to use multi-threading in this application, because, like all chat systems any number of users must be able to communicate concurrently (Hamza-Lup, Bot, Salomie). Multiple threads within a program all share the same data and resources which makes it easier for clients to share data in a real time communication environment. However it also makes it easier for them to cause problems for one another if say two clients are attempting to use the same data or resource at the same time (Chen, 2003).

The below code demonstrates how extra code was wrote to combat such a scenario arising, whereby two clients are attempting to submit a message at exactly the same time.

```

protected void broadcast (String message)
{
    synchronized (handlers)
    {
        Enumeration e = handlers.elements ();
        while (e.hasMoreElements ())
        {
            ChatHandler client = (ChatHandler) e.nextElement ();
            try
            {
                synchronized (client.outstreamMessage)
                {
                    client.outstreamMessage.writeUTF (message);
                }
                client.outstreamMessage.flush ();
            }
            catch (Exception ex)
            {
                client.stop();
            }
        }
    }
}

```

**Figure 4.21 Multi-threading environment depicting the ChatHandler class broadcasting a message to clients**

In Java programming each object has a lock which can be obtained by using the 'synchronized' keyword.

Methods, or synchronized blocks of code, can only be executed by one thread at a time for a given instantiation of a class, because that code requires obtaining the object's lock before execution (Roetter, 2001).

#### 4.5.1.4 ChatClient

The ChatClient class is the main client class which controls all the functionality for the end user. It ties in the client interface classes, the speech classes, and takes care of all the multithreading involved in order for the application to run. This class acts as the central hub for the client, making appropriate calls to the various client classes when needed. It was important to separate the design classes, namely the LoginScreen and ChatScreen, and the speech classes, namely the SpeakSynthesizerAPI, from the main ChatClient class, which resulted in being better able to maintain the code and classes.

Once this class is invoked, the class first makes a call to the LoginScreen call. As mentioned in section 4.5.1.1, the LoginScreen captures the user's login details and submits them to the ChatServer. If the user successfully logs in, an instance of the ChatScreen class is created. This interface displays the main chat screen which incorporates the main functionality of the system.

Once the user has successfully reached this stage in the application, a new thread is created for this user which handles all incoming and outbound messages to the server. On figure 4.12,



depicting the development class diagram, this is represented by the link between the ChatClient and the ChatHandler class. Calls to the speech class's methods are made from within this class. Additional functionality to reduce cognitive load for the visually impaired, as discussed in section 4.7, is also handled within the ChatClient class.

```
public void run ()
{
    try
    {
        while (true)
        {
            String line = instream.readUTF();
            chatTextArea.append(line + "\n");
            speakChat.speakMsg(line);
        }
    }
}
```

**Figure 4.22 ChatClient thread reading incoming message from the ChatHandler class**

The above code snippet is the implementation behind the Thread run method, which when called, starts the thread for the ChatClient. This piece of code creates a thread which allows the continuous communication with the server while the user can still input commands to the system in a multi-threading environment (Hilderink, Broenink.. 1997). Once the thread is running, it will continue to listen for any incoming data from the ChatHandler broadcast method (discussed in 4.5.1.3). Any data received will be appended to the user's main screen in the form of a conversational message. The third line of code will synthesize this message so that the visually impaired user can comprehend the conversation.

#### 4.5.1.5 LoginScreen

The user interface classes, LoginScreen and ChatScreen, will be discussed in more detail in section 4.6. Both this section and section 4.7 will discuss the type of functionality each class has to offer the end user. The LoginScreen interface is not nearly as technically developed as the ChatScreen class.

The LoginScreen class is composed of a couple of text fields, a couple of labels, and a few buttons displayed on a picture background to enhance its appeal to the user. The text fields capture the user's username and password which are then transmitted to the server when the user clicks the 'login' button. The other two buttons are called 'Clear', and 'Close'. As the names suggest, the 'login' button submits the user's details to the server for validation, the 'clear' button clears the text fields of any data, and the 'close' button closes the interface.

These three buttons can be triggered in more than one way. The event behind each button will be triggered if mouse is clicked when the cursor is over the button. The LoginScreen implements the ActionListener interface so that it can receive action events. When an action event occurs, that object's actionPerformed method is invoked.

The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method (Sun Website).

In the code below we can see the ActionListener's actionPerformed method implementation within the LoginScreen class.

```
public void actionPerformed(ActionEvent e)
{
    if ("login".equals(e.getActionCommand()))
    {
        submitDetails();
    }
    if ("clear".equals(e.getActionCommand()))
```

**Figure 4.23 ActionListener's actionPerformed method implementation in the LoginScreen class**

This action is then registered with the instance 'login' of the JButton class;

```
login.setActionCommand("login");
```

**Figure 4.24 ActionListener's actionPerformed event registered with 'login' component**

An event will also be triggered if the 'enter' key on the keyboard is pressed. This is an alternative way of accepting input from the user and all buttons have this dual command acceptance.

```
login.addKeyListener(new KeyAdapter()
{
    public void keyPressed(KeyEvent e)
    {
        int key = e.getKeyCode();
        if (key == KeyEvent.VK_ENTER)
        {
            submitDetails();
        }
    }
});
```

**Figure 4.25 KeyListener code executed once the 'Enter' key is pressed**

The KeyAdapter is an abstract adapter class which is used to receive keyboard events. It creates the keyListener objects, such as the 'login' button in this example, using the addKeyListener method. When any key is pressed to the object, the generated event is received via the KeyPressed method.

Once the event is triggered the details are submitted to the server via the submitDetails method. This method opens a two way communication with the server, whereby the username and password are sent to the server, and a response of either 0 or 1 is returned from the server. This was also mentioned in section 4.5.2.1.

```
try
{
    Socket = new Socket ("localhost", 9200);
    DataOutputStream outLogin = new DataOutputStream
        (new BufferedOutputStream (s.getOutputStream()));
    DataInputStream loginResponse = new DataInputStream
        (new BufferedInputStream (s.getInputStream ()));
    outLogin.writeUTF (usr+","+pwd);
    outLogin.flush();
    validLogin = loginResponse.readInt();
}
```

**Figure 4.26 LoginScreen submits user login details to server on port 9200**

A '0' returned from the server indicates the user was unsuccessful in logging in. The user will then be alerted with a pop up prompt asking if they would like to try again. The code used to generate this is displayed below.

```
if (validLogin == 0)
{
    String message = "Your login attempt was unsuccessful."
        += "Would you like to try again? ";
    int reply = JOptionPane.showConfirmDialog(null, message,
        title, JOptionPane.YES_NO_OPTION);
    if (reply == JOptionPane.YES_OPTION)
    {
        usr = logintext.getText();
        pwd = loginpwd.getText();
        loginFrame.pack();
        loginFrame.setSize(image.getIconWidth()+5, image.getIconHeight()+30);
        loginFrame.setVisible(true);
    }
}
```

**Figure 4.27 Pop up prompt prompting the user to try logging in again**

If the user is successful on logging in a '1' is returned and the user is then presented with the main chat screen.

```
if (validLogin == 1)
{
    try
    {
        receiveAndSaveImage(s);
        new ChatScreen(s.getInputStream(), s.getOutputStream(), usr);
    }
}
```

**Figure 4.28 User successfully logs in and is presented with main Chat Screen**

The image received from the server and saved locally on the client's machine. A new instance of a Thread and ChatScreen are created for the client.

#### 4.5.1.6 ChatScreen

The ChatScreen interface enables the user to gain access to the majority of the system functions.

It allows the user to:

- view who is currently logged into the system
- submit a message to the main chat lobby
- control the speech synthesizer to best suit their needs
- re-synthesize old messages (discussed in 4.7.1)
- maintain a list of acronyms, initialisms, and abbreviations (discussed in 4.7.3)
- save a message to the database (discussed in 4.7.2)

Needless to say, the main function of the system is to chat with other users of the system. In order to submit a message to the lobby area, the data is first sent to the ChatHandler class which then relays this message to all users connected at the time. From the code below, we can see that if the user clicks the 'send' button, the entire message is captured via a getText method of the JTextField class. This value is then transmitted to an already opened connection to the ChatHandler object via the writeUTF function of the DataOutputStream class.

```
// Send button
buttonListener = new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        // Request a connection initiation
        if (e.getActionCommand().equals("send"))
        {
            try
            {
                outstream.writeUTF(chatLine.getText());
                outstream.flush();
            }
        }
    }
}
```

**Figure 4.29 Message submitted to the lobby**

The next fragment of code demonstrates how the user is informed of what they have typed. The speech synthesizer synthesizes any character that the user types. If the character typed is a comma, space, semi-colon etc, then the synthesizer will treat this as the end of a word, and synthesize the whole word to the user. This is to ensure the visually impaired user is constantly aware of what they have typed, and reduces the number of typos being submitted to the conversation.

```
chatLine.addKeyListener(new KeyAdapter()
{
    public void keyPressed(KeyEvent e)
    {
        int key = e.getKeyCode();
        word = word + (char)key;
        speakChat.speakMsg((char)key + "");
        if ((char)key == ' ')
        {
            speakChat.speakMsg(word);
            word = "";
        }
        if (key == KeyEvent.VK_ENTER)
        {
            try
            {
                outstream.writeUTF(chatLine.getText());
                outstream.flush();
            }
            catch (IOException ex)
            {
                ex.printStackTrace();
                listener.stop();
            }
            chatLine.setText("");
            mainFrame.repaint();
        }
    }
});
```

**Figure 4.30 Character / word typed synthesized before being submitted to the lobby**

In order for the user to control the speech synthesizer, four sliders which allow the user to set the pitch, range, volume, and speed of the speech synthesizer accordingly have been provided. As each slider re-used code, there was a need to create a method called createSlider which returned a slider with different settings, given the dimensions via its parameters. The method is called like so:

```
pitchSlider = createSlider(0,200,pitchSliderCounter,470,25,25,50,"Pitch Control");
```

**Figure 4.31 createSlider method called to initialize pitchSlider**

The implementation behind the createSlider method is given below. This returns an instance of the JSlider class. The parameters will vary accordingly given the type of slider that needs to be created.

```

private JSlider createSlider(int bottom, int top, int begin,
                             int locwidth, int locheight, int minorTickSpacing,
                             int majorTickSpacing, String titleRollover)
{
    JSlider slider = new JSlider(JSlider.VERTICAL, bottom, top, begin);
    slider.setBounds(locwidth, locheight, 50, 125);
    slider.putClientProperty("JSlider.isFilled", Boolean.TRUE);
    slider.setMinorTickSpacing(minorTickSpacing);
    slider.setMajorTickSpacing(majorTickSpacing);
    slider.setPaintTicks(true);
    slider.setPaintLabels(true);
    slider.setToolTipText(titleRollover);
    return slider;
}

```

**Figure 4.32 Slider created to control the synthesizer settings**

The user can control the sliders via one of three different ways:

- the mouse
- the up and down arrows (once the slider has focus)
- access keys

*Access keys* allow operators to select or execute an item by pressing a key combination, even when that item is not the currently focused item. For example, a button labeled 'Open' with an access key of 'O' can be activated at any time by pressing **Alt+O**. Access keys are applicable to menu items, push buttons, checkboxes, radio buttons and tab pages. All access keys within a window should be unique, including keys used in the top level of the pull-down menu (typically F for File, E for Edit, etc.).

Accelerators are keyboard shortcuts for actions that are frequently performed, for example, **Ctrl+P** for Print. Keyboard shortcuts allow users to bypass opening the menu by using a specific combination of keystrokes that perform the same function as a corresponding menu item. Instead of pressing **Ctrl+F**, then **S**, to activate menu item File-Save, a user can just press **Ctrl+S** to execute the same function. Accelerator keys are desirable but not mandatory and are listed in Keyboard Help at runtime.

The use of access keys would be very important for the end user, especially if they were blind. For example, if the user held down the 'Alt' key and pressed 'P', the pitch slider would raise and thus increase the pitch of the synthesizer. Similarly if the user held down the 'Ctrl' key and typed 'P' then the pitch would decrease. The other three sliders were designed in the same way except for speed the letter used was 'S', for range it was 'R' and for volume 'V' was used.

In the code displayed below an example of how this was done is presented. An ActionListener was created called setPitchHigher. Any time the action event was triggered for this, the pitch was increased and set as the new pitch to be used by the synthesizer. The highest the pitch can be set to is 200. If any of the sliders are being set to a value out of the valid range then the user will be notified through synthesized speech. An instance of the keystroke class called pitchIncrement was also created that triggered an event if the keys 'Alt' and 'P' were pressed at the same time. Finally, the setPitchHigher ActionListener and pitchIncrement Keystroke object were registered with a component of the ChatInterface.

```

ActionListener setPitchHigher = new ActionListener()
{
    public void actionPerformed(ActionEvent actionEvent)
    {
        if (pitchSliderCounter < 199)
        {
            pitchSliderCounter++;
            pitchSlider.setValue(pitchSliderCounter);
            System.out.println(pitchSliderCounter);
        }
        else
            System.out.println("Cannot set pitch any higher - out of pitch range");
    }
};

KeyStroke pitchIncrement =
    KeyStroke.getKeyStroke(KeyEvent.VK_P, KeyEvent.ALT_MASK);
chatLine.registerKeyboardAction(setPitchHigher,
    pitchIncrement, JComponent.WHEN_FOCUSED);

```

**Figure 4.33 ActionListener used to set the pitch slider higher using access keys ('Alt' and 'P')**

Other KeyStroke and ActionListener instances were created to cater for incrementing and decrementing the range, volume, and speed of the synthesizer.

#### 4.5.1.7 UserListServer

The UserListServer class's main function is to run as a Thread in the background that updates each client's user list with all the users currently logged into the system. The Thread is set to sleep for one second, it then wakes up and updates all users logged into the system. The Thread was specifically designed this way so that processor and memory were not used up unnecessarily (Pederson, 2003)

The UserListServer class was not considered as part of the design stage as it was presumed the ChatHandler class could have handled this extra functionality early on. Once development commenced a number of bugs and issues were encountered. The ChatHandler class could not have two threads running within the same class. The ChatHandler class already had an existing Thread running for handling the broadcasting of message to all clients. As a result there was a need to tweak the initial design in order to combat this issue.

The UserListServer class is invoked from the ChatHandler class as follows:

```
UserListServer userlistServer = new UserListServer();  
userlistServer.start();
```

**Figure 4.34 UserListServer Thread class created and started**

Once the Thread wakes up, it accepts a connection from the client (UserClientList class). Once this ServerSocket receives a connection, the rest of the code will be executed in sequence. The user list is stored in a vector variable and as a result, an instance of the ObjectOutputStream class needed to be created in order to send the list of users to the clients since a Vector is also an object.

```
public void run()  
{  
    while(true)  
    {  
        try  
        {  
            Thread.sleep(1000);  
            sock = new ServerSocket(serverPort);  
            clientSocket = sock.accept();  
            vectorObj = new ObjectOutputStream(clientSocket.getOutputStream());  
  
            synchronized (vectorObj)  
            {  
                vectorObj.writeObject(userlist);  
            }  
            clientSocket.close();  
            sock.close();  
            System.out.println("Latest list sent to Client");  
        }  
        catch (Throwable e)  
        {  
            System.out.println("Error " + e.getMessage());  
            e.printStackTrace();  
        }  
    }  
}
```

**Figure 4.35 Code used to send list of users logged in to the clients once every second**

#### 4.5.1.8 UserListClient

The UserListClient class communicates with the UserClientServer class. Like the UserClientServer class, a Thread is also created which sleeps for just over one second. Once the thread wakes up, the class attempts to make a connection to the UserClientServer. Once a connection is made, the UserClientServer proceeds to execute the code following the acceptance of the connection, which submits the object (userlist) to the UserListClient. On the UserListClient side, the object is read which needs to be cast to a Vector object. An instance of the enumeration class is then created and assigned the list of elements in this vector. The list of user names are iterated through and each value is appended to the user list area on the client's interface. This process is repeated once every second. As the project evolves, and more memory is being used by the application, the time intervals of suspending the Thread can be delayed even further.



```

public void run()
{
    while(true)
    {
        try
        {
            Thread.sleep(1500);
            sock = new Socket(serverIPname,serverPort);
            vectorObj = new ObjectInputStream(sock.getInputStream());

            userlist = (Vector)vectorObj.readObject();
            Enumeration en = userlist.elements();
            String name = "";
            userListArea.setText("User List");
            while(en.hasMoreElements())
            {
                name = (String)en.nextElement();
                userListArea.append("\n"+name);
            }
        }
        catch (Throwable e)
        {
            System.out.println("Error " + e.getMessage());
            e.printStackTrace();
        }
    }
}

```

**Figure 4.36 Code used to receive list of users logged in from the UserListServer class once every second**

#### 4.5.1.9 SpeakSynthesizerAPI

Each instance of the SpeakSynthesizerAPI is created with default values as can be seen in figure 4.37. The synthesizers pitch is set to 100, the volume is set to 10, the speed or word per minute is set to 150, and the pitch range is set to 10. In the code below a general domain synthesizer for US English is being used. In order to find this synthesizer the Central class of the JSAPI is called to find a Synthesizer. The Central class expects to find a speech.properties file in user.home or java.home/lib. The user has the choice to control the various synthesizer settings via access keys. This was achieved by parameter passing control values to public methods coded by the author of this thesis.

```

public SpeakSynthesizerAPI()    throws Exception
{
    try
    {
        SynthesizerModeDesc generalDesc = new SynthesizerModeDesc(
            null,           // engine name
            "general",      // mode name
            Locale.US,      // locale
            null,           // running
            null);          // voice
        synthesizer1 = Central.createSynthesizer(generalDesc);
        allocate();
        synthesizer1.getSynthesizerProperties().setPitch(100);
        synthesizer1.getSynthesizerProperties().setSpeakingRate(150);
        synthesizer1.getSynthesizerProperties().setVolume(10);
        synthesizer1.getSynthesizerProperties().setPitchRange(10);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

```

**Figure 4.37 SpeakSynthesizerAPI constructor**

## **4.6 Interface Screen Shots**

The graphical user interface (GUI) is one of the most important components of any system. It acts as a gateway to all functionality of any system. It is therefore needless to say that the design and layout of any user interface is hugely important for any application. Since the first interfaces designed in the late 60s (Engelbart & English, 1968), GUIs have evolved immensely into graphical icons and visual indicators of software applications today. In the early days there was no interface for users, the user had their punch cards which were read in to a reader which in turn sent the results to the printer (Constantine, 2000). The idea behind GUIs stems from the fact that using the mouse to point to a desired command on the interface is far easier than having to remember hundreds of commands in operating systems like AIX, Unix, and MS Dos. Using the mouse to gain functionality from a system is far easier than using the keyboard. The idea of pointing to something is more of a human gesture (Gladden, 2000) whereas using the keyboard is more machine like. As a result an extra effort was made to make the interaction process with the keyboard more human like since this application will be used by the visually impaired.

### **4.6.1 User Interface Guidelines**

Smith and Mosier (1986) noted, a single flaw in the user interface may not result in system failure, but a combination of defects may result in system failure, poor performance, and user complaints. As a result, there was a need for a set of guidelines or usability principles to be used in this thesis, so that common design pitfalls were less likely to be encountered. These usability principles can then be used to analyze an interface for usability issues.

Nielsen's (1994) "Ten Usability Heuristics" were used in the interface design of this system to perform heuristic evaluations.

#### **Nielsen's Heuristic 1: Visibility of System Status**

'The system should always keep users informed about what is going on, through appropriate feedback within reasonable time'

This heuristic applies to this speech enabled chat application, typically through synthesized speech and visual displays. This is achieved through issuing constant feedback to the user. Anytime a user logs in/out of the system, the end user is notified about this. The user is also given feedback regarding the key the user pressed though synthesized speech. Finally, through the use of access keys, the user is also notified about key information. This will be discussed more in section 4.6, additional functionality.

#### **Nielsen's Heuristic 2: Match between system and the real world**

‘The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order’

Through the use of user interface metaphors or analogies to the real world users can better comprehend how to navigate through the environment and interact with other users via conversing.

### Nielsen's Heuristic 3: User control and freedom

‘Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo’

If the user feels restricted, they will most likely become frustrated, and the result can be disinterest in the application (Norman, 1990). Therefore, the user needs to feel that they are in control in any part of the application. An ‘emergency exit’ can be achieved either by clicking the x on the top right hand corner of the application or by clicking the ‘logout’ button on the interface. In order to cater for the visually impaired, the user has the ability to logout of the system through the use of access keys. If the user types ‘Alt’ and ‘L’, the user will be logged out of the system.

```

        ActionListener chatLogoutListener = new ActionListener()
        {
            public void actionPerformed(ActionEvent actionEvent)
            {
                try{SpeakChat.speakMsg("You are now logging out of the system");}
                catch(Exception e) {e.printStackTrace();}
                mainFrame.dispose();
                mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                System.exit(0);
            }
        };

        KeyStroke chatLogout = KeyStroke.getKeyStroke(KeyEvent.VK_L, KeyEvent.ALT_MASK);
        disconnectButton.registerKeyboardAction(chatLogoutListener,
                                                chatLogout, JComponent.WHEN_IN_FOCUSED_WINDOW);
    }
}
```

**Figure 4.38 Access Keys ‘Alt’ and ‘L’ to logout of the system**

### Nielsen's Heuristic 4: Consistency and standards

*‘Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions’*

The chat application user interface, as with all the interfaces, should be consistent throughout. All the button texts, labels, alt tags, frame texts etc mean exactly as they intend to eliminate any confusion for the end user.

### Nielsen's Heuristic 5: Error prevention

*'Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action'*

Error detection and exception handling is catered for in this application for roughly 80% of the possible bugs that may arise. This is discussed in more detail in section 4.7.

The users are presented with confirmation options just before various actions are committed. For example, if the user wants to save a conversation, then the user will be prompted to confirm before the conversation is saved. Such alert prompt include 'Are you sure you want to save this conversation?' and 'Your login attempt was unsuccessful. Would you like to try again?'. An example of the code of such a confirmation concept can be seen in figure 4.39.

```

ActionListener saveMsgListener = new ActionListener()
{
    public void actionPerformed(ActionEvent actionEvent)
    {
        int answer = JOptionPane.showConfirmDialog(null,
            "Are you sure you want to save this message?",
            "Save Conversation Confirmation", JOptionPane.YES_NO_OPTION);
        try{speakChat.speakMsg("Are you sure you want to save this message?");}
        catch(Exception ex) {ex.printStackTrace();}
        if ( answer == JOptionPane.YES_OPTION)
        {
            DatabaseHelper dbconnect = new DatabaseHelper(usr,pwd);
            dbconnect.saveMsg(chatTextArea.getText());
            try{speakChat.speakMsg("This conversation has been successfully saved");}
            catch(Exception ex) {ex.printStackTrace();}
        }
    }
};
KeyStroke saveMsg = KeyStroke.getKeyStroke(KeyEvent.VK_S,KeyEvent.ALT_MASK
    | InputEvent.SHIFT_MASK);
saveButton.registerKeyboardAction(saveMsgListener, saveMsg,
    JComponent.WHEN_IN_FOCUSED_WINDOW);

```

**Figure 4.39 Access Keys 'Shift', 'Alt' and 'S' to save a conversation**

### Nielsen's Heuristic 6: Recognition rather than recall

*'Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate'*

In line with this heuristic, instructions for the system can be synthesized to the user through the use of access keys. If the user types 'Alt' and 'I' then the list of instructions and commands will be reiterated to the user through synthetic speech (Appendix B). Also, given the end user of the system could be blind, an additional piece of functionality was added to the application. This includes synthesizing previous messages from other users to the end users since the visually impaired do not have the option of scrolling up on the conversation pane.

#### **Nielsen's Heuristic 7: Flexibility and efficiency of use**

*'Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions'*

This chat application was designed to be simple by nature, and no degree of experience or technicality is required in order to use this application.

#### **Nielsen's Heuristic 8: Aesthetic and minimalist design**

*'Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility'*

As mentioned in the previous heuristic, this application is simple by design, and no additional needless information is seen or can be used from the interface.

#### **Nielsen's Heuristic 9: Help users recognize, diagnose, and recover from errors**

*'Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution'*

This heuristic is relevant to the chat applications user interface which is able to assist the user in preventing errors from arising initially. If an error is generated it will be handled accordingly through exception handling (as discussed in Nielsen's Heuristic 5: Error Prevention).

#### **Nielsen's Heuristic 10: Help and documentation**

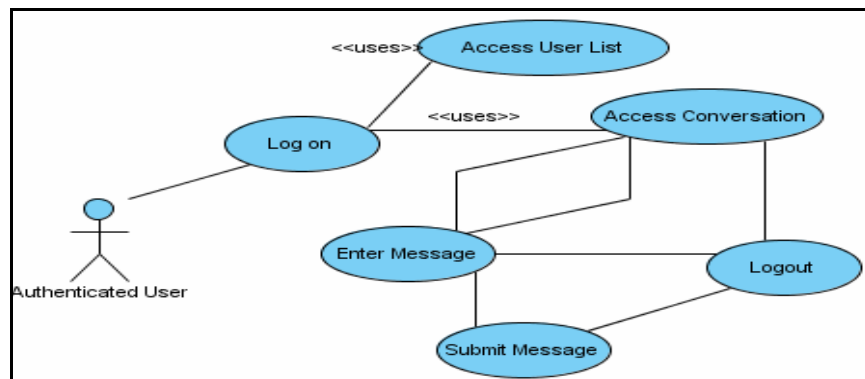
*'Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large'*

As this chat application is designed to be simple, no tutorial is required for this application. Nevertheless, help can be obtained through appendix B which outlines the list of access key commands to control the system.

#### 4.6.2 Evolution of Screenshots

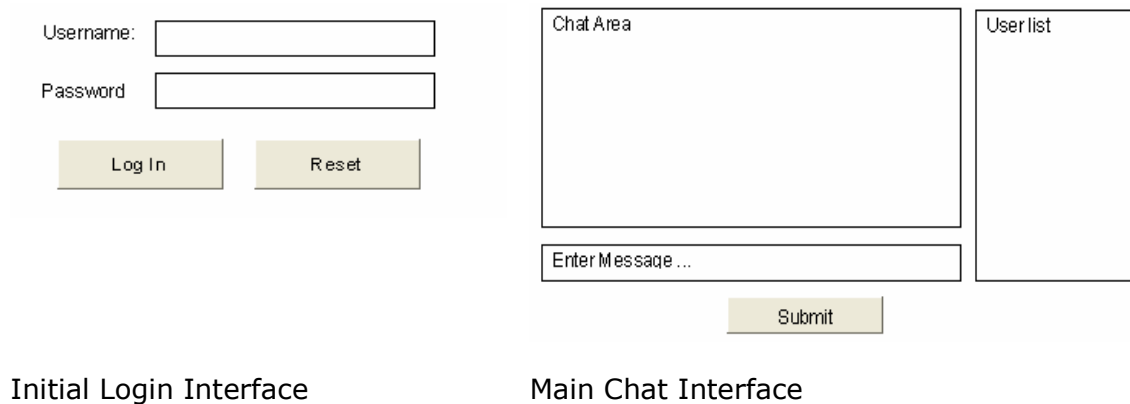
In section 3.3.3.1 use cases were discussed and the initial functionality required of the system, and how each use case represented specifies a complete functionality within the system. Alistair Cockburn's described use cases as 'a sequence of transactions between external actors and the system' (Cockburn, 1997). With this in mind, we can see why people map use cases to user interfaces, as use cases are just user interface descriptions. The design of the user interface comes at a later stage in the development life cycle after the goals and objectives have been clearly defined through the various use cases.

The initial requirements were first captured through a use case diagram. As we seen in the design stage, this use case describes the main areas of functionality in the chat system. The user of the chat system logs on to the system where they can then access messages typed by other members of the chat room as they appear on the screen. They then have the option to log out or to enter a message and post it to the room. This can be done many times before logging out of the room.



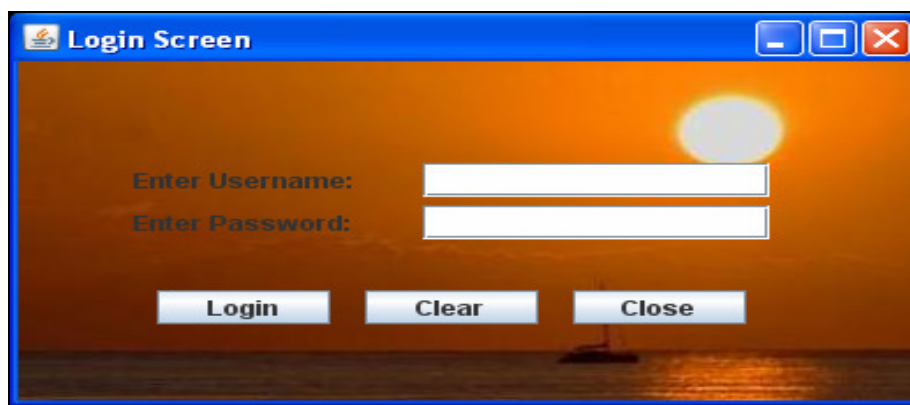
**Figure 4.40 Use case from Section 3.3.3.1**

This use case mapped nicely with the initial prototype screen shots designed on Microsoft Powerpoint. As seen in section 3.5.4, the initial prototype screen shots were designed in order to gain vision and focus of what the developed application interface would look like.



**Figure 4.41 Prototype screen shots from section 3.5.4**

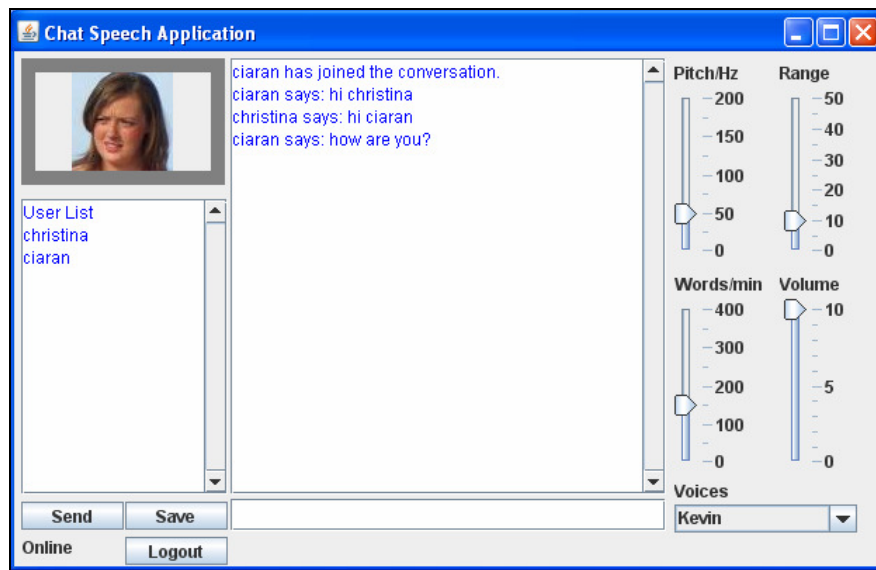
The initial prototypes (seen above and in section 3.5.4) have since evolved into working applications. Their appearance has been greatly altered and their appearance enhanced. The initial prototype login screen has evolved into a functionally working login screen depicted below.



**Figure 4.42 Login Screen Application Screenshot**

Similarly, the main chat application has evolved into what is now the main chat application. This interface has been redesigned incrementally in numerous mini releases, as with all applications developed under Crystal Clear (Stojanovic, Dahanayake, Sol, 2003) with added functionality appended with each release.





**Figure 4.43 Main Chat Lobby Application Screenshot**

## ***4.7 Additional Functionality***

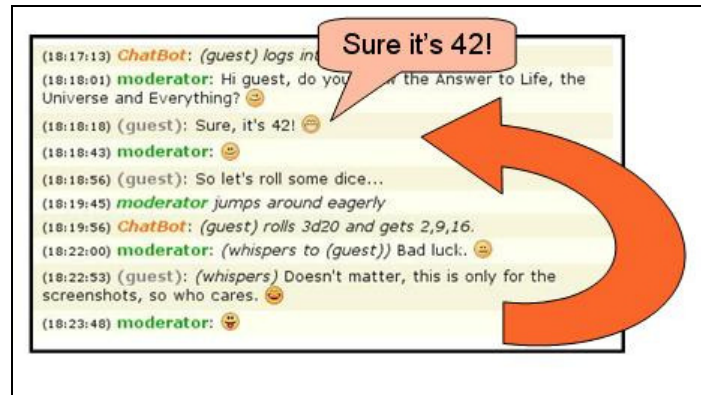
Once the application was developed, it was clear that extra functionality was needed. Since the methodology used to develop this application was Crystal Clear, it paved the way to add additional functionality to the system without too many problems.

### **4.7.1 Reduce Cognitive Load**

The term cognitive load refers to the load on working memory during instruction. During the development process the need to reduce the amount of cognitive load for the end user was reviewed. In a normal chat application, to view previous messages typed by other users the user needs only to scroll up. The visually impaired have no way of revisiting previous messages which undoubtedly would increase cognitive load for them when trying to remember everything said in the conversation. The cognitive load would undoubtedly be even higher with multiple people conversing concurrently. Even if the speech can be presented at a loud enough level to be heard - and this may not always be possible - understanding this speech will provide a greater cognitive load than would be exerted on a person with normal hearing (Newell, 2003).

Cognitive load naturally increases with the amount of information to process. With the advent of high internet usage, cognitive load has become more of an issue (Quiroga, Crosby & Iding, 2004).

This research attempts to combat this issue in such a way that the visually impaired are able to gain access to what a sighted person has access to in a chat application environment. Such a tool would alleviate the pressure on the end user of trying to remember previous messages while typing at the same time.



**Figure 4.44 Recall Function**

A storage mechanism of some sort was needed to hold the messages for each user. Initially I had thought about using a 'Hash Table' for this purpose. In computer software a 'Hash Table' is a data structure that associates key and value pairs. For example, in this instance the key could be the user name and the value could hold the different messages for this key or username.

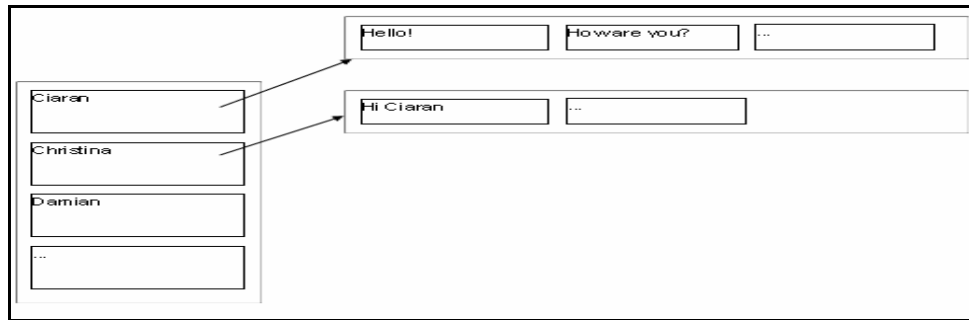
Ciaran	Hello!
Christina	Hi Ciaran
Ciaran	How are you?

**Figure 4.45 Hash Table Key (username) - value (message) pairs. [source: author using MS PowerPoint].**

The development on this concept had started; however, this approach did not work as initially planned. This was because after each user sent a message, the original message by that same user was overwritten by the most recent message. So, for example, the key-value pair 'ciaran','Hello!' was overwritten by 'ciaran','how are you?'. Other data structures such as 'Linked Lists' and 'Array Lists' were researched for this purpose but similar issues were encountered.

Through online aid from Devshed forums, a fellow developer suggested the use a 'Hash Table' but instead of using the string data type for a value, a 'Vector' could be used. Now

there would be a unique username because every key in the 'Hash Table' must be unique, and a list of messages for each username. This can be better explained in figure 4.46.



**Figure 4.46 Hash Table Key (username) - value (vector of messages) pairs. [source: author using MS PowerPoint].**

The messages can then be retrieved by accessing the individual messages from the vector with a username and an index. The username will first select the key value pair, and the index will retrieve the relevant message for that user in the vector list. Below how such a concept was achieved is explained using the relevant coding commands. The code snippet below has grouped to relevant hash table coding techniques into one section of code for demonstration purposes.

```
Vector msgs = new Vector();
Hashtable <String,Vector> userMsgs = new Hashtable<String,Vector>();

if(joinedOrLeft.equals("joined"))
{
    chatTextArea.append(line + "\n");
    userMsgs.put(usr,new Vector());
}
else if(joinedOrLeft.equals("left"))
{
    chatTextArea.append(line + "\n");
    if (userMsgs.containsKey(usr))
    {
        userMsgs.remove(usr);
    }
}

userMsgs.get(usr).addElement(line);
```

**Figure 4.47 Code to demonstrate the handling of synthesizing previous messages**

Anytime an instance of the ChatClient class is created, an instance of a Vector and Hashtable are also created. Once a user joins the conversation, a key-value pair is added to the Hashtable. The key will be set to the username and a new empty vector will be inserted for the corresponding value. Similarly, when a user leaves the conversation, the key and its value for that user will be removed from the Hashtable. Any inbound string message from the ChatHandler will be sectioned so that the username and message can be identified.

As can be seen in figure 4.48, any inbound message is of the form “Ciaran says: Hi Christina”. This string is then stripped down by an extra method that was created called `getRememberedLine`. The `getRememberedLine` takes a string parameter which will be the inbound message from `ChatHandler`, analyzes this message accordingly, and then returns the relevant line to be synthesized to the user.

```
public String getRememberedLine(String line)
{
    String[] splittLine = line.split("\\\\,");
    String username,command,lineToSpeak = "";

    if (line.contains("<") && line.contains(">"))
    {
        int start = line.indexOf('<');
        int end = line.indexOf('>');
        command = line.substring(start, end+1);
    }

    int indexMsg = 0;
    int validMsg = 0;

    Integer.parseInt(splittLine[1].trim());
    indexMsg = Integer.parseInt(splittLine[1].trim());
    username = splittLine[2].trim();
    validMsg = 1;

    if (command.equals("<remember>"))
    {
        lineToSpeak = userMsgs.get(username).get(userMsgs.get(username).size()-indexMsg).toString();
    }
    return lineToSpeak;
}
```

**Figure 4.48 Code to retrieve the previous message**

In order to synthesize a particular message for a particular user, the command given to the system naturally needs to be in a certain form so that the system understands how to action it. The command issued to the system needs to be of the following syntax: ‘<remember>,index,username’. This command was only used to demonstrate that such functionality was achievable.

For example, if the user wishes to re-synthesize the third most recent message sent by user ‘ciaran’ then the following command would need to be entered onto the interface: ‘<remember>,3,ciaran’.

From code snippet displayed in figure 4.48, the message accepted as a parameter to the method, is first divided so that relevant command, index and username can be obtained for processing. After these details have been obtained, the relevant message to be synthesized can be retrieved from the `HashTable` instance. The piece of code in order to achieve this is highlighted with a green oval in figure 4.48.

### 4.7.2 Additional Access Keys

Access keys are keys used on the keyboard to navigate the system. Access keys are particularly useful for handheld devices, laptops, and visually impaired users. Not all users can access graphical environments with a mouse. Some users depend on alternative methods of communicating with systems either through speech or the keyboard in order to navigate

the system at ease. Horton (2005) found that these shortcuts improve keyboard navigation and can be a real timesaver for disabled users.

It stands to reason that designing an application for keyboard access will improve the overall design appeal of the application. Navigating through an application and gaining focus of key components on the interface can be faster through the use of access keys (Buzzi, Andronico, Leporini, 2004). Easier navigation aside, there are times when the end user has no other choice but to use access keys. Therefore, it was more important that such functionality was enabled for the end user of this system. The visually impaired don't have the ability to use a pointing device such as a mouse but can, however, use a keyboard. Wright (2004) noted that when designing and developing an application, one should consider proper use of keyboard tabbing and access keys with these users in mind.

#### 4.7.2.1 Logout of Chat Application

As mentioned in section 4.6, to comply with Nielsen's Heuristic 3: 'User control and freedom', an easy exit option was necessary for all users of the system. All users of the system include the visually impaired users too. Since there exists no easy exit option for the visually impaired users through the click of a mouse, an alternative exit route was necessary. This functionality was achieved through the use of sticky keys, whereby if the user pressed 'Alt' and 'L' the user was logged out of the system.

```
ActionListener chatLogoutListener = new ActionListener()
{
    public void actionPerformed(ActionEvent actionEvent)
    {
        mainFrame.dispose();
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        System.exit(0);
    }
};

KeyStroke chatLogout = KeyStroke.getKeyStroke(KeyEvent.VK_L, KeyEvent.ALT_MASK);
disconnectButton.registerKeyboardAction(chatLogoutListener, chatLogout,
                                       JComponent.WHEN_IN_FOCUSED_WINDOW);
```

**Figure 4.49 Logging out of System via Access Keys**

An ActionListener object called chatLogoutListener has been set up to listen for the actionEvent, which when triggered will log the user out of the system. If the key stroke obtained is a combination of 'Alt' and 'L' is found then the actionEvent is actioned.

### 4.7.2.2 Saving a Conversation

The user also has the option to save the current conversation as any time. If the user clicks the 'save' button or uses access keys 'Alt', 'Shift' and 'S' together, an instance of the DatabaseHelper class is created, and the saveMsg method of this helper class is invoked. The conversation is passed as a parameter to the method, which is then saved to a table in the database along with a unique message id, and the timestamp of when the conversation was saved.

```
if (e.getActionCommand().equals("save"))
{
    System.out.println("Saving this msg to the db... " + chatTextArea.getText());
    DatabaseHelper dbconnect = new DatabaseHelper(usr,pwd);
    dbconnect.saveMsg(chatTextArea.getText());
}
```

Figure 4.50 Saving Conversation via Mouse

```
ActionListener saveMsgListener = new ActionListener()
{
    public void actionPerformed(ActionEvent actionEvent)
    {
        System.out.println("Saving this msg to the db... " + chatTextArea.getText());
        DatabaseHelper dbconnect = new DatabaseHelper(usr,pwd);
        dbconnect.saveMsg(chatTextArea.getText());
    }
};

KeyStroke saveMsg = KeyStroke.getKeyStroke(KeyEvent.VK_S,KeyEvent.ALT_MASK
| InputEvent.SHIFT_MASK);
saveButton.registerKeyboardAction(saveMsgListener, saveMsg,
JComponent.WHEN_IN_FOCUSED_WINDOW);
```

Figure 4.51 Saving Conversation via Access Keys

### 4.7.2.3 Synthesizing list of Logged in Users

An instance of the SpeakSyntheizerAPI is first created so that its methods can be referenced from within the ChatClient class. As described in the other access key functions, the ActionListener and(ActionEvent) classes are initiated, and once triggered via key strokes 'Alt' and 'U', the list of users who are currently logged in are synthesized to the user. The 'userlist' is iterated through and each element in synthesized to the end user.

```

SpeakSynthesizerAPI speakUpdate = new SpeakSynthesizerAPI("kevin");
speakUpdate.allocate();

ActionListener userListListener = new ActionListener()
{
    public void actionPerformed(ActionEvent actionEvent)
    {
        speakUpdate.speakMsg("The following users are currently online: ");
        for (int i = 0; i < userList.size(); i++)
        {
            speakUpdate.speakMsg(userList.elementAt(i).toString());
        }
    }
};

KeyStroke speakUserlist = KeyStroke.getKeyStroke(KeyEvent.VK_U, KeyEvent.ALT_MASK);
chatLine.registerKeyboardAction(userListListener, speakUserlist, JComponent.WHEN_IN_FOCUSED_WINDOW);

```

**Figure 4.52 Synthesizing logged in users via access keys**

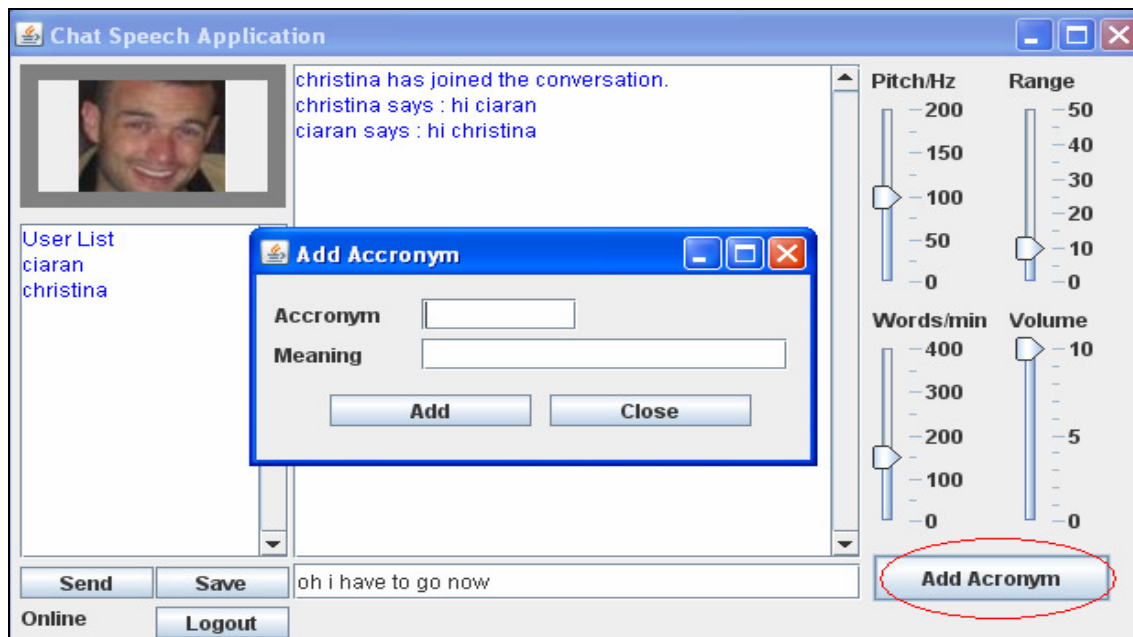
### 4.7.3 Acronyms interpreted

With the popularity and rise in usage of real time text-based communications, such as instant messaging, e-mail, Internet and online gaming chat rooms, discussion boards and cell phone text messaging (SMS), came the emergence of a new language tailored to the immediacy and compactness of these new communication media.

As a result it was only natural to integrate such a concept into this application. This will not only further reduce cognitive load on the visually impaired end user but it will also quicken the conversation for both sender and received of any message.

As mentioned previously, each letter the user types is synthesized and when the end of each word is reached the entire word is synthesized. If that word is saved on the database as an acronym, then the actual meaning of this word will be synthesized instead. For example, if the user typed 't tyl', then the meaning of this acronym 'talk to you later' will be synthesized instead, provided this acronym is already added to the list.

The user has control of what acronyms they would like added to the database. This functionality is highlighted in Figure 4.53. This functionality can also be accessed via access keys 'Alt' and 'A' for acronym.



**Figure 4.53 Add Acronym GUI**

When the button 'Add Acronym' is accessed the user is then presented with a smaller Acronym GUI. Once the user adds the acronym it this acronym is then appended to the list in the database table called tblAcronym. As from the main chat, the user can also access this add button via access keys 'Alt' and 'A' when the Acronym GUI is in focus. If the user clicks 'Alt' and 'C' the Acronym GUI will close. The user will be back on the main screen ready to start chatting with the enter message field already set in focus. The user is aware of this through synthesized speech

```
if(acronymtext.getText().trim().length() > 0 && meaningtext.getText().trim().length() > 0)
{
    Hashtable <String,String> userMsgs = new Hashtable<String,String>();
    DatabaseHelper dbhelper = new DatabaseHelper();
    userMsgs = dbhelper.getAcronyms();
    if(!userMsgs.containsKey(acronymtext.getText().trim()))
    {
        dbhelper.addAcronym(acronymtext.getText().trim(), meaningtext.getText().trim());
    }
    else
    {
        try(speakChat.speakMsg("This abbreviation has already been entered.
                               Please enter another abbreviation "));
        catch(Exception ex) {ex.printStackTrace();}
        acronymtext.setText("");
        meaningtext.setText("");
        acronymtext.requestFocus();
    }
}
```

**Figure 4.54 Code to add Acronym**



The above code snippet adds an acronym to the database. It first checks whether the acronym has already been entered on the database and if it hasn't it adds it. Once it has successfully been added, the user is ready to add another acronym.

If the user enters a blank space for either 'Acronym' or 'Meaning' the system will notify the user of this through the synthesizer. Similarly, if the user enters an acronym that is already entered on the database then the user will also be notified of this via synthetic speech.

```
if ((char)key == ' ' || (char)key == ';' || (char)key == '.')
{
    try
    {
        DatabaseHelper dbhelper = new DatabaseHelper();
        Hashtable <String,String> getAccronyms = new Hashtable<String,String>();
        getAccronyms = dbhelper.getAccronyms();
        if(getAccronyms.containsKey(word.toLowerCase().trim()))
        {
            speakChat.speakMsg(getAccronyms.get(word.toLowerCase().trim()));
        }
        else
            speakChat.speakMsg(word);
    }
}
```

**Figure 4.55 Synthesizes meaning of acronym**

The code snippet in Figure 4.55 depicts the system substituting and synthesizing the meaning of the acronym. If the acronym is not found then just the word typed by the user is synthesized.

## ***4.8 Error Detection and Exception Handling***

Although all software has defects or bugs (Weise & Baer, 2008), there are ways of reducing the number of bugs in software or dealing with the bugs that cannot be avoided. This can be achieved through error detection and exception handling. Exception handling is the process of handling abnormal system events and exception handling features enable developers to declare exceptions (Romanovsky, 2001). Developers can implement a piece of code which will be executed once an error in the code is encountered. The normal sequential execution of the code stops at the exception and a corresponding handler is then called and executed in place of the error.

Unexpected errors can arise from almost any part of the code. For example, you could be trying to open a file that no longer exists, or you are trying to save data to a read only device, or you are trying to use an object that has not been correctly initialized.

It was found that between 1% and 5% of program text in a study (Weimer & Nacula, 2004) was comprised of error-handling catch and finally blocks and between 3% and 46% of the program text could be reached from catch and finally blocks. These stats provide enough evidence that error handling is a significant part of any application.

The following piece of code illustrates a simple exception handler.

```

try
{
    file = new DataInputStream(new InputStream("example.txt"));
}
catch( IOException e )
{
    System.out.println("This example.txt file does not exist");
}

```

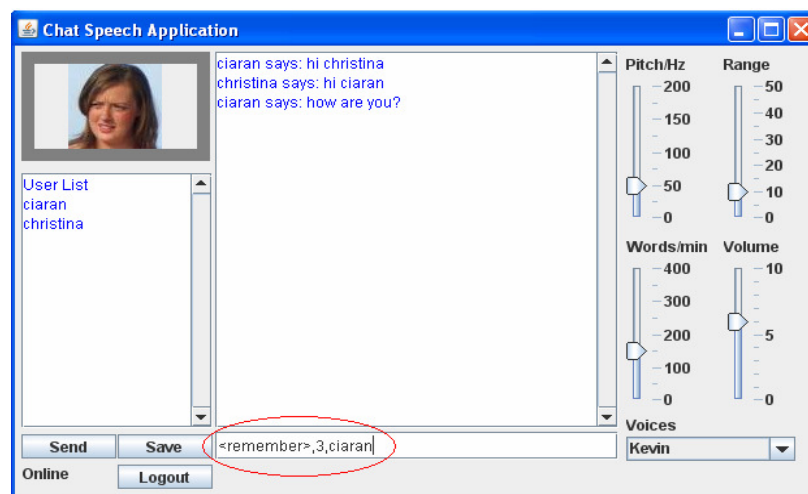
**Figure 4.56 Example of Error Handling**

Typically, an exception handler consists of two parts or blocks of code enclosed in curly brackets. The try block encloses some code that may generate an error or throw an exception, and the catch block specifies what action needs to be taken if the exception is thrown. If an error is generated exception handling provides a means of exiting the application in a graceful manner. In this application roughly 80% of the catastrophic errors have been catered for which allows a relatively smooth interaction between the system and end user.

## 4.8.1 Examples

### 4.8.1.1 ArrayIndexOutOfBoundsException

According to Sun this exception is thrown at runtime when an array is indexed with a value less than zero or greater than or equal to the size of the array. For example, in the screenshot below we can see that the user is trying to retrieve a value which does not exist. In the main chat lobby, the user 'ciaran' has contributed only two messages to the conversation. However, the command used by the user is trying to retrieve the third last message by the user 'ciaran' which does not exist. This is clearly demonstrated below in figure 4.57.



**Figure 4.57 Interface Command to generate Array Index Out Of Bounds Exception**

At runtime, when such a command was submitted to the system and exception was thrown and the application terminated abnormally. The below error was returned to the console command prompt and the interface refused any further input from the user.

```

java.lang.ArrayIndexOutOfBoundsException: -1
    at java.util.Vector.get(Unknown Source)
    at ChatClient.getRememberedLine(ChatClient.java:278)
    at ChatClient.run(ChatClient.java:201)
    at java.lang.Thread.run(Unknown Source)

```

**Figure 4.58 Console error - Array Index Out Of Bounds Exception**

In order to cater for such an error been thrown, a try-catch block was needed. If the code in the try block, i.e. attempting to retrieve a previous message, generates an error then this piece of code stops executing and the exception block is executed. The piece of code to handle such scenarios is given below in figure 4.59.

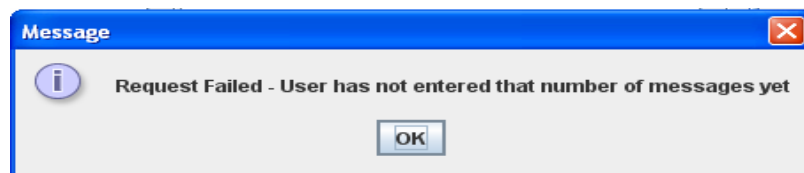
```

try
{
    lineToSpeak = userMsgs.get(username).get(userMsgs.get(username).size()-indexMsg).toString();
    System.out.println(lineToSpeak);
}
catch (Exception e)
{
    lineToSpeak = "Request Failed - User has not entered that number of messages yet";
    String title = "Error";
    JOptionPane.showMessageDialog(null, lineToSpeak);
    speakChat.speakMsg(line);
}

```

**Figure 4.59 Exception Handling Code - Array Index Out Of Bounds Exception**

This piece of code returned a user friendly alert as seen in figure 4.60 outlining the problem to the user while also synthesizing the error to the user so that full comprehension is achieved by the end user.

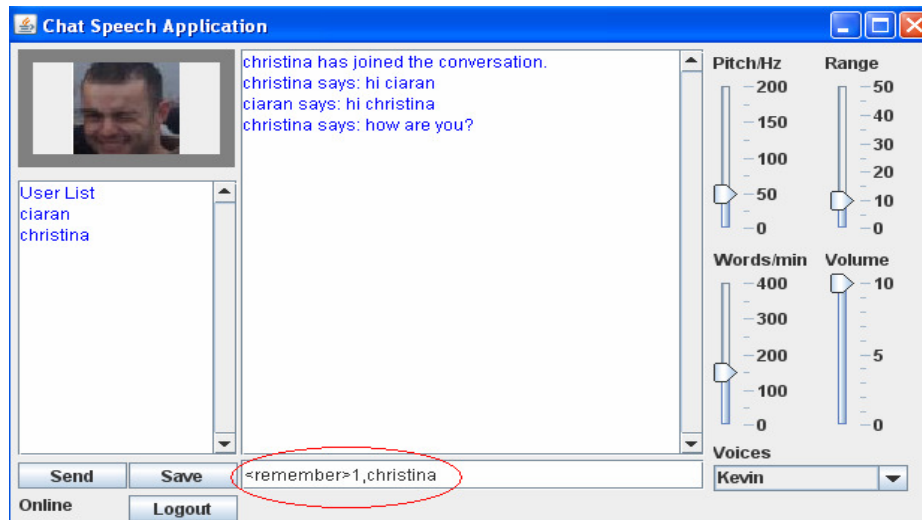


**Figure 4.60 User Friendly Error - Array Index Out Of Bounds Exception**

### 4.8.1.2 NumberFormatException

This error is thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format. In this application, such an exception can be thrown if the command does not contain the right syntax, as in the previous example. For example, the second parameter in the command '<remember>,2,ciaran' is a numeric character. This command is divided up into three separate strings separated by a comma for processing. If the wrong syntax is entered, say for

example, a comma is omitted then an exception is thrown. As in the interface below I have demonstrated how this is possible.



**Figure 4.61 Interface Command to generate Number Format Exception**

This command would be divided into two strings, '<remember>1' and 'christina'. The program would then try to convert the string '<remember>1' into an integer to be used as an index for hash table vector. Before any exception handling was introduced, the below error was generated from the command prompt.

```
ciaran says: <remember>1,christina
java.lang.NumberFormatException: For input string: "christina"
    at java.lang.NumberFormatException.forInputString(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at ChatClient.getRememberedLine(ChatClient.java:255)
    at ChatClient.run(ChatClient.java:201)
    at java.lang.Thread.run(Unknown Source)
```

**Figure 4.62 Console error - Array Number Format Exception**

The below exception handling code prevented the application from terminating abnormally. The program attempts to convert the second parameter in the command to an integer, and on failure the catch block is executed instead.

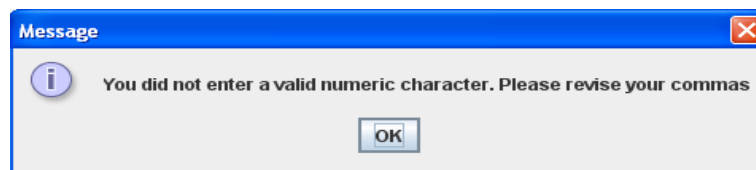
```

try
{
    Integer.parseInt(splittLine[1].trim());
    indexMsg = Integer.parseInt(splittLine[1].trim());
    username = splittLine[2].trim();
}
catch (Exception e)
{
    lineToSpeak = "You did not enter a valid numeric character. Please revise your commas";
    JOptionPane.showMessageDialog(null, lineToSpeak);
    speakChat.speakMsg(line);
}

```

**Figure 4.63 Exception Handling Code – Number Format Exception**

The above code again enabled a user friendly error returned to the user without the application freezing. The result of the above catch block being invoked can be seen below in figure 4.64



**Figure 4.64 User Friendly Error – Number Format Exception**

## **4.9 Conclusions**

In this chapter it was clearly demonstrated and proven that a speech synthesis chat application can be developed using the architecture and tools outlined in section 4.4. The various classes used in developing this system, and how each of these classes is associated with each other, were addressed separately in section 4.5. Using the methodology outlined in section 4.2, it has also been demonstrated that in an iterative development process, the additional functionality as outlined section 4.7 can be appended to the system with little or no implications to the rest of the system. An overlap of use case diagrams and the prototype interfaces outlined in chapter 3 and the actual screen shots of the working application was clearly demonstrated in section 4.6, highlighting the relationship from the initial requirements through to the design and into the implementation of these requirements. Finally section 4.8 looked at the importance of using proper error handling in any system.

## **5. Testing and Evaluation**

### **5.1 Introduction**

In the previous chapter, we saw how it is possible to develop a speech enabled chat application. We saw how it is possible to integrate ‘network communication mediums and open source speech synthesis’ in order to produce a functionally working real time communication medium for the visually impaired to use at ease. In this chapter, testing and evaluation, we will first see the different methods used for testing the system. We will see the types of testing methods carried out, a test plan, example test cases with inputs and results, the features tested, and environmental needs. In the latter part of this chapter we will discuss the evaluation process used to qualitatively determine the success of the implementation carried out in chapter 4. This will be achieved using questionnaires to be answered by end users of the system and then evaluating the system’s performance on these answers.

### **5.2 Testing**

Sami Zahran stated in his book ‘Software Process Improvement’ that roughly 4,000 people have died over the last 15 years due to software defects (Zahran, 1998). Such atrocities in military combat, aviation and the aerospace industry could have been prevented with proper testing plans and documentation in place. With software continuing to play a significant role in every day life, it stands to reason that there is more of a need that the software developed is of a sound nature with as few defects as possible. After all there is not point in producing software with some of the latest tools and technologies if it does not work. Software engineers are acknowledging that software should not be left to some one else, and that software testing should not begin once the development ends. (Ogilvy) believes that integrating testing directly into your development process can produce more robust and error-free code.

#### **5.2.1 Development of Test Plan**

Over the years various different test plans were thought up by companies and individuals to allow the control of testing for all phases of testing. As a result this has caused confusion among software testers as there was no agreed set of document standards and templates for testing. The IEEE (Institute of Electrical and Electronics Engineers, Inc.) have provided the 829 Standard for Software Test Documentation specifically for all types of software testing. The IEEE 829 standard was applied to this application as a basis for structuring the testing process. Although there are nineteen clauses to consider when building a successful test plan (IEEE 829), I have used the most suitable clauses for the testing of this application.

While this standard has a similar scope to and is based on the principles defined in IEEE 829, this standard has customized these principles for the chat application environment. The test

plan document for the chat application environment, and in accordance with IEEE 829, shall have the following structure:

- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Approvals

### **5.2.2 Test Plan Features**

The Test Plan for the speech enabled chat application was drafted by the author of this thesis and it describes the testing strategy to validate the quality of this product prior to release. It also contains various resources required for the successful completion of this project. The test plan will outline the scope, approach, and the schedule of intended testing activities. The test plan will identify the test items to be tested, the features that will be tested, and who will test each unit or item.

### **5.2.3 Schedules and Resources**

The developer (author) will test each unit or component as it is built. In practice though, as each code component is built, it is handed over to the testing team for finer grained testing before released to UAT. The developer does not usually test their own work.

Once the use cases have been tested and confirmed as working, they can be escalated to the UAT environment for the end user to test. Any errors encountered will be recorded in a log. The resource for such recording of errors will be MS Access. Once the defects have been

ironed out, the normal test suite can begin again. It should be noted that some of these testing stages will need to be repeated and/or overlapped.

Generally speaking, once the UAT team has signed off, the code is deployed to the production environment in the next release. For the first week of the new system being in a live environment, the development team works with the production support team in aiding the usage of the system for the end user. As there are no automatic test scripts needed for the testing of this application, the main resource needed is the tester's time.

#### **5.2.4 Testing Approach**

Unit testing will be undertaken by the developer (author of this thesis). Approval of unit testing will also be done by the developer (author of this thesis).

Integration testing will be performed by the test developer (author of this thesis). No specific test tools are available or required for this project. Programs will enter into integration test after all critical defects have been removed from unit testing.

Acceptance testing will be performed by the developer (author of this thesis) and by the actual end users with the assistance of the author of this thesis. The acceptance test will be done in parallel with the existing test script generated by the author of this thesis. As mentioned under the methodology section (section 4.2), the adoption of the Crystal Clear methodology enables the author of this thesis to play the role of the sponsor, senior designer, programmer, user, and tester.

#### **5.2.5 Features to be tested**

At the end of each development life cycle, all use cases as outlined in chapter 3 will be tested. For example, the use cases to be tested for an authenticated user are as follows:

- Log on
- View user list
- Enter message
- Submit message
- View conversation
- Retrieve Previous Message
- Acronyms Interpreted
- 'Alt' + 'L' – Logout of system
- 'Alt' + 'S' – Saving a conversation to the database
- 'Alt' + 'U' – Synthesizing list of logged in users



- Logout

As an example, I will now detail a specific use case for clarity. The test item to be explored further is for the use case 'log on'.

Log on test case in accordance with IEEE 829 test case specification template.

Test item	This test case validates the user's loginid and password and determines whether the use is authenticated to use the system or not
Input Details	
Input items	Data items to be used  username  password
Ordering	The username is entered first, followed by the password.
Values	No aschii character validation has been done as this system is a prototype
States	Test case executed once details are submitted from LoginScreen.java
Response Time	Response time should be no more than 4 seconds.
Output Details	
Message Responses	'Login Success' or 'Login Failed. Would you like to try again?'
Response Time	Response time should be no more than 4 seconds.

### 5.2.6 Environmental Needs

**Hardware:** properly configured hardware and appropriate arrangement of connection to the network for both the server and client.

**Software:** the correct version of Java, FreeTTS, MySQL (including jdbc driver), Sphinx-4, class paths and libraries configured as outlined in section 4.3.3.1 (setting up environment)

### 5.2.7 Item Pass/Fail Criteria

Since this is still a small application compared to industry standards, for each individual test item to pass it is a requirement that no defects are reported. Also, the result returned from all test items tested combined must also return no defects (see test-script). Such a tight constraint is feasible since this project is still relatively small and contained.

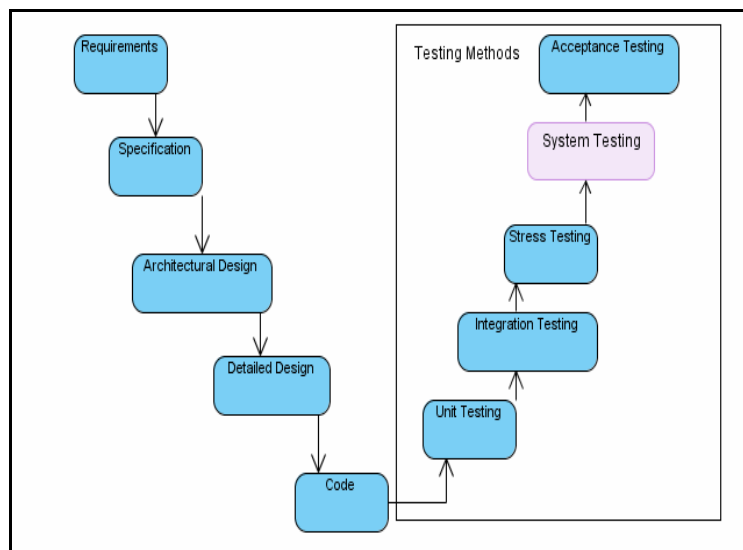
### 5.2.8 Test Deliverables

The following will be delivered as part of the test plan:

- Test plan document.
- Test cases.
- Test design specifications.
- Error logs and execution logs.
- Problem reports and corrective actions.

### 5.3 Types of Testing

In the diagram below we can see how the testing phase fits into the design and development phases. The design phase was a prelude to the development phase which in turn is a prelude to the testing phase. We can see in figure 5.1 the different types of testing which was done in this application, and highlighted is the ‘system testing’ to show that although it was not used in the test plans for this system, this is where it would fit in if it was used. This diagram was produced in accordance with the V model (V-model).



**Figure 5.1 Integration of Testing methods into Design and Development of System**  
[source: author using Visual Paradigm]

#### 5.3.1 Unit Testing

Unit testing is a method of testing and verifying that each unit of code works as designed before proceeding with further development. A unit is the smallest testable part of an application that developers can discuss as an independent entity (Redmonk, 1999). In

computer programming, this may be an individual class, method, or any piece of code that delivers functionality to the end user. The source code is tested in isolation from the main application. Unit testing proved to be critical in the design of this system, as it was important to be able to test certain parts without having to wait (Rajendran) on user / developer feedback on other parts. Such feedback related to advice being given from fellow developers and on performance and functionality feedback from typical users of the system.

### 5.3.2 Integration Testing

Integration testing is the next stage in the testing phase. Once all the units have passed testing in isolation, the individual units were tested together as a whole. Although modules of code may work in isolation, they may not always work in an integrated environment. This was the case for this application because this application involved numerous individual units calling various different classes, some even on the other end of the network.

For example, it was important to test that the ChatServer and ChatClient classes worked in isolation but it was pivotal for such a chat communication application to work as a whole. Integration testing unveiled further defects in the code with regard to Socket programming issues, Connection refused exceptions; Connection reset exceptions, and Socket binding connections. In the screenshots below we can see three examples of how ‘integration testing’ revealed bug defects. In the first example, we can see the Client attempting to connect to the Server but the connection is refused. Although both units and classes worked in isolation, connection concerns were raised during integration testing.

```
C:\Program Files\Java\jdk1.6.0_06\thesis-test>java ChatClient
java.net.ConnectException: Connection refused: connect
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.PlainSocketImpl.doConnect(Unknown Source)
    at java.net.PlainSocketImpl.connectToAddress(Unknown Source)
    at java.net.PlainSocketImpl.connect(Unknown Source)
    at java.net.SocksSocketImpl.connect(Unknown Source)
    at java.net.Socket.connect(Unknown Source)
    at java.net.Socket.connect(Unknown Source)
    at java.net.Socket.<init>(Unknown Source)
    at java.net.Socket.<init>(Unknown Source)
```

**Figure 5.2 ConnectException (generated by developer).**

The second socket programming exception occurred by the client ending its connection to the server ungracefully. Such an exception was handled by closing all open stream connections to the server before closing the instance. Such an exception was handled using the close method in socket.close().

```
java.net.SocketException: Connection reset
    at java.net.SocketInputStream.read(Unknown Source)
    at java.io.BufferedInputStream.fill(Unknown Source)
    at java.io.BufferedInputStream.read(Unknown Source)
    at java.io.DataInputStream.readUnsignedShort(Unknown Source)
    at java.io.DataInputStream.readUTF(Unknown Source)
    at java.io.DataInputStream.readUTF(Unknown Source)
    at ChatHandler.run(ChatHandler.java:54)
```

**Figure 5.3 SocketException (generated by developer).**

The final of examples where defects were discovered during 'integration testing' involved multiple instances of the ChatHandler object listening for incoming connections on the same port. This resulted in the below error being returned on the console because two sockets cannot connect to the same port on the same machine simultaneously (SunManagers).

```
Error Address already in use: JVM_Bind
java.net.BindException: Address already in use: JVM_Bind
    at java.net.PlainSocketImpl.socketBind(Native Method)
    at java.net.PlainSocketImpl.bind(Unknown Source)
    at java.net.ServerSocket.bind(Unknown Source)
    at java.net.ServerSocket.<init>(Unknown Source)
    at java.net.ServerSocket.<init>(Unknown Source)
    at ChatHandler$UserListServer.<init>(ChatHandler.java:112)
    at ChatHandler.<init>(ChatHandler.java:33)
    at ChatServer.main(ChatServer.java:43)
```

**Figure 5.4 BindException (generated by developer).**

### 5.3.3 Stress Testing

The idea of stress testing is to push the boundaries of the system on all fronts to the point of failure. As this application is a prototype 'stress testing' was not the main method of testing but a certain element of testing was used. The 'stress testing' carried out on this application involved setting up a few different users on the system by adding new records to the database tblUsers table. I then ran a few different instances of the application in a local environment (on one IP and multiple IPs). I envisioned how a normal conversation would be carried out among five different users on a typical chat application and proceeded to create such an environment. The results showed that the ChatHandler could handle and broadcast all incoming messages from any of the clients. The Speech Synthesizer worked perfectly too, however when I used just the one IP address I noticed there was an echo from the synthesizer. This just meant that all instances of the ChatClient calling the speech synthesizer were working. When more than one IP was used the echo was removed and the synthesizer worked on all remote machines under the same network.

### 5.3.4 System Testing

System testing was not involved in the testing of this application as this type of testing generally only applies to projects within the industry that have separate Q&A teams.

### 5.3.5 Acceptance Testing

The final phase during testing is the 'acceptance testing' which is carried out by the users of the system. 'Acceptance testing' is also sometimes referred to as User Acceptance Testing (UAT) or Beta Testing. For this thesis, this involved allowing user's of the system test the application. The users in turn confirmed if the requirements and functionality as outlined in the design stage have been met or not.

## 5.4 Software Evaluation

### 5.4.1 Introduction

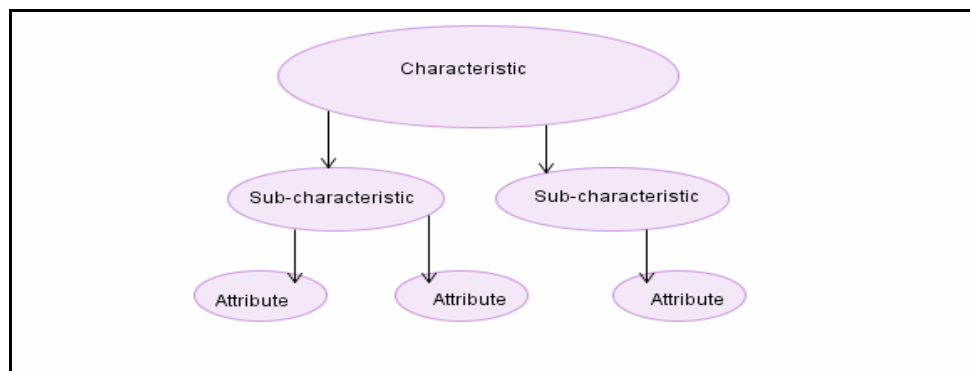
In this section the chat application product's quality is determined and evaluated. Software evaluation is a process which helps to improve the software product quality. The ISO (International Organization for Standardization) is the world's largest developer and publisher of international standards. In evaluating the chat application as an end product, the evaluation process used was based on ISO 14598 (ISO 14598) and in determining the quality of the application as an end product it was based on ISO 9126 (ISO 9126).

#### ISO 14598

The ISO 14598 standard can be used by various different types of users to evaluate software. Such users can be acquirers, users or evaluators, and developers (ISO 14598). For this particular application to be evaluated successfully we are only concerned with the evaluation being done by evaluators or the end users. In standard ISO 14598, to achieve software evaluations the various entities, components, and attributes need to be quantitatively measured using validated metrics. In order to evaluate the software, the ISO 14598-5 (process for evaluators) standard was used in conjunction with the ISO 9126 standards which describe the product quality characteristics and metrics.

#### ISO 9126

According to ISO 9126 there are six quality characteristics which can be applied to a software product, they are functionality, reliability, usability, efficiency, maintainability, and portability. Each of these characteristics can be subdivided into sub-characteristics which in turn can be divided further into elementary quality attributes that can be measured and rated (ISO 9126).



**Figure 5.5 ISO 9126-1 Quality Model Characteristics [source: author using Visual Paradigm].**

These individual elementary subsections of the software can then be measured and rated using customized metrics. The metrics adopted for this product was the making of a checklist

where the various components of the software and its attributes were transformed into question format to be answered by the evaluator.

Since the quality characteristics cannot be measured in a quantitative manner, there needs to exist some sort of quality metric that maps the software quality characteristic to a set of numeric terms. Each measure of quality, presented in question format, that the end user answers will be a numerical value between 1 and 10. This new value can then be interpreted to indicate how satisfied the end user is with the application.

The feedback obtained from the user can then be assessed and the software quality evaluated. The result will indicate whether the software product meets the quality requirements.

### **5.4.2 Software Evaluation Process**

The research methodologies used to gather the information needed to address the evaluation objectives were as follows:

- Primary research: interviews and questionnaires
- Secondary research: review of literature
- Critical Analysis
- Data collection

In this thesis, interviews and questionnaires were the major sources of data to provide quantitative and qualitative data. Such methods of acquiring data provided the opportunity to allow the end user, some visually impaired, to comment on how they felt the speech enabled chat application was for them. The visually impaired users had very positive and powerful feedback, more so than the sighted users.

The questions used in the questionnaire were designed to be simple and open ended to allow the evaluators to express clearly and openly their impressions of the system. I felt such feedback would be more beneficial in going forward with future redesign and additional functionality.

### **5.4.3 Rubric Provided**

The researcher of this thesis developed an evaluation rubric which is a paper based instrument to assess and evaluate the quality of the software system developed. According to Wikipedia, rubrics allow for 'standardized evaluation according to specified criteria' in order to 'assess criteria which are complex and subjective and also provide ground for self-evaluation, reflection and peer review'.

The rubric provided to the interviewees was composed of a number of questions which allowed them to express clearly and openly their thoughts of the system. The idea of using open ended questions meant that more open and fruitful answers could be obtained. I felt this was more important in gaining an overview to properly evaluate the success of researching, designing, and implementing such a voice enabled chat application for the visually impaired. In successfully evaluating the software system developed, four interviews were conducted. I have used a coding scheme, given below in figure 5.6, throughout this research to conform to confidentiality and anonymity.

	<b>Gender</b>	<b>Code Throughout Research</b>
Interview 1	Female	F1
Interview 2	Female	F2
Interview 3	Male	M1

**Figure 5.6 Interviewees Coded.**

Since this research is about visual issues, it was imperative that the information obtained from the rubric was obtained from visually impaired users, and / or users with sight impairments. All three of the interviewees had a strong background in using computers (F1, F2, M1). One particular interviewee (F3) also had touch typing skills. Since this system was designed for the visually impaired it was important that most of the interviews were conducting with people who had visual impairments.

#### **5.4.4 Evaluator Feedback**

The characteristics for the four interviewees were as follows:

- F1 represents a female teenager who wore spectacles with a strong lens
- F2 represents a female in her early 50s who has been blind since birth.
- M1 represents a male in his late 50s who has no visual impairments. The interviewee M1 filled in the rubric twice. The first time represented the results of M1 using the system normally, while the second set of answers (interview 3b) reflected M1's thoughts of using the system whilst blind-folded.

The results obtained from the rubric were far more beneficial in evaluating the speech enabled chat application.

### **What are your initial impressions of the system?**

The first impressions of any system are very important to any user as this can have a lasting effect for them.

When this question was put to F1 the answers obtained were not as positive as were the answers from the other interviewees. F1 stated *"I use MSN a lot but it would be better if this tool had more of the features that MSN and the likes had such as changing the background to an image of my choice, games, and image icons"*.

In reviewing this answer, this application is just a prototype and more development can and will be done, but as it's a communication medium for the visually impaired there should not be as much emphasis placed on entertainment features that other main stream applications facilitate.

F2 had a completely different and more optimistic view on this application. F2 stated *"Wow, this is a really good idea. I'm surprised I haven't heard of this idea before."* They also found the system easily navigable. This answer was very welcoming as such a concept was a concern from the beginning. It was no surprise such a more positive first impression came from a blind user.

When this question was put to M1, he stated *"I had problems initially starting off but as soon as I was in the main chat lobby it was easier to interact with the system."* He also mentioned how he thought that it was good idea to integrate IM software with speech tools. M1 felt that it was *"good for visually impaired users to socialize via the internet as they probably wouldn't socialize as much as sighted people like over a beer etc"*. (Kinght et al, 2002) mentioned that 54% of disabled people found that the internet was a necessity which could be to do with a socializing concept more so than anything else such as online shopping.

### **Did you find it difficult starting off?**

M1 stated *"as well as finding it difficult to first get into the application, I also found it difficult to use the shortcuts provided since I'm so used to using the mouse"*.

F1 was not pleased with the quality of the synthesizer and found it a little robotic. F1 asked if there *"were better synthesizers we could use instead"*.

However, on the contrary to F1's view on the synthesizer's quality, F2 was not surprised and that it sounded similar to most synthesizers she had heard before. She also reiterated M1's opinion on finding it straight forward with regard navigating throughout the system and 'being in control'.



### **What features did you like best about the system?**

F2 stated “*I loved the idea of synthesizing old messages as this is always an issue for blind users*”. Such a remark comes as no surprise from a blind user. This user told me that trying to remember previous content just synthesized was ‘very frustrating when using the web’ in particular. F2 also found that it was great to have such technology free of charge, as other screen readers such as JAWS was expensive. This related well to what (Fichten, 2001) highlights when they found that cost was a huge factor for adaptive computer technologies.

F1 replied to this question saying “*I love the idea of a computer talking to me, it’s so cool*”. For those who do not usually hear speech synthesis in their everyday life this was no surprise.

M1 appreciated the concept of how each key was synthesized and appreciated how such a concept would be of great benefit to the visually impaired. M1 demonstrated this by using a blind-fold and of roughly 100 words typed no mistake was made. This experiment proved that this particular piece of functionality worked perfectly.

### **How do you think it could be improved in the future?**

M1 had some very interesting ideas regarding this question. M1 wanted to know whether the synthesizer control settings were saved so that every time he logged back on he wouldn’t have to reset them each time. Having given time to think about this idea, it would be possible to add this functionality onto the system. This could be achieved by saving the preferred settings to the tblUsers table on the database. Then each time the user logged on these settings could be retrieved and applied to the interface.

F1 suggested the option to allow the user enable or disable the speech synthesis.

F2, the blind user, suggested synthesizing the component, button, or control that the mouse was placed over on the interface. With a little bit of work this could be achieved through mouse coordinates. If the mouse coordinates were within a specified range around a particular component, then the synthesizer could synthesize this particular component.

### **Would you like to add any further comments about the system?**

F2 stated she found the system very user friendly and “*always felt in the loop*” when discussing the system and this was achieved through ‘constant synthesized feedback’. This was great to hear because (Pew, 2003) found that people with disabilities had a more negative opinion on technology than able bodied people which is related to software not being entirely user friendly.

M1 went as far as requesting ‘full time access to the system’ and how does one set oneself up.

## ***5.5 Conclusion***

We have learned that this speech enabled chat application is critically important to these users. It is liberating. For many of our participants, this application has brought a greater sense of independence than ever before. It allows them to "read" a conversation for the first time. It gave them an easy way to socialize and interact. It made them less dependent on others to tell them what people had said if they were chatting using one of the main stream chat applications.

## **6. Conclusions**

### **6.1 Introduction**

In this conclusion chapter we will revisit this dissertation and the research involved, give a summary of the findings found in the previous chapters, list the problems encountered in the research and investigation, and outline plausible suggestions for future work in this area of research.

### **6.2 Key Findings**

#### **6.2.1 Review of Literature**

In chapter 2, the literature review, there were numerous papers, books, conference readings, and journals on software accessibility, assistive technologies, instant messaging software, and open source software. It was not one of these, but all these topics, which combined gave some belief and structure in recommending a path to combat the issues regarding accessible instant messaging software for the visually impaired to use.

It is clear that more and more people are using computers in today's world which is due to improvements in computing technology, better high-speed broadband connections, and the reduced cost of these technologies. Despite the advancements in software and technology, software accessibility remains a significant issue for the visually impaired. This has been documented in a paper 'Is the web truly accessible to the Disabled?' by McGrane (2000) where 98% of all websites pose some form of accessibility constraint. Such negative statistics has caused difficulty or even prevented roughly 10 million blind or visually impaired users from participating in the cyber society of the modern age (AFB – American Foundation for the blind).

Instant Messaging (IM) software is a relatively new technology compared to most software applications. Most websites have been forced to conform to W3C guidelines. It is a new form of social interaction that has gained acceptance by people from all generations for many different purposes ranging from socializing (Oblinger, 2004), education, and corporate related agendas (Tang, Yankelovich, Begole, 2001).

(Pew, 2003) found that people with disabilities had a negative opinion on technology. This could well be due to the fact that software, as mentioned above, is not entirely accessible by all. The literature review also revealed that assistive technology is not always readily available which is due to inadequate funding for adaptive computer technologies (Fichten, 2001). This is the case because adaptive technologies have been specifically designed for a particular segment for the market. Also documented in the literature review was that most of these technologies, such as JAWS, are closed source meaning that the software is not distributed with the executable to be reused again in other applications. This highlights a need to use existing open source software such as speech enabled technologies with open

source speech tools such as Java to develop create a new application that the blind would benefit from.

The above literature combined paves the way for developing and instant messaging chat socializing tool coupled with speech enabled technologies for the visually impaired to use. IM allows people to discuss real experiences in virtual spaces, where often physical accessibility barriers may prevent real life community situations occurring for the disabled.

### **6.2.2 Review of design**

The design stage proved very important in this thesis as it helped develop the right system for the users. UML is ideally suited for the design of this system. The various UML diagrams gave structure to what classes and methods were needed for the development of this project. The domain analysis provided a direct mapping from the objects in the real world to the various components of the system. The state and activity diagrams outlined the various states the objects could be in given a scenario. Finally the sequence diagrams proved very useful with regard to parameter passing among the various classes and methods of the system. The sequence diagrams also helped in documenting the sequence of steps which must be completed in order for the use case return a completed function to the end user. However, the collaboration diagram, sometimes used instead of the sequence diagram provided no more value to the design as they only documented what the sequence diagram had already documented.

The prototype screenshots helped clarify in my own mind how the UI was going to work. The initial prototype screenshots were designed to be very basic as it was important to not give any false indication of progress to the end user (Fowler, 1998). Such initial prototype GUIs gave direction in what the end system would look like. It was important to have an initial plan that could be visualized from early on.

### **6.2.3 Review of Development**

Having completed the development of this application, it has been established that such an accessible communication tool can be implemented using the Java programming language and open source speech enabled technologies. It has also been shown that such technologies can be easily integrated into one system as the programming language for all components in this thesis was Java. The scientific conclusions also highlight that the unlimited tuning of open source software can be performed in order to deliver the exact system requirements to the end user.

## **6.2.4 Review of Testing and Evaluation**

It was interesting to investigate standards associated with both software testing and software evaluation. IEE 829 proved to be a very comprehensive test plan in controlling the testing of this application. The different types of testing proved exceptionally important in this project as not one, but the various types of testing together, proved very influential in removing bugs from the system.

In evaluating this project as an end product, the evaluation process, based on ISO 14598, and the process of determining the quality of the application, based on ISO 9126, proved very important in gaining substantial feedback with regard to quality and success of this project.

## **6.3 Problems Encountered**

There were numerous difficulties and obstacles encountered throughout this thesis from the offset through to completion. There were numerous technical problems encountered during the development phase of this thesis. Coding of any project is never straight forward and there were times when it proved impossible to deliver certain functionality in a certain way. This was largely because it was beyond the capabilities technologies used. In these circumstances, such problems were addressed by requesting the aid from professionals in the area of Java programming through online forums such as Sun and DevShed. Another difficulty was that these experts delayed a few days before responding which delayed development.

Environmental issues and configuration settings also caused many problems when certain libraries and dll were being used. To overcome such difficulties involved editing the classpath variable a few times.

One of the difficulties associated with researching instant messaging accessibility was that access to material on this specific topic was only of a marginal amount. There was a lot of material which covered software accessibility but this was mainly confined to websites, and of the instant messaging literature that was found there was very little which addressed the concept of accessibility. However this problem was addressed by reading a variety of literature sources of all aspects of this dissertation, and also by engaging in discussions with experienced users in order to acquire direction and assistance on this dissertation.

The copying of certain images from visual paradigm, the software IDE tool used in the creation of system diagrams, to this document proved difficult. For example, the detailed class diagram depicting the whole diagram of the system was too big to select as one image. In order to address this issue, the diagram needed to be copied as two separate diagrams which were then rejoined using MS Paint before being copied.

There were performance issues with the synthesizer as some users felt the synthesis was not of a high enough quality. Again this was beyond the capabilities of modern speech synthesizers which has to cater for such a large vocabulary. However this system was

designed so that other voices could be imported once they become available. The synthesizer also behaved differently to what was expected when attempting to synthesize certain keys that were pressed. For example, when the numlock functionality was enabled it synthesized a - i when 1 – 9 were pressed. When the numlock was disabled the word ‘hash’ was synthesized for 1, the word ‘percent’ was synthesized for 4, the word ‘dollar’ was synthesized for 7, and the word ‘ampersand’ was synthesized for the number 8. The remaining numbers from 1-9 returned nothing. There were also issues with the synthesizer pronouncing words with great confusion. This issue could almost be addressed in another thesis as it is so broad.

## **6.4 Conclusions**

This research thesis set out to review current software accessibility, in particular Instant Messaging (IM) Software. A great deal of desk-based research was undertaken in order to complete this thesis. The relevant literature was researched such as software accessibility, instant messaging, and open source software in a quest to combining this selection of existing literature and attempting to provide something new and beneficial for the visually impaired.

It has been clearly demonstrated and proven in this thesis that an accessible communication and socializing tool can be developed for the visually impaired. It has also been clearly demonstrated that open source speech enabled technologies can be fine tuned and used to cater for one’s desires. Such a concept proved exceptionally important for this application as each incremental release provided more speech synthesis functionality. In order to evaluate the development undertaken in this thesis a rubric was created which invited user’s thoughts about their use of the system. The results obtained from section 5.3 help confirm that this newly developed speech enabled chat application specifically designed for the visually impaired represents an important building step in software accessibility.

## **6.5 Future Work**

### **6.5.1 Speech Synthesis**

This thesis has undergone a huge amount of research to help completing a functionally working chat enabled communication medium. A great deal more work can be done in the area of speech synthesis, as the additional functionality section proved. There is an endless amount of work and experimentation that can be applied to the speech synthesis of this particular application. by importing voices from Festival, it would be interesting to try to allocate different voices to different people in the conversation. For example, if the conversationalist was female then a female synthetic voice could be allocated for this particular user.

### **6.5.2 Speech Recognition**

Speech recognition could be integrated into this chat application in a similar manner to the speech synthesis approach. By integrating such a feature, this would really make this

application a truly speech operated device for those who not only having visual difficulties but also for those who dislike using the keyboard as the primary input.

Social-psychological research has shown that people treat media the same way as they treat other people. Professors Byron Reeves and Clifford Nass from Cambridge University say that a speech recognition system is viewed as a social actor by the users of the system (Reeves, 1996). With this in mind it seems only natural to enable the use of speech recognition in applications. As already mentioned in this thesis it is important to treat computers more as humans rather than machines.

Speech recognition technology is constantly improving. Today's speech recognition systems can understand more than just one word or phrase at a time. A typical North American English speech recognition system is so good that it can match a single spoken utterance against a list of 80,000 items with upwards of 95% accuracy while still being able to respond to the caller with the next question or statement in tenths of a second (Kotelly, 2003).

### **6.5.2.1 Previous Work**

Speech recognition technology has thus far been deployed in a number of applications in a number of different industries from air transport (United Airlines) to freight forwarding (Fedex) (Juang and Rabiner, 2004). Voice dialling applications such as these can be used in a number of applications seeking to retrieve various types of information for the user. Car navigation is one of the latest speech recognition systems on the market day (Muthusamy, 1999). The VODIS (voice operated driver's information systems) project (Pouteau, 1997) was launched in Europe to investigate a robust speech interface for command and control applications of car functions. These types of applications face obvious problems such as cost, and accuracy. Naturally word recognition accuracy would deteriorate in the car environment due to noises coming from the engine, the wind, the car radio, the horn, etc.

Voice dialing is a simple task and everybody believes that in the car environment the use of voice dialing can improve driver safety. However, a study in Canada (Redelmeier and Tibshirani, 1997) has demonstrated that the risk of having a car accident while using a telephone is comparable to the risk arising from the consumption of alcohol. Furthermore, the dangers are similar whether the telephone in use is of a hands free design or not.

In other words, the risk is associated with the verbal task in which the driver is engaged, and not necessarily with the physical manipulation of the telephone handset. Furthermore, (Boves and den Os, 1999) observed that voice dialing cannot be considered a successful application. They found that most ignore this add on to existing applications and only 10 to 20% of users actually use it. Although if the user of the system is blind and may have no other choice in communicating with the system then this figure would certainly not be as low.

### **6.5.2.2 Performance Issues**

Earlier studies (Hirsch, 1989) have shown that the recognition rates of word recognition systems deteriorate considerably for a hands free speech input in reverberant environments. The reason for this is the masking of the spectral features of certain phonemes. The

dominating vowels mask the following phonemes with less energy and less duration. Many users today use a hands free system to communicate either via mobile, Skye, Msn Messenger etc. If the end user of the system is blind then the chances are even greater that they will use a hands free speech as the primary input. Having said that, noise suppression algorithms have been developed to improve the noisy speech in such communication situations (Compernelle, 1989).

Since potential recognition to be used in this project would not involve the speech being transmitted across the network, there would be no bandwidth or line noise and distortion issues and therefore accuracy of the recognizer should not be affected. Every time someone speaks into a telephone microphone, the signal is compressed and transmitted across the telephone network, inevitably losing quality along the way (Kotelly, 2003). By the time the signal reaches the other end it can be very difficult for the recognizer to understand the muffled signal.

### **6.5.2.3 Trained Data Concerns**

Acoustic models used in speech recognizers are trained using data from people in the age range of 15-60. This then naturally poses an age group problem when one considers the interaction of the system and speech from other groups such as children and the elderly.

It has been proven that the word error rate among these groups is much higher than other age groups (Wilpon and Jacobsen, 1996) and reports have shown that children's speech is slower than that of adult speech and that children exhibit higher variability as compared to adults in speaking rate, vocal effort and degree of spontaneity (Lee et al, 1997).

Another area of recognition performance which one must not ignore when designing a system is the speaking rate of the user, and the consequences this will have on promising performance statistics (Walker, 2004). It was found that as people spoke more quickly, word error rate increased as the confusion between the phones also increased. It has been observed that for fast speakers error rate is more than double as compared to average speakers (Pallett, 1994).

Research has shown that the performance levels of speech recognizers are greatly reduced when non native speakers use recognizers which have been trained on native speech. This is because non-native speakers tend to use sounds that do not belong to the main language that trained the recognizer. Such "foreign" sounds are sometime called xenophones (Eklund and Lindström, 1998) and depending on how these xeneophones are dealt with, recognition performance will be affected.

### **6.5.2.4 Potential Speech Recognizer**

The Sphinx-4 speech recognizer was developed entirely in the Java programming language by researchers at the Carnegie Mellon University. As with the FreeTTS speech synthesizer, this speech recognizer is also open source meaning this recognizer could be altered and



designed with a dictionary that would suit the vocabulary needed by an instant messaging chat application.

According to the developers of the Sphinx-4 speech recognition system, the success and modularity was largely due to the use of the Java programming language. Wille Walker stated, 'the ability of the Java platform to load code at run time permits simple support for the pluggable framework'. He also said that the Java platform also provides Sphinx-4 with a number of other advantages such as: built-in support for multithreading makes it simple to experiment with distributing decoding tasks across multiple threads' (Walker, 2004).

However, (Moreno and Stern, 1994) found additive stationary noise, impulse noise, and low-frequency tones increased recognition errors when they used a commercial telephone channel simulator together with the Sphinx speech recognition system.

## Appendix A:

# Test Script

Date	Version	Author	Remarks
10/09/2008	0.01	Ciaran Reidy	Initial Draft

<b>Functional Spec Ref:</b>				
Step	Test Purpose / Sequence	Expected Result	Actual Result	Pass / Fail

<b>Functional Spec Ref:</b>		<b>Ensure that the new User has been created on the database.</b>		
Step	Test Purpose / Sequence	Expected Result	Actual Result	Pass / Fail
1.	Log in to the system	'Login Successful' or 'Login Failed – Would you like to try again?'	'Login Successful'	Pass

<b>Functional Spec Ref:</b>		<b>Ensure that all application functionality is working</b>		
Step	Test Purpose / Sequence	Expected Result	Actual Result	Pass / Fail

Functional Spec Ref:		Ensure that all application functionality is working		
Step	Test Purpose / Sequence	Expected Result	Actual Result	Pass / Fail
2.	View User List	The User List is updated every 1 second with the latest users logged in	As Expected – User List updated as users logged in / out system	Pass
3.	Enter Message	The ChatLine is enabled to allow the user to type message	As Expected – User successfully able to type	Pass
4.	Submit Message	The users message is sent to the server for broadcast and appended to ChatArea	As Expected – All Interfaces connected to server were updated.	Pass
5.	View Conversation	The correct messages in sequence are displayed to user screen	As Expected – User can view all messages	Pass
6.	Retrieve Previous Message	Previous Message synthesized	As Expected – The correct user and message retrieved	Pass
7.	Acronyms Interpreted	The correct meaning given an acronym is displayed and synthesized	As Expected – acronyms worked fine	Pass
8.	Logout	'You have successfully logged out of the system'	As Expected – connection to server closed gracefully	Pass

Functional Spec Ref:		Ensure all access keys function as expected.		
Step	Test Purpose / Sequence	Expected Result	Actual Result	Pass / Fail
9.	'Alt' + 'L' – Logout of system	'You have successfully logged out of the system'	As Expected – connection to server closed gracefully	Pass

Functional Spec Ref:		Ensure all access keys function as expected.		
Step	Test Purpose / Sequence	Expected Result	Actual Result	Pass / Fail
10.	'Alt' + 'S' – Saving a conversation to the database	You have successfully saved this conversation	As Expected – Message saved to the database	Pass
11.	'Alt' + 'U' – Synthesizing list of logged in users	'The following users are currently logged in...'	As Expected – User notified of the list of users logged in	Pass
12.				

Sign Off: Ciaran Reidy

## **Appendix B:**

# **Rubric Provided**

**What are your initial impressions of the system?**

**Did you find it difficult starting off?**

**What features did you like best about the system?**

**How do you think it could be improved in the future?**

**Would you like to add any further comments about the system?**

## Appendix C:

# System Commands

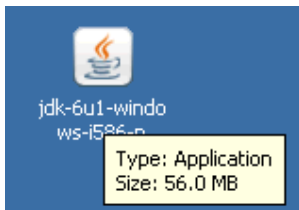
'Alt', 'Shift', 'S'	Save Conversation
'Alt' 'L'	Logout
'Alt' 'U'	Synthesize list of users logged in
'Alt' 'C'	Get ready to start chatting
'Alt' 'C'	Close out of Acronym box / Get ready to start chatting
'Alt' 'P'	Increase pitch of synthesizer
'Ctrl' 'P'	Decrease pitch of synthesizer
'Alt' 'S'	Increase pitch of synthesizer
'Ctrl' 'S'	Decrease pitch of synthesizer
'Alt' 'V'	Increase volume of synthesizer
'Ctrl' 'V'	Decrease volume of synthesizer
'Alt' 'R'	Increase pitch range of synthesizer
'Ctrl' 'A'	Add an abbreviation



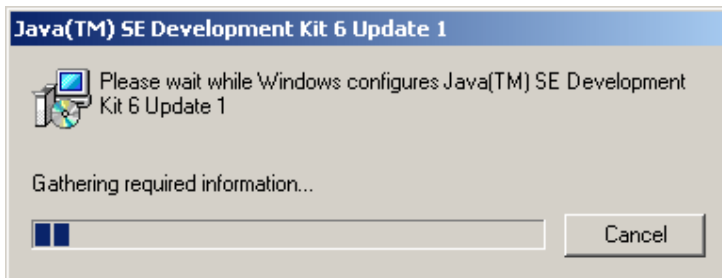
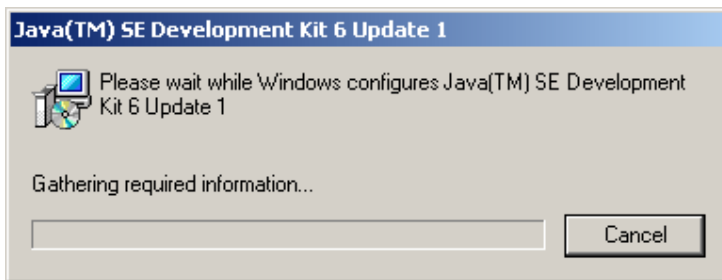
## Appendix D:

# Java Installation Guide

**Step 1.** Double Click the icon just downloaded from Sun's Website.

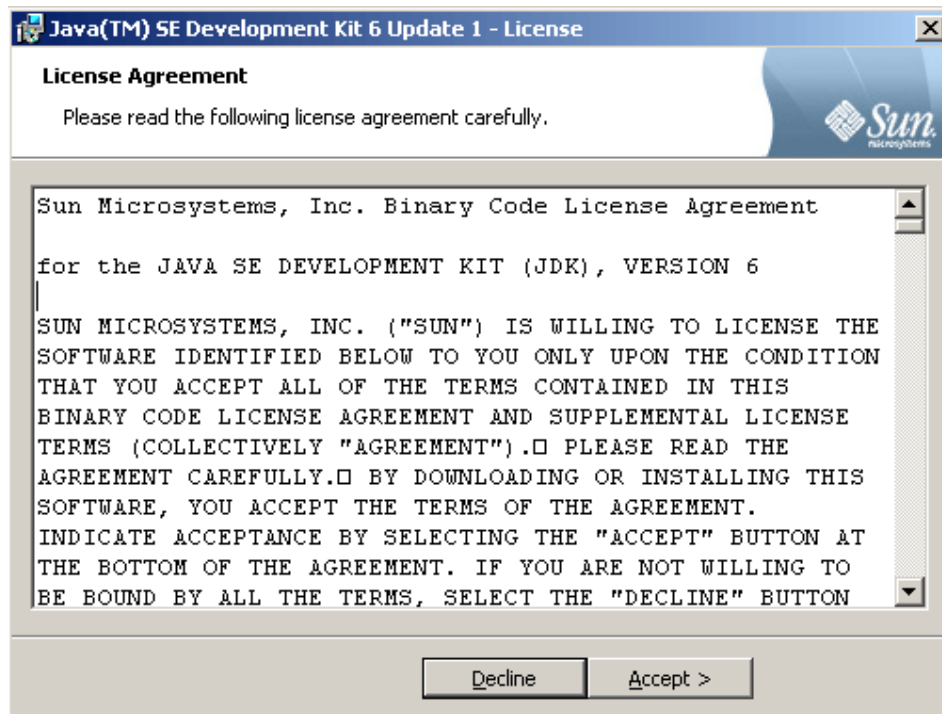


You will see jdk 6 update 1 window as shown below.

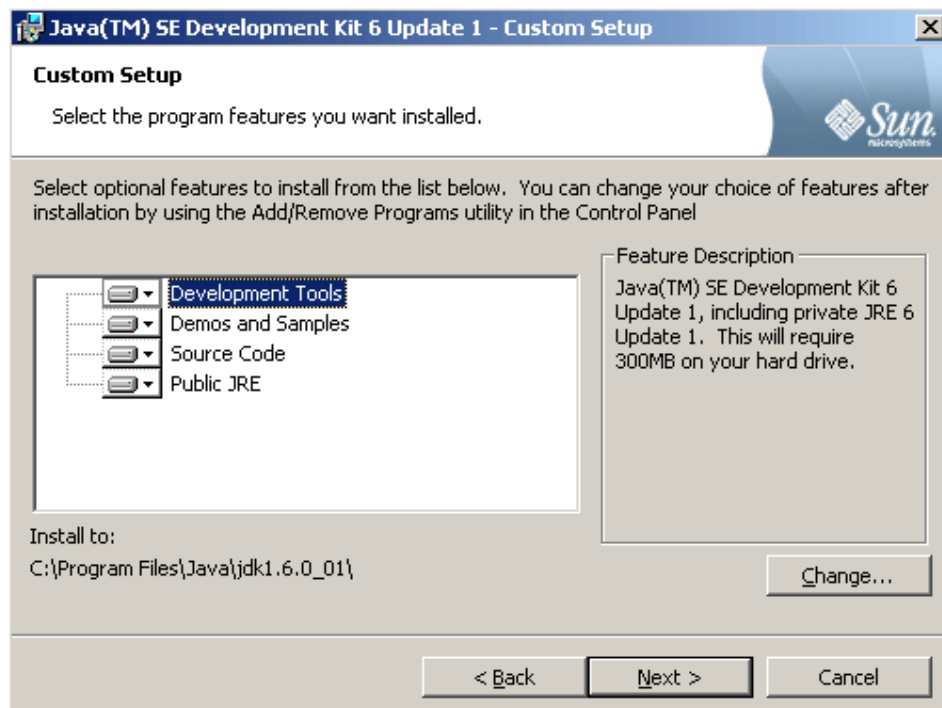


**Step 2:** Now a "License Agreement" window opens. Just read the agreement and click the "Accept" button to accept and proceed to the ext step.

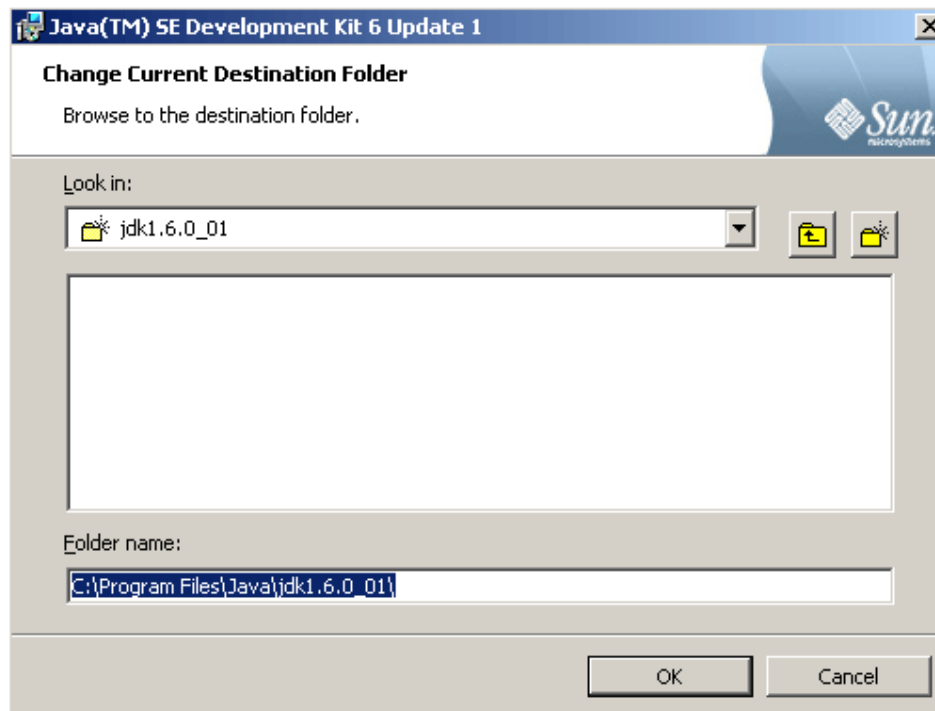




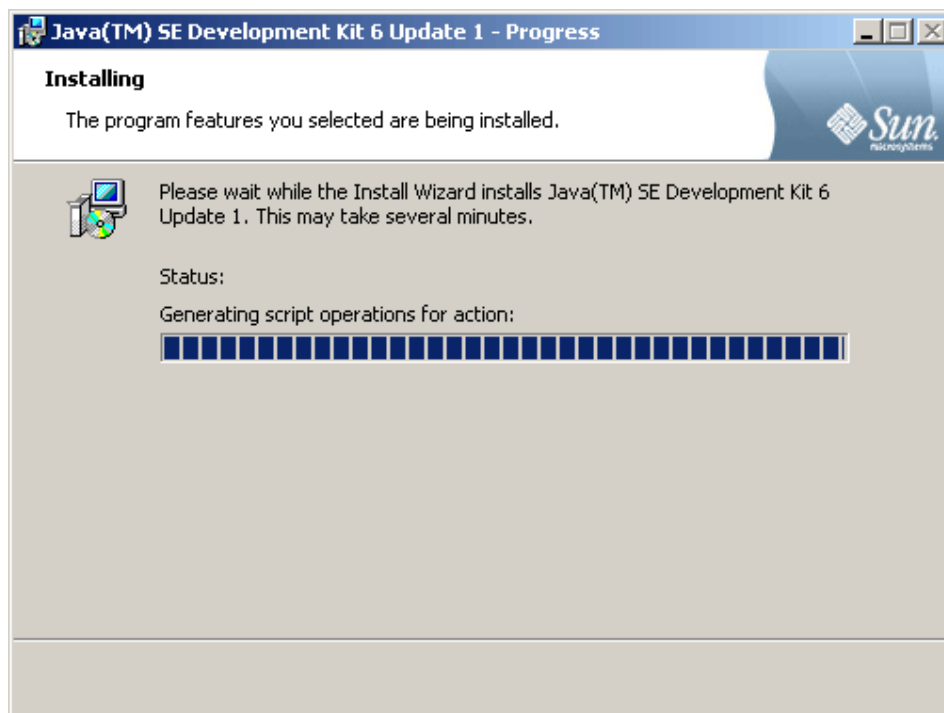
**Step 3:** Now a "Custom Setup" window opens.

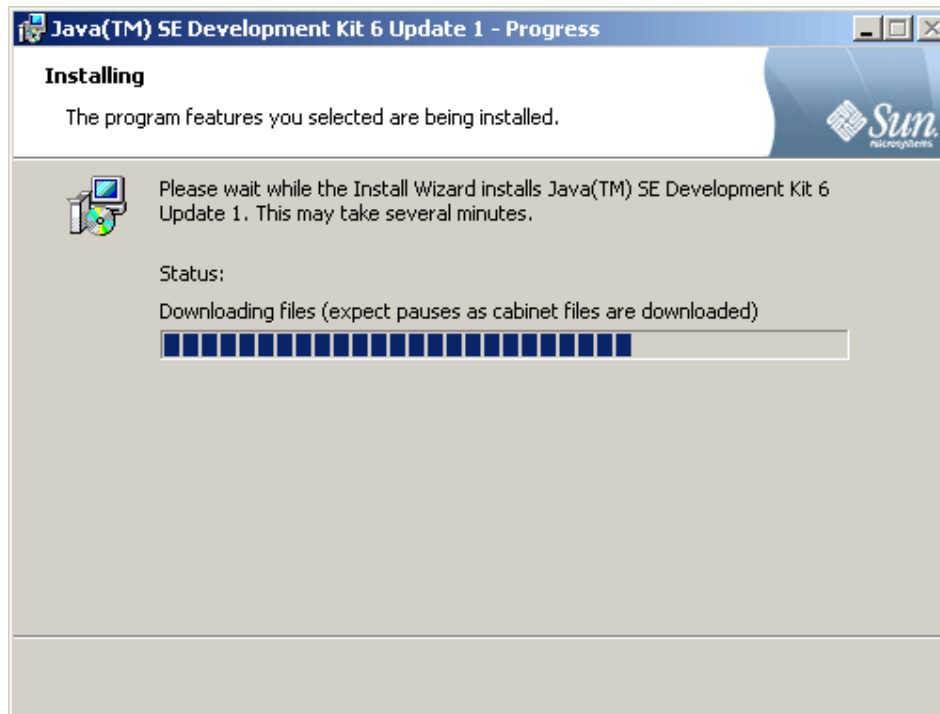


**Step 4:** Click on "Change" button to choose the installation directory. Here it is "C:\Program Files\Java\jdk1.6.0\_01". Now click on "OK" button.

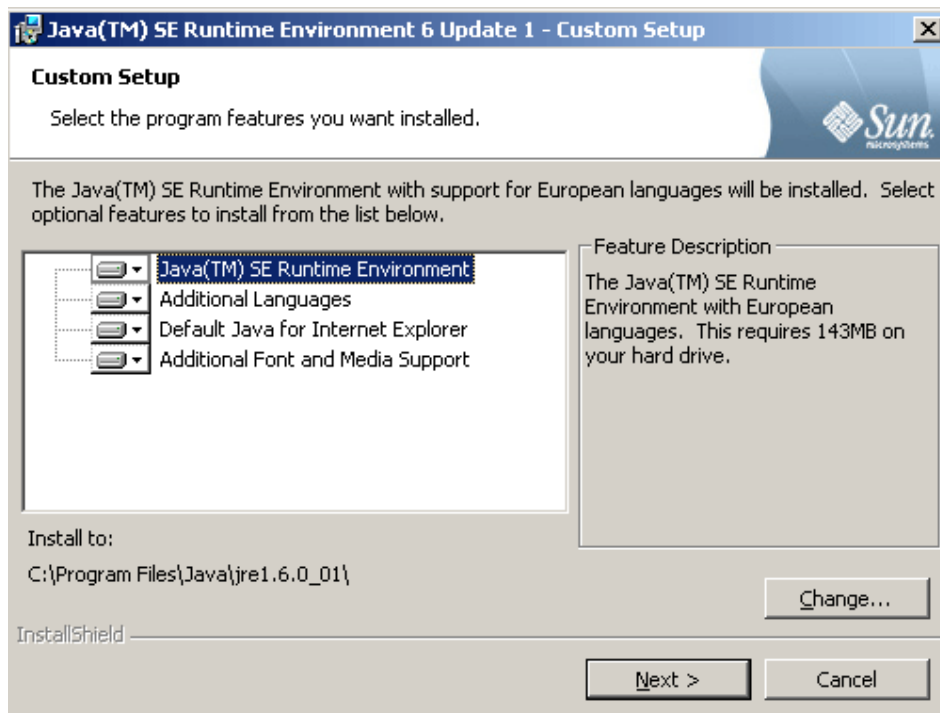


Clicking the "OK" button starts the installation. It is shown in the following figure.

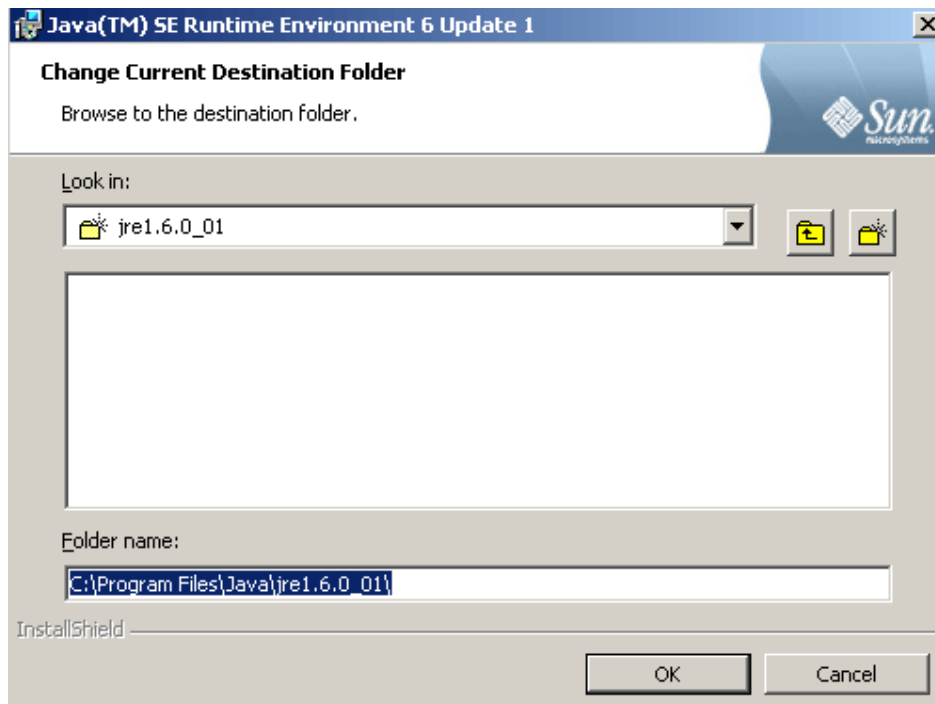




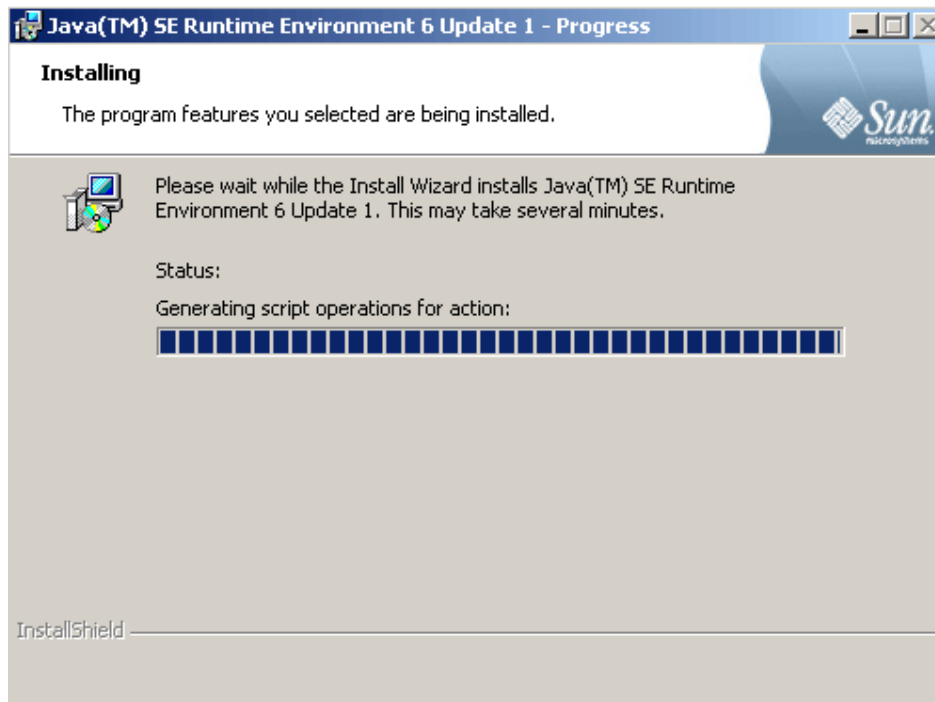
**Step 5:** Next window asks to install **Runtime Environment**.



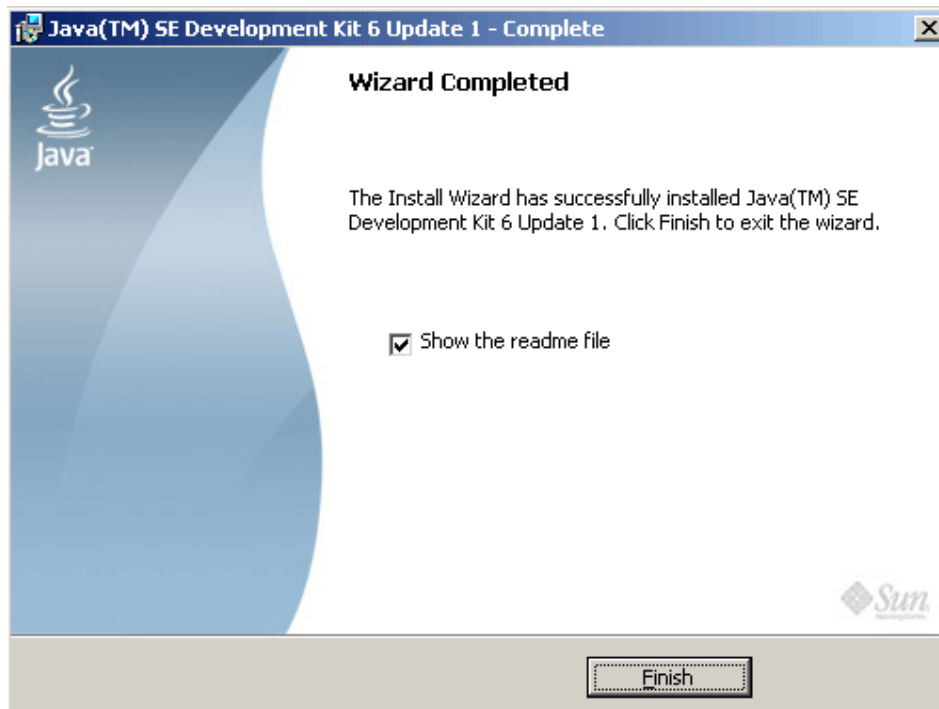
Click the "**Change**" button to choose the installation directory of Runtime Environment. It is best not to change it. So click "**OK**" button.



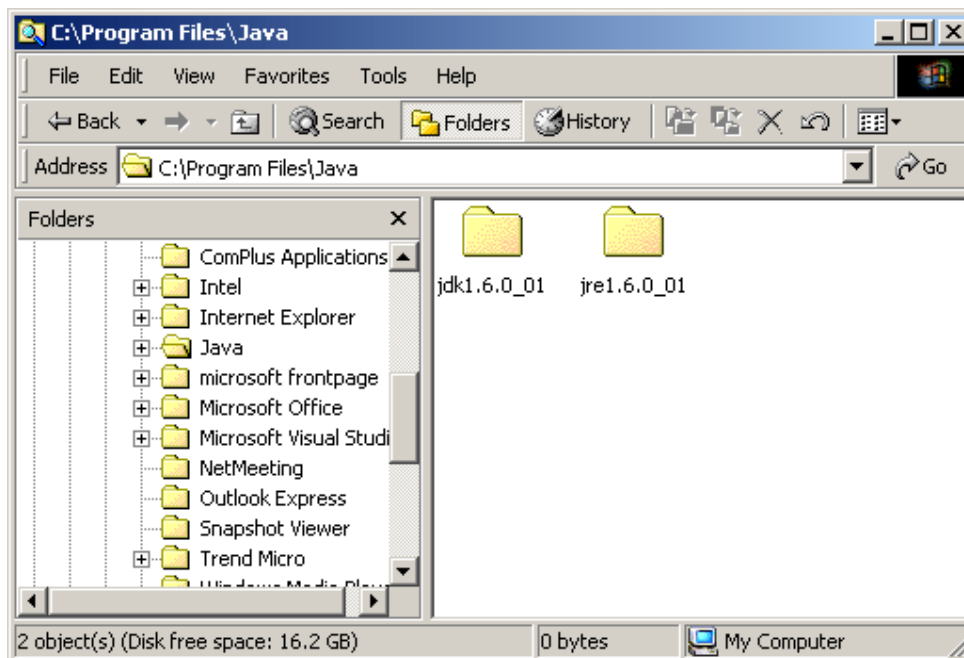
**Step 6:** Click "OK" button to start the installation.



**Step 7:** Now "Complete" window appears indicating that installation of jdk 1.6 has completed successfully. Click "Finish" button to exit from the installation process.



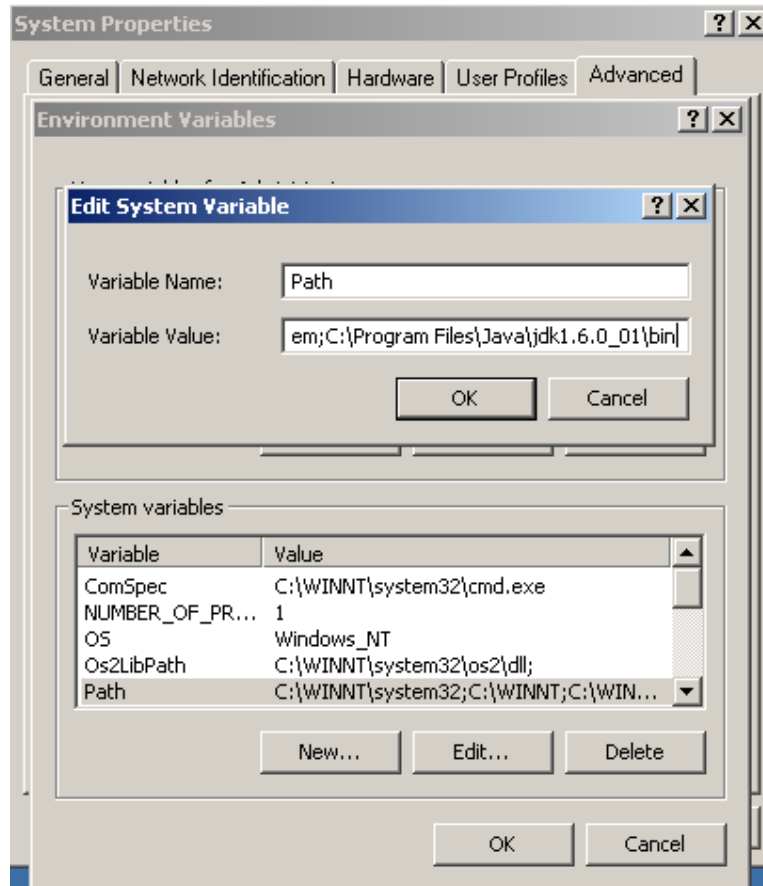
**Step 8:** The above installation will create two folders "**jdk1.6.0\_01**" and "**jre1.6.0\_01**" in "**C:\Program Files\java**" folder.



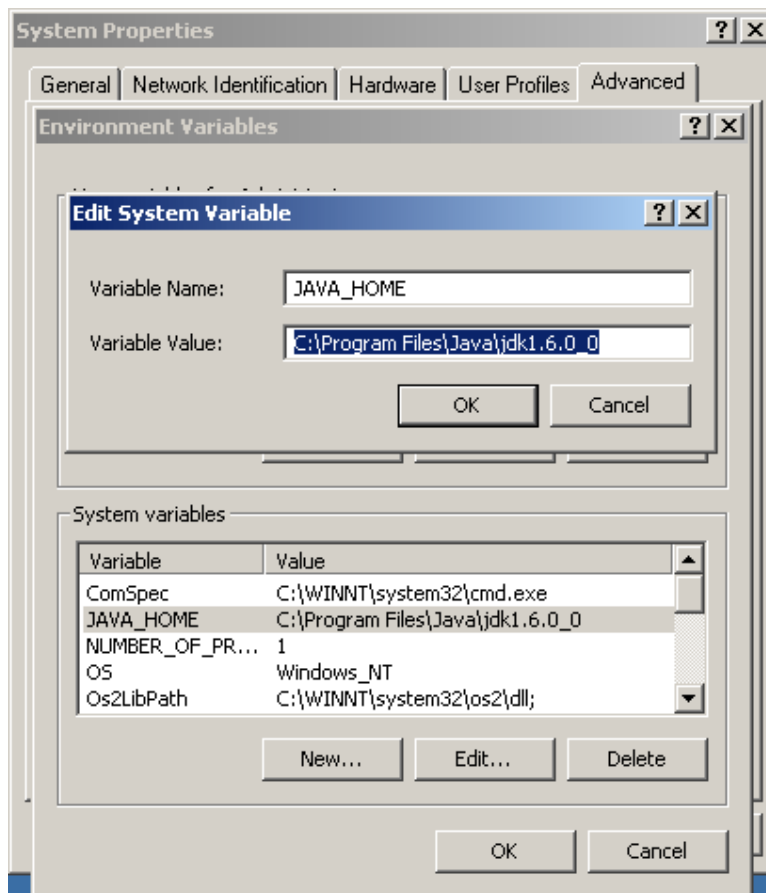
**Step 9:** To make available Java Compiler and Runtime Environment for compiling and running java programs, set the system environment variables. This is also discussed in section 4.4.4.

First of all select "**My Computer**" icon and right click the mouse button. Now click on "**system Properties**" option. It provides the "**System Properties**" window; click the

'Advanced' tab. Then Click the "Environment Variables" button. It provides "Environment Variables" window. Now select the **path** in **System variables** and click 'Edit' button. The "Edit System Variable" window will open. Add "**c:\Program Files\Java\jdk1.6.0\_01\bin**" to 'variable value' and click 'Ok', 'Ok' and 'Ok' buttons.



**Step 10:** Now set the **JAVA\_HOME** variable and set its value to " **C:\Program Files\Java\jdk1.6.0\_01** ". If this variable has not been declared earlier then create a new system variable by clicking on "New" button and give variable name as "**JAVA\_HOME**" and variable value as " **C:\Program Files\Java\jdk1.6.0\_01** ". Now click "OK". This variable is used by other applications to find jdk installation directory. For example, Tomcat server needs "**JAVA\_HOME**" variable to find the installation directory of jdk.



**Step 11:** Now this is the final step to check that you have installed jdk successfully and it is working fine. Just go to the command prompt and type **javac** and hit enter key you will get the screen as shown below:

```
Command Prompt
C:\>javac
Usage: javac <options> <source files>
where possible options include:
  -g               Generate all debugging info
  -g:none          Generate no debugging info
  -g:{lines,vars,source} Generate only some debugging info
  -nowarn          Generate no warnings
  -verbose         Output messages about what the compiler is doing
  -deprecation     Output source locations where deprecated APIs are used
  -classpath <path> Specify where to find user class files and annotations
  -bootclasspath <path> Specify where to find bootstrap class files
  -sourcepath <path> Specify where to find input source files
  -extdirs <dirs> Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -proc:{none,only} Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path> Specify where to find annotation processors
  -d <directory> Specify where to place generated class files
  -s <directory> Specify where to place generated source files
  -implicit:{none,class} Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding> Specify character encoding used by source files
  -source <release> Provide source compatibility with specified release
  -target <release> Generate class files for specific VM version
  -version         Version information
  -help           Print a synopsis of standard options
  -Akey[=value] Options to pass to annotation processors
  -X             Print a synopsis of nonstandard options
  -J<flag>        Pass <flag> directly to the runtime system

C:\>
```

Now you can create, compile and run java programs.



## Bibliography

Access-board.gov. The Rehabilitation Act Amendments (Section 508) – Online at:

<http://www.access-board.gov/sec508/guide/act.htm>

Access Now. Miami Daily Business Review 10/7/02 - The ADA and The Internet

<http://www.adaaccessnow.org/internet.htm>

AFB - American Foundation for the blind Online at: <http://afb.org/Section.asp?SectionID=15>

Allen, J., Hunnicut, M. and Klatt, D. (1987) From Text to Speech: The MITalk system, Cambridge University Press

Americans with Disability Act (ADA). Doe v. Mutual of Omaha Insurance Company (1999)

<http://www.ada.gov/briefs/doebr.doc>

AOL Launches Real-Time Instant Messaging Targeted to Deaf and Hard of Hearing Users

Online at: <http://www.pr-inside.com/aol-launches-real-time-instant-messaging-targeted-r388334.htm>

Apache HTTP Web Server. Online at: [http://httpd.apache.org/ABOUT\\_APACHE.html](http://httpd.apache.org/ABOUT_APACHE.html)

Arons B, (1993) SpeechSkimmer: Interactively skimming recorded speech, Proceedings of the Sixth ACM Symposium on User Interface Software and Technology, Atlanta, USA, 3-5th November 1993, 187-196.

Bednar, J. and Robertson, D (2005) Development Methodologies. Online at:

[http://www.inf.ed.ac.uk/teaching/courses/seoc2/2004\\_2005/slides/methodologies.pdf](http://www.inf.ed.ac.uk/teaching/courses/seoc2/2004_2005/slides/methodologies.pdf)

Bell, T. W. (2002) Access Now, Inc. v. Southwest Airlines, Co

<http://www.tomwbell.com/NetLaw/Ch02/AccessNow.html>

Berinato, S. (2001). "The Secret to Software Success". Retrieved article from www.cio.com.

Online at: [http://www.cio.com/article/30341/The\\_Secret\\_to\\_Software\\_Success?page=1](http://www.cio.com/article/30341/The_Secret_to_Software_Success?page=1)

Bigham, J., Aller, M., Brudvik, J., Leung, J., Yazzolino, L. and Ladner, R. - Inspiring Blind High School Students to Pursue Computer Science with Instant Messaging Chatbots. Online

at: <http://webinsight.cs.washington.edu/papers/sigcse.pdf>

Blanck, P. and Sandler, L. (2000) ADA Title III and the Internet: Technology and Civil Rights. Online at:

[http://disability.law.uiowa.edu/lhpdc/publications/documents/blancketaldocs/ADA\\_Title\\_3\\_and\\_Internet.pdf](http://disability.law.uiowa.edu/lhpdc/publications/documents/blancketaldocs/ADA_Title_3_and_Internet.pdf)

Bovs, L. and den Os, E., (1999) "Applications of Speech Technology: Designing for Usability". Proc. IEEE Workshop on ASR and Understanding, 353-362

Bretthauer, D. (2001) - Open Source Software: A History. UConn Libraries Published Works. Online at:

[http://digitalcommons.uconn.edu/cgi/viewcontent.cgi?article=1009&context=libr\\_pubs](http://digitalcommons.uconn.edu/cgi/viewcontent.cgi?article=1009&context=libr_pubs)

Brunet, P., Feigenbaum, B., Harris, K., Laws, C., Schwerdtfeger, R. and Weiss, L. (2005). Accessibility requirements for systems design to accommodate users with vision impairments. Online at: <http://www.research.ibm.com/journal/sj/443/brunet.pdf>

Buhler, P, Starr, C., Schroder, W. and Vidal, J. (2004) Preparing for Service-Oriented Computing: a composite design pattern for stubless Web service invocation. Online at: <http://jmvidal.cse.sc.edu/papers/buhler04b.pdf>

Buzzi, M, Andronico, P. and Leporini, B. (2004) Accessibility and Usability of Search Engine Interfaces: Preliminary Testing. Online at: [http://www.ui4all.gr/workshop2004/files/ui4all\\_proceedings/adjunct/accessibility/58.pdf](http://www.ui4all.gr/workshop2004/files/ui4all_proceedings/adjunct/accessibility/58.pdf)

Cabinet Office. Online Annual Report, 2002. Online at: [http://archive.cabinetoffice.gov.uk/e-envoy/reports-annrep-2002-pdf/\\$file/annualreport02.pdf](http://archive.cabinetoffice.gov.uk/e-envoy/reports-annrep-2002-pdf/$file/annualreport02.pdf)

Chen, Z. (2003) Building Eclipse Instance Messenger. Online at: <http://www.scs.carleton.ca/~arpwhite/documents/honoursProjects/paul-chen-2003.pdf>

Cockburn, A. (1997). Structuring Use Cases with Goals. Online at: <http://atlas-project-tdaq-awg.web.cern.ch/atlas-project-tdaq-awg/Documents/StructuringUseCasesWithGoals.pdf>

Coffin, R. and Lane, D. - A Practical Guide to Seven Agile Methodologies, Part 2 Article posted on devx.com. Online at: [www.devx.com/architect/Article/32836/1954](http://www.devx.com/architect/Article/32836/1954)

Comino, S. and Manenti, F. (2003) - Open Source vs. Closed Source Software - Public Policies in the Software Market. Online at: <http://opensource.mit.edu/papers/cominomanenti.pdf>

Compernelle, V (1989) "Speech Recognition in noisy environments with the aid of microphone arrays", Proc. European Conference on Speech Communication and Technology, Paris, pp. 657-660

Constantine, L. (2000) What do users want? Engineering usability into Software. Online at: <http://www.foruse.com/articles/whatusers.pdf>

Dirk, G. and Dusty, S. (2003) The Linux Kernel & the File System Subsystem : An Architectural Overview. Online at: <http://azungu.dyndns.org/files/gevpaper.pdf>

Disability Discrimination Act 1995. Online at: [http://www.opsi.gov.uk/acts/acts1995/ukpga\\_19950050\\_en\\_1](http://www.opsi.gov.uk/acts/acts1995/ukpga_19950050_en_1)

Dolan, D. (2004) The academic journal of the Institute of Technology Blanchardstown. Special Edition for ITB Research Conference 2004 Conference papers 22/23 April 2004

Dubinsky, Y. and Hazzan, O. (2004) Roles in Agile Software Development Teams. Online at: [http://edu.technion.ac.il/Courses/cs\\_methods/eXtremeProgramming/XP\\_Papers/Dubinsky&HazzanXP2004includingAppendix.pdf](http://edu.technion.ac.il/Courses/cs_methods/eXtremeProgramming/XP_Papers/Dubinsky&HazzanXP2004includingAppendix.pdf)

Edwards, K. (2001) Epistemic Communities, Situated Learning and Open Source Software Development. Online at: <http://opensource.mit.edu/papers/kasperedwards-ec.pdf>

Eklund, R. and Lindström, A., (1998). How to handle "foreign" sounds in swedish text-to-speech conversion: Approaching the "Xenophone" problem. 5th International Conference on Spoken Language Processing, Sydney, Australia November 30 - December 4, 1998

Elliott, M. (2008) Examining the success of computerization movements in the ubiquitous computing era: Free and Open Source Software Movements. Online at: <http://www.crito.uci.edu/si/resources/elliott.pdf>

Elliott, M and Scacchi, W. (2003) - Free Software Development: Cooperation and Conflict in A Virtual Organizational Culture. Revised version submitted to: S. Koch (ed.), Free/Open Source Software Development, IDEA Publishing, 2004 Online at: <http://www.ics.uci.edu/~wscacchi/Papers/New/Elliott-Scacchi-BookChapter.pdf>

Engelbert, D. and English, W (1968). A research center for augmenting human intellect. Online at: <http://www.bootstrap.org/augdocs/friedewald030402/researchcenter1968/ResearchCenter1968.html>

Fichten, S., Asuncion, J., Barile, M., Fossey, M., Robillard, C. and Wolforth, J. (2001). Computer technologies for postsecondary students with disabilities II: Resources and recommendations for postsecondary service providers. Journal of Postsecondary Education and Disability, 15(1), 59-82.

Fowler, M. (1998) Use and abuse cases. Online at: <http://martinfowler.com/distributedComputing/abuse.pdf>

FreeTTS 1.2 - A speech synthesizer written entirely in the Java™ programming language. Online at: <http://freetts.sourceforge.net/docs/index.php>

Garrigue, J. (2000) Code reuse through polymorphic variants. Online at: <http://www.math.nagoya-u.ac.jp/~garrigue/papers/variant-reuse.pdf>

Georgia Northern District. Martin v. Metro Atlanta Rapid Transit Authority, 2001 <http://www.gand.uscourts.gov/documents/1001cv3255TWTinj.pdf>

Gibson Research Corp: Steve Gibson: Without XPDite, Microsoft's Patch, or XP's Service Pack 1, clicking on a simple, but malicious, URL can delete the entire contents of your directories. Online at: <http://www.grc.com/xpdite/xpdite.htm>

Gladden, J. (2000) History of Computer Graphics. Online at:  
<http://www.gladdengraphics.com/academics/GradCourses/ComputerGraphicsHistory/ResearchPaper/parcgui01.pdf>

GNU. Overview of the GNU Operating System - <http://www.gnu.org/gnu/gnu-history.html>

Hamza-Lup, F., Bot, R. and Salomie, I. (1999) Virtual University of Cluj-Napoca, A Web Based Educational Framework. Online at:  
[http://www.cs.ucf.edu/~fhamza/papers/VirtualUniversity\\_Hamza-Lup.pdf](http://www.cs.ucf.edu/~fhamza/papers/VirtualUniversity_Hamza-Lup.pdf)

Hann, I., Roberts, J., Slaughter, S., Fielding, R. (2002) - Why Do Developers Contribute to Open Source Projects? First Evidence of Economic Incentives. Online at:  
<http://opensource.ucc.ie/icse2002/HannRobertsSlaughterFielding.pdf>

Harisha, K. (2007) Software Development Process and Its Importance posted on articlesbase.com. Online at: <http://www.articlesbase.com/softwarearticles/software-development-process-and-its-importance-155037.html>

Hars, A. and Ou, S. (2001) - Working for Free? – Motivations of Participating in Open Source Projects. Proceedings of the 34th Hawaii International Conference on System Sciences – 2001. Online at:  
<http://csdl2.computer.org/comp/proceedings/hicss/2001/0981/07/09817014.pdf>

Healy, K. and Schussman, A. (2003) - The Ecology of Open-Source Software Development. Online at: <http://opensource.mit.edu/papers/healyschussman.pdf>

Herbsleb, J., Atkins, D., Boyer, D., Handel, M and Finholt, T. (2002). Introducing Instant Messaging and Chat in the Workplace. Online at:  
[http://www.crew.umich.edu/Technical%20reports/Herbsleb\\_Atkins\\_Boyer\\_Handel\\_Finholt\\_Introducing\\_instant\\_messaging\\_12\\_10\\_01.pdf](http://www.crew.umich.edu/Technical%20reports/Herbsleb_Atkins_Boyer_Handel_Finholt_Introducing_instant_messaging_12_10_01.pdf)

High Tech Center Training Unit. High Tech Center Training Unit of the Chancellor's Office of California Community Colleges. (1999, August). Distance education: Access guidelines for students with disabilities. Online at:  
[http://www.htctu.net/publications/guidelines/distance\\_ed/disted.htm](http://www.htctu.net/publications/guidelines/distance_ed/disted.htm)

Hilderink, G., Broenink, J., Vervoort, W. and Bakkers, A. (1997) Communicating Java Threads. Online at: <http://www.ce.utwente.nl/javapp/cjt/CJT-paper.PDF>

Hirsch, H.G., (1989) Speech Recognition in the Noisy Car Environment, Proc. European Conference on Speech Communication and Technology, Paris, pp. 652-655

Horton, S. (2005) Accessible Design Guidelines. Online at:  
<http://www.dartmouth.edu/~webteach/resources/download/guidelines5.pdf>

IEEE 829. Test Plan Outline (IEEE 829 Format). Foundation Course in Software Testing. Online at: [itisfun.tistory.com/attachment/jk2.pdf](http://itisfun.tistory.com/attachment/jk2.pdf)

Jayaswal, B. and Patton, P. (2006) *Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software*. Chapter 1: Software Development Methodology Today

Johnson, J. (2001) Economics of Open Source Software. Online at:  
<http://opensource.mit.edu/papers/johnsonopensource.pdf>

Jorgensen, N. (2001) Putting it All in the Trunk. Incremental Software Development in the FreeBSD Open Source Project. Online at:  
<http://webhotel2.ruc.dk/nielsj//research/publications/freebsd.pdf>

Junqua, J. (2000) *Robust Speech Recognition in Embedded Systems and PC applications*, Chapter 3, Page 99.

Karaali O, Corrigan G, Gerson I, (1996) "Speech Synthesis with Neural Networks." Invited paper, World Congress on Neural Networks, San Diego, September 1996, pages: 45-50. Online at: [http://arxiv.org/PS\\_cache/cs/pdf/9811/9811031v1.pdf](http://arxiv.org/PS_cache/cs/pdf/9811/9811031v1.pdf)

KDE, The KDE Instant Messenger. Online at: <http://kopete.kde.org/>

King, A., Evans, G. and Blenkhorn, P. (2004) - WebbIE: a Web Browser for Visually Impaired People. Online at: <http://rehab-www.eng.cam.ac.uk/cwuaat/04/08-pat-cmc-king-webbie.pdf>

Koch, N., Baumeister, H., Hennicker, R. and Mandel, L. (2000) - Extending UML to Model Navigation and Presentation in Web Applications. Online at:  
<http://www.pst.informatik.unimuenchen.de/personen/koehn/ExtendingUML.pdf>

Kotelly, B. (2003) *The art and business of speech recognition*. Chapter 1, Page 9.

Kuan, J. (2002) - Open Source Software as Lead User's Make or Buy Decision: A Study of Open and Closed Source Quality. Online at:  
[http://idei.fr/doc/conf/sic/papers\\_2002/osskuan.pdf](http://idei.fr/doc/conf/sic/papers_2002/osskuan.pdf)

Kuhn, Thomas S (1961) "The Function of Measurement in Modern Physical Science", Isis, Vol. 52, No. 2 (Jun., 1961), pp. 161-193

Lee, S., Potamianos, A., and Narayanan, S., (1997). "Analysis of Children's Speech: Duration, Pitch and Formants". Eurospeech, pages 473-476

Lethbridge, T. and Laganriere, R. (2001) *Object-Oriented Software Engineering, Practical software development using UML and Java*. Ch 7. P 243.

Lyons, K. (2004) The Agile Approach. Online at:  
<http://www.agilealliance.org/system/article/file/1324/file.pdf>

Madey, G., Freeh, V. and Tynan, R. (2002) - The Open Source Software Development Phenomenon: An Analysis Based On Social Network Theory. Eighth Americas Conference on Information Systems.

McConnell, S. (1999) Open-Source Methodology: Ready for Prime Time? Online at: <http://www.stevemcconnell.com/ieeesoftware/OpenSource.pdf>

McGrane, S. (2000) Is the web truly accessible to the Disabled? Article on cnet.com. Online at: [http://www.cnet.com/4520-6022\\_1-105104-1.html](http://www.cnet.com/4520-6022_1-105104-1.html)

Microsoft Corporation (2004 - Quarterly Period Ended March 31) - United States Securities and Exchange Commission. Online at: [http://www.microsoft.com/msft/download/FY04/MSFT\\_3Q2004\\_10Q.doc](http://www.microsoft.com/msft/download/FY04/MSFT_3Q2004_10Q.doc)

Miller, M., Fredriksen, L. and So, B. (1989) - An Empirical Study of the Reliability of UNIX Utilities. Online at: [ftp://ftp.cs.wisc.edu/paradyn/technical\\_papers/fuzz.pdf](ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz.pdf)

Moreno, P. and Stern, R., (1994). Sources of Degradation of Speech Recognition in the Telephone Network. ICASSP, pages I.109-I.112.

Mozilla – online at: <http://www.mozilla.org/mozorg.html>

MSN Messenger Connect Service . Online at: <http://www.microsoft.com/presspass/press/2002/nov02/11-13corporateimpr.msp>

Muthusamy, Y., Agarwal, R., Gong Y., Viswanathan, V. (1999). Speech enabled Information Retrieval in the Automobile Environment. ICASSP. pp 2259-2262

MySQL – Online at: <http://en.wikipedia.org/wiki/MySQL#History>

National Telecommunications and Information Administration. A nation online - How Americans Are Expanding Their Use of the Internet. Online at: <http://www.ntia.doc.gov/ntiahome/dn/anationonline2.pdf>

Nelson, G. (2007) Reexamining the Waterfall Model. Online at: [http://www.fiercekitten.com/blog/papers/635ResearchPaper1\\_GNelson.pdf](http://www.fiercekitten.com/blog/papers/635ResearchPaper1_GNelson.pdf)

Netcraft Statistics. Market Share for Top Servers Across All Domains August 1995 - June 2008. Online at: <http://news.netcraft.com/archives/2008/06/>

Neumeyer, L., Franco, H., Abrash, V., Julia, L., Ronen, O., Bratt, H., Bing, J., Digalakis, V. and Rypa, M. (1998) WebGraderTM: A Multilingual Pronunciation Practice Tool. Online at: <http://www.speech.sri.com/people/leo/papers/still98-webgrader.pdf>

Newell, A. (2003) Spoken Language and e-inclusion. Euro speech 2003 – Geneva. Online at: <http://www.computing.dundee.ac.uk/projects/UTOPIA/publications/Eurospeech%202003.pdf>

New Rowley TechView. Online at: [http://www.newrowley.com/2006/08/introducing\\_the\\_wicket\\_a\\_new\\_a.html](http://www.newrowley.com/2006/08/introducing_the_wicket_a_new_a.html)

NextUp Talker. Online at: <http://www.nextup.com/>

Nielsen, J. (1994) Ten Usability Heuristics. Online at: [http://www.useit.com/papers/heuristic/heuristic\\_list.html](http://www.useit.com/papers/heuristic/heuristic_list.html)

- Nuvolari, A. (2003) - Open Source Software Development: Some Historical Perspectives. Online at: <http://opensource.mit.edu/papers/nuvolari.pdf>
- Oblinger, D. (2004). The Next Generation of Educational Engagement. Journal of Interactive Media in Education, 2004, Special Issue on the Educational Semantic Web. Online at: <http://www-jime.open.ac.uk/2004/8/oblinger-2004-8.pdf>
- Ogilvy, D. *Confessions of an Advertising Man*, Chapter 5 Preventing Bugs with Unit Testing, P 69
- Pallett, D.S., (1994) Benchmark tests for the ARPA Spoken Language Program. ARPA's Human Language Technology workshop, pages 49-74
- Pederson, D. (2003) A Look into C# Threads. Online at: <http://people.msoe.edu/~taylor/cs384/pedersod.pdf>
- Peeling, N and Satchell J, (2001) Analysis of the Impact of Open Source Software. Online at: [http://www.govtalk.gov.uk/documents/QinetiQ\\_OSS\\_rep.pdf](http://www.govtalk.gov.uk/documents/QinetiQ_OSS_rep.pdf)
- Perriot, F. and Szor, P. (2003) - An Analysis of the Slapper Worm Exploit. Online at: <http://www.symantec.com/avcenter/reference/analysis.slapper.worm.pdf>
- PHP – Online at: <http://en.wikipedia.org/wiki/PHP#History>
- Pitt, I., Edwards, A., (2003) *Design of Speech-based Devices*
- Preece, J., Maloney-Krichmar, D. and Abras, C. (2003). History and emergence of online communities. Online at: <http://www.ifsm.umbc.edu/~preece/paper/6%20Final%20Enc%20preece%20et%20al.pdf>
- Quiroga, L., Crosby, M. and Iding, M. (2004) Reducing Cognitive Load. Online at: <http://csdl2.computer.org/comp/proceedings/hicss/2004/2056/05/205650131a.pdf>
- Rabiner, L.R., Juang, B.H., (2004) “Automatic Speech Recognition – A Brief History of the Technology Development” , Georgia Institute of Technology, Atlanta
- Ragragio, I. (2007) - Full Disclosure – IT Security. Article on IT Security. Online at: <http://www.itsecurity.com/features/full-disclosure-032907/>
- Rajendran, R. (2004) White paper on Unit Testing, Director, Deccanet Designs Ltd. Online at: <http://www.mobilein.com/WhitePaperonUnitTesting.pdf>
- Raymond, E. (2000) The Cathedral and the Bazaar. Online at: <http://gnuwin.epfl.ch/articles/en/cathedralbazaar/cathedral-bazaar.pdf>
- Redmonk. Agile QA Automation. A Redmonk Briefing Document, Nov 6 2006. Online at: <http://redmonk.com/public/agile/agileqatools.pdf>
- Reeves, B. and Nass, C. *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places* (New York: Cambridge University Press, 1996).



- Reich S (1980) "Significance of pauses for speech perception", Journal of Psycholinguistic Research., 9(4), 379-389
- Reis, C. and Mattos Fortes, R. (2002) - An Overview of the Software Engineering Process and Tools in the Mozilla Project. Online at:  
<http://www.async.com.br/~kiko/papers/mozse.pdf>
- Resig, J., Dawara, S., Homan, C. and Teredesai, A. (2004) Extracting Social Networks from Instant Messaging Populations. Online at:  
<http://www.cs.cmu.edu/~dunja/LinkKDD2004/John-Resig-LinkKDD-2004.pdf>
- Robinson CP and Eberts RE (1987) Comparison of speech and pictorial displays in a cockpit environment, Human Factors, 29(1), 31-44.
- Roetter, A. (2001) Writing multithreaded Java applications. Online at:  
<http://www.ibm.com/developerworks/library/j-thread.html>
- Romanovsky, A. (2001) Exception Handling in Component-Based System Development. Online at: [http://rogue.ncl.ac.uk/file\\_store/trs/724.pdf](http://rogue.ncl.ac.uk/file_store/trs/724.pdf)
- Russell N. and Stafford, N. (2002) Trends in ICT Access and Use. Available online at:  
[www.dfes.gov.uk/research/data/uploadfiles/RR358.doc](http://www.dfes.gov.uk/research/data/uploadfiles/RR358.doc)
- Sauer, G., Hochheiser, H., Feng, J and Lazar, J. (2008) Towards a Universally Usable CAPTCHA Online at: <http://cups.cs.cmu.edu/soups/2008/SOAPS/sauer.pdf>
- Schroeter, J. (2006) *Text to-Speech (TTS) Synthesis*.
- Shankland, S. (2004) – Novell eyes move into embedded Linux. Article on cnet.com. Online at: <http://news.cnet.com/>
- Smith, S. and Mosier, J. (1986) Guidelines for designing user Interface Software. Online at: <http://hcibib.org/sam/>
- Sphinx-4 - A speech recognizer written entirely in the Java™ programming language. Online at: <http://cmusphinx.sourceforge.net/sphinx4/>
- Stallman, R. - Why "Open Source" misses the point of Free Software. Online at:  
<http://www.gnu.org/philosophy/open-source-misses-the-point.html>
- Stallman, R. The GNU Project. Online at: <http://www.gnu.org/gnu/thegnuproject.html>
- Stojanovic, Z., Dahanayake, A. and Sol, H. (2003). Modeling and Architectural Design in Agile Development Methodologies. Online at:  
<http://www.emmsad.org/2003/Final%20Copy/27.pdf>
- Stotts, D. (2002) - The Perl Scripting Language. Online at: <http://rockfish-cs.cs.unc.edu/comp144/ieperl.pdf>



Sunmanagers.org - Bind error - address already in use Online at:

<http://www.sunmanagers.org/archives/1993/1164.html>

Tang, J., Yankelovich, N., Begole, J., Kleek, M., Li, F. and Bhalodia, J. (2001) ConNexus to Awarenex: Extending Awareness to mobile Users. Online at:

<http://people.csail.mit.edu/emax/papers/connexus.pdf>

Teppo, A. and Vuorimaa, P. (2001) Speech Interface Implementation for XML Browser. Proceedings of the 2001 International Conference on Auditory Display, Espoo, Finland, July 29-August 1, 2001. Online at:

<http://www.acoustics.hut.fi/icad2001/proceedings/papers/teppo.pdf>

thesportsjournal.org. The PGA Tour vs. Casey Martin, 1998 -

<http://www.thesportjournal.org/article/pga-tour-vs-casey-martin>

Thompson, T. (2005) - Universal Design & The Web: Techniques, Problems, and Solutions.

Online at: [http://tcc.kcc.hawaii.edu/2008/tcc/presentation-format2\\_files/sample\\_paper.pdf](http://tcc.kcc.hawaii.edu/2008/tcc/presentation-format2_files/sample_paper.pdf)

Times Daily -

[http://www.timesdaily.com/apps/pbcs.dll/section?category=NEWS&template=wiki&text=open\\_source\\_software](http://www.timesdaily.com/apps/pbcs.dll/section?category=NEWS&template=wiki&text=open_source_software)

University of Pennsylvania. ENIAC's recessive. Online at:

<http://www.upenn.edu/computing/printout/archive/v12/4/pdf/gene.pdf>

Walker W, Lamere P, Kwok P (2002) FreeTTS - A Performance Case Study. Online at: [research.sun.com/techrep/2002/abstract-114.html](http://research.sun.com/techrep/2002/abstract-114.html)

Warren, S. - Advantages Disadvantages Of Internet Marketing: The Good The Bad And The Ugly. Article on ESL Teachers Board. Online at: <http://www.eslteachersboard.com/cgi-bin/website/index.pl?read=132>

Weber, S. (2000) – The Political Economy of Open Source Software. BRIE Working Paper 140, E-economy Projectä Working Paper 15, June 2000. Online at: <http://e-conomy.berkeley.edu/publications/wp/wp140.pdf>

West, J. and Dedrick, J. (2001) - Open Source Standardization: The Rise of Linux in the Network Era. Proceedings of the 34th Hawai'i International Conference on System Sciences, Jan. 2001. Online at: <http://www.cob.sjsu.edu/OpenSource/Research/WestDedrick2001.pdf>

Weimer, W. and Necula, G. (2004) Finding and Preventing RunTime Error Handling Mistakes. Online at: <http://www.cs.virginia.edu/~weimer/papers/WN-FindingAndPreventing.pdf>

Weise, T. and Baer, P. (2008) 1st Kassel Student Workshop on Security in Distributed Systems - KaSWoSDS'08. Online at: [https://kobra.bibliothek.uni-kassel.de/bitstream/urn:nbn:de:hebis:34-2008041421155/3/Technicalreport2008\\_1.pdf](https://kobra.bibliothek.uni-kassel.de/bitstream/urn:nbn:de:hebis:34-2008041421155/3/Technicalreport2008_1.pdf)

Williams, L. (2007) A Survey of Agile Development Methodologies. Online at:  
<http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf>

Wilpon, J., and Jacobsen, C., (1996). “A study of speech recognition for children and the elderly”, ICASSP, pages 3-9

Wolak, W. (2001) System Development: Research Paper 1. SDLC on a Diet. Online at:  
<http://codecourse.sourceforge.net/materials/System-Development-Life-Cycle.html>

W3C. Authoring Tool Accessibility Guidelines 1.0. Online at: <http://www.w3.org/TR/WAI-AUTOOLS/>

W3C. User Agent Accessibility Guidelines 1.0. Online at:  
<http://www.w3.org/TR/WAIUSERAGENT/>

W3C. Web Content Accessibility Guidelines 1.0. Online at:  
<http://www.w3.org/TR/WCAG10/>

Wikipedia – Rovert Tappan Morris. Online at:  
[http://en.wikipedia.org/wiki/Robert\\_Tappan\\_Morris](http://en.wikipedia.org/wiki/Robert_Tappan_Morris)

Zahran, S. (1998) *Software Process Improvement*, Addison-Wesley, p 4